UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Gašper Grom

Integracijski portal podatkov o zdravem in aktivnem življenju

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2019

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Gašper Grom

Integracijski portal podatkov o zdravem in aktivnem življenju

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Marko Bajec

Ljubljana, 2019

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Cilj naloge je proučiti obstoječe platforme za zajem podatkov o aktivnem in zdravstvenem življenju posameznikov ter razviti portal, ki bo združeval podatke na enem mestu. Primerov takšnih platform je veliko. Med najbolj znanimi in neodvisnimi od pametnih naprav sta platformi Google Fit in Apple Health Kit, ki podpirata množico različnih pametnih naprav. Poleg tega obstajajo tudi platforme, ki jih ponujajo proizvajalci pametnih naprav, npr. Fitbit, Garmin ipd. Z nalogo želimo pokazati, kako je možno podatke o zdravem in aktivnem življenju integrirati v poljubno novo aplikacijo, brez da bi za to morali komunicirati z vsako pametno napravo posebej. V okviru naloge bomo proučili podatkovne modele izbranih ponudnikov platform in razvili skupen podatkovni model, v katerem bodo na voljo entitete, ki jih je moč pridobiti iz vseh izbranih platform. Podatkovni model bo sledil standardu FHIR, v kolikor bo to mogoče.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Gašper Grom sem avtor diplomskega dela z naslovom:

Integracijski portal podatkov o zdravem in aktivnem življenju

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Bajca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) in ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu prek univerzitetnega spletnega arhiva.

V Ljubljani, dne 2. januarja 2019 Podpis avtorja:

Kazalo

Povzetek

Abstract

1. Uvod 1

2. Arhitektura sistemov za zajem podatkov o aktivnem in zdravem življenju 3

2.1 Naprave za zajem podatkov 3

2.2 Mobilne aplikacije za prenos podatkov do zalednega sistema 4

2.3 Zaledni sistem 5

2.4 Aplikacije za prikaz podatkov 5

3. Naprave za zajem podatkov o aktivnem in zdravem življenju 9

3.1 Pametne ure 9

3.2 Pametne zdravstvene merilne naprave 10

4. Platforme za hranjenje podatkov o aktivnem in zdravem življenju 11

4.1 Google Fit 11

4.2 Apple Health Kit 12

4.3 Garmin Connect 13

4.4 Fitbit 14

5. Tehnologije rešitve 16

5.1 Vue.js 16

5.2 Nuxt.js 16

5.3 Node.js 16

5.4 Express.js 17

5.5 MongoDB 17

6. Izdelava integracije platform za prikaz podatkov o aktivnem in zdravem življenju 19

6.1 Pridobivanje dostopa do podatkov uporabnika 19
6.1.1 Google Fit 19
6.1.2 Apple Health Kit 22
6.1.3 Garmin Connect 22
6.1.4 Fitbit 22
6.2 Pridobivanje podatkov 24
6.2.1 Google Fit 24
6.2.2 Fitbit 25
6.3 Načrtovanje skupnega podatkovnega modela 27
6.3.1 Google Fit 27
6.3.2 Apple Health Kit 28
6.3.3 Garmin 28
6.3.4 Fitbit 29
6.3.5 Združeni podatkovni model 29
6.4 Združitev podatkov 31
6.5 Priključitev nove platforme 32
7. Končna rešitev 33
8. Sklepne ugotovitve 37

Seznam uporabljenih kratic

Kratica	Angleško	Slovensko
HTML	Hyper Text Markup Language	Jezik za označevanje nadbesedila
API	Application programming interface	Aplikacijski programski vmesnik
SDK	Software development kit	Programska knjižnica

Povzetek

Diplomsko delo prikazuje izdelavo integracijskega portala za pregled podatkov o zdravem in aktivnem življenju. Opisane so štiri najbolj uporabljene platforme za beleženje teh podatkov in tehnologije, ki so bile uporabljene za izdelavo rešitve. Podan je tudi postopek izdelave rešitve, od pridobivanja dovoljenj do pridobivanja podatkov, ter združevanje teh v enotno obliko.

Ključne besede: integracijski portal, zdravje, aktivnosti.

Abstract

This thesis outlines the development of an integration portal used to overview data about a healthy and active lifestyle. Firstly, the four platforms that were integrated into the portal are described. Then there is presented the development process, which includes acquiring permissions and data as well as combining them into a singular entity.

Key words: integration portal, health, activities

1. Uvod

Veliko svetovnih podjetij je ustvarilo platforme, s katerimi je možno beležiti podatke o zdravem in aktivnem življenju. Omogočajo tudi pregled nad njimi, da lahko uporabniki vidijo spremembe v svojem zdravju ali napredek pri aktivnostih. Platforme pridobivajo podatke s pomočjo pametnih telefonov in pametnih naprav, ki so specializirane za merjenje določenih vrst meritev.

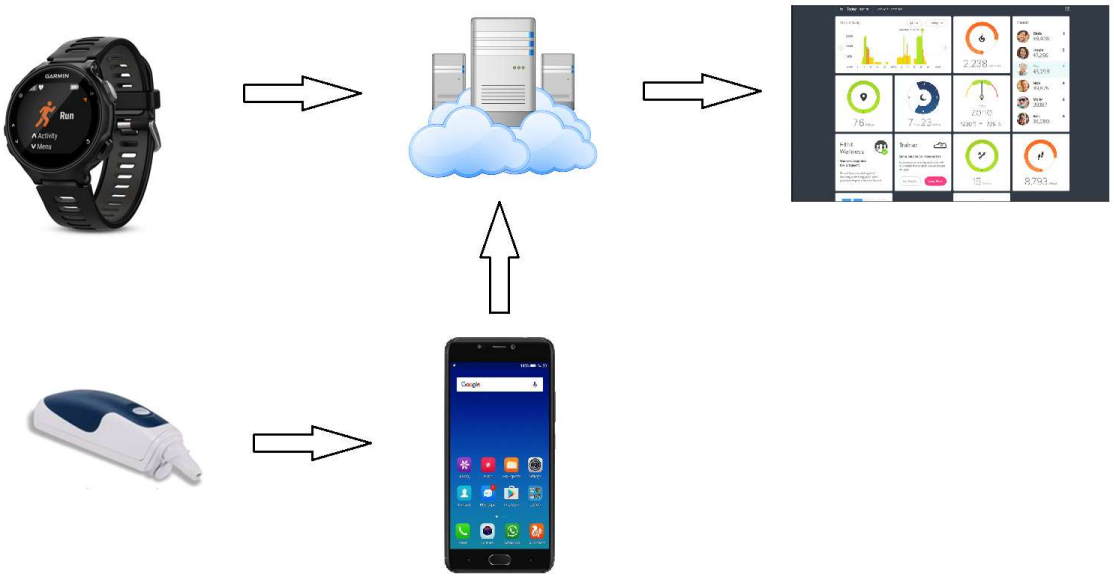
Problem pri teh platformah se pojavlja pri njihovi različnosti in njihovih omejitvah. Določene platforme so vezane le na določen operacijski sistem, kot recimo Apple Health Kit, ki omogoča aplikacijo le na operacijskem sistemu iOS, Google Fit pa le na operacijskem sistemu Android. Tako uporabnik nima enotnega sistema za pregled svojih meritev. Problem nastane tudi, če uporabnik uporablja več platform za merjenje podatkov o aktivnem in zdravem življenju in nima platforme, ki bi mu omogočala pregled združenih podatkov iz vseh platform.

Cilj diplomskega dela je izdelava integracijskega portala, ki pridobiva podatke iz večjega števila platform za beleženje podatkov o aktivnem in zdravem življenju. Te podatke potem združuje v enotno obliko in omogoča enoten pregled nad podatki vseh platform. Sam integracijski portal mora omogočati tudi priključevanje novih platform.

V diplomskem delu je najprej opisana splošna arhitektura sistemov za zajem podatkov o zdravem in aktivnem življenju, nato so opisane naprave za zajem takšnih podatkov in njihove vrste, sledi opis obstoječih platform za beleženje takšne vrste podatkov. Podana je tudi izdelava rešitve oz. integracijskega portala. Najprej so opisane tehnologije, ki so bile uporabljene za izdelavo rešitve, nato pa postopek pridobivanja podatkov, načrtovanje skupnega podatkovnega modela in združevanje podatkov. Na koncu je opisan postopek dodajanja nove platforme v integracijski portal in podan prikaz končne rešitve.

2. Arhitektura sistemov za zajem podatkov o aktivnem in zdravem življenju

Sistemi za zajem podatkov o aktivnem in zdravem življenju so razdeljeni na štiri glavne dele. To so naprave za zajem podatkov, mobilne aplikacije za prenos podatkov do zalednega sistema, zaledni sistem in aplikacije za prikaz podatkov. V naslednjih podpoglavjih je podan bolj podroben opis posameznih delov arhitekture takih sistemov.



Slika 2.1: Splošna arhitektura sistemov

2.1 Naprave za zajem podatkov

Prvi del arhitekture sistemov za zajem podatkov o aktivnem in zdravem življenju in hkrati tudi najpomembnejši so naprave za zajem podatkov. To so naprave, katerih naloga je zajemanje podatkov in posredovanje le-teh. Takšne naprave ne merijo le podatkov iz okolja, temveč tudi korake, srčni utrip, krvni pritisk, kalorijsko porabo itd. Primeri takih naprav so lahko tehtnice, merilniki pritiska, merilniki krvnega sladkorja, števec na kolesu itd. Na Sliki 2.2 je primer naprave za zajem podatkov proizvajalca Garmin, ki meri aktivnost pri teku

ali kolesarjenju. Meri čas, razdaljo, hitrost in porabljene kalorije. Kot že omenjeno, je glavna naloga teh naprav merjenje podatkov iz okolice. Te naprave morajo nato podatke posredovati do zalednega sistema, direktno ali prek pametnega mobilnega telefona. Direktno lahko posredujejo podatke zalednemu sistemu le, če naprava vsebuje kartico SIM in je tako lahko naprava direktno povezana v omrežje ter lahko pošilja podatke zalednemu sistemu. Če pa naprava ne vsebuje kartice SIM, je edini možni način prenosa podatkov prek pametne mobilne naprave, kjer prek povezave Bluetooth pošlje podatke na mobilno napravo, ta pa nato podatke posreduje v zaledni sistem.

Slika 2.2: Naprava za zajem podatkov proizvajalca Garmin

2.2 Mobilne aplikacije za prenos podatkov do zalednega sistema

Drugi del arhitekture takšnih sistemov je mobilna aplikacija za prenos podatkov do zalednega sistema. Naloga mobilne aplikacije je pridobitev podatkov iz naprav za zajem podatkov in posredovanje le-teh do zalednega sistema. Aplikacija mora najprej poskrbeti, da je s samo napravo za zajem podatkov vzpostavljena povezava Bluetooth. Najprej mora poskrbeti, da ima telefon vključeno povezavo Bluetooth. To lahko stori tako, da uporabnika opozori, da mora pred uporabo vključiti povezavo, lahko pa aplikacija sama vključi Bluetooth z uporabnikovim dovoljenjem. Nato mora aplikacija zagotoviti, da je vzpostavljena povezava s primerno napravo za zajem podatkov. Ko je vse to storjeno in aplikacija pridobi podatke iz naprav, jih lahko posreduje do zalednega sistema. Da lahko sama aplikacija posreduje podatke do zalednega sistema, mora poslati zahtevek HTTP, za katerega je potrebna internetna povezava. Če te povezave ni, potem mora aplikacija sama zagotoviti rešitve. Tukaj ima

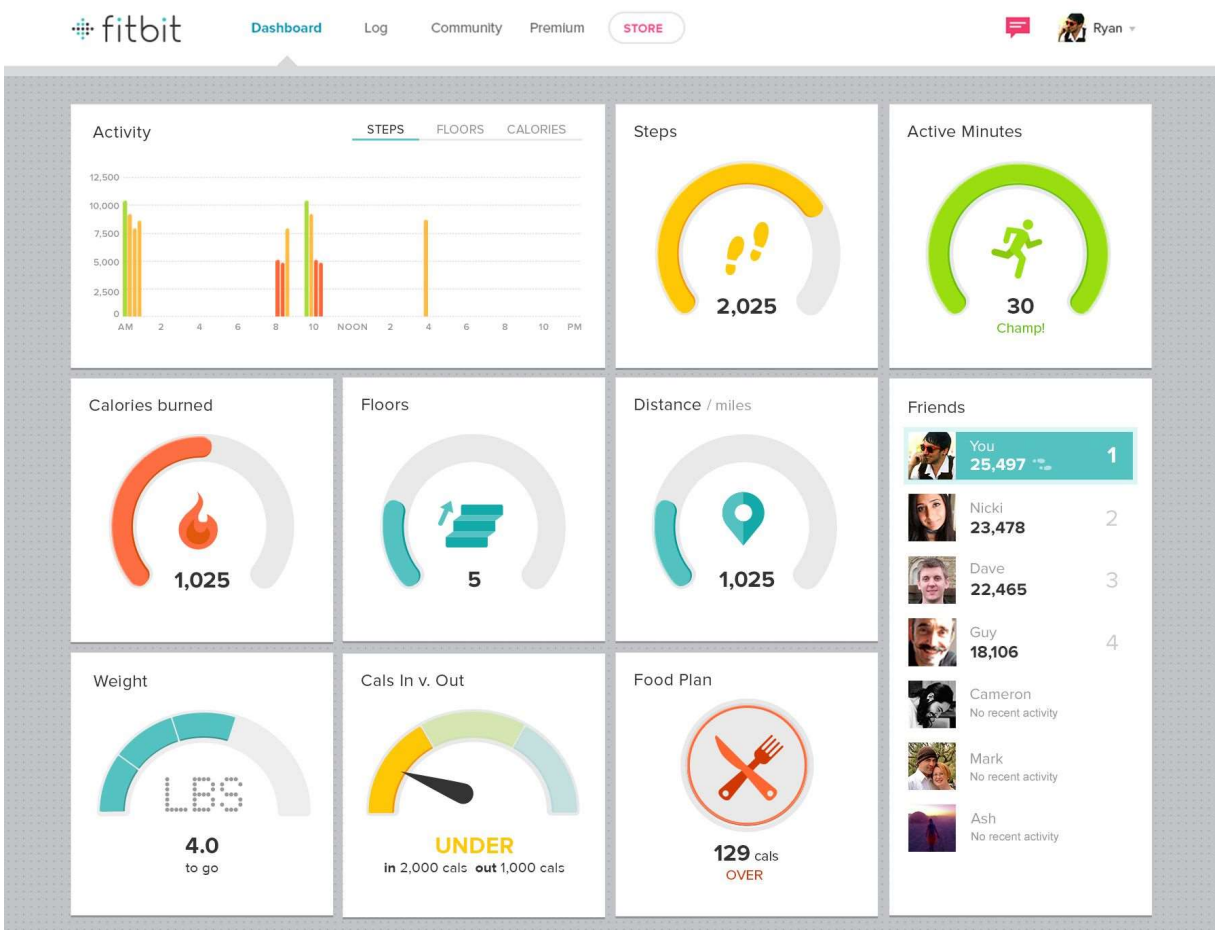
aplikacija dve možni rešitvi, in sicer uporabnika lahko obvesti, da mora vklopiti internetno povezavo, lahko pa podatke začasno shrani in jih pošlje ob ponovni vzpostavitvi povezave.

2.3 Zaledni sistem

Zaledni sistem prejme podatke mobilne aplikacije iz pametnega telefona. Njegova naloga je pravilna hramba podatkov, saj so podatki različni in uporabljajo različne enote. Sam zaledni sistem mora omogočati tudi pridobivanje podatkov iz podatkovne baze, tako da lahko platforme pridobijo podatke in jih primerno prikažejo.

2.4 Aplikacije za prikaz podatkov

Zadnja komponenta arhitekture takšnih sistemov so aplikacije za prikaz podatkov. Njihova naloga je pridobiti podatke in jih na najbolj smiseln način prikazati uporabniku. Uporabnik ima na teh aplikacijah možnost pregleda podatkov o svojem zdravem in aktivnem življenju in tako lahko hitro opazi spremembe v svojem zdravju ali napredek pri določenih aktivnostih. Nekatere platforme vsebujejo tudi druge funkcionalnosti, kot na primer priporočila glede prehrane in gibanja ali prijavljanje za donatorje. Na Sliki 2.3 je prikazan primer platforme za prikaz podatkov proizvajalca Fitbit.



Slika 2.3: Nadzorna plošča proizvajalca Fitbit

3. Naprave za zajem podatkov o aktivnem in zdravem življenju

Naprave, katerih namen je zajem podatkov, se razlikujejo med seboj po vrsti podatkov, ki jih merijo. Kar nekaj podjetij se ukvarja s prodajo pametnih naprav za zajemanje podatkov. Med večjimi sta Garmin in Fitbit, ki omogočata vrsto pametnih naprav, kot so pametne ure in naprave za merjenje zdravstvenih meritev. V naslednjih podpoglavjih je podanih nekaj primerov naprav za zajem podatkov in njihov namen.

3.1 Pametne ure

Pametne ure so najbolj pogosto uporabljene naprave za zajem podatkov. Njihov namen je beleženje podatkov o aktivnostih, kot so tek, kolesarjenje, pohodništvo, spanje, omogočajo tudi merjenje srčnega utripa. Na Sliki 3.1 je primer pametne ure proizvajalca Garmin.

Slika 3.1: Pametna ura proizvajalca Garmin

3.2 Pametne zdravstvene merilne naprave

Pametne zdravstvene merilne naprave so naprave, katerih namen je meriti zdravstvene podatke pacienta. Vsaka vrsta naprave je specializirana za merjenje določenega tipa podatkov. Na Sliki 3.2 je nekaj primerov pametnih zdravstvenih merilnih naprav. S temi napravami lahko merimo podatke o telesu, kot so temperatura, krvni sladkor, krvni pritisk, teža itd.

Slika 3.2: Pametne zdravstvene merilne naprave

4. Platforme za hranjenje podatkov o aktivnem in zdravem življenju

4.1 Google Fit

Google Fit je platforma za beleženje meritev zdravja in aktivnosti ter omogoča pregled nad njimi. Ustvaril jo je Google in je dosegljiva na operacijskih sistemih Android in Wear OS. Google Fit omogoča beleženje podatkov s pametnimi napravami, kot so pametni telefoni in pametne ure, in na tak način beleži aktivnosti, kot so tek, hoja, kolesarjenje in mnoge druge. Aplikacija omogoča tudi vnos krvnega pritiska, glukoze, utripa srca in druge zdravstvene informacije. Google Fit je bil prvič oznanjen na konferenci Google I/O leta 2014. Aplikacija je bila objavljena proti koncu istega leta. Avgusta 2018 so oznanili, da bodo v novo verzijo aplikacije vključili tudi priporočila glede na aktivnost uporabnika, ki so jih ustvarili skupaj z ameriško zvezo za srce in Svetovno zdravstveno organizacijo. Google Fit omogoča aplikacijski programski vmesnik (v nadaljevanju API) za aplikacije in naprave, da lahko dostopajo do podatkov in jih hranijo. Platforma omogoča pridobivanje podatkov prek senzorjev na napravah android in drugih napravah, kot so pametne ure, merilci srčnega utripa itd. Platforma je omejena le na mobilno napravo in majhno število drugih merilnih naprav. Uporabnik lahko tudi izbira, kdo lahko vidi njegove fitnes podatke, in si tudi sam zastavi želene cilje. Google Fit omogoča API, ki je dostopen vsakomur, ki prej registrira svojo aplikacijo v Googlovem sistemu za razvijalce. Google Fit v svoji platformi sledi standardu FHIR s področja zdravstva.

Slika 4.1: Aplikacija Google Fit

4.2 Apple HealthKit

Apple HealthKit je platforma, zgrajena za beleženje podatkov o aktivnem in zdravem življenju. Izdelal jo je Apple in vključuje mobilno aplikacijo Health in aplikacijo za pametne ure HealthKit. Mobilna aplikacija je bila javno objavljena leta 2014 in je namenjena le za naprave, ki jih poganja operacijski sistem iOS, in za pametne ure, ki jih poganja operacijski sistem watchOS. Celotna platforma omogoča beleženje podatkov prek pametne ure ali pametnega telefona in jih pošilja na zaledni sistem. Od julija 2016 aplikacija tudi omogoča, da se uporabniki lahko prijavijo kot donatorji organov, oči in tkiva. Apple HealthKit omogoča funkcionalnost za avtentikacijo uporabnika in pridobivanje podatkov le v programski knjižnici (v nadaljevanju SDK) iOS, tako da je mogoče to platformo uporabljati le v namene izgradnje aplikacij za operacijski sistem iOS. Na začetku je bila aplikacija zelo skritizirana zaradi njene počasnosti in ker ni omogočala beleženja glukoze v krvi, saj je to pomembna lastnost aplikacije, ker je ljudi s sladkorno boleznijo veliko. Podatka, kateremu standardu sledi Apple HealthKit, ni bilo na voljo.

Slika 4.2: Aplikacija Apple Health

4.3 Garmin Connect

Garmin Connect je še ena izmed platform, ki na enak način omogoča pridobivanje podatkov uporabnikov o zdravem in aktivnem življenju. Platforma je bila zgrajena v namene pridobivanja podatkov iz pametnih ur proizvajalca Garmin in hranjenja teh v njihovem zalednem sistemu. Platforma omogoča hranjenje podatkov in pregled nad njimi. Za razliko od aplikacij Google Fit in Apple HealthKit ta platforma omogoča aplikacijo na obeh operacijskih sistemih, tako na Androidu kot iOS-u. Garmin API omogoča uporabo njihovega zalednega sistema v namene hranjenja in pridobivanja izmerjenih podatkov iz izdelanih aplikacij. Vendar za dostop do API zahtevajo 5000 dolarjev, zato API ni dostopen vsakomur. Podatka, kateremu standardu sledi Garmin Connect, ni bilo na voljo.

Slika 4.3: Aplikacija Garmin

4.4 Fitbit

Fitbit je platforma, ki omogoča beleženje podatkov o zdravem in aktivnem življenju. Sama platforma beleži podatke prek pametnih ur in pametnih telefonov. Fitbit za razliko od drugih platform pokriva tudi aplikacije za računalnike. Platforma omogoča aplikacije na mobilnih operacijskih sistemih, kot so Android, iOS in Windows, prav tako tudi na računalnikih, ki imajo operacijske sisteme Windows ali MacOS. Platforma omogoča tudi pregled nad meritvami na spletni aplikaciji. Prednost te platforme pred drugimi je predvsem njena razširljivost, ki skoraj vsakemu omogoča uporabo njihove platforme in tako uporabnik ni omejen na določene naprave. Fitbit API je odprt vsakomur, ki predhodno registrira aplikacijo na Fitbitovi platformi za razvijalce. Ko registriramo aplikacijo, dobimo potrebne podatke za implementacijo avtentikacije uporabnika na aplikaciji in pridobivanje podatkov. Fitbit v svoji platformi sledi standardu FHIR s področja zdravstva.

Slika 4.4: Aplikacija Fitbit

5. Tehnologije rešitve

5.1 Vue.js

Vue.js ali bolj poznano Vue je odprtokodna knjižnica JavaScript za izdelavo uporabniških vmesnikov. Vue se največ uporablja kot knjižnica za izdelavo spletnih aplikacij. Z njim lahko izdelamo od preprostih spletnih aplikacij do zahtevnejših enostranskih spletnih aplikacij. Znan je predvsem po preprostosti in lahkotnosti. Vue je ustvaril Evan You leta 2014, in sicer po tem, ko je delal za Google na mnogih projektih z AngularJS. Vključil je vse stvari, ki so mu bile všeč pri aplikaciji AngularJS, in zgradili svojo lahkokodno knjižnico Vue.js. Za uporabo te knjižnjice sem se odločili, saj lahko z njo na zelo preprost in fleksibilen način pišemo enostranske spletne aplikacije, sama knjižnica pa omogoča tudi ogromno funkcionalnosti, s katerimi je v veliki prednosti pred ostalimi knjižnicami, kot sta Angular in React.

5.2 Nuxt.js

Nuxt.js je odprtokodna knjižnica za spletne aplikacije, ki bazira na knjižnicah Vue.js, Node.js in Express.js. Sama struktura omogoča hitrejšo in preprostejšo izdelavo spletnih aplikacij. Omogoča tudi, da se določeni podatki generirajo v aplikacijo že na strežniku, še preden se aplikacija pošlje uporabniku. Glavna prednost uporabe knjižnice Nuxt.js je predvsem pretvorba drevesne strukture datotek v strukturo URL usmerjanja komponent aplikacije, izdelane v knjižnici Vue.js. Prednost je tudi uporaba ureditve strani, ki omogoča lažjo strukturo same aplikacije. Nuxt.js sem izbral zaradi njegove funkcionalnosti, saj omogoča, da se v enostranski spletni aplikaciji nekateri podatki iz zalednega sistema vnesejo v sam pogled, še preden se zahtevek pošlje brskalniku uporabnika.

5.3 Node.js

Node.js je odprtokodno okolje JavaScript, ki omogoča izvajanje kode JavaScript izven samega brskalnika. Node.js omogoča, da se lahko z okoljem JavaScript izvaja skripte ukazne vrstice in skripte na zalednem sistemu. Prednost okolja Node.js pred ostalimi je definitivno njegova preprostost in hitrost. Najbolj se uporablja za izdelavo zalednih sistemov in pri avtomatizaciji procesa programiranja. Z njim si programerji olajšajo delo in imajo napisane skripte, ki jim pomagajo pri procesu programiranja. To so razne skripte za optimizacijo kode, preverjanje kode, testov itd. S paketnim managiranjem npm se je uporaba okolja Node.js še

bolj razširila, saj so programerji lahko zlahka naložili skripte drugih programerjev in tako na lahek način ustvarili sebi prilagojene skripte. Za okolje Node.js smo se odločili zaradi njegove hitrosti in dobre kombinacije s knjižnico Vue.js v knjižnici Nuxt.js. Node.js sem izbral tudi zaradi zmožnosti obvladovanja večjega števila zahtevkov, kar poveča razširljivost same aplikacije.

5.4 Express.js

Express.js je knjižnica za okolje Node.js, ki omogoča lažjo izdelavo zalednega sistema. Vsak zahtevek, ki pride na zaledni sistem, gre najprej čez osrednji del, v katerem se lahko uporabi različne module iz NPM, ki sprocesirajo zahtevke ali omogočijo določeno funkcionalnost. Nato gre zahtevek še na usmerjanje, kjer se glede na URL zahtevka izvede specifična koda.

5.5 MongoDB

MongoDB je odprtokodna objektno orientirana podatkovna baza. Spada pod podatkovne baze NoSQL in shranjuje podatke v formatu JSON. Podatkovna baza MongoDB je primerna za uporabo, kadar aplikacija potrebuje obdelavo večje količine podatkov, saj se sama v takih primerih obnaša najbolj optimalno. MongoDB sem izbral zato, ker omogoča strukturo NoSQL, zaradi česar je sama podatkovna baza zmožna hraniti ogromne količine podatkov in jih zelo hitro procesirati.

6. Izdelava integracije platform za prikaz podatkov o aktivnem in zdravem življenju

6.1 Pridobivanje dostopa do podatkov uporabnika

V tem poglavju bom opisal postopek pridobivanja dostopa do uporabnikovih podatkov iz posameznih platform.

6.1.1 Google Fit

Za pridobivanje podatkov iz platforme Google Fit je treba najprej pridobiti uporabnikovo dovoljenje za posredovanje podatkov aplikaciji. Najprej moramo registrirati aplikacijo na svojem Googlovem računu za razvijalce. Ko aplikacijo registriramo, zraven dobimo še uporabniški identifikator aplikacije, s katerim se aplikacija identificira Googlovi platformi, da vidi, katera aplikacija želi pridobiti uporabnikove podatke.

V aplikaciji moramo v kodo HTML vključiti dve stvari, in sicer uporabniški identifikator in katera območja podatkov želimo uporabiti. Uporabniški identifikator vključimo v aplikacijo tako, da v glavo dokumenta HTML vključimo metaatribut »google-signin-client_id«, kateremu dodamo vrednost uporabniškega identifikatorja naše aplikacije. Nato moramo v aplikacijo vnesti vsa območja podatkovnega modela, kjer želimo pridobivati podatke ali jih vnašati. Za to moramo v HTML dodati metaatribut »google-signin-scope«, kateremu za vrednost dodamo seznam vseh območij, ločenih s presledkom. V programu za to diplomsko delo smo uporabili naslednja območja podatkovnega modela:

- profile,
- email,
- openid,
- https://www.googleapis.com/auth/fitness.activity.read,
- https://www.googleapis.com/auth/fitness.blood_glucose.read,
- https://www.googleapis.com/auth/fitness.blood_pressure.read,
- https://www.googleapis.com/auth/fitness.body.read,
- https://www.googleapis.com/auth/fitness.body_temperature.read,

- https://www.googleapis.com/auth/fitness.location.read,
- https://www.googleapis.com/auth/fitness.nutrition.read,
- https://www.googleapis.com/auth/fitness.oxygen_saturation.read,
- https://www.googleapis.com/auth/fitness.reproductive_health.read,
- https://www.googleapis.com/auth/fitness.activity.read.

V HTML-ju smo v aplikacijo dodali to kot:

```
<meta name="google-signin-client_id" content="913956411366-
3cuvoc61vi0861thgj4fjl5267gb9s85.apps.googleusercontent.com"/>

<meta name="google-signin-scope" content=" profile email openid
https://www.googleapis.com/auth/fitness.activity.read
https://www.googleapis.com/auth/fitness.blood_glucose.read
https://www.googleapis.com/auth/fitness.blood_pressure.read
https://www.googleapis.com/auth/fitness.body.read
https://www.googleapis.com/auth/fitness.body_temperature.read
https://www.googleapis.com/auth/fitness.location.read
https://www.googleapis.com/auth/fitness.nutrition.read
https://www.googleapis.com/auth/fitness.oxygen_saturation.read
https://www.googleapis.com/auth/fitness.reproductive_health.read
https://www.googleapis.com/auth/fitness.activity.read"/>
```

V glavo dokumenta je treba vključiti tudi Googlovo knjižnico JavaScript, ki je potrebna za vso funkcionalnost same prijave. V HTML vključimo naslednje:

```
<script src=https://apis.google.com/js/platform.js async defer />
```

Treba je bilo dodati še gumb na primerno mesto in funkcijo, ki naj se kliče ob pridobitvi uporabnikove privolitve. Gumb za prijavo z Googlom dodamo tako, da na želeno mesto v dokumentu vstavimo kodo HTML:

```
<div class="g-signin2 id="google-signin-button" data-
theme="dark"></div>
```

Vstaviti je treba tudi kodo Javascript za prikaz gumba in dodajanje funkcije ob pridobitvi potrebnih podatkov. V naslednjem primeru se bo ob uspešnem pridobivanju podatkov klicala funkcija onGoogleSignIn, ki bo dobila en parameter, v katerem so vsi podatki uporabnika.

```
gapi.signin2.render('google-signin-button',{
    onsuccess: onGoogleSignIn
})
```

V povratni funkciji dobimo več informacij, in sicer vse podatke o uporabniku in žeton za dostop do preostalih podatkov. Spodaj je primer podatkov, ki jih dobimo v povratnem klicu.


```
{
	El: "104453565262630834538"
	Zi: {
		access_token: "ya29.GlxFBj37QT-C1K6c7b_3zS5Upf8N-
		Zijbe6FiTFcGGdd71R2cxqjq8n6CEH1XCJMGfCd8jjOesx7FhTyTQ3yxO-
		cxqqytL4oxuLfRkW2aHyqNlwdpV9qCWlNt_7UfA",
		expires_at: 1540813968831,
		expires_in: 1236,
		first_issued_at: 1540810368831,
		id_token: "eyJhbGciOiJSUzI1NiIsImtpZCI6IjgyODlkNTQyODBiNzY3MTJkZTQxY2QyZWY5NT
		k3MmIxMjNiZTlhYzAiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJhY2NvdW50cy5nb29nbGUuY29tIiwi
		YXpwIjoiOTEzOTU2NDExMzY2LTNjdXZvYzZsdmkwODYxdGhnajRmamw1MjY3Z2I5czg1LmFwcHMuZ
		29vZ2xldXNlcmNvbnRlbnQuY29tIiwiYXVkIjoiOTEzOTU2NDExMzY2LTNjdXZvYzZsdmkwODYxdG
		hnajRmamw1MjY3Z2I5czg1LmFwcHMuZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwic3ViIjoiMTA0NDU
		zNTY1MjYyNjMwODM0NTM4IiwiZW1haWwiOiJnYXNwZXIuZ3JvbUBnbWFpbC5jb20iLCJlbWFpbF92
		ZXJpZmllZCI6dHJ1ZSwiYXRfaGFzaCI6InkwaTlPWUhkR0NVMzI3dXlwTmliYkEiLCJuYW1lIjoiR
		2HFoXBlciBHcm9tIiwicGljdHVyZSI6Imh0dHBzOi8vbGgzLmdvb2dsZXVzZXJjb250ZW50LmNvbS
		8tWklsZGdJX1NtZXMvQUFBQUFBQUFBQUkvQUFBQUFBQUFCRmcveFZUcG5UYklaLVUvczk2LWMvcGh
		vdG8uanBnIiwiZ2l2ZW5fbmFtZSI6IkdhxaFwZXIiLCJmYW1pbHlfbmFtZSI6Ikdyb20iLCJsb2Nh
		bGUiOiJzbCIsImlhdCI6MTU0MDgxMDM3MCwiZXhwIjoxNTQwODEzOTcwLCJqdGkiOiI0YTNiOTM4N
		2E5NDQzNmQ2ZjA0ZGUwZmZkODIyMDQ4ZmY0MjdjNDI3In0.WOkOHfIU_UEC2uZ-7Sx35IBPT-
		N299E-HcV5wJf5-XYJDyiSDnxC_e8udRnOSQCIJXWaIuB-QL20sMrpQLA6T2Cuv1Ltmy-
		cPLmg7ojmc7SyAzEBW69ED4z-_kM_eMQE61AmSDlZh6NAnKxCUh6X1VbDY4tqzl_2Q-
		hroK3tMOYzY8pnvPzqbYczFXOFDgR8ank9Tz6P7eDyT1xG_f0LqpmsJobtG6YHFSbiSE56QXQRgU_
		T_Lpimb8Qb1Hb0xNpap4Fkt_f-dbpQpEqKk21KBqfJd-YCH306LJTA37ZZZuyymsjDwhdVsSY--
		ZHB8N3nyAklDuoqTPu6xAC9r8yyw",
		idpId: "google",
		login_hint: "AJDLj6JUa8yxXrhHdWRHIV0S13cAhjQDFdTKju3wfB__kOJ6o8NxuDlMBNDdav-
		n9l3ku6bI0WoCiXYo4OwxXJAcGyRADZsVMQ",
		scope: "openid email profile
		https://www.googleapis.com/auth/fitness.oxygen_saturation.read
		https://www.googleapis.com/auth/fitness.body.read
		https://www.googleapis.com/auth/fitness.reproductive_health.read
		https://www.googleapis.com/auth/userinfo.email
		https://www.googleapis.com/auth/fitness.activity.read
		https://www.googleapis.com/auth/fitness.blood_glucose.read
		https://www.googleapis.com/auth/plus.me
		https://www.googleapis.com/auth/userinfo.profile
		https://www.googleapis.com/auth/fitness.blood_pressure.read
		https://www.googleapis.com/auth/fitness.body_temperature.read
		https://www.googleapis.com/auth/fitness.location.read
		https://www.googleapis.com/auth/fitness.nutrition.read",
		}
		token_type: "Bearer"
	w3: {
		Eea: "104453565262630834538",
		Paa: https://lh3.googleusercontent.com/-
		ZIldgI_Smes/AAAAAAAAAAI/AAAAAAAABFg/xVTpnTbIZ-U/s96-c/photo.jpg,
		U3: gasper.grom@gmail.com,
		ig: "Gašper Grom",
		ofa: "Gašper",
		wea: "Grom"
	}
}
```

6.1.2 Apple HealthKit

Po pregledu celotne dokumentacije Apple HealthKit sem ugotovil, da ta platforma ne omogoča pridobivanja podatkov s spletnih strani, vendar le za aplikacije, zgrajene na platformi iOS. Ena možnost bi sicer bila, da bi ustvaril mobilno aplikacijo, ki bi pridobivala podatke in jih posredovala na zaledni sistem naše aplikacije, vendar moje znanje žal ne obsega mobilnih aplikacij za platformo iOS, da bi to lahko implementiral v obsegu tega diplomskega dela.

6.1.3 Garmin Connect

Po pregledu celotne strani Garmin Connect API sem ugotovil, da te platforme ne bom mogel vključiti v diplomsko delo. Podjetje Garmin zahteva za dostop do zalednega sistema 5000 dolarjev. Garminu sem poslal prošnjo za odobritev dostopa v namene diplomskega dela, vendar odgovora nisem dobili.

6.1.4 Fitbit

Pri platformi Fitbit je treba najprej pridobiti dovoljenje uporabnika za uporabo njegovih podatkov iz platforme Fitbit v naši aplikaciji. Najprej moramo registrirati samo aplikacijo v platformo Fitbit, kjer je treba določiti ime aplikacije in območja, katere podatke bo aplikacija zajemala. Platforma vrne generirano povezavo, na katero moramo preusmeriti uporabnika za pridobitev njegove privolitve. Povezava za našo aplikacijo je videti takole:

```
https://www.fitbit.com/oauth2/authorize?response_type=code&client_id=22
D4N8&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Ffitbit&scope=activity
%20heartrate%20location%20nutrition%20profile%20settings%20sleep%20soci
al%20weight&expires_in=604800
```

Povezava ima najprej glavni del, to je `https://www.fitbit.com/oauth2/authorize`, pri tem ima dodane parametre. Parameter `response_type` nam pove, kaj želimo pridobiti nazaj iz same potrditve, v tem primeru smo želeli dobiti kodo. `client_id` je identifikator, s katerim se lahko naša aplikacija identificira na platformi Fitbit, njegovo vrednost dobimo ob registraciji aplikacije na platformi Fitbit. `scope` nam pove, katera območja podatkov želimo pridobivati. Kot vrednost dodamo seznam vseh območij, ločenih s presledkom. Zadnji pomemben parameter je `redirect_uri`, ki nam pove, kam naj aplikacija Fitbit preusmeri uporabnika po odobritvi uporabe podatkov.

Po uporabnikovi potrditvi o uporabi njegovih podatkov nas platforma Fitbit preusmeri na dano povezavo, ki smo jo vključili. Na tej povezavi moramo urediti še nekaj stvari. Kot

parameter povezave `code` dobimo kodo, ki jo uporabimo za dostop do uporabnikovega dostopnega žetona in preostalih podatkov.

V aplikaciji se to stori tako, da se parameter najprej zajame prek okolja JavaScript. Nato se pošlje zahtevek HTTP prek metode AJAX. To je metoda v okolju JavaScript, ki omogoča, da se brez osvežitve strani pridobi podatke z zahtevkom HTTP. Povezava za zahtevek HTTP je videti takole:

```
https://api.fitbit.com/oauth2/token?clientId=22D4N8&grant_type=authoriz
ation_code&code={{code}}
```

Poslati je bilo treba zahtevek POST na povezavo `https://api.fitbit.com/oauth2/token`, ki smo mu morali dodati naslednje parametre: `clientId` je identifikator naše aplikacije. Tako kot pri prejšnjem zahtevku mu damo vrednost identifikatorja naše aplikacije na platformi Fitbit. `grant_type` je parameter, kateremu povemo, kaj želimo dobiti v odgovoru na zahtevek, tu damo vrednost `authorization_code`. Zadnji parameter je `code`, kateremu dodamo vrednost, ki jo dobimo iz parametra povezave. Pri tem zahtevku je pomembno, da nastavimo glavo samega zahtevka, in sicer dodati moramo del glave `Authorisation`, kateremu damo vrednost `Basic + kodo avtorizacije`, ki jo dobimo ob registraciji aplikacije.

Na ta zahtevek dobimo iz platforme Fitbit odgovor v obliki JSON, z vsemi žetoni, potrebnimi za dostop do podatkov aplikacije.

V nadaljevanju je primer odgovora, ki ga dobimo kot odgovor na naš zahtevek. Dobimo `access_token`, s katerim lahko dostopamo do preostalih podatkov iz platforme, `refresh_token`, s katerim osvežimo `access_token`, ko le-ta poteče, in `user_id`, ki je identifikator uporabnika.

```
{
	access_token: "eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIyMkQ0TjgiLCJzdWIiOiI2VFRDQ00iLCJpc3Mi
	iJGaXRiaXQiLCJ0eXAiOiJhY2Nlc3NfdG9rZW4iLCJzY29wZXMiOiJyc29jIHJhY3QgcnNldCBybG9jIHJ3Z
	WkgcmhyIHJwcm8gcm51dCByc2xlIiwiZXhwIjoxNTQwOTIzODA1LCJpYXQiOjE1NDA4OTUwMDV9.V8exikKh
	3EwKeBlGRRs6QPxNWZPbiV33xHKv-HNDafs",
	expires_in: 28800,
	refresh_token: "7fb73cdea8788a59512d69fe1bc0568e4e335c79297170221403807b3136620d",
	scope: "location weight profile social nutrition heartrate activity settings sleep",
	token_type: "Bearer",
	user_id: "6TTCCM"

}
```

6.2 Pridobivanje podatkov

6.2.1 Google Fit

Za pridobivanje podatkov iz platforme Google Fit moramo najprej pridobiti seznam vseh podatkovnih virov, ki jih ima uporabnik. To storimo tako, da pošljemo zahtevek z metodo `GET` na povezavo `https://www.googleapis.com/fitness/v1/users/me/dataSource`. Pri tem zahtevku moramo dodati v glavo še dodaten parameter Authorization, kateremu damo vrednost Bearer in uporabnikov dostopni žeton.

Kot vrnjeno vrednost dobimo seznam vseh podatkovnih virov in njihove podrobnosti, kot na primer podatkovne tipe, ki jih platforma prejme iz njih. Primer odgovora na zahtevek:

```
dataSource: [
	{
		application: {
			packageName: "com.google.android.gms"
		},
		dataQualityStandard: [],
		dataStreamId: "derived:com.google.active_minutes:com.google.android.gms:from_
		activity<-merge_activity_segments",
		dataStreamName: "from_activity<-merge_activity_segments",
		dataType: {
			field: [
				{
				    1.  format: "integer"
				    2.  name: "duration"
				}
			],
			name: "com.google.active_minutes"
		},
		type: "derived"
	}
]
```

Te podatke nato uporabimo za pridobitev posameznih podatkov, ki jih potrebujemo. To storimo tako, da na povezavo

https://www.googleapis.com/fitness/v1/users/me/dataSources/{{ dataStreamId }}/datasets/{{ minStartTime }}-{{ maxEndTime }}

pošljemo zahtevek GET. V samo povezavo moramo vstaviti tri parametre. Prvi parameter je dataStreamId, ki je identifikator podatkovnega vira, ki ga dobimo v prejšnjem odgovoru na zahtevek v parametru dataStreamId posameznega podatkovnega vira. Zadnja dva parametra pa sta časovno obdobje, v katerem želimo pridobiti podatke. V glavo zahtevka ne smemo pozabiti vstaviti še parametra Authorization, kateremu damo vrednost Bearer in uporabnikov dostopni žeton.

```
{
        dataSourceId: "derived:com.google.activity.samples:com.google.android.gms:HUAWEI:ALE
        -L21:91101c87:phone_default_dnn_v14",
        maxEndTimeNs: "1542033361281",
        minStartTimeNs: "1542033361281",
        point: [
            {
                startTimeNanos: "1542033361281",
                originDataSourceId: "",
                endTimeNanos: "1542033361281",
                value: [
                        {
                                intVal: 8
                        }
                ],
                dataTypeName: "com.google.active_minutes",
            }
        ]

}
```


Zgoraj je primer odgovora na naš zahtevek. Najprej imamo v odgovoru vse parametre, ki smo jih podali v zahtevku, nato pa v parametru point vse meritve iz danega obdobja, ki smo ga podali. Za vsako meritev imamo podane čas začetka in konca merjenja, vrednosti, izmerjene v tem časovnem obdobju, in tip meritve.

6.2.2 Fitbit

Da pridobimo podatke iz platforme Fitbit, moramo poslati na povezavo

https://api.fitbit.com/1/user/{{ userId }}/activities/list.json

zahtevek z metodo GET. V naslov povezave moramo vstaviti en parameter, identifikator uporabnika. Če želimo pridobivati podatke od trenutno prijavljenega uporabnika, potem uporabimo v tem polju le znak "-". V glavo samega zahtevka moramo pod polje Authorization dodati vrednost Bearer in uporabnikov dostopni žeton za dostop do platforme Fitbit. Dodati moramo štiri parametre. Prvi parameter je beforeDate, ta nam pove, od katerega datuma dalje želimo pridobiti informacije. Vrednost prvega parametra je dodana v obliki yyyy-mm-dd, na primer 2018-11-05. Naslednji parameter je sort, kateremu podamo vrednost asc, če želimo imeti sortirane podatke po vrstnem redu, ali desc, če jih želimo meti sortirane v obratnem vrstnem redu. V našem primeru smo jih želeli imeti sortirane po pravem vrstnem redu. Naslednji obvezni parameter je limit, ki nam pove, koliko podatkov naenkrat naj nam pošlje nazaj platforma Fitbit. Maksimalna številka za ta parameter je 20, to smo uporabili tudi v našem primeru. Zadnji parameter je offset, ki nam pove, katerih 20 podatkov po vrsti naj nam pošlje. Če je vneseno 0, potem nam prikaže prvih 20 podatkov, če vnesemo 1, potem je vnesenih naslednjih 20 podatkov, in tako naprej.

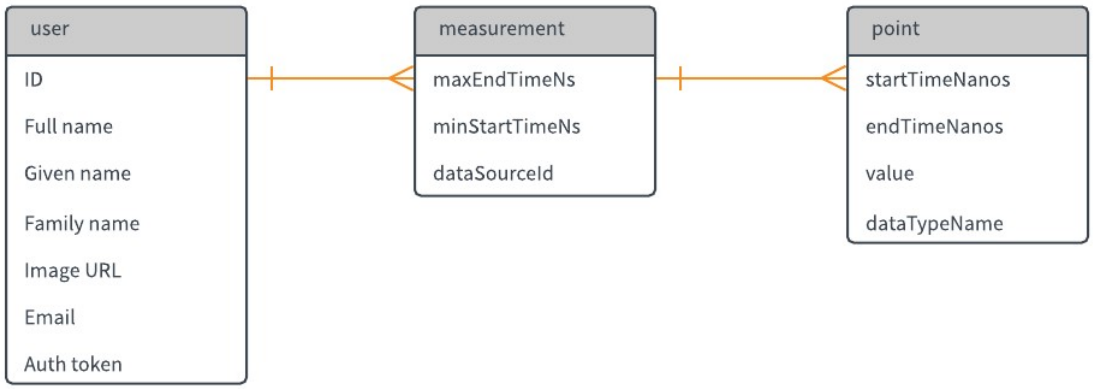
Odgovor na zahtevek je videti nekako takole:

```
activities: [
   {
          activeDuration: 1734000,
          activityLevel: [
                 {
                        minutes: 4,
                        name: "sedentary"
                 }
          ]
          activityName: "Running",
          activityTypeId: 17589491,
          averageHeartRate: 94,
          calories: 136,
          distance: 1.071811,
          distanceUnit: "Kilometer",
          source: {
                 id: "228T4Z",
                 name: "friFit",
                 type: "app",
                 url: http://www.frifit.com
          }
          speed: 2.2252131487889275,
          startTime: "2016-11-06T08:45:00.000-07:00",
          steps: 1354
   }
]
pagination:
       beforeDate: "2018-11-05",
       limit: 20,
       next: "https://api.fitbit.com/1/user/-
       /activities/list.json?limit=20&sort=asc&beforeDate=2018-11-05&offset=1"
       sort: "asc"
```

Odgovor nam poda dva glavna parametra, activities in pagination. Activities nam poda seznam vseh aktivnosti danega obdobja z vsemi podrobnosti, kot so enote, dolžina, čas, hitrost, čas začetka, koraki, srčni utrip. Pagination pa nam pove, kateri parametri so bili podani v samem zahtevku, in zraven doda še naslov povezave, s katero lahko dostopamo do naslednje serije podatkov.

6.3 Načrtovanje skupnega podatkovnega modela

6.3.1 Google Fit

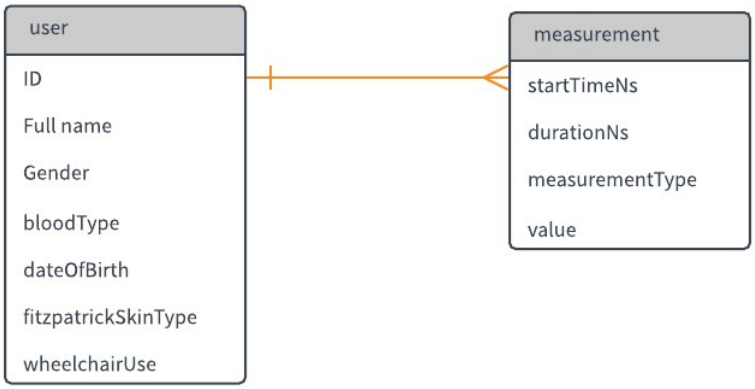


Slika 6.1: Podatkovni model Google Fit

Na Sliki 6.1 je prikazan podatkovni model platforme Google Fit. Podatke o uporabniku dobimo pri uporabnikovi avtorizaciji na platformi, podatke o meritvah pa nato, ko te podatke zahtevamo. Vse podatke o meritvah, popiti vodi in količini zaužitih prehranskih vsebnosti pridobivamo v enakem formatu.

Za vsakega uporabnika dobimo ime, priimek, sliko profila, e-pošto in dostopni žeton. Vsak uporabnik ima lahko več meritev. Za vsako meritev dobimo čas začetka, čas konca in ID izvorne naprave. Vsaka meritev vsebuje tudi več točk meritev, vsaka točka pa vsebuje čas začetka, čas konca, vrednost in tip meritve.

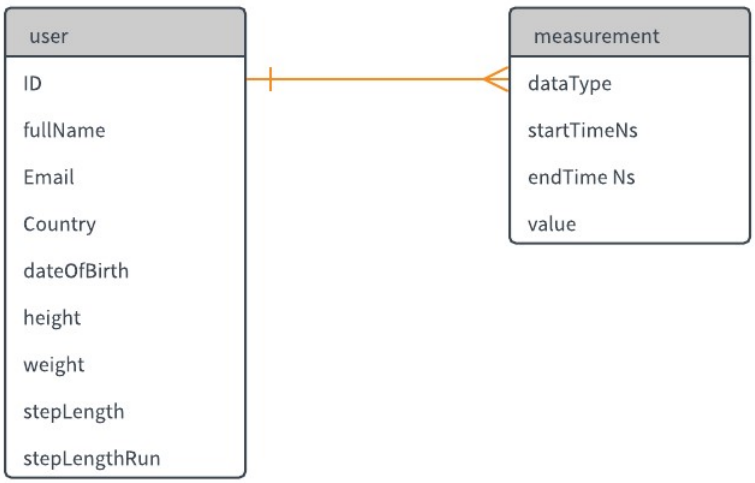
6.3.2 Apple HealthKit



Slika 6.2: Podatkovni model Apple HealthKit

Zaradi omejitev modela AppleHealth Kit nisem mogel iz vrnjenih podatkov narediti podatkovnega modela. V dokumentaciji ni bilo primerov odgovorov na zahtevke, vendar le razredi, kako so strukturirani. Na Sliki 6.2 je prikazan podatkovni model, ki mi ga je uspelo razbrati iz dokumentacije Apple HealthKit. V modelu lahko pridobimo več informacij o uporabniku, predvsem tip krvi, datum rojstva, tip kože in ali uporabnik uporablja invalidski voziček.

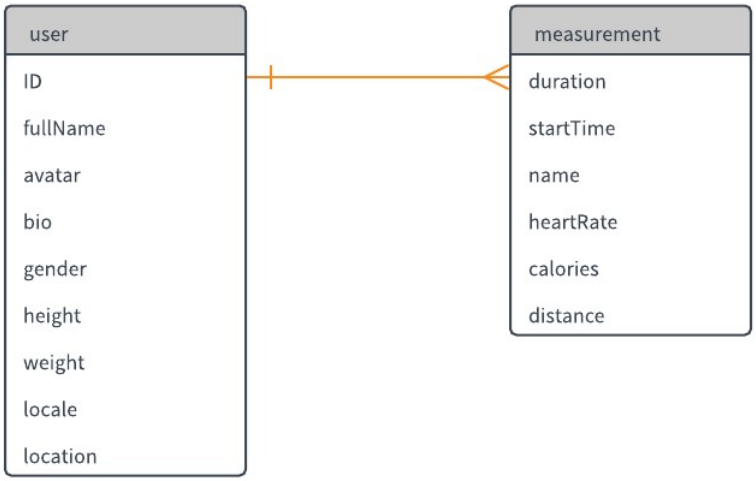
6.3.3 Garmin



Slika 6.3: Podatkovni model Garmin

Ker Garmin zahteva za dostop do API in njegove dokumentacije 5000 dolarjev, sem naredil podatkovni model po podatkih, ki sem jih lahko poizvedel iz same aplikacije za pregled meritev. Garminov podatkovni model se predvsem razlikuje v tem, da določene podatke, ki jih druge platforme merijo pri meritvah, ta beleži v uporabniški entiteti. To so na primer višina, teža, dolžina koraka, dolžina koraka med tekom itd. Garminov podatkovni model predpostavlja, da se višina in teža uporabnika ne bosta spreminjali oz. spremembe uporabnik spremeni v uporabniških nastavitvah, nima pa uporabnik konkretnega pregleda nad spremembami teh meritev.

6.3.4 Fitbit



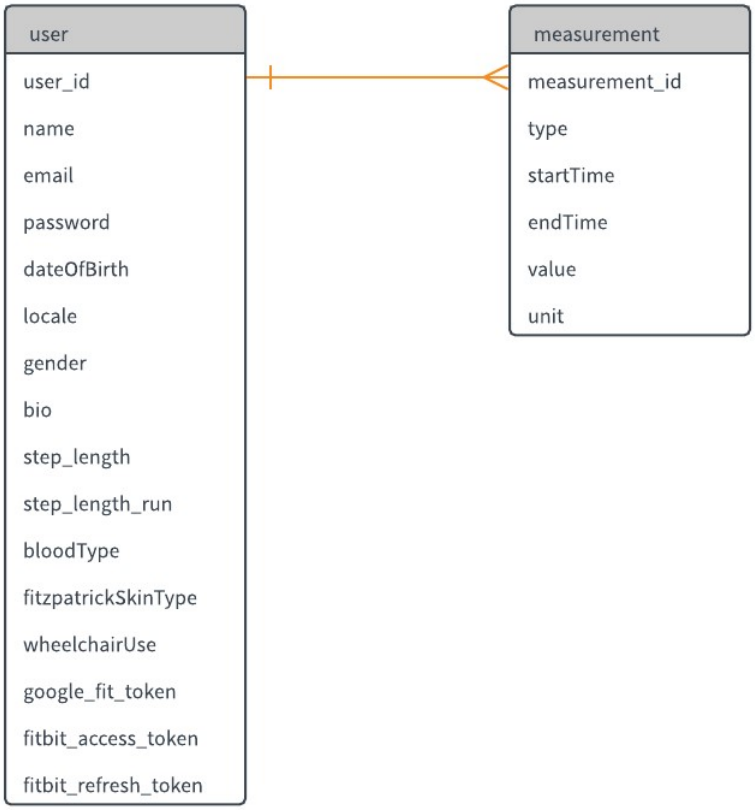
Slika 6.4: Podatkovni model Fitbit

Podatkovni model Fitbit se od preostalih razlikuje predvsem v tem, da za vsako meritev beleži več podatkov. Za vsako meritev zraven, če je le mogoče, vnese tudi povprečni utrip srca, kalorije in razdaljo. Če teh meritev ni, je na tem mestu vnesen null.

6.3.5 Združen podatkovni model

Da bi vse te podatke lahko združil, sem naredil preprost združen podatkovni model, s katerim lahko zajamemo vse podatke iz vseh platform. Ker Fitbit vrača bistveno več podatkov kot Google, smo se odločili narediti tako, da preostale meritve in aktivnosti, kot so razdalja,

število korakov, srčni utrip, porabljene kalorije, dodamo v več instanc posameznega modela meritve. Na Sliki 5.6 je prikazan diagram ER združenih podatkov.



Slika 6.5: Diagram ER združenega podatkovnega modela

Prva entiteta prikazuje uporabnika. Za vsakega uporabnika hranimo ime, e-pošto in geslo. Ti podatki so pomembni predvsem za avtentikacijo uporabnika v integracijskem portalu in prikaz njegovega profila. Nato v entiteti hranimo tudi preostale podatke, ki jih lahko pridobimo iz platform, kot so datum rojstva, jezik, spol, opis, dolžina koraka pri hoji in teku, tip krvi, tip kože in če uporabnik uporablja invalidski voziček. Na koncu v entiteti uporabnika hranimo tudi dostopne žetone za dostop do platform. Za Google Fit moramo hraniti le dostopni žeton, za platformo Fitbit pa dostopni in osvežilni žeton.

Entiteta, ki je tukaj ključnega pomena, je entiteta meritve. Vsaka meritev vsebuje tip meritve, katerega se poenoti iz vseh aplikacij, da lahko prepoznavamo tipe meritev in tako lahko združimo podatke več platform ter prikažemo na posameznih grafih. Potem imamo čas začetka in konca v milisekundah, saj tako lažje hranimo podatke v podatkovni bazi ter tudi lažje obdelujemo in prikazujemo primerno. Na koncu vsebuje tudi podatka vrednost in enota, da se lahko vzame vrednost in se po potrebi pretvori v ustrezne enote oz. prikazuje v različnih enotah glede na nastavitve uporabnika. Tak podatkovni model omogoča veliko fleksibilnost

glede na platforme in omogoča dober pregled nad podatki. Zaradi splošnosti podatkovnega modela lahko preprosto filtriramo podatke. Če želimo podatke filtrirati po tipu, gledamo samo atribut type. Če želimo dobiti podatke le v določenem časovnem obdobju, vrnemo le tiste, ki imajo čas začetka in konca v tem obdobju. Podatke lahko filtriramo tudi po vrednostih. Pridobimo lahko tudi več podatkovnih tipov naenkrat. Pri tem podatkovnem modelu nastane problem le v primeru, da neka meritev zahteva shrambo večjega števila vrednosti, v takem primeru v atribut value shranimo objekt z vsemi vrednostmi.

Povezava med obema entitetama je vidna na diagramu. Vsak uporabnik ima lahko mnogo meritev ali nobene, vsaka meritev pa ima lahko le enega uporabnika.

6.4 Združitev podatkov

Zadnji korak v okviru te aplikacije je združitev podatkov. Treba je bilo narediti enotno strukturo za aktivnosti, ki jo bo možno napolniti s podatki iz katere koli platforme. Model za posamezno aktivnost v podatkovni bazi smo ustvarili s petimi atributi, to so ime aktivnosti, čas začetka v milisekundah, čas konca v milisekundah, vrednost aktivnosti in enota, v kateri je bila izmerjena aktivnost. Podatkovni model je videti takole:

```
mongoose.Schema({
      type: {type: String, required: true},
      startTime: {type: Number, required: true},
      endTime: {type: Number },
      value: {type: Number, required: true},
      unit: {type: String, required: true},
})
```

Pri združevanju podatkov je bilo treba implementirati tudi nekaj določenih funkcij za pretvorbo parametrov v enotne vrednosti, saj imajo platforme določene podatke različno vnesene, kot na primer datum je na platformi Google Fit vnesen v milisekundah, medtem ko je na Fitbitu vnesen v formatu yyyy-mm-dd.

Najprej je bilo treba poenotiti imena aktivnosti. Za to smo morali narediti za vsako platformo objekt, ki je v ključih hranil ime aktivnosti iz platforme, kot vrednost pa enotno ime aktivnosti v naši aplikaciji. To nam je omogočalo, da so vsa imena aktivnosti poenotena v celotnem sistemu, saj lahko tako združujemo podatke.

Naslednja stopnja je bila pretvorba podatkov za parameter začetnega časa aktivnosti. Iz obeh platform za posamezno aktivnost lahko dobimo čas začetka aktivnosti, vendar sta v različnih

formatih. V tem primeru moramo poenotiti format. Iz platforme Google Fit dobimo vrnjen začetni čas v milisekundah, zato tukaj ni treba nič pretvarjati, saj enak format uporabljamo tudi v naši aplikaciji. Za platformo Fitbit pa dobimo začetni čas aktivnosti vrnjen v nizu, zato moramo tega pretvoriti v milisekunde. To storimo preprosto z objektom Date(), ki vzame niz in ga pretvori v datum. Na tem objektu nato uporabimo le še funkcijo getTime(), ki pa nam vrne dani datum v milisekundah.

Naslednji parameter, pri katerem moramo zagotoviti enotnost, je čas konca aktivnosti. Iz platforme Google Fit dobimo vrnjen čas konca aktivnosti že kot parameter posamezne aktivnosti. Ta parameter je tudi v formatu, ki ga zahteva naša aplikacija. Iz platforme Fitbit pa pridobimo dva parametra, ki ju lahko uporabimo za pridobitev konca aktivnosti, to sta čas začetka aktivnosti in čas trajanja aktivnosti. Čas začetka aktivnosti je podan v nizu, zato ga moramo najprej pretvoriti v format milisekund, tako kot je že opisano v prejšnjem odstavku. Tem vrednostim nato le dodamo trajanje aktivnosti, ki jo dobimo vrnjeno v milisekundah in tako dobimo končni čas aktivnosti.

Naslednji parameter je vrednost aktivnosti. Platforma Google Fit nam lahko za posamezno aktivnost vrne tudi več vrednosti, zato v naši aplikaciji, da bo enotno, naredimo več instanc aktivnosti v primeru večjega števila vrednosti. Platforma Fitbit pa nam za vsako aktivnost vrže več podatkov o aktivnosti. Za ta primer moramo narediti iz ene aktivnosti, ki jo dobimo vrnjeno iz te platforme, več instanc različnih aktivnosti. Recimo za aktivnost tek bomo dobili vrnjeno število korakov, kalorije, srčni utrip itd. in za vsako posebej moramo narediti novo instanco modela aktivnosti v podatkovni bazi.

Zadnji parameter našega podatkovnega modela aktivnosti je enota. Za vsako vrednost v aktivnosti obstaja več enot, zato smo morali v aplikaciji implementirati tudi objekt, ki je imel vrednosti za pretvorbo med enotami.

Tako so sedaj vsi podatki združeni in pripravljeni za uporabo v aplikaciji in za njihov prikaz.

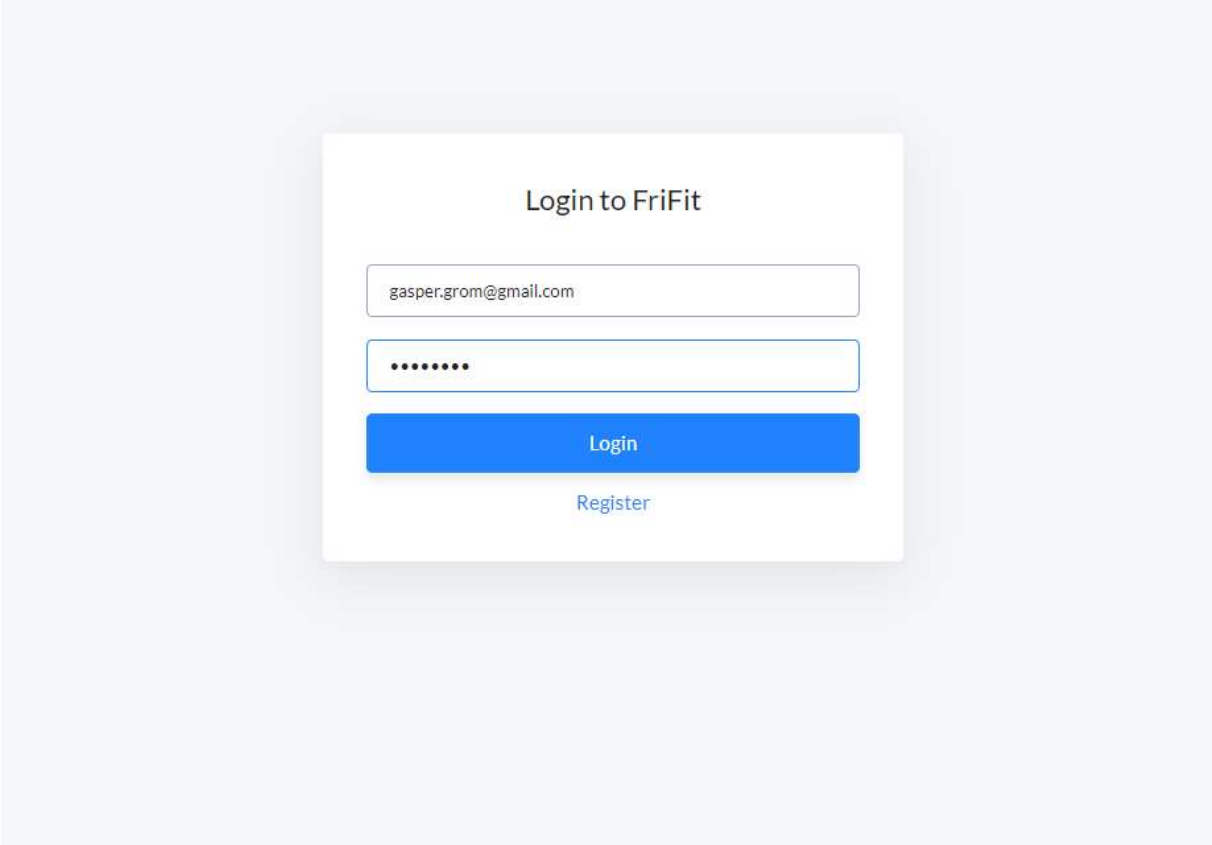
6.5 Priključitev nove platforme

Če želimo v integracijski portal priključiti novo platformo, moramo slediti določenim korakom priključitve. V integracijski portal moramo najprej dodati v vpogled uporabniškega profila novo platformo in dodati še gumb za povezovanje platforme z integracijskim portalom. Ta gumb mora omogočati, da uporabnika preusmeri na stran za odobritev pridobivanja podatkov in tako integracijski portal dobi dostopni žeton nove platforme, s katerim dostopa do preostalih podatkov. Nato moramo v model uporabnika dodati še polje,

kjer lahko hranimo dostopni žeton do nove platforme. Ko imamo vse to narejeno, uredimo še pridobivanje podatkov in pretvorbo teh v enotno obliko, ki se lahko hrani v naši podatkovni bazi.

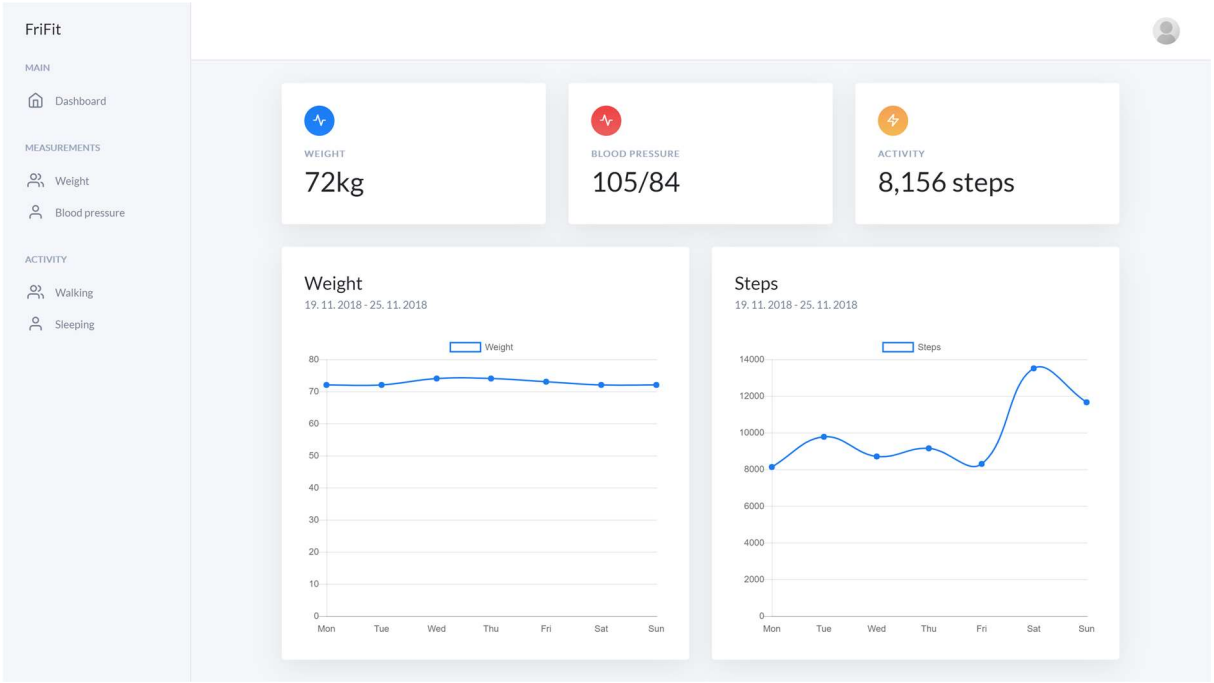
7. Končna rešitev

Končna rešitev tega diplomskega dela je bil integracijski portal, ki integrira podatke iz več platform, v našem primeru platform Google Fit in Fitbit.



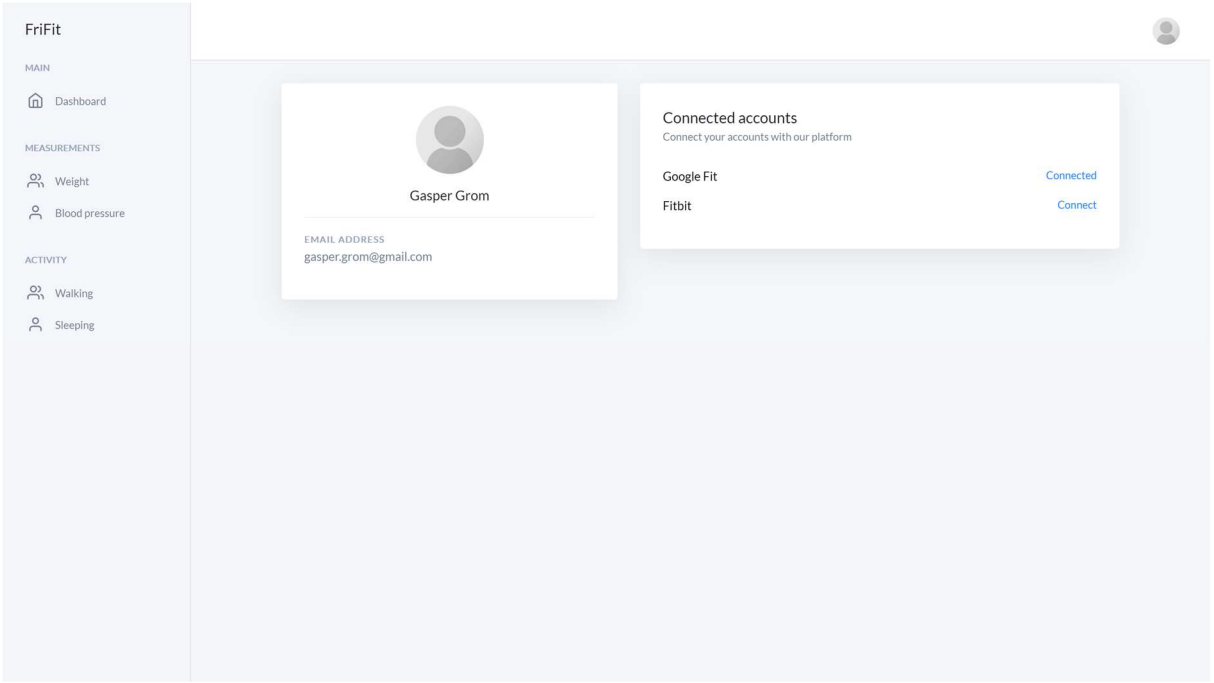
Slika 7.1: Prijavna zaslonska maska aplikacije

V aplikacijo se lahko registriramo in prijavimo z e-pošto in izbranim geslom. Na Sliki 7.1 je vidna zaslonska maska prijavnega zaslona, kjer se uporabnik prijavi v integracijski portal.



Slika 7.2: Nadzorna plošča v aplikaciji

Ko se prijavimo, nas aplikacija privede na glavni zaslon, kjer imamo pregled nad meritvami. Celotna glavna stran je nastavljiva po meri uporabnika, tako da si sam ustvari, katere meritve želi imeti na glavni strani. Na levi strani je še glavni meni, kjer uporabnika privede do pregleda posameznih tipov meritev. V zgornjem desnem kotu se s klikom na sliko profila uporabnika odprejo možnosti za pregled profila in odjavo iz portala.



Slika 7.3: Pregled profila v aplikaciji

Zadnja zaslonska maska prikazuje pregled uporabniškega profila. Na tej strani imamo pregled nad uporabniškim profilom. Uporabnik lahko na te strani poveže platforme z integracijskim portalom. Na desni strani ima prikazan seznam vseh platform, s katerimi lahko poveže naš integracijski portal. S klikom na »Connect« bo uporabnika preusmerilo na primerno stran za avtorizacijo pridobivanja podatkov. Z uporabnikovo odobritvijo integracijski portal shrani uporabnikov dostopni žeton in tako lahko začne s pridobivanjem uporabnikovih podatkov iz povezanih platform.

8. Sklepne ugotovitve

Za izdelavo integracijskega portala je treba pred tem dobro premisliti strukturo aplikacije in podatkovne baze. Za vsakega uporabnika moramo v podatkovni bazi hraniti dostopne žetone, s katerimi dostopamo do njegovih podatkov na platformah, iz katerih beremo same podatke. Integracijski portal zahteva dobro premišljeno strukturo in ogromno dela s prilagajanjem pridobljenih podatkov v enotno obliko. Pri integracijskem portalu se pojavi tudi težava, ko imajo platforme omejene načine dostopa. Na primer do podatkov iz platforme Apple HealthKit lahko dostopamo le z mobilnimi aplikacijami, izdelanimi za operacijski sistem iOS. Problem je tudi pri nekaterih platformah, ki ne omogočajo dostopa za vsakogar, vendar zahtevajo plačilo za dostop do platforme, kot na primer Garmin. Integracijski portali so zelo uporabni, vendar tudi zelo zahtevni za izdelavo.

Kazalo slik

Slika 2.1 splošna arhitektura sistemov 3
Slika 2.2 Naprava za zajem podatkov Garmin 4
Slika 2.3 Nadzorna plošča Fitbit 6
Slika 3.1 pametna ura Garmin 9
Slika 3.2 pametne zdravstvene merilne naprave 10
Slika 4.1 aplikacija Google Fit 12
Slika 4.2 aplikacija Apple Health 13
Slika 4.3 aplikacija Garmin 14
Slika 4.4 aplikacija Fitbit 15
Slika 6.1 podatkovni model Google Fit 27
Slika 6.2 podatkovni model Apple Health Kit 28
Slika 6.3 podatkovni model Garmin 28
Slika 6.4 podatkovni model Fitbit 29
Slika 6.5 ER diagram združenega podatkovnega modela 30
Slika 7.1 prijavna zaslonska maska aplikacije 33
Slika 7.2 nadzorna plošča v aplikaciji 34
Slika 7.3 pregled profila v aplikaciji 35

Literatura

[1] Google Fit. *Google* [Online]. Dosegljivo: https://www.google.com/fit/.

[2] Google Fit. *Google developers* [Online]. Dosegljivo: https://developers.google.com/fit/.

[3] Google Fit. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Google_Fit.

[4] HealthKit. *Apple Developer Documentation* [Online]. Dosegljivo: https://developer.apple.com/documentation/healthkit.

[5] Health (Apple). *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Health_(Apple).

[6] Garmin Connect [Online]. Dosegljivo: https://connect.garmin.com.

[7] Garmin. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Garmin.

[8] Fitbit Official Site [Online]. Dosegljivo: https://www.fitbit.com/eu/home.

[9] Web API. *Fitbit* [Online]. Dosegljivo: https://dev.fitbit.com/build/reference/web-api/.

[10] Vue.js [Online]. Dosegljivo: https://vuejs.org.

[11] Nuxt.js – Universal Vue.js Applications [Online]. Dosegljivo: https://nuxtjs.org/.

[12] Node.js [Online]. Dosegljivo: https://nodejs.org/en/.

[13] Express – Node.js web application framework [Online]. Dosegljivo: https://expressjs.com/.

[14] MongoDB: Open Source Document Database [Online]. Dosegljivo: https://www.mongodb.com/.