jamk.fi

Harjoituspäiväkirjasovelluksen toteuttaminen mobiililaitteille

Olli Korhonen

Opinnäytetyö
Huhtikuu 2019
Tekniikan ja liikenteen ala
Tieto- ja viestintätekniikan tutkinto-ohjelma
Mediatekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Korhonen, Olli	Opinnäytetyö, AMK	Huhtikuu 2019
	Sivumäärä 47	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Harjoituspäiväkirjasovelluksen toteuttaminen mobiililaitteille

Tutkinto-ohjelma

Tieto- ja viestintätekniikka

Työn ohjaaja(t)

Manninen Pasi, Niemi Kari

Toimeksiantaja(t)

-

Tiivistelmä

Opinnäytetyön aihe valikoitui kiinnostuksesta kuntosaliharjoitteluun ja halusta luoda toimiva sovellus kuntosaliharjoitusten kirjanpitoon.

Opinnäytetyön tavoitteena oli toteuttaa hyvin käytettävä sovellus, jota käyttäjän on mukava ja helppo käyttää kuntosalilla ollessa, ilman treenin häiriintymistä.

Opinnäytetyössä tutkittiin olemassa olevia harjoituspäiväkirjasovelluksia Android- sekä iOS-puhelimiin, koska haluttiin selvittää, onko markkinoilla tarvetta uusille sovelluksille. Opinnäytetyössä tutkittiin myös eri cross-platform-tekniikoita, jotta saatiin selville, minkälaisia tekniikoita cross-platform-sovelluskehitykseen on tarjolla ja mikä olisi sopivin toteutukseen.

Sovelluksen suunnitteluvaiheessa tehtyjen rautalankamallien tekemiseen käytettiin Microsoft Visio -sovellusta. Sovelluksen tekoon valikoitui React Native sekä Native Base. Sovelluksessa oleva tab-navigointi toteutettiin React Navigation JavaScript-kirjastolla. Useamman kielen saamiseksi sovellukseen käytettiin i18next JavaScript-kirjastoa sekä react-native-device-info JavaScript-kirjastoa. Sovelluksen sisäistä tilaa hallinnoitiin Redux-kirjastolla. Sovelluksen tietokantana toimi Realm-tietokanta.

Opinnäytetyön tuloksena saatiin hyvä alku harjoituspäiväkirjamobiilisovellukseen, jota on hyvä lähteä jatkokehittämään julkaisua varten.

Avainsanat (asiasanat)

React Native, Native Base, JavaScript, Redux, Realm, i18next, Cross-platform, Mobiilikehitys

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Korhonen, Olli	Bachelor's thesis	April 2019
		Language of publication: Finnish
	Number of pages 47	Permission for web publication: x

Title of publication

Implementation of training diary application for mobile devices

Degree programme

Information and communication technology

Supervisor(s)

Manninen Pasi, Niemi Kari

Assigned by

-

Abstract

The subject for this study was selected due to an interest in gym training and a desire to create a workable application to write down the results of gym exercises.

The objective was to create a well-used application which is comfortable and easy for the users to make use of in the gym without disturbing the exercise.

The assignment was to investigate existing training diary applications for Android and iOS phones in order to find out if there is a niche to a new application on the market. The study also investigated different cross-platform techniques to find out what kind of techniques there are available for cross-platform-development and what would be the most appropriate for the implementation.

Microsoft Visio application was used to create wireframes during the design period of the application. React Native and Native Base were selected to create the application. Tab-navigation of the application was created with React Navigation JavaScript library. To get Multilanguage to the application a i18next JavaScript library and react-native-device-info JavaScript library were used. The State of the application was managed with Redux library. The database of the application was Realm database.

The study resulted in a good start to a training diary mobile application which can be further developed for publication.

Keywords/tags (subjects)

React Native, Native Base, JavaScript, Redux, Realm, i18next, Cross-platform, Mobile Development

Miscellaneous

Sisältö

1 Johdanto 5

2 Olemassa olevien sovelluksien kartoitus 6

2.1 Yleistä 6

2.2 Vertailu 7

2.2.1 GymRun 7

2.2.2 Simple Workout Log 8

2.2.3 FitNotes 9

2.2.4 Gym Diary 10

2.2.5 RepCount Gym Log 10

2.2.6 Gym Log+ 11

2.3 Vertailun tulokset 13

3 Cross-platform-mobiilikehitys 15

3.1 Yleistä 15

3.2 Vertailu 15

3.2.1 Lähtökohta 15

3.2.2 Xamarin.Forms 16

3.2.3 React Native 17

3.2.4 Ionic 19

3.2.5 Framework7 21

3.3 Vertailun tulokset 23

4 Harjoituspäiväkirja 24

4.1 Suunnittelu 24

4.1.1 Harjoitusohjelmat 24

4.1.2 Harjoitusohjelman suunnittelu 24

4.1.3 Tuloksien kirjaus 27

4.1.4 Treenihistoria 28

4.1.5 Asetukset 29

4.1.6 Rakenne 30

4.2 Toteutus 31

4.2.1 Navigaatio 31

4.2.2 Kieli 33

4.2.3 Realm-tietokanta 35

4.2.4 Redux 36

4.2.5 Harjoitusohjelmat 37

4.2.6 Harjoitusohjelman suunnittelu 38

4.2.7 Tuloksien kirjaus 40

4.2.8 Historia 43

5 Yhteenveto 44

5.1 Tulokset 44

5.2 Pohdinta 44

Lähteet 47

Kuviot

Kuvio 1. GymRun, treenirutiinit, harjoitussivu ja historiasivu 7
Kuvio 2. Simple Workout Log, harjoitussivu ja historiasivu 8
Kuvio 3. FitNotes, pääsivu ja harjoituksen kirjaus..... 9
Kuvio 4. Gym Diary, viikkorutiini, tuloksien kirjaus -sivu ja historiasivu 10
Kuvio 5. RepCount Gym Log, harjoitukset, rutiinin teko ja tuloksien kirjaus..... 11
Kuvio 6. Gym Log+, rutiini, tuloksien kirjaus ja historiasivu..... 12
Kuvio 7. Gym Log+, asetukset 13
Kuvio 8. Xamarin.Forms, asennusvalikko 16
Kuvio 9. Xamarin.Forms, asennusvalikko 2 17
Kuvio 10. Framework7-sovelluksen luominen 22
Kuvio 11. Harjoitusohjelmat-sivu 24
Kuvio 12. Harjoitusohjelman suunnittelu -sivu 25
Kuvio 13. Valittavat liikkeet, liikkeen lisäys sekä liikkeen muokkaus..... 26
Kuvio 14. Harjoitussivu 27
Kuvio 15. Ajastinasetukset ja ajastin 28
Kuvio 16. Historia ja liikkeen muokkaus..... 29
Kuvio 17. Asetukset 30
Kuvio 18. Sovelluksen rakenne..... 31
Kuvio 19. Tab-navigaation rakenne..... 32
Kuvio 20. Oma sovellus, harjoitusohjelmat-sivu 38
Kuvio 21. Oma sovellus, harjoitusohjelman suunnittelu -sivu ja liikkeet -sivu..... 39
Kuvio 22. Oma sovellus, sarjojen lisäys -sivu 40
Kuvio 23. Oma sovellus, harjoitussivu ja ajastin asetukset..... 41
Kuvio 24. Oma sovellus, harjoitussivu..... 42
Kuvio 25. Oma sovellus, ajastin..... 42
Kuvio 26. Oma sovellus, historiasivu 43

Taulukot

Taulukko 1. Sovelluksien vertailutaulukko 14
Taulukko 2. Sovelluksien tiedot 14
Taulukko 3. Tietoa tekniikoista 23
Taulukko 4. Esimerkkiharjoitusohjelma 25

1 Johdanto

Opinnäytetyön aiheeksi valikoitui harjoituspäiväkirjamobiilisovelluksen kehittäminen Android- sekä iOS-laitteille. Aiheen valinta oli helppo kuntosaliharrastuksen myötä. Myös halu haastaa itseään valmistamalla helppokäyttöinen harjoituspäiväkirjasovellus, jonka voisi valmistuttuaan laittaa sovelluskauppoihin kaikkien kuntosaliharrastajien saataville, vaikutti aiheen valintaan.

Sovelluskauppoja tutkittaessa huomattiin, että ilmaisia sovelluksia, jotka toimisivat molemmilla alustoilla, ei loppujen lopuksi paljon ole. Löytyi monia hyvänoloisia sovelluksia, mutta niissäkin usein oli rajoitettu ilmaisversion ominaisuuksia. Kun kriteerinä oli, että sovelluksesta tulisi löytyä myös suomen kieli, vaihtoehdot alkoivat olla todella vähissä. Sovellus haluttiin tarjota myös suomalaisille, ja monet suomalaiset eivät halua käyttää englanninkielistä sovellusta joko osaamisen puutteen takia tai periaatteesta, joten suomenkieliselle sovellukselle olisi tarvetta.

Työn tavoitteena oli valmistaa cross-platform-mobiilisovellus työkalulla, joka valittiin ”Cross-platform-mobiilikehitys” -luvussa 3 tehdyn vertailun perusteella. Tavoitteena oli valmistaa hyvin käytettävä sovellus, jota käyttäjän on mukava ja helppo käyttää kuntosalilla ollessa ilman treenin häiriintymistä. Tarkoituksena oli tarjota sovellus suomen, englannin sekä ruotsin kielellä, jotta käyttäjäkunnan on mahdollista laajentua myös Suomen rajojen ulkopuolelle. Sovellus tulee olemaan ilmainen, mutta mahdollisesti sisältää mainoksia, jotka häiritsevät mahdollisimman vähän, eivätkä riko käyttökokemusta. Sovelluksen helppouden varmistamisella on tavoitteena saada paperisten treenivihkojen käyttäjät huomaamaan mobiilisovelluksien hyödyt treenattaessa.

2 Olemassa olevien sovelluksien kartoitus

2.1 Yleistä

Opinnäytetyössä lähdettiin kartoittamaan olemassa olevia sovelluksia, jotta saatiin selville, onko tarpeellista tehdä uutta sovellusta kuntosaliharjoitusten kirjanpitoon. Sovelluksen vaatimukset olivat seuraavat:

- Ilmaisuus
- Tarjolla sekä Androidille että iOS:lle
- Ei sisällä häiritseviä mainoksia
- Suomen kieli
- Tyylikäs ulkoasu
- Eri teemoja
- Käytettävä

Sovelluksesta tulisi löytyä kaikki olennaiset toiminnot hyvän treenikirjanpidon aikaansaamiseksi. Olennaiset toiminnot ovat seuraavat:

- Harjoitusohjelman teko
- Tuloksien kirjaus
- Valmiit liikkeet
- Liikkeiden lisäys
- Ajastin lepoajan mittaamiseen
- Sekuntikello treeniajan mittaamiseen
- Edellisen treenikerran tuloksen tuloksia merkatessa
- Treenihistoria
- Yksikköjärjestelmän muuttaminen
- Supersettien lisäys

Treenisovelluksia on sovelluskaupoissa todella paljon. Tällä hetkellä maailmalla vallitsevan fitnessbuumin myötä myös sovelluksia on lukuisia moneen eri treenimuotoon. Nykyään ihmiset seuraavat sosiaalisen median kautta yhä enemmän myös ihmisiä, jotka eivät ole niin sanottuja julkkiksia, ja löytävät motivaatiota treenaamiseen sitä kautta. Tämän ansiosta myös treenisovelluksien kysyntä kasvaa. Sovelluksia löytyy esimerkiksi täydellisten vatsalihasten saamiseen, laihduttamiseen, kotona treenaamiseen kehon painolla, maratonin juoksemiseen sekä lihasten kasvatukseen.

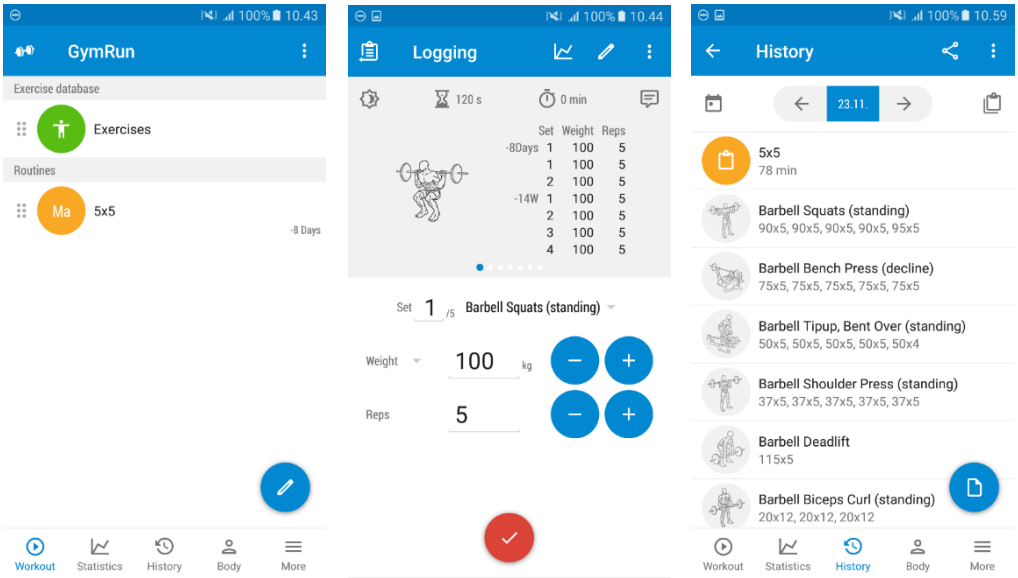
Opinnäytetyössä keskityttiin kuntosaliharjoitustulosten ylös kirjaamiseen, joten lähdettiin etsimään tähän tarkoitukseen soveltuvia sovelluksia sekä Android- että iOS-

laitteille. Jotta vertailtavia sovelluksia ei olisi liian paljon, sovelluksien määrä päätettiin rajata kuuteen sovellukseen: kolme Android- sekä kolme iOS-laitteille. Vertailuun valittiin löydetyistä sovelluksista ne, joita voitaisiin itse käyttää tuloksien kirjaamiseen.

2.2 Vertailu

2.2.1 GymRun

GymRun oli helppokäyttöinen ja hyvä sovellus. Sovelluksessa pystyi suunnittelemaan treenirutiinin, johon valittiin tehtävät liikkeet, ja tulosten kirjaus oli selkeä. Sovelluksessa oli myös cardio-treeni kirjausmahdollisuus. Tuloksia kirjatessa löytyi ajastin, joka lähti automaattisesti käyntiin, kun sarja tallennettiin. Ajastin olisi voinut olla isompi, jotta sen olisi nähnyt myös kauempaa. Tästä näkymästä löytyi myös tehtävien liikkeiden historia sekä sekuntikello treeniajan mittaamiseen. Premium-version toimintoja olivat teemaan muuttaminen, graafi tuloksista sekä kommentin lisääminen treeniin. Sovelluksessa oli myös selkeä historiasivu. Historiasivulla pystyi suodattamaan treenejä rutiinin perusteella sekä valitsemaan halutun päivän kalenterista tai painamalla edellinen- tai seuraava-nuolinäppäimiä. (Ks. kuvio 1.)

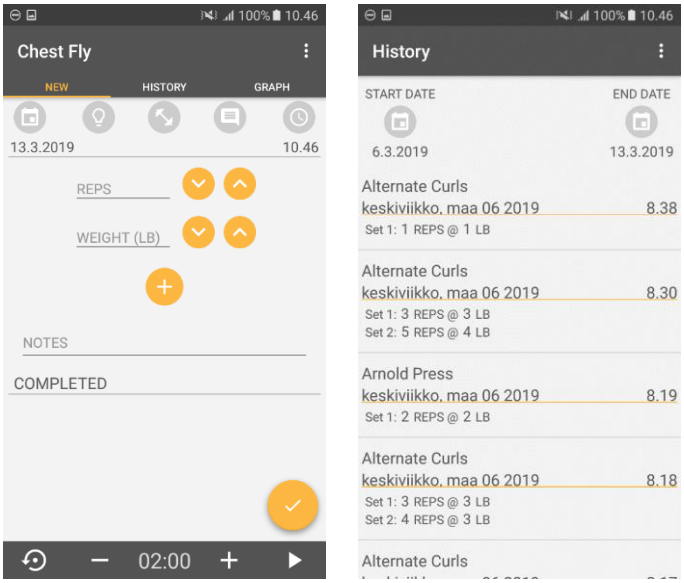


Kuvio 1. GymRun, treenirutiinit, harjoitussivu ja historiasivu

Mittayksikköjärjestelmää pystyi muuttamaan eurooppalaisen ja amerikkalaisen järjestelmän välillä. Tuloksia pystyi tuomaan ulos sovelluksesta esimerkiksi muistioon, mikä on hyvä ominaisuus. Sovelluksen ulkoasu oli hyvin tehty, eikä ollut häiritsevän näköinen. Sovelluksessa oli paljon ominaisuuksia, jotka ovat käytössä vain premium-versiossa. Parempi olisi, jos ilmaisversiosta olisi piilotettu premium-version ominaisuudet ja esitetty omalla sivulla, jotta ne eivät häiritsisi niin paljon. Sovelluksesta ei myöskään löytynyt suomen kieltä.

2.2.2 Simple Workout Log

Simple Workout Log -sovellus ajoi asiansa. Ulkoasua ei ollut ihan loppuun asti mietitty, mutta tarvittavat toiminnallisuudet löytyivät. Sovelluksessa ei ollut premium-versiolla aukeavia toimintoja häiritsemässä. Treenirutiinin läpikäyminen oli selkeä ja ajastin lepoajan mittaamiseen löytyi. Sovelluksessa pystyi kommentoimaan treeniä ja tuloksista löytyi myös graafi. Treenaussivulla pystyi näkemään, mitä painoja kannattaa käyttää, kun haluttu paino tiedettiin. Toiminto oli ihan hyvä, jos ei jaksa itse miettiä painoja, mutta melko turha. Sovelluksessa oli myös yksittäisen harjoituksen ja cardio-treenin lisäysmahdollisuus. Historiasivulla ei pystynyt suodattamaan treenejä rutiinin perusteella, mikä olisi hyvä toiminto. Sivulla pystyi valitsemaan aloituspäivän ja lopetuspäivän, missä aikaikkunassa tallennetut harjoitukset näytetään. (Ks. kuvio 2.)

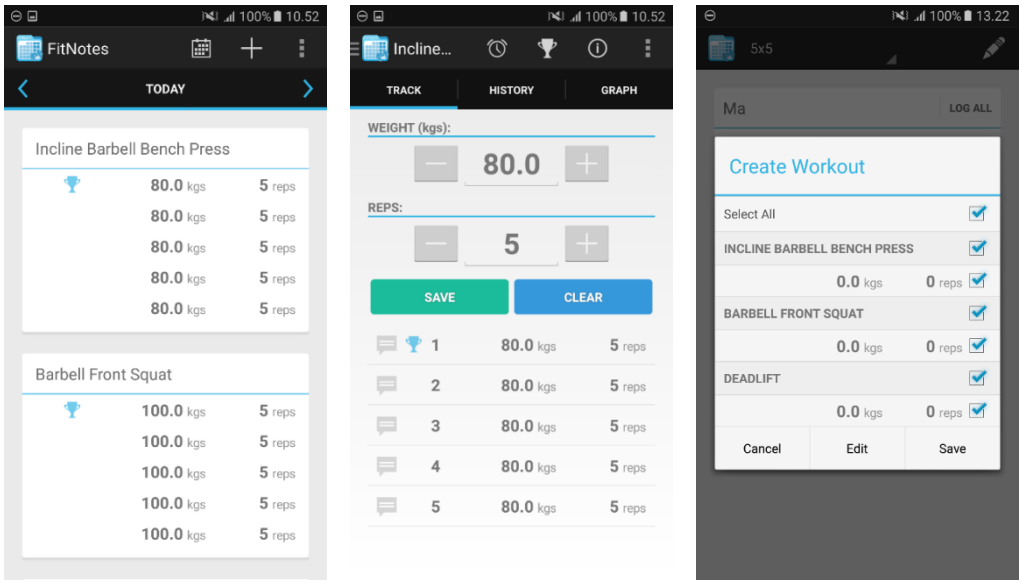


Kuvio 2. Simple Workout Log, harjoitussivu ja historiasivu

Nettiyhteyden ollessa päällä sovelluksessa oli yksi mainos alareunassa, joten se ei ollut liian häiritsevä. Suomen kieltä ei löytynyt myöskään tästä sovelluksesta.

2.2.3 FitNotes

Sovelluksen pääsivu oli toteutettu historiasivuna niin, että päivän treenit näkyivät heti. Sivulle liu'uttamalla päivä vaihtui. Uuden treenin sai lisättyä treenihistoria-osasta, jos treenejä ei vielä sinä päivänä ollut merkattuna, muussa tapauksessa ylä-palkissa olevasta plusnäppäimestä. Treenin kirjaussivulla pystyi valitsemaan ohjelman, jonka haluaa kirjata. Treenit oli mahdollista kopioida edellisistä harjoituksista tai kirjata samalla tyylillä kuin paperille kirjatessa. Tällöin ajastin ei ollut käytössä. Vaihtoehtoisesti treenin pystyi kirjaamaan treenin aikana, jolloin pystyi käyttämään ajastinta lepoajan mittaamiseen. (Ks. kuvio 3.)

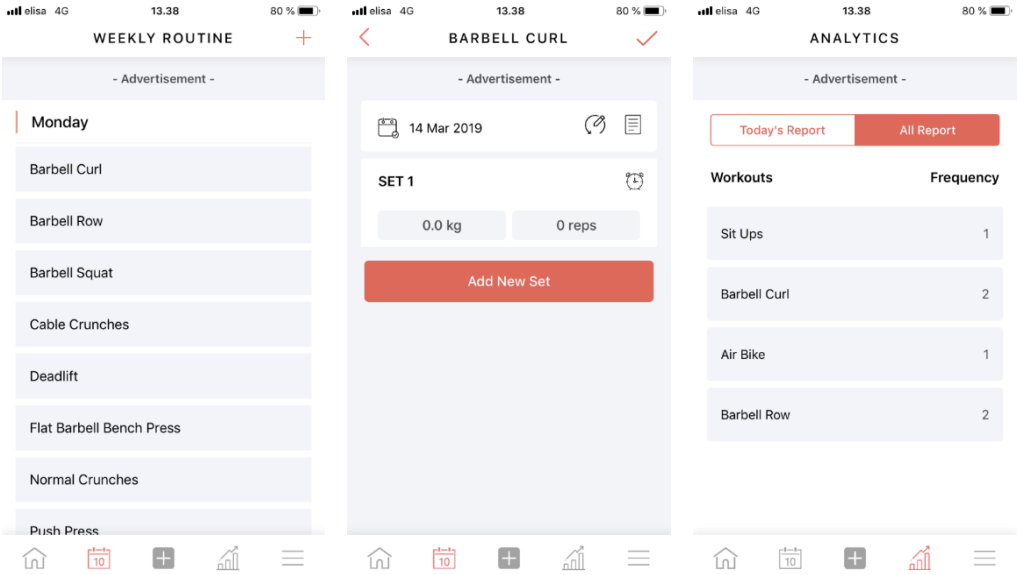


Kuvio 3. FitNotes, pääsivu ja harjoituksen kirjaus

Sovelluksessa oli hyvä graafi, jossa pystyi näkemään treenivolyymin, sarjojen määrän sekä toistojen määrän eri aikaikkunoilla. Sovelluksessa näki myös ennätykset eri liikkeille. Sovelluksessa ei ollut mainoksia eikä premium-versiota vaativia ominaisuuksia. Suomen kieltä ei sovelluksesta löytynyt.

2.2.4 Gym Diary

Gym Diary -sovelluksessa ei pystynyt tekemään ohjelmaa, jonka voisi suorittaa joka päivä, vaan siinä lisättiin viikoittainen rutiini. Tämä on vähän huono tyyli, jos haluaa tehdä eri treenikokonaisuuksia samana päivänä. Tuloksienkirjaus-sivu oli selkeä, mutta kaikki liikkeet kirjattiin erikseen ilman, että käytäisiin automaattisesti koko treenikokonaisuus läpi, jolloin sovelluksessa ei ollut treeniajan mittaamista. Sovelluksessa ei myöskään ollut ajastinta lepoajan mittaamiseen, vaan ainoastaan sekuntikello, jolla pystyi mittaamaan jokaisen sarjan pituuden. Historiasivulla näki harjoitukset joko päiväkohtaisesti tai kaikki kerralla. Tallennettuja harjoituksia pystyi myös muokkaamaan. Kaikkien harjoitusten ollessa näkyvissä, harjoitusta painamalla aukesi sivu, josta näki graafin kyseisen harjoituksen aiemmista suorituksista. (Ks. kuvio 4.)



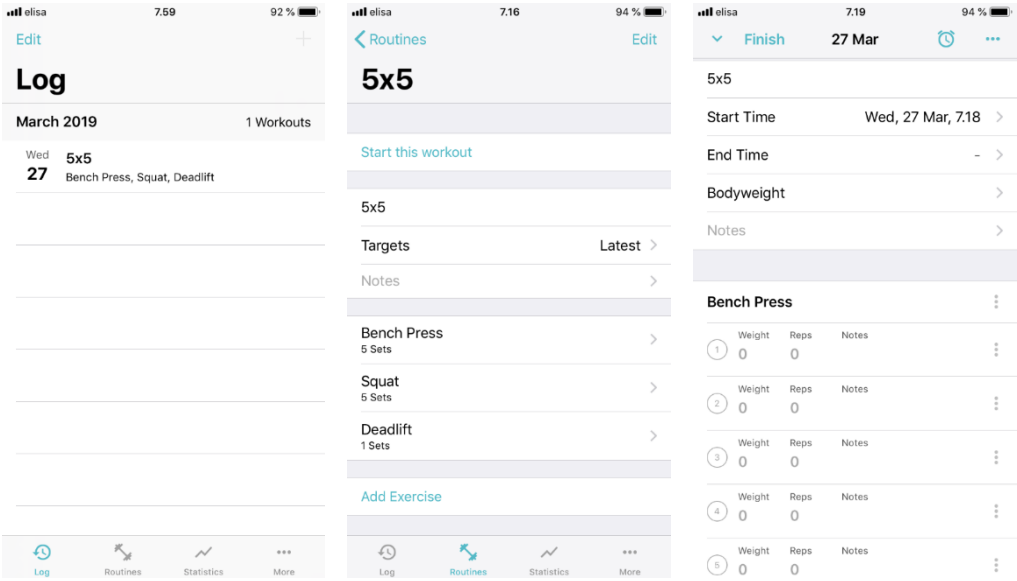
Kuvio 4. Gym Diary, viikkorutiini, tuloksien kirjaus -sivu ja historiasivu

Ylälaidassa oli yksi mainos, joka häiritsi, koska ensimmäisellä kerralla yläpalkki jäi huomaamatta tämän takia. Välillä sovelluksessa pomppasi myös pop-up-mainos näkyviin. Ulkoasu oli tyylikkään näköinen.

2.2.5 RepCount Gym Log

Sovelluksen pääsivut olivat treenihistoriasivu, rutiinisivu, statistiikkasivu sekä sivu, jossa oli erilaisia asetuksia. Sovelluksessa ei ollut mainoksia, mutta monet toiminnot

olivat käytössä vain premium-versiossa. Esimerkiksi statistiikkasivu ei ollut ilmaisena käytössä ollenkaan. Historiasivu oli yksinkertainen, jossa listattiin vain kaikki treenit kuukausittain. Treenejä ei voinut suodattaa esimerkiksi rutiinin perusteella, vaan kaikki olivat listassa aikajärjestyksessä. Sovelluksessa pystyi luomaan treenirutiinin. Tuloksienkirjaussivu oli selkeä, mutta sovelluksessa ei saanut automaattisesti ajastinta käyntiin sarjan kirjaamisen jälkeen, joten ajastinsivun joutui aina avaamaan ja sulkemaan, jos sitä halusi käyttää, mikä oli huono. Tulokset tulivat automaattisesti edellisestä harjoituksesta, mikä helpotti kirjausta. (Ks. kuvio 5.)

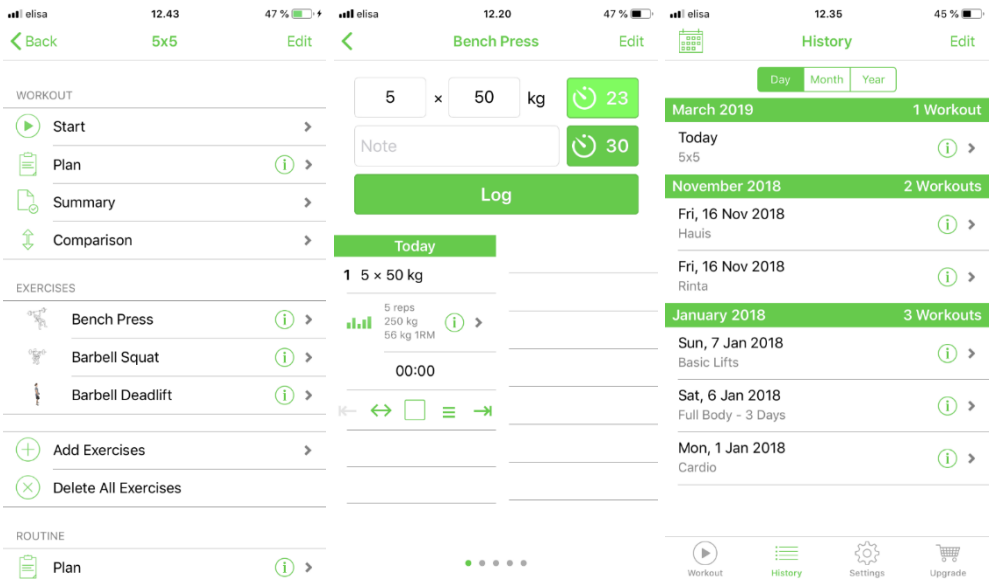


Kuvio 5. RepCount Gym Log, harjoitukset, rutiinin teko ja tuloksien kirjaus

2.2.6 Gym Log+

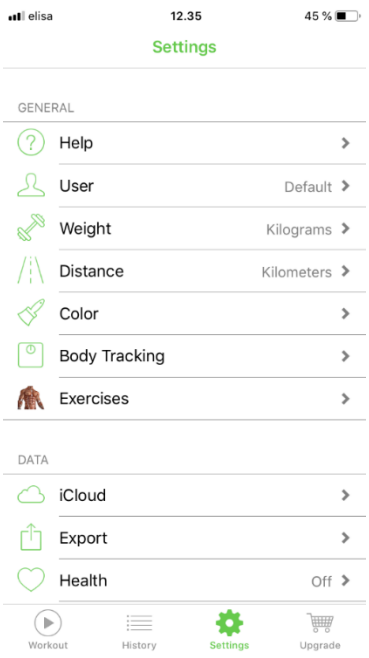
Sovelluksen pääsivut olivat treenisivu, historiasivu, asetukset-sivu sekä sivu, josta pystyi ostamaan ominaisuuksia sovellukseen esimerkiksi treeniohjelmia. Treenisivulla luotiin rutiini, johon lisättiin halutut liikkeet. Tuloksienkirjaussivulla oli ajastin, joka lähti käyntiin tuloksen kirjattua. Sovellus laski myös kokonaistreeniajan sekä sarjojen keston. Ajastimen aikaa pystyi säätämään vain asetuksista, joka oli huono juttu. Tuloksien kirjauksen yhteydessä näki yhteenvedon treenistä, sekä pystyi vertailemaan treeniä toisen päivän treeniin. Historiasivulla oli kolme vaihtoehtoa näyttää tuloksia. Päivän ollessa valittuna treenit listattiin kuukausittain uusimmasta vanhimpaan. Treeniä klikatessa aukesi sivu, josta näki treenitulokset ja halutessaan tuloksia pystyi

muokkaamaan tai poistamaan. Kuukausi- ja vuosinäkymissä näki tilastotietoja kuukauden- tai vuoden treeneistä. Esimerkiksi treenien määrän, treeniajan sekä sarjojen- ja toistojen määrän. Historia sivulla oli myös kalenteri, mistä oli mahdollista valita haluttu päivä nähdäkseen kyseisen päivän treenit. (Ks. kuvio 6.)



Kuvio 6. Gym Log+, rutiini, tuloksien kirjaus ja historiasivu

Asetukset sivulla oli perusasetuksia, kuten paino- ja mittayksikön muuttaminen. Sovelluksessa pystyi muokata ulkoasua vaihtamalla tekstiväriä, pohjan väriä, sekä painikkeiden väriä. Asetukset sivulla näki statistiikkaa treeneistä, esimerkiksi treenipäivien määrän, treeneihin käytetyn ajan, sarjojen- ja toistojen määrän. Sivulla oli myös ennätykset osio. (Ks. kuvio 7.)



Kuvio 7. Gym Log+, asetukset

2.3 Vertailun tulokset

Sovelluksien ominaisuuksista luotiin vertailutaulukko (ks. Taulukko 1), jotta voitiin nähdä nopealla silmäyksellä, mitä samoja ja eri toimintoja kyseisissä sovelluksissa on. Vertailun perusteella todettiin, että yksikään vertailuun valituista sovelluksista ei täyttänyt kaikkia vaadittuja kriteerieitä. Esimerkiksi mikään näistä sovelluksista ei ollut saatavilla kummallekin alustalle, ja suomen kieli puuttui kaikista sovelluksista. Sovelluksia on todella paljon saatavilla, ja vertailuun valittiin yksinkertaisia sovelluksia, joista löytyy helposti tarvittavat toiminnot, eikä ole liikaa turhia toimintoja. Vertailun perusteella todettiin myös, että tähän kategoriaan mahtuu vielä uusia sovelluksia lisäämään kilpailua markkinoille.

Taulukko 1. Sovelluksien vertailutaulukko

	GymRun	Simple Workout Log	FitNotes	GymDiary	RepCount Gym Log	Gym Log+
Ilmainen	✓	✓	✓	✓	✓	✓
Suomen kieli	✗	✗	✗	✗	✗	✗
Android	✓	✓	✓	✗	✗	✗
iOS	✗	✗	✗	✓	✓	✓
Ei mainoksia	✓	✗	✓	✗	✓	✓
Tyylikäs ulkoasu	✓	✗	✓	✓	✗	
Harjoitusohjelman teko	✓	✓	✓	✗	✓	✓
Tuloksien kirjaus	✓	✓	✓	✓	✓	✓
Valmiita liikkeitä	✓	✓	✓	✓	✓	✓
Liikkeiden lisäys	✓	✓	✓	✓	✓	✓
Ajastin	✓	✓	✓	✓	✓	✓
Sekuntikello	✓	✗	✗	✗	✗	✓
Edelliset tulokset	✓	✓	✓	✓	✓	✓
Treenihistoria	✓	✓	✓	✓	✓	✓
Yksikön muuttaminen	✓	✓	✓	✓	✗	✓
Supersetit	✓	✗	✗	✗	✗	✓
Vaihdettava teema	✗	✓	✗	✓	✓	✓
Arvosana 1-5	4,5	3	3,5	3,2	3,2	3

Taulukossa 2 on esitetty tietoa sovelluksista. iPhonelle tehdyistä sovelluksista ei ollut saatavilla asennusmääriä ja arvostelutkin olivat vähäisiä verrattuna Android-sovelluksiin.

Taulukko 2. Sovelluksien tiedot

	Asennukset	Arvostelut	Tähdet
GymRun	500 000+	8012	4,6
Simple Workout Log	500 000+	9165	4,6
FitNotes	1 000 000+	18 598	4,6
GymDiary	ei tiedossa	16	4,8
RepCount Gym Log	ei tiedossa	81	4,7
Gym Log+	ei tiedossa	5	4,8

3 Cross-platform-mobiilikehitys

3.1 Yleistä

Natiivin mobiilisovelluksen kehitys on kovaa työtä, ja kehittäjien täytyy osata käyttää erilaisia kehitysympäristöjä ja eri kieliä halutessaan kehittää sovelluksia eri alustoille. Kehittäjien onneksi on kehitetty cross-platform-tekniikoita, jotka mahdollistavat kirjoitetun koodin käyttämistä useammalla alustalla. Cross-platform-tekniikoissa käytetään kieliä, kuten JavaScript ja C#, mitkä mahdollistavat esimerkiksi web-ohjelmoijien siirtymisen mobiilikehityksen pariin helpommin ja vaivattomammin, kun pystyy hyödyntämään jo olemassa olevaa taitoa. (Holland 2018.)

Vaihtoehtoja cross-platform-kehitykselle on useita, mutta mikä tekniikka olisi hyvä kuhunkin tarkoitukseen? Osa tekniikoista tekee sovelluksesta natiivin, mikä tarkoittaa sitä, että käyttöliittymät käännetään natiiveiksi käyttöliittymäkomponenteiksi ja kaikki laiteohjelmointirajapinnat ovat käytettävissä, kuten esimerkiksi kamera. Osa tekniikoista tekee sovelluksesta hybridisovelluksen, mikä tarkoittaa sitä, että sovellus luodaan HTML-tekniikoilla, kuten verkkosivut, mutta sovellukset on pakattu ja ne näyttävät natiiveilta sovelluksilta. Hybridisovellus ajetaan web-selaimessa ilman web-selaimen ominaisuuksia, ja sovelluksissa on rajoitettu pääsy laiteohjelmointirajapintoihin. (Holland 2018.)

3.2 Vertailu

3.2.1 Lähtökohta

Tekniikoiden vertailuun valittiin neljä eri cross-platform-sovelluskehystä, joilla pystyy valmistamaan sovelluksen sekä Androidille että iOS:lle. Tekniikoiden vertailussa tarkoituksena oli selvittää, mikä sovelluskehys olisi järkevin valinta sovelluksen kehittämiseen. Sovelluskehyksen valinnassa otettiin huomioon sovelluskehyksen

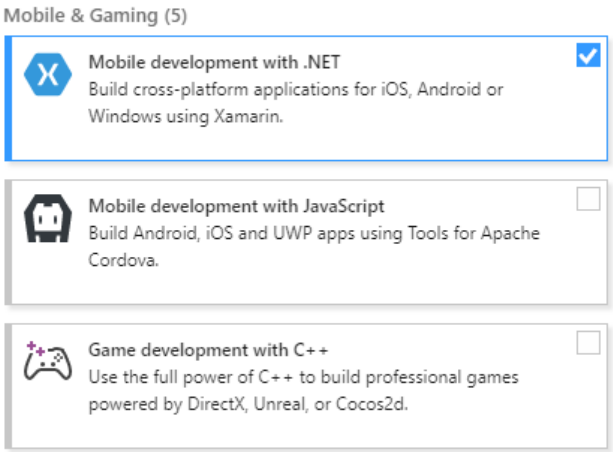
- laajuus
- käyttäjäkunta
- helppokäyttöisyys
- ympäristön pystytys
- työmarkkinat.

3.2.2 Xamarin.Forms

Xamarin esitteli Xamarin.Formsin toukokuussa 2014 (Petzold 2016, 8). Xamarin.Forms mahdollistaa kehittäjien rakentaa cross-platform-sovelluksia Android- , iOS- sekä Windows-alustoille (Xamarin.Forms n.d.).

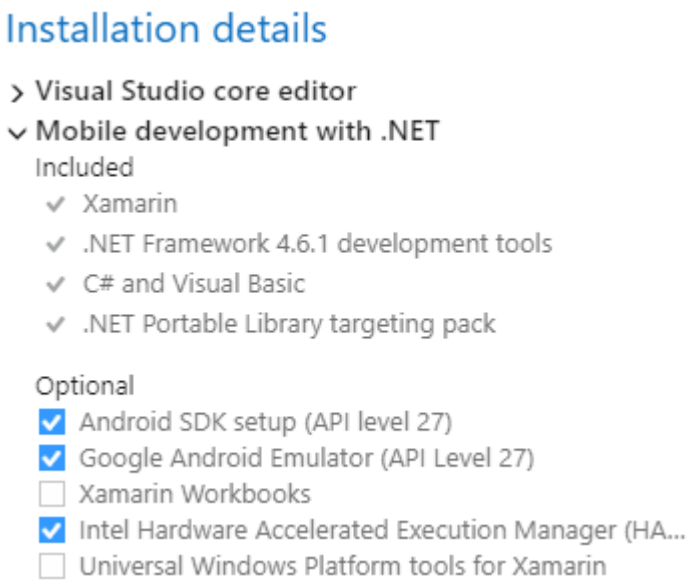
Xamarin.Formsissa koodi ja käyttöliittymä on jaettu jokaiselle alustalle. Tämä tarkoittaa sitä, että koodi kirjoitetaan vain kerran ja se käännetään natiiviksi jokaiselle alustalle. Xamarin.Formsissa ohjelmointikielenä on C# ja käyttöliittymä tehdään joko XAML-merkintäkielellä tai ohjelmallisesti käyttäen C#-kieltä. Xamarin.Forms-ohjelman arkkitehtuurissa on .NET-projekti, johon kirjoitetaan kaikki käyttöliittymät ja toiminnallisuudet. Tämän lisäksi on iOS- , Android- ja Windows-projektit, joihin tulevat kaikki koodit, joilla halutaan tehdä jotain vain tietylle alustalle. (Xamarin.Forms Quickstart Deep Dive 2018.)

Xamarin.Forms-kehitysympäristön asennus oli todella helppoa ja melko nopeaa. Asentaakseen ympäristön tarvitsi asentaa vain Visual Studio ja valita mobiilikehitystyökalut asennettavaksi. (Ks. kuvio 8.)



Kuvio 8. Xamarin.Forms, asennusvalikko

Tämän jälkeen tarkistettiin, että Android SDK on valittuna asennuspakettiin (ks. kuvio 9).



Kuvio 9. Xamarin.Forms, asennusvalikko 2

Visual Studio oli melko raskas ohjelma ja Xamarin.Formsilla kehittäminen oli melko hidasta. Yksinkertaisen sovelluksen kääntämiseen puhelimeen meni yli 20 sekuntia ja sovelluksen kasvaessa myös tämän aika kasvoi, jopa yli minuutin mittaiseksi, riippuen tietenkin koneen tehosta. Muutoksen jälkeen uudestaan kääntämiseen ja ohjelman suorittamiseen meni hiukan vähemmän aikaa. Visual Studiossa oli kuitenkin hyvä debugger, mikä helpotti virheiden löytymistä sekä graafinen käyttöliittymä Git-versionhallintaan.

3.2.3 React Native

React Native on Facebook Inc:n kehittämä cross-platform-mobiilisovelluskehys Android- ja iOS-alustoille. React Native mahdollistaa natiivin sovelluksen kehittämisen JavaScriptillä, sekä Reactilla. React Native toimii kuten React, joten sovellus koostuu komponenteista. Jos osaa Reactia, niin on helppo siirtyä React Nativen pariin niiden yhtäläisyyksien takia. React Nativella kehittäessä tehdään paljon uusia komponentteja. Kaikki mitä näytöllä näkyy, on eräänlainen komponentti. Komponentit voivat olla todella yksinkertaisia. Komponentti vaatii vain render-funktion, joka palauttaa JSX:n, joka renderöidään näytölle. (Learn the Basics 2018.)

React Nativea käyttäessä sovellus ei ole web-, HTML5- tai hybridisovellus, vaan oikea natiivisovellus. Koska React Native käyttää samoja käyttöliittymäkomponentteja, kuin Android- ja iOS-sovellukset, niin React Nativella rakennetut sovellukset on melko mahdotonta erottaa Objective-C:llä, Swiftillä, Javalla tai Kotlinilla rakennetuista sovelluksista. React Nativessa nämä käyttöliittymäkomponentit asetetaan yhteen käyttämällä JavaSciptiä ja Reactia. (How React Native works 2018.)

React Nativella sovelluksen rakentaminen on nopeaa. Sovellusta testatessa uudelleen kääntämisen sijasta muutokset ladataan välittömästi, joten sovellusta on nopea testata muutoksia tehdessä. React Native komponentit yhdistyvät sulavasti natiivisti kirjoitettujen komponenttien kanssa. React Nativella voi helposti kirjoittaa osan ohjelmasta natiivisti, jos on tarvetta ja osan React Nativella. Painopiste React Nativessa on kehittäjien tehokkuudessa kirjoittaa ohjelmia kaikille alustoille, tarkoittaen sitä, että sama koodi toimii jokaisella alustalla. Facebook käyttää React Nativea monissa omissa sovelluksissa ja jatkaa siihen investoimista. Tästä johtuen voidaan päätellä, että React Native tulee olemaan käytössä myös tulevaisuudessa. (How React Native works 2018.)

Tuhansia sovelluksia on tehty käyttäen React Nativea. Muutamia tunnettuja esimerkkejä ovat Facebook, Instagram, Skype ja Tesla. (Who’s using React Native? 2018)

React Nativelle on tehty monia kirjastoja helpottamaan sovelluksien käyttöliittymien tekemistä. Esimerkiksi Native Base on hyvä ja monipuolinen käyttöliittymäkomponenttikirjasto, joka mahdollistaa ominaisuuksien tekemisen, mitä React Nativesta ei suoraan löydy ilman, että tarvitsisi itse tehdä natiivisti.

Kun React Nativella aloitetaan kehittämään sovellusta, niin pitää tietää haluaako itse tehdä komponentteja natiivisti eri alustoille. Jos haluaa, niin täytyy käyttää React Native CLI -kehitysympäristöä saadakseen käyttöön natiivit kehitystyökalut. Jokaiselle alustalle pitää asentaa erikseen kehitysympäristö. iOS-kehitykseen tarvittavat työkalut toimivat vain Applen omissa käyttöjärjestelmissä, joten sovelluksen kääntäminen ja asennuspaketin tekeminen iOS-puhelimeen ei ole Windows-koneella mahdollista. (Getting Started 2018.)

Jos natiivien komponenttien tekemiseen ei ole tarvetta, niin helppo tapa on aloittaa kehitys Expo CLI -ympäristössä. Expo ei vaadi kehittäjältä mitään natiiviin ohjelmointiin liittyviä ohjelmia tai konfiguraatioita. Expolla on mahdollista testata sovellusta myös iOS-puhelimella, mutta asennuspakettia ei voi tehdä myöskään tällä ilman OS-käyttöjärjestelmällistä tietokonetta. Expon asennus on todella helppo ja nopea verrattuna React Native CLI -ympäristön pystyttämiseen. Vaikka Expolla ei voi itse kirjoittaa natiivisti eri alustoille, niin siitä huolimatta Expo tekee sovelluksesta natiiviin. (Getting Started 2018.)

Expossa on kuitenkin rajoituksia joiden johdosta ei välttämättä sovellu kaikkiin projekteihin. Jos haluaa pystyä ajamaan sovellusta taustalla, esimerkiksi soittaa musiikkia tai käyttää push-notifikaatioita, niin Expolla nämä eivät onnistu. Myöskään Bluetoothin käyttäminen ei toimi Expolla. (Why not Expo? N.d.)

3.2.4 Ionic

Ionic on perustettu vuonna 2012. Silloin natiivien sovelluksien kehitys webtekniikoilla oli vielä alussa. Ionic perustettiin halusta luoda parempi tapa web-kehittäjille luoda sovelluksia osaamillaan taidoilla. Tänä päivänä Ionic on yksi suosituimmista cross-platform-teknologioista ja sillä on luotu yli 5 miljoonaa sovellusta. Ionicilla on yli 5 miljoonan kehittäjän yhteisö yli 200 maasta. (The dev-friendly app platform for building cross-platform apps with one codebase, for any device, with the web 2019.)

Ionic on avoimen lähdekoodin käyttöliittymätyökalu, joka mahdollistaa kehittämään suorituskykyisiä ja laadukkaita mobiilisovelluksia sekä tietokonesovelluksia käyttäen HTML:ää, CSS:ää ja JavaScriptiä. Ionicissa keskitytään käyttöliittymän käyttökokemukseen, tai sovelluksen käyttöliittymän vuorovaikutukseen. Ionic on helppo oppia ja se integroituu hyvin muiden kirjastojen ja sovelluskehysten, kuten esimerkiksi Angularin kanssa. Tällä hetkellä Ionic on virallisesti integroitu Angularin kanssa, mutta tuki Vuen ja Reactin kanssa on kehitteillä. (Lucas & Wiegert 2019.)

Ionic native on TypeScript-kääre Cordova/PhoneGap-lisäosalle, jolla voi lisätä helposti natiiveja toimintoja esimerkiksi kameran käytön Ionicilla tehtyyn mobiilisovellukseen (Ionic Native n.d.).

Muutama esimerkki Ionicin käyttäjistä ovat MarketWatch, Airbus helicopters ja Sworkit. Ionic-kehitysympäristö oli helppo asentaa, mutta jos halusi käyttää natiiveja ominaisuuksia, joutui asentamaan natiivin kehitysympäristön, jolloin asennuksessa meni aikaa hieman enemmän. Ionic käyttää npm-paketteja, joten aluksi asennettiin node. Noden asennuksen jälkeen Ionic asennettiin komennolla

```
npm install -g ionic
```

Ionic-sovellus luotiin komennolla

```
ionic start myApp tabs
```

Sovelluksen nimi oli myApp ja tabs oli sovelluspohja, joka haluttiin asentaa. Ionicissa oli kolme valmista sovelluspohjaa: tyhjä, tab-navikointi ja sivunavikointi. Lisäksi koneella täytyi olla määritettynä ANDROID_HOME- ja JAVA_HOME-ympäristömuuttujat. ANDROID_HOME viittaa Android SDK:n sijaintiin tietokoneella ja JAVA_HOME Java JDK:n sijaintiin tietokoneella. Lisäksi tarvitsi lisätä PATH-muuttujaan polut, joista löytyivät Android platform-tools ja tools. Ionic-sovellus ajettiin selaimessa komennolla

```
ionic serve
```

Tämä komento avasi sovelluksen selaimeen, jolloin pystyi avaamaan selaimen debug-ikkunan ja valita näytöksi puhelimen. Ionicilla sovelluksen kehittäminen oli todella nopeaa, koska muutoksien jälkeen ohjelma suoritettiin selaimessa natiiviksi kääntämisen sijasta. Puhelimessa ajettaessa piti olla kehityspaketit asennettuna. Kun kehityspaketit oli asennettu, niin komennolla

```
ionic cordova run android –device
```

projektiin asentui kaikki androidin tarvittavat paketit ja sovellus ajettiin puhelimen kautta. Nyt käytössä oli myös laiteohjelmointirajapinnat eli esimerkiksi kameraa pystyisi käyttämään sovelluksessa.

3.2.5 Framework7

Framework7 on ilmainen avoimenlähdekoodin sovelluskehys mobiili- tai web-sovelluksien kehitykseen HTML-, CSS- ja JavaScript-tekniikoilla. Sen on kehittänyt Vladimir Kharlampidi. Ensimmäinen versio julkaistiin vuonna 2014. Framework7:llä kehitys on yhtä helppoa kuin web-sivujen teko. Framework7:n syntaksi on lähellä JQueryn syntaksia. Alun perin Framework7 rakennettiin iOS-käyttäjille, mutta myöhemmin hyväksyttiin myös Androidille. (Matthews 2018.)

Framework7 käyttää Cordovaa tehdäkseen hybridisovelluksen, jonka voi ladata Googlen ja Applen sovelluskauppaan (Kharlampidi 2019).

Framework7 on MIT-lisensoitu avoimen lähdekoodin projekti, jonka kehittämisen on mahdollistanut sponsorit. Sponsoreita ovat Yakaz, AppValley, Tommy, SecuCom, Active Waiter ja Wappler. (Framework7 2019.)

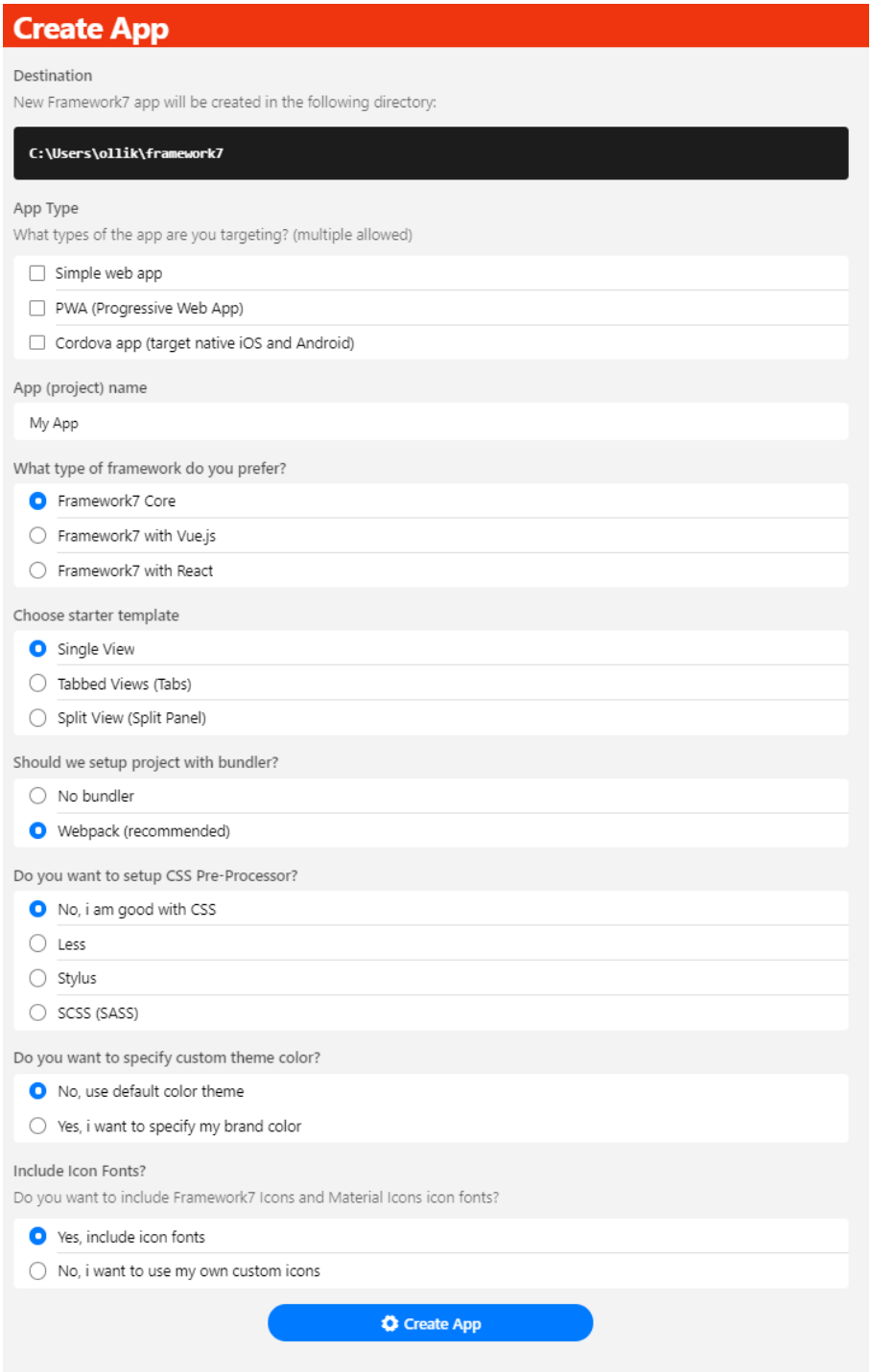
Framework7:n alkuun pääsi helposti ja nopeasti. Framework7 asennettiin komennolla

```
npm i framework7-cli cordova -g
```

Asennuksen jälkeen ajettiin komento

```
framework7 create --ui
```

jolloin localhostiin aukesi sivu, josta valittiin minkä tyyppisen sovelluksen halusi luoda (Ks. kuvio 10).



Kuvio 10. Framework7-sovelluksen luominen

Sovellustyyppejä oli kolme erilaista: Framework7 Core, Framework7 with Vue.js ja Framework7 with React. Komennolla

```
npm start
```

sovellus käynnistettiin ja localhostiin aukesi sivu, josta näki sovelluksen ja kehittämisen pystyi aloittamaan.

3.3 Vertailun tulokset

Tekniikoiden tutkimisen perusteella React Native oli paras ja mukavin vaihtoehto, johtuen dokumentaation selkeydestä ja kehittämisen nopeudesta. GitHub-arvioiden perusteella React Native oli helposti suosituin. Vertaillessa React Nativea ja Xamarin.Formsia, React Nativella oli sovelluksien kehittäminen nopeampaa. React Nativella kehittäessä käytettiin JavaScript-ohjelmointia, joka oli tutumpi kieli, kuin Xamarin.Formsissa käytetty C#. Dokumentaatioita vertaillessa React Nativen dokumentaatio oli selkeämpi, Xamarin.Formsin dokumentaation ollessa todella laaja, josta asioita oli vaikeampi löytää. React Nativea vertaillessa Ioniciin ja Framework7: ään, React Native oli ainut, joka teki sovelluksen käyttöliittymästä natiivin hybridisovelluksen sijasta. Koska hybridisovelluksissa on rajoitettu pääsy ohjelmistorajapintoihin, tämmöistä tekniikkaa ei haluttu käyttää jatkoa ajatellen. Myös työpaikkailmoituksia selaillessa, on huomattu Reactin ja React Nativen osaamisen tarve. React Nativesta oli myös aiempaa kokemusta. Mainituista syistä React Native tuntui parhaimmalle ja miellyttävimmälle, jolloin valinta osui siihen. Yksityiskohtia tekniikoista on esitetty taulukossa 3.

Taulukko 3. Tietoa tekniikoista

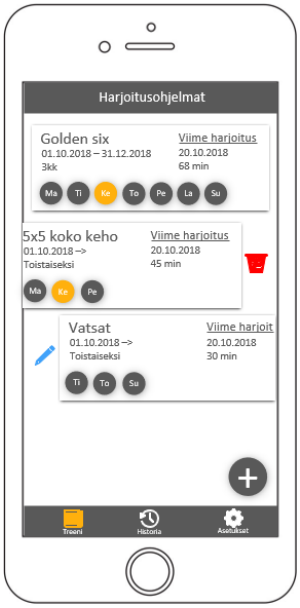
	React Native	Xamarin.Forms	Ionic	Framework7
Helppo asentaa	✓	✓	✓	✓
Nopea uudelleen lataus	✓	✗	✓	✓
Natiivi sovellus	✓	✓	✗	✗
Github tähdet	75,659	3,164	37,546	14,090
Ohjelmointikieli	JavaScript	C#	JavaScript	JavaScript

4 Harjoituspäiväkirja

4.1 Suunnittelu

4.1.1 Harjoitusohjelmat

Harjoitusohjelmat tulevat olemaan pääsivuna treeni nimisessä tab-navigointi sivussa. Harjoitusohjelmat sivulla listataan kaikki käytössä olevat harjoitusohjelmat. Ohjelmat listautuvat sivulle kortteina, jossa näkyvät ohjelman nimi, aloitus ja lopetus päivä, jäljellä oleva kesto ohjelmalle, viimeisen harjoituksen päivä ja kesto sekä päivät, joina on tarkoitus treenata. Korttia vasemmalle liu'uttamalla tulee esille ohjelman poistamisnappi ja oikealle liu'uttamalla tulee esille muokkausnappi. Oikeassa alakulmassa on harjoitusohjelmien lisäys-nappi. (Ks. Kuvio 11.)



Kuvio 11. Harjoitusohjelmat-sivu

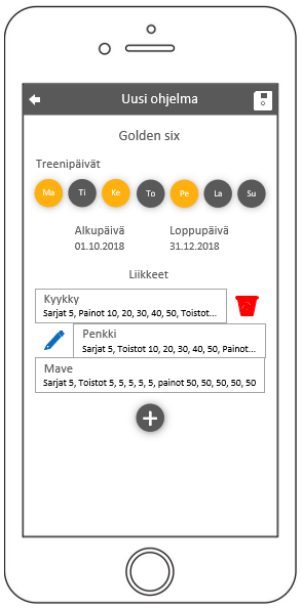
4.1.2 Harjoitusohjelman suunnittelu

Sovelluksessa pitää pystyä suunnittelemaan harjoitusohjelma. Ohjelmaan kirjataan tehtävät liikkeet, sarjamäärät, toistomäärät sekä käytettävä paino. Ohjelmaan pitää pystyä merkitsemään, jos jotkin liikkeet tehdään supersarjana, eli toistetaan peräkkäin ilman lepoa. Yksinkertainen harjoitusohjelma on esitetty taulukossa 4.

Taulukko 4. Esimerkkiharjoitusohjelma

Liike	Sarjat	Toistot	Paino/kg
Kyykky	5	5	100
Penkki	5	5	70
Leuat	5	5	20

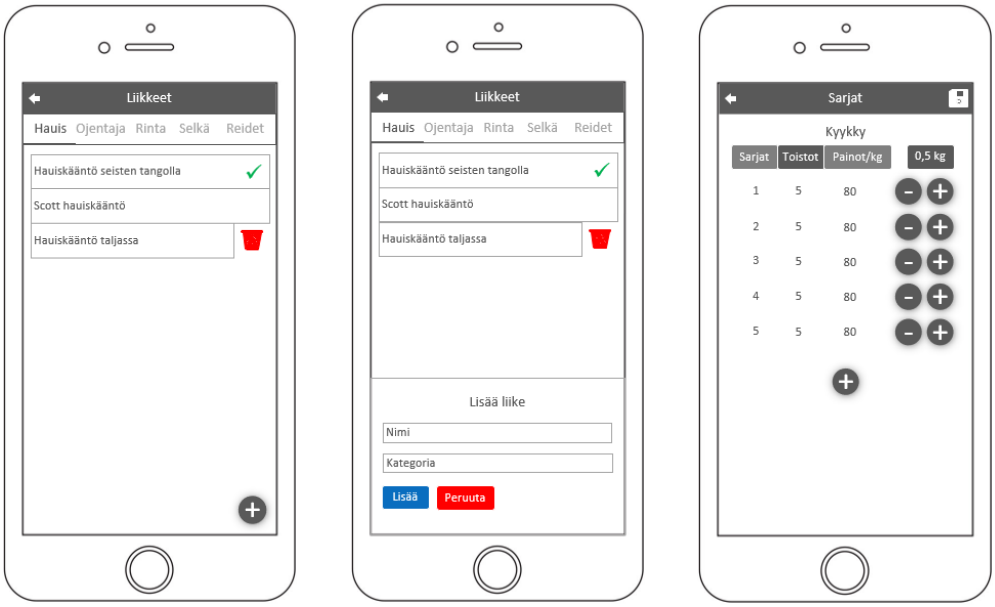
Harjoitusohjelman suunnittelu -sivulla Laitetaan harjoitusohjelmalle nimi, valitaan päivät, joina on tarkoitus treenata, lisätään ohjelman alkamis- ja loppumispäivä sekä valitaan lisättävät liikkeet. Liikkeet-otsikon alla on plusnappi, josta päästään sivulle, mistä valitaan liikkeet ohjelmaan. Kun liike on valittu ja tullaan takaisin ohjelman lisäys -sivulle, liikkeet ilmestyvät listana liikkeet osioon. Listaelementtiä vasemmalle liu'uttamalla tulee esiin nappi, josta päästään valitun liikkeen sivulle, missä asetetaan ohjelmassa tehtävät sarjat, toistot, ja painot joilla, on tarkoitus aloittaa treeni. Kun liikkeen tiedot on kirjattu, ne ilmestyvät listaelementtiin liikkeen nimen alapuolelle. Listaelementtiä vasemmalle liu'uttamalla tulee esiin liikkeen poistamisnappi. Sivun oikeassa yläkulmassa on ohjelman tallennusnappi. (Ks. kuvio 12.)



Kuvio 12. Harjoitusohjelman suunnittelu -sivu

Liikkeiden valinnan helpottamiseksi liikkeet sivulla on tab-navigointi eri lihasryhmille, josta löytyvät tehtävät liikkeet kyseiselle lihakselle. Liikkeen valinnan jälkeen listaelementtiin ilmestyy ”valittu”-merkki kyseisen liikkeen kohdalle. Listaelementtiä vasemmalle liu’uttamalla tulee esille nappi, josta saa poistettua kyseisen liikkeen. Vain itse lisäämät liikkeet on mahdollista poistaa. Oikeassa alakulmassa on nappi, josta saa lisättyä liikkeen. Nappia painamalla sivun alalaitaan tulee osio, josta liikkeen lisäys onnistuu. (Ks. kuvio 13.)

Liikkeen muokkaus -sivulle otsikoksi tulee liikkeen nimi ja otsikon alapuolelle otsikot sarja, toistot, painot sekä lisäysmäärä. Toistot ja painot otsikoista painamalla valinta aktivoituu kyseiselle osalle, jolloin plus- ja miinusnappeja painamalla arvo muuttuu. Lisäys määrää painamalla lisäyksen määrä muuttuu. Lisäys määrät ovat 0,25, 0,5, 1 sekä 10. Otsikoiden alapuolella on plus-nappi, josta saa lisättyä sarjoja. Oikeassa yläkulmassa on sarjojen tallennusnappi. (Ks. kuvio 13.)

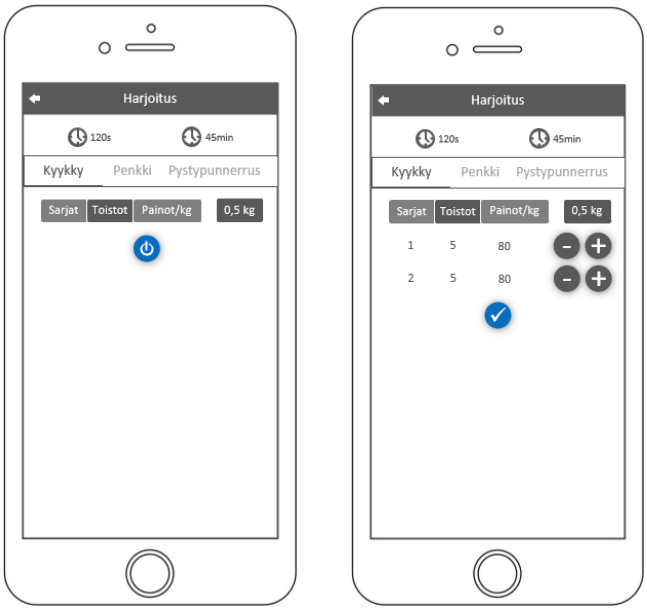


Kuvio 13. Valittavat liikkeet, liikkeen lisäys sekä liikkeen muokkaus

4.1.3 Tuloksien kirjaus

Tuloksiin kirjataan suoritettu liike, käytetty paino, tehdyt sarjat sekä tehdyt toistomäärät sarjassa. Tuloksien kirjaus kohdassa pitää näkyä edellisen treenin tulokset. tuloksien kirjaamisen yhteydessä tulee olla ajastin lepoajan mittaamiseen sarjan jälkeen, sekä sekuntikello treeniajan mittaamiseen.

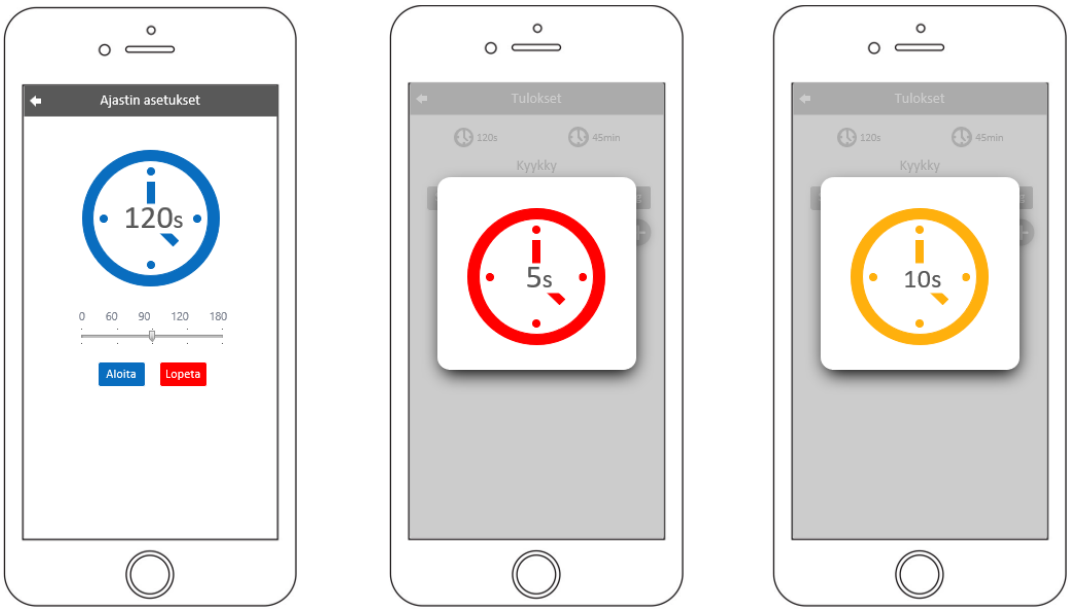
Harjoitusohjelmat-sivulla ohjelmaa napauttamalla päästään harjoitussivulle, jossa treenin kirjaus tapahtuu. Sivun ylälaidassa on ajastinnappi, josta näkee valitun ajan ja sitä painamalla avautuu ajastinasetukset-sivu. Ajastimen vieressä on sekuntikello, joka mittaa treenin keston, kun treeni on aloitettu. Kellojen alapuolella näkyy treenissä tehtävät liikkeet tab-navigaationa. Liikkeiden alapuolella näkyy sarjan lisäyksen otsikot. Otsikoiden alapuolella on treenin käynnistys nappi. Kun treeni on aloitettu, ensimmäinen sarja tulee näkyviin. Sarjaan lisätään ohjelman suunnittelussa määritetyt arvot. Kun sarja on tehty, alapuolella olevasta napista tallennetaan sarja, jolloin ajastin käynnistyy. Ajastimen loputtua toinen sarja on ilmestynyt näkyviin tallentamista varten. Plus- ja miinusnapeista saa muutettua arvoja. Kun ensimmäinen harjoitus kerta on suoritettu, seuraavan harjoitukseen ladataan sarjojen arvot edellisestä harjoituksesta. Tällöin aina edellinen treeni näkyy tuloksia merkatessa. (Ks. kuvio 14.)



Kuvio 14. Harjoitussivu

Ajastinasetukset sivulla valitaan haluttu lepoaika. Sivulla on kello-ikoni, jonka keskellä on ajastimen arvo. Arvoa painamalla aktivoituu tekstikenttä, jolloin halutun arvon voi itse kirjoittaa. Vaihtoehtoisesti kellon alapuolella on liukusäädin ajan valitsemiseen. Ajastinsivulla on myös mahdollista käyttää ajastinta. (Ks. kuvio 15.)

Sarjan lisättyä ilmestyy ajastin isolla näkyviin, jolloin ajastimen näkee myös kauempaa. Kun ajastimessa on jäljellä kymmenen sekuntia, ajastin muuttuu keltaiseksi, jolloin voi jo alkaa keskittymään seuraavaan sarjaan. Viiden sekunnin kohdalla ajastin muuttuu punaiseksi. (Ks. kuvio 15.)



Kuvio 15. Ajastinasetukset ja ajastin

4.1.4 Treenihistoria

Sovelluksessa tulee olla selkeä treenihistoria listamuodossa sekä graafi. Graafista pitää pystyä näkemään treenien nousujohteisuus. Treenihistorian yhteydessä pitää olla myös lista ennätyksistä.

Historiasivulla voi valita halutun listausnäkymän harjoituksille. Valittavana on päivän, viikon tai kuukauden aikana tehdyt harjoitukset. Listauksessa näkyy ohjelman nimi, treenin kesto sekä päivämäärä. Tietojen alle listautuu tehdyt liikkeet, sarjamäärät, tehdyt toistot sekä käytetyt painot. Historiaan on tarkoitus tulla myös kalenteri, josta

voi katsoa tietyn päivän treenit, graafi tuloksista sekä tuloksien ulosvienti sovelluksesta. Historian listaelementtiä painamalla aukeaa sivu, josta voi muokata kyseistä liikettä, jos tuloksien kirjauksessa on tapahtunut virhe. (Ks. kuvio 16.)

Kuvio 16. Historia ja liikkeen muokkaus

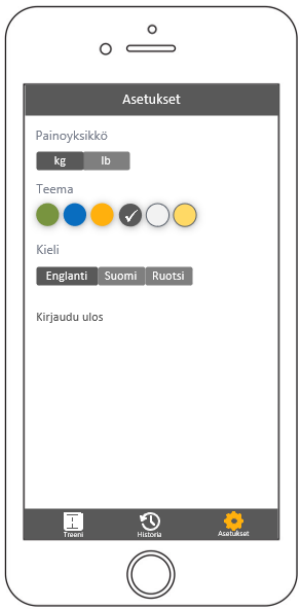
4.1.5 Asetukset

Sovelluksen ulkoasun tulee voida vaihtaa valittavana olevista teemoista. Valittavana on erivärisiä teemoja, jotta jokainen voi vaihtaa sovelluksen ulkoasun itselleen miellyttävämmäksi.

Sovelluksessa tulee olla valittavana painomäärät eurooppalaisen ja amerikkalaisen yksikön välillä.

Sovelluksen kieli tulee olla valittavana englannin, suomen ja ruotsin välillä, jotta sovelluksesta saa mahdollisimman monelle käyttäjälle mukavan käyttää. Tällöin myös käyttäjäkunnalla on mahdollisuus kasvaa isommaksi.

Asetukset-sivulla näkyy painoyksikön muokkaus napit kilogramman ja paunan välillä. Sivulla on myös mahdollista vaihtaa sovelluksen teemaa sekä kieltä. (Ks. kuvio 17.)

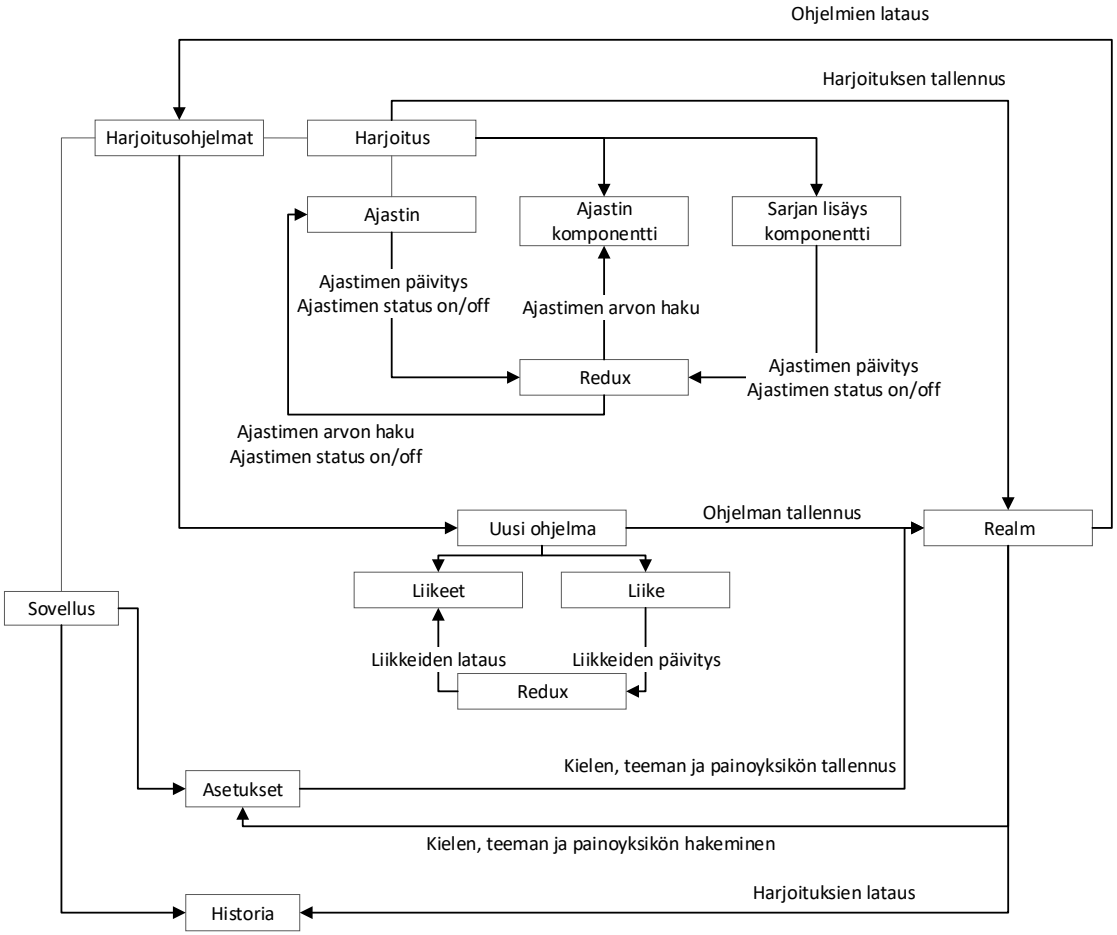


Kuvio 17. Asetukset

4.1.6 Rakenne

Sovelluksen rakenne koostuu eri sivuista, joista jokainen sivu on oma React-komponentti. Sivujen sisällä olevat toistuvat elementtikokonaisuudet tehdään omana erillisenä komponenttina, minimoidakseen uudelleen kirjoittaminen. Esimerkiksi samanlaisista listaelementeistä tehdään yksi komponentti, jota käytetään kaikkialla. Sovelluksessa on tab-navigointi, joten sovelluksen pääsivut ovat tab-sivut. Tab-sivujen sisällä olevat sivut määritetään stack-navigaationa. (Ks. kuvio 18.)

Redux JavaScript-kirjastolla hallinnoidaan sovelluksen tietoja, joihin on pääsy usealla eri komponentilla. Liikkeiden lisäys -sivulla olevat liikkeet merkataan valituksi Reduxin tilan sisällä, jolloin ohjelman teko sivulla voidaan Reduxin tilasta ladata kaikki merkityt liikkeet. Myös ajastimen arvoa ja käynnissä oloa hallinnoidaan Reduxin kautta. Puhelimessa olevan kielen valitseminen tapahtuu automaattisesti puhelimen kielen mukaan, mutta halutessaan kielen voi vaihtaa, jolloin valinta kulkee tietokannan kautta. Myös teeman ja painoyksikön muuttaminen tapahtuu tietokannan avulla. (Ks. kuvio 18.)



Kuvio 18. Sovelluksen rakenne

4.2 Toteutus

4.2.1 Navigaatio

Teknologiaksi sovelluksen valmistamiseen valikoitui React Native. Sovelluksen valmistamien alkoi tab-navigoinnin tekemisellä. Navigointiin käytettiin React Navigation nimistä JavaScript-kirjastoa. React Navigation asennettiin npm-pakettina komennolla

```
npm install --save react-navigation
```

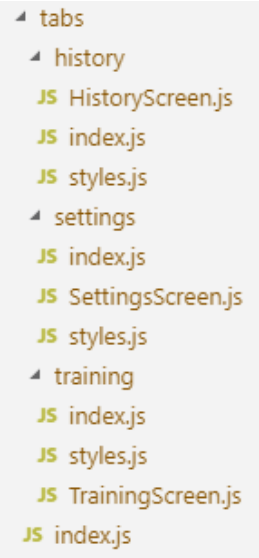
Koska sovelluksessa käytettiin React Native Cli -kehitysympäristöä expon sijasta, asennuksen yhteydessä asennettiin myös React Native Gesture Handler komennolla

```
npm install --save react-native-gesture-handler
```

Lopuksi yhdistettiin natiivit riippuvuudet komennolla

```
react-native link react-native-gesture-handler
```

Komennon ajamisen jälkeen täytyi tarkistaa android/settings.gradle-tiedostosta, että yhdistäminen onnistui. Tällä hetkellä link-komento laittaa kauttaviivat kenoviivoiksi, jolloin sovellusta ajaessa tulee virhe. Tab-navigaation rakenne sovelluksessa on näkyvissä kuviossa 19.



Kuvio 19. Tab-navigaation rakenne

Tabs-kansiossa olevassa index-tiedostossa luotiin TabNavigator, jossa määritetään mitä navigointi pitää sisällään. Tässä tapauksessa navigointi tuli sivun alalaitaan. Alapuolella on esitetty osa TabNavigatorista. Muut sivut ovat määritettynä samalla tavalla.

```
const TabNavigator = createBottomTabNavigator(
    {
      Training: {
          screen: TrainingStack,
          navigationOptions: () => ({
              title: I18n.t('Tabs.TrainingTabText')
          })
        },
```

TabNavigatorin sisällä määritettiin myös mitä tabeissa tulee lukemaan ja mikä ikoni tulee näkymään. Jokaisen tabissa olevan sivun index-tiedostossa luotiin StackNavigator, jotta voitiin navigoida tietyssä tabissa sivulta toiselle. Alapuolella esitetty osa StackNavigatorista. Muut sivut ovat määritettynä samalla tavalla.

```
const Stack = createStackNavigator({
    Training: {
        screen: TrainingScreen,
        navigationOptions: () => ({
            title: I18n.t('TrainingScreen.TrainingTitle'),
```

4.2.2 Kieli

Sovelluksesta tehtiin monikielinen käyttämällä i18next JavaScript-kirjastoa. Kirjaston tueksi tarvitsi myös React-i18next paketin. Paketit asennettiin projektiin npm-paketteina komennolla

```
npm install react-i18next i18next --save
```

i18next-kirjastolla ei pysty näkemään, mikä kieli puhelimessa on käytössä, joten tähän tarkoitukseen asennettiin react-native-device-info komennolla

```
npm install --save react-native-device-info
```

Kirjasto täytyi vielä yhdistää natiivien riippuvuuksien kanssa komennolla

```
react-native link react-native-device-info
```

Tämän jälkeen puhelimen kieli saatiin sovelluksen käyttöön määrittämällä muuttuja, arvolla DeviceInfo.getDeviceLocale(). Sovellukseen tehtiin tiedosto i18n, jossa otettiin käyttöön i18next-paketti. Tiedostossa määritettiin resurssit, jossa määritettiin mitä kieliä on valittavana. Kaikki sovelluksen tekstit kirjoitettiin jokaiselle kielelle omaan JSON-tiedostoon, josta käännökset haettiin. Init-osiossa oleva lng: DeviceInfo.getDeviceLocale() palautti puhelimen käytössä olevan kielen. Jos kieltä ei ollut resursseissa, niin palautettiin oletuskielenä englanti.

I18n-tiedosto näytti tältä:

```
import I18n from 'i18next';
import { reactI18nextModule } from 'react-i18next';

import fi from './finnish.json';
import en from './english.json';

import DeviceInfo from 'react-native-device-info';

const resources = {
    en: {
        translation: en
    },
    fi: {
        translation: fi
    }
};

I18n
    .use( reactI18nextModule)
    .init({
        resources,
        lng: DeviceInfo.getDeviceLocale(),
        fallbackLng: 'en',

        interpolation: {
            escapeValue: false
        }
    });

export default I18n;
```

Koodissa kielet haettiin i18n-tiedoston kautta kirjoittamalla tekstin sijasta I18n.t('haluttu käännös').

```
<Text
 style={styles.LanguageTitle}
>
    {I18n.t('SettingsScreen.LanguageTitle')}
</Text>
```

Kielen vaihtaminen oli tarkoitus tehdä mahdolliseksi myös manuaalisesti, jotta sovellusta olisi mahdollista käyttää eri kielellä kuin puhelinta, mutta tällä hetkellä sitä ei ole toteutettu. Sovellukseen on toteutettu tällä hetkellä suomen ja englannin kielet.

4.2.3 Realm-tietokanta

Tiedon tallentamiseen käytettiin Realm-tietokantaa. Realm asennettiin projektiin npm-pakettina komennolla

```
npm install --save realm
```

Realm npm-paketti yhdistettiin natiivien riippuvuuksien kanssa komennolla

```
react-native link realm
```

Realmissa luodaan tietokantamalleja (eng. schema), relaatiotietokannoissa käytettävien taulujen sijasta, joissa määritetään käytettävät ominaisuudet. Tässä projektissa luotiin seuraavat mallit: Program, Exercise, Days, Move ja Set, joissa määritettiin tarvittavat ominaisuudet kullekin mallille. Esimerkiksi Exercise-schema näytti tältä:

```
export const Exercise = {
  name: 'Exercise',
  primaryKey: 'id',
  properties: {
    id: 'string',
    programName: 'string',
    timestamp: 'date',
    during: 'int',
    moves: 'Move []'
  }
}
```

Sovelluksessa määritettiin databaseOptions, jossa asetettiin tietokantatiedoston polun nimi, sovelluksessa käytettävät schemat sekä schemaversio.

```
const databaseOptions = {
  path: 'workoutApp.realm',
  schema: [Exercise, Program, Move, Set, Days, Theme],
  schemaVersion: 0
}
```

Esimerkiksi ohjelmanlisäys tapahtui insertProgram-funktiolla seuraavasti:

```
export const insertProgram = newProgram =>
new Promise((resolve, reject) => {
  Realm.open(databaseOptions).then(realm => {
    realm.write(() => {
      realm.create('Program', newProgram);
      resolve(newProgram);
```

```
    })
  }).catch((error) => reject(error));
});
```

4.2.4 Redux

Redux otettiin käyttöön sovellukseen asentamalla npm-pakettina komennolla

npm install –save redux

Lisäksi asennettiin React-Redux komennolla

npm install --save react-redux

Reduxissa koko sovelluksen tila varastoidaan Redux-varastoon. Tilan muuttaminen on mahdollista lähettämällä toiminta (eng. action), reducerille, jossa määritetään, miten tilaa muutetaan. (Abramov 2015.)

Sovelluksen juuressa sijaitsevaan index-tiedostoon luotiin redux-varasto, johon annettiin parametrina reducerit. Varasto otettiin koko sovelluksen käyttöön määrittämällä Provider, jonka sisällä määritettiin sovelluksen juurikomponentti App, jolloin varasto oli mahdollista ottaa käyttöön kaikissa App-komponentin sisällä olevissa komponenteissa.

```
const store = createStore(reducer);

const AppContainer = () =>
    <Provider store={store}>
        <App/>
    </Provider>
```

Sovelluksessa määritellyt reducerit olivat movesReducer sekä timerReducer. Reducerit liitettiin yhteen index-tiedostossa.

```
import { combineReducers } from 'redux';
import movesReducer from './movesReducer';
import {timerStatus, timerValue} from './timerReducer'

export default combineReducers({
    moves: movesReducer,
    timerStatus: timerStatus,
    timerValue: timerValue,
});
```

Reducerin sisällä määritettiin, miten tilaa muutetaan. Esimerkiksi timerReducerissa, sovelluksessa olevan ajastimen näkyvissä oloa muutettiin timerStatus-funktiossa

```
export const timerStatus = (state = initialState , action) => {
    switch(action.type) {
        case SHOW_TIMER:
            return{...state, timerStatus: true};
        case HIDE_TIMER:
        return{...state, timerStatus: false};
        case TIMER_RUNNING:
          return {...state, timerRunning: action.value};
        default:
            return state;
    }
}
```

Redux-tilan arvot oli mahdollista ottaa komponentin käyttöön mapStateToProps-funktiossa

```
const mapStateToProps = (state) => {
   return {
     timerStatus: state.timerStatus.timerStatus,
     timerRunningValue: state.timerStatus.timerRunning,
```

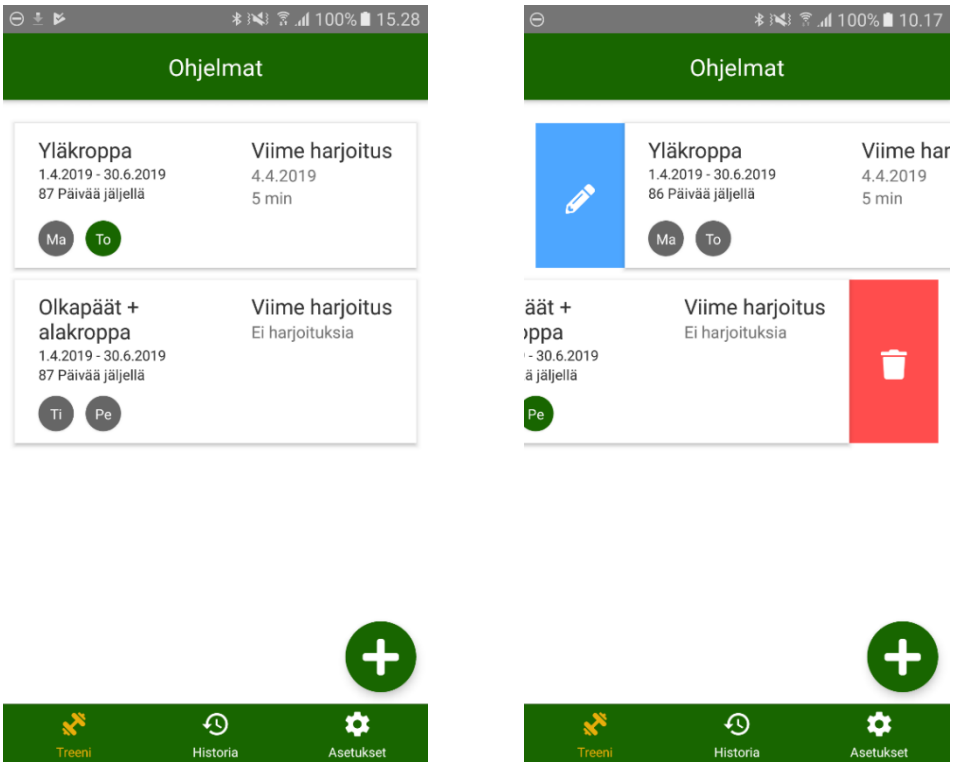
Toiminnat määritettiin mapDispatchToProps-funktiossa, jossa määritettiin toiminnan tyyppi, sekä mahdolliset arvot, jota haluttiin käyttää tilan muuttamisessa.

```
function mapDispatchToProps(dispatch) {
  return {
    showTimer: () => dispatch({type:'SHOW_TIMER'}),
    hideTimer: () => dispatch({type:'HIDE_TIMER'}),
    timerRunning: (value) => dispatch({
       type:'TIMER_RUNNING', value: value
    }),
```

4.2.5 Harjoitusohjelmat

Harjoitusohjelmat sivulla ladataan ohjelmat tietokannasta ja näytetään näytöllä kortteina, joissa on hieman yksilöllistä tietoa. Korteissa olevat tiedot ovat ohjelman nimi, ohjelman aloitus- ja lopetuspäivä, ohjelman voimassaolopäivät, viime harjoituksen päivä ja kesto sekä päivät, joina on tarkoitus treenata. Nykyinen päivä näkyy vihreällä. Kortissa on sivulle liu'utus toiminto, joka toteutettiin käyttämällä Native Base-kirjaston Swipe Row -komponenttia. Vasemmalle liu'uttamalla tulee esiin ohjelman

poistamisnappi ja oikealle muokkausnappi. Tällä hetkellä ohjelman on mahdollista poistaa, mutta muokkaus ei ole vielä toiminnassa. Sivu päivitetään aina, kun sivulle tullaan joko harjoitussivulta tai harjoitusohjelman suunnittelu -sivulta. Korttia painamalla aukeaa harjoitussivu, jossa tulokset kirjataan. (Ks. kuvio 20.)

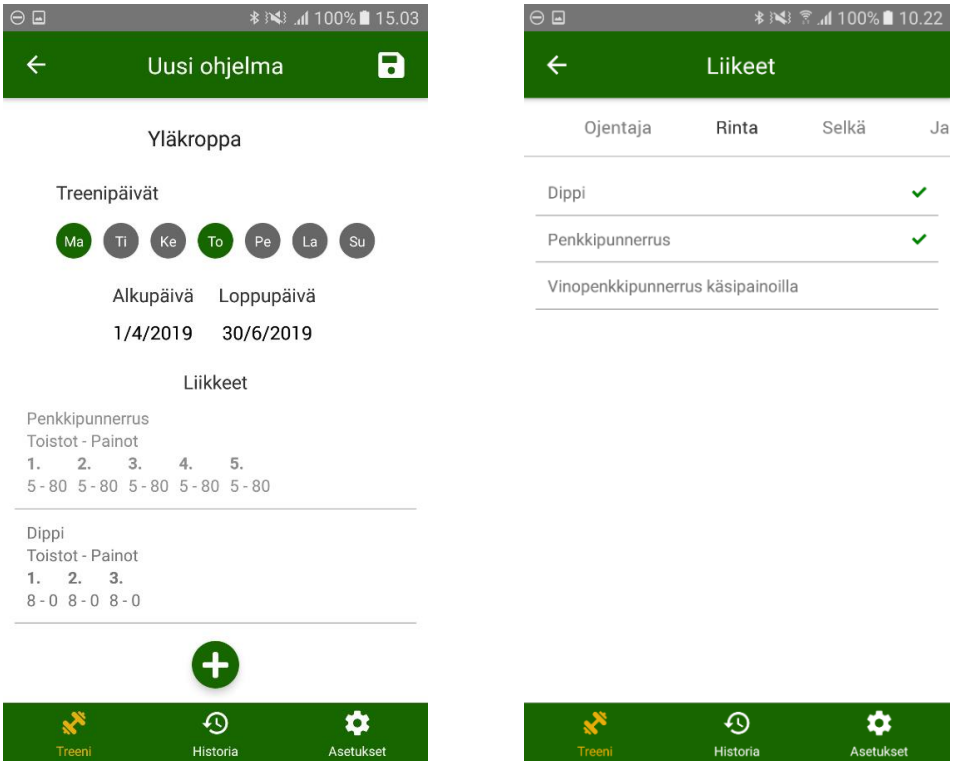


Kuvio 20. Oma sovellus, harjoitusohjelmat-sivu

4.2.6 Harjoitusohjelman suunnittelu

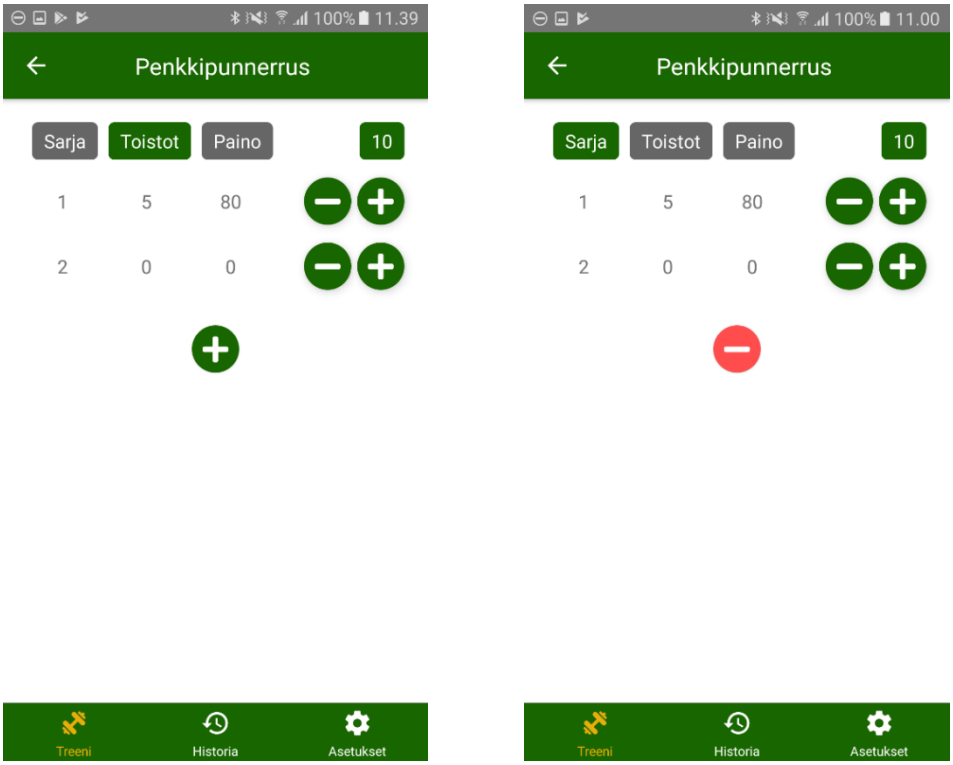
Harjoitusohjelman suunnittelu -sivulla laitetaan ohjelmalle nimi, valitaan harjoituspäivät, alku- ja loppupäivä sekä ohjelmassa tehtävät liikkeet. Aikavalitsimet toteutettiin Native Base -kirjaston Date Picker -komponenteilla. Liikkeet lisätään plus-näppäimestä, jolloin näkymä siirtyy liikkeet sivulle, jossa on eri lihasryhmille tab-sivut. Liikettä painaessa ilmestyy oikeaan laitaan valittu merkki. Tällä hetkellä liikkeet ovat kovakoodattuna sovellukseen. Liikkeen valinta on toteutettu Redux-kirjastolla, siten että liikkeet ovat objekteina Reduxin tilassa ja objekteissa on selected-valinta. Kun liikettä painaa niin selected-arvo muutetaan arvoon true. Kun sivulta mennään takaisin uusi ohjelma sivulle, käydään läpi Reduxista liikkeet ja kaikki liikkeet missä selected on true näytetään listassa. Tällä hetkellä listasta ei suoraan saa poistettua ohjelmassa

olevaa liikettä, vaan ne on mahdollista poistaa, kun käy laittamassa liike sivulla valinnan pois. (Ks. kuvio 21.)



Kuvio 21. Oma sovellus, harjoitusohjelman suunnittelu -sivu ja liikkeet -sivu

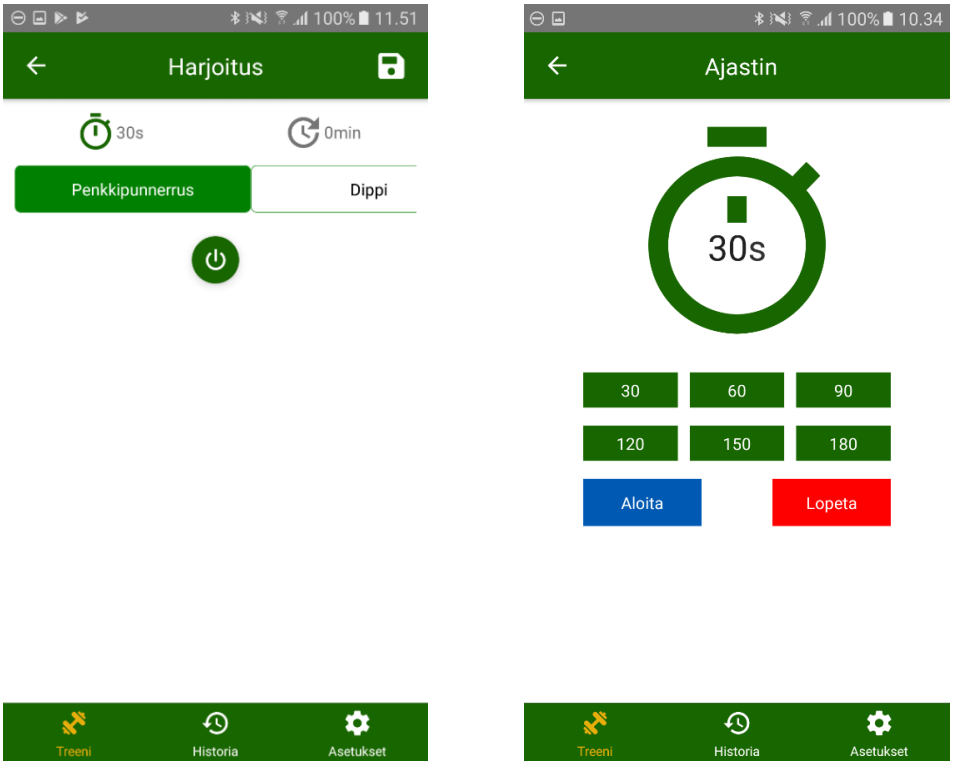
Harjoitusohjelman suunnittelu -sivulla, liikettä painamalla aukeaa liikkeen oma sivu, josta saa lisättyä tehtävien sarjojen määrän, sarjoissa tehtävät toistot sekä käytettävät painot. Toistot ja paino -napin ollessa aktiivinen on kyseisen arvon muuttaminen mahdollista oikeassa laidassa olevista plus- ja miinusnapeista ja sarjojen lisäys onnistuu alhaalla olevasta plus-napista. Sarjanapin ollessa aktiivinen ilmestyy miinusnappi, josta saa poistettua viimeisenä olevan sarjan. Oikeassa laidassa olevasta numeronapista vaihdetaan lisäysarvon määrää. Vaihdettavat arvot ovat 0.25, 0.5, 1 ja 10. (Ks. kuvio 22.)



Kuvio 22. Oma sovellus, sarjojen lisäys -sivu

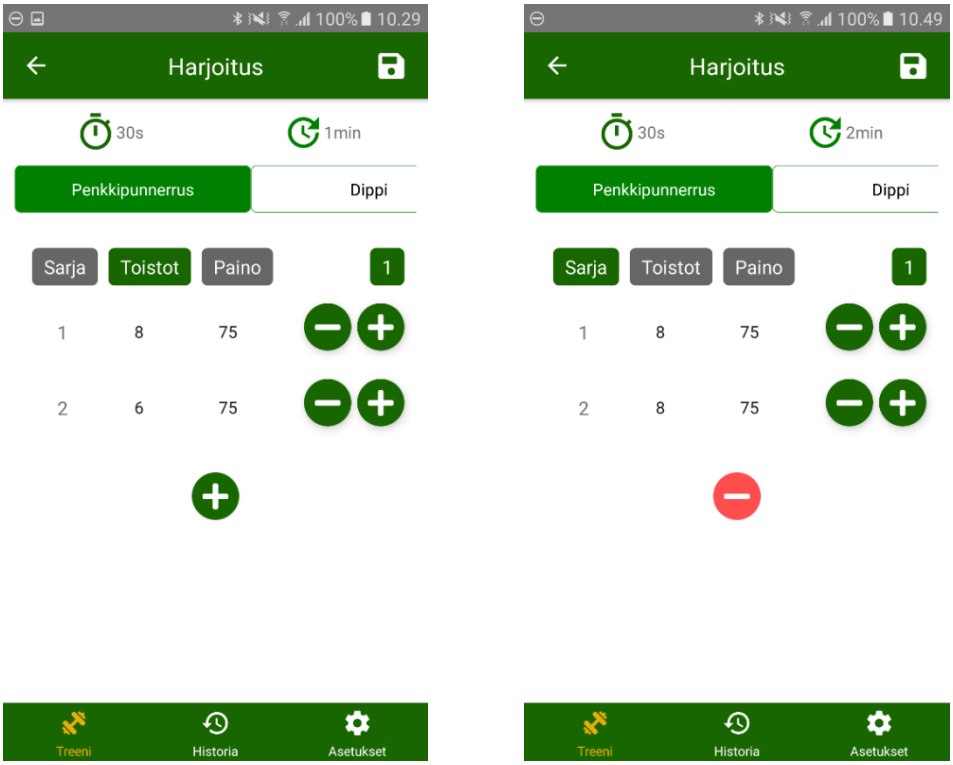
4.2.7 Tuloksien kirjaus

Harjoitussivu toteutettiin hieman suunnitelmista poikkeavasti. Sivun ylälaidassa on ajastin sekä sekuntikello. Sekuntikello lähtee käyntiin, kun ohjelman aloitus -nappia painetaan. Ajastimesta painamalla aukeaa ajastinsivu, josta on mahdollista muuttaa ajastimen aikaa sekä käyttää ajastinta manuaalisesti. Suunnitelmissa oleva liukusäädin ajan valitsemiseen päätettiin toteuttaa eri tavalla. Toiminto toteutettiin käyttämällä nappeja, joiden arvot ovat yleiset lepoajat. Lepoajat ovat 30-, 60-, 90-, 120-, 150- sekä 180 sekuntia. Jos mikään näistä ajoista ei vastaa haluamaa aikaa, niin ajan on mahdollista muuttaa myös, kun painaa sivulla olevassa kellossa olevasta arvosta, jolloin tekstikenttä aktivoituu. Ajan muuttaminen on toteutettu käyttämällä Redux-kirjastoa, jolloin valinta oli helppo saada muuttumaan harjoitus sivulla olevaan ajastin kuvakkeeseen. (Ks. kuvio 23.)



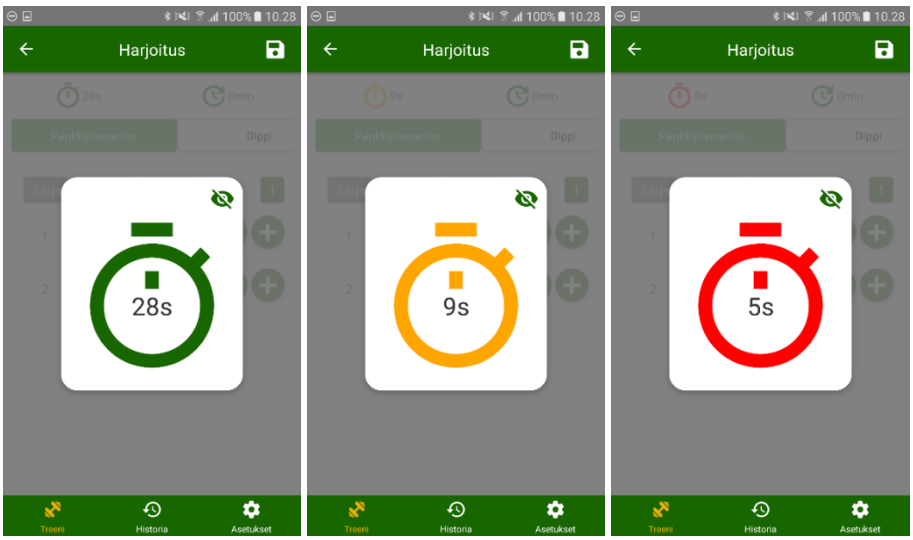
Kuvio 23. Oma sovellus, harjoitussivu ja ajastin asetukset

Harjoitussivulla olevat liikkeet oli suunniteltu tab-navigointina, mutta päätettiin toteuttaa nappeina. Tietokannasta ladataan harjoituksessa käytettävät liikkeet ja näytölle renderöidään Scrollview-elementin sisälle napit jokaiselle liikkeelle. Scrollview asetettiin horisontaaliin asentoon, jolloin napit saatiin vierittämään hyvin sivulta sivulle. Kun ohjelman aloitus -nappia painaa, niin jokainen ohjelmassa tehtävä liike ladataan Reduxin tilaan ja asetetaan yksi tyhjä sarja tai ensimmäinen sarja edellisestä treenistä valmiiksi ja käynnistysnappi muuttuu plusnapiksi. Plusnapista painamalla lisätään aina uusi sarja ja laitetaan ajastin näkymään isona näytöllä. Supersarjat on mahdollista toteuttaa, koska liikkeet on mahdollista suorittaa siinä järjestyksessä kuin haluaa, mutta supersarjoina tehdyt liikkeet eivät näy historiassa supersarjoina. Tällä hetkellä harjoitusta ei pysty tallentamaan treenin välillä, vaan tallennettuaan harjoituksen ohjelma keskeytyy. Ohjelma tallennetaan oikeassa yläkulmassa olevasta tallennusnapista. (Ks. kuvio 24.)



Kuvio 24. Oma sovellus, harjoitussivu

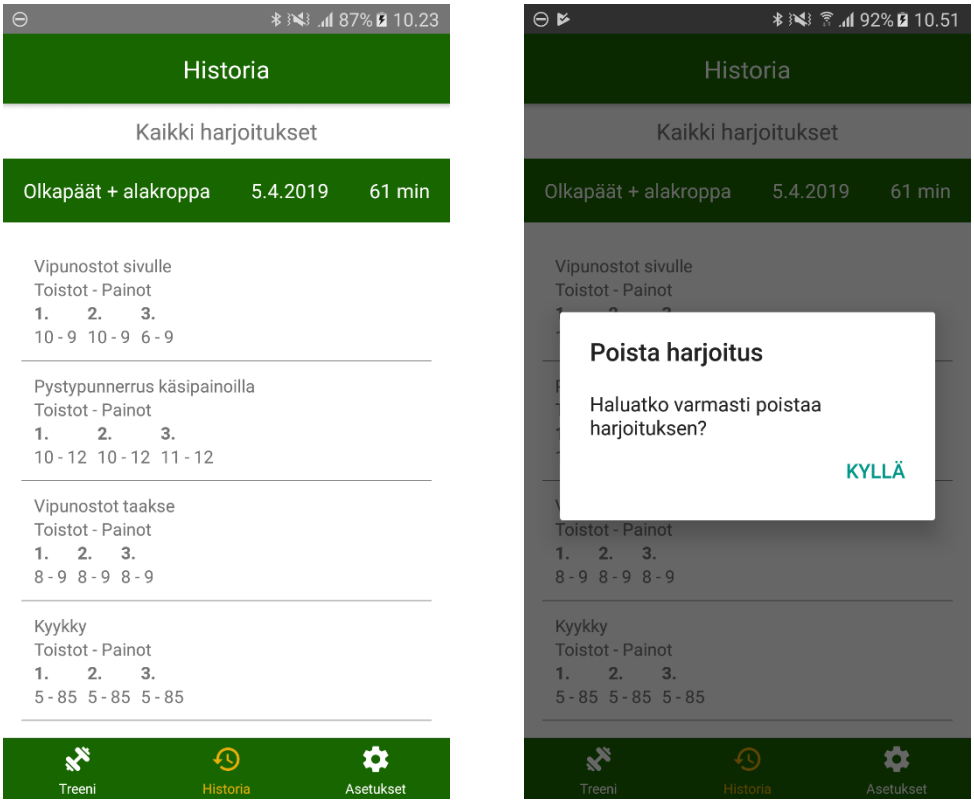
Suunnitelmista poiketen, isoksi aukeavan ajastimen on mahdollista piilottaa näkyvistä halutessaan. Kun aika umpeutuu, ajastin katoaa automaattisesti. Ajastin toteutettiin kuten suunnitelmissa määritettiin, eli alussa ajastin on sovelluksen päävärin värinen, 10 sekunnin kohdalla ajastimen väri muuttuu oranssiksi ja 5 sekunnin kohdalla punaiseksi. (Ks. kuvio 25.)



Kuvio 25. Oma sovellus, ajastin

4.2.8 Historia

Historiasivulle ladataan kaikki harjoitukset tietokannasta. Harjoitukset näytetään listana uusimmasta vanhimpaan. Harjoituksen otsikosta näkee ohjelman nimen millä kyseinen treeni on suoritettu, päivämäärän sekä harjoituksen keston minuutteina. Harjoituksen liikkeet näytetään listaelementissä. Datan näyttämistyyli poikkeaa hieman suunnitellusta, jotta kaikki sarjat sai hyvin ja selkeämmin mahtumaan näytölle. Harjoituksen otsikkoelementistä pitkään painamalla saa kyseisen harjoituksen poistettua. (Ks. kuvio 26.)



Kuvio 26. Oma sovellus, historiasivu

5 Yhteenveto

5.1 Tulokset

Opinnäytetyön tuloksena saatiin toteutettua harjoituspäiväkirjamobiilisovelluksen suunnitelma, sekä valmistettua suunnitelman pohjalta toimiva sovellus. Sovellus ei ole täydellinen, eikä sisällä kaikkia suunnitelman toiminnallisuuksia, mutta sovelluksesta saatiin hyvä pohja, jota on helppo lähteä jatkokehittämään tuotantoversioksi. Sovelluksessa on mahdollista luoda harjoitusohjelma, sekä kirjata tulokset ohjelman liikkeiden mukaisesti. Ohjelma on mahdollista myös poistaa. Kaikki harjoitustulokset näytetään listana historiasivulla ja jokainen harjoitus on mahdollista poistaa erikseen. Jos puhelimen kieli on suomi tai englanti, tulee kyseinen kieli automaattisesti sovellukseen, muussa tapauksessa sovelluksen kieli on englanti.

Työn aikana saatiin parannettua osaamista React Nativen parissa ja saatiin lisää ymmärrystä suunnittelun tärkeyden osalta. Tuloksena saatiin myös suppea vertailu olemassa olevista harjoituspäiväkirja-sovelluksista, sekä cross-platform-tekniikoista.

5.2 Pohdinta

Opinnäytetyön tavoitteena oli valmistaa toimiva harjoituspäiväkirja-sovellus, jota olisi helppo ja mukava käyttää kuntosalissa ollessa. Osittain työn tavoite saavutettiin, koska tuloksena saatiin valmistettua toimiva sovellus, jota oli helppo ja hyvä käyttää. Kaikkia suunniteltuja ominaisuuksia ei vielä ole sovellukseen toteutettu ja osaksi suunnitelmat muuttuivat kehitysvaiheessa parempien ideoiden syntyessä, mutta sovellusta on hyvä lähteä jatkokehittämään julkaisukelpoiseksi sovellukseksi.

Opinnäytetyöprosessin aikana huomattiin, miten tärkeää hyvä suunnittelu on ja miten paljon asioita pitää ottaa huomioon tällaista sovellusta tehdessä. Jatkokehityksessä on parannettava harjoitusohjelman lisäystä vastaamaan mahdollisimman monenlaisen harjoitusohjelman tarvetta. Nyt suunnitteluvaiheessa ei tullut otettua huomioon riittävän laajasti ohjelman suunnittelua. Esimerkiksi, jos haluaa tehdä joka toinen harjoituskerta eri liikkeitä, joutuu tekemään kaksi erillistä ohjelmaa, jotka ovat eri kokonaisuuksia. Sovellus toimisi paremmin, jos ohjelmassa olisi mahdollista lisätä

liike A ja liike B, jotka tehtäisiin, joka toinen treenikerta. Harjoitusohjelmaa tulee pystyä muokata, jos esimerkiksi halutaan vaihtaa liikkeitä ohjelmassa. Liikkeiden lisäys olisi myös oltava ohjelmassa, koska treeniliikkeitä on todella paljon ja todennäköisesti aina joku liike puuttuu sovelluksesta. Harjoitteluosiossa tulisi saada tallennettua harjoitus myös harjoituksen aikana, koska tällä hetkellä sovelluksen kaatuessa tulokset menetetään. Historiaosiossa jäi vielä paljon tehtävää, jotta sovellus olisi julkaisukelpoinen. Harjoituksia tulisi pystyä suodattamaan päivien ja ohjelmien perusteella. Tällä hetkellä sovelluksessa näytetään kaikki harjoitukset. Jatkokehityksessä tulee mahdollistaa tuloksien ulosvienti sovelluksesta. Monikielisyys saatiin toteutettua automaattisena puhelimen kielen mukaan, mutta mahdollisuutta itse valita kieltä ei saatu vielä toteutettua. Myöskään teeman valinta ei ole vielä mahdollista. Vielä on paljon tehtävää ennen, kuin sovellus on valmis julkaistavaksi.

Sovellusta tehdessä saatiin parannettua osaamista React Nativen parissa, vaikka siitä oli hieman kokemusta ennestään. Realm-tietokannan käyttö tuli täysin uutena asiana, joten myös tietokannan osalta tuli lisää osaamista jatkoa ajatellen. Alun perin tietokannaksi ajateltiin SQLite-tietokantaa ja sitä tulikin kokeiltua sovelluksessa, mutta jostain syystä koko sovellus hajosi, joten lopulta päädyttiin suunnittelemaan tietokanta uudestaan Realm-tietokannaksi. Kokeilun myötä myös relaatiotietokannan käyttö palautui mieleen, joka on tärkeää tulevaisuuden kannalta. Opinnäytetyön aikana saatua kokemusta pystyy hyödyntämään mahdollisissa omissa projekteissa sekä tulevissa työpaikoissa.

Teknologian valintaan oltiin tyytyväisiä. Jos ei ole kokemusta React Nativesta, se voi tuntua vaikealta, mutta oppimisen myötä React Nativella saa hyvää jälkeä aikaiseksi. Kokonaisuuksien komponentteihin jakamisessa ja koodin selkeämmin kirjottamisessa on vielä parannettavaa. React Nativeen löytyi sovelluksessa tarvittavat lisäosat, joten natiivin koodin kirjoittamiselta vältyttiin.

Omaa sovellusta tehdessä osaa arvostaa muiden kehittäjien sovelluksia, koska tietää mitä kehittäminen vaatii ja miten paljon resursseja on käytettävä hyvään ja toimivaan sovellukseen. Sovellus täytyy miettiä tarkasti, jotta on mahdollista saavuttaa tuhansia ja jopa miljoonia latauskertoja ilman, että sovellus menee poistoon ensimmäisen käyttökerran jälkeen. Vaikka opinnäytetyön aikana valmistuneeseen sovelluksen

alkuun on oltava tyytyväinen, niin on tiedostettava, että matkaa parhaimpien sovelluksien joukkoon on vielä paljon jäljellä. Tavoite on jatkokehityksellä mahdollistaa sovelluksen leviäminen moneen maahan lukuisien treenaajien käyttöön.

Lähteet

Abramov, D. 2015. Getting Started with Redux. Redux-dokumentaatiosivulla. Viitattu 9.4.2019. https://redux.js.org/introduction/getting-started.

Framework7. 2019. Framework7.io-verkkosivulla. Viitattu 3.2.2019. https://framework7.io/.

Getting Started. 2018. React Native -dokumentaatiosivulla. Viitattu 3.12.2018. https://facebook.github.io/react-native/docs/getting-started.

Holland, B. 2018. NativeScript and Xamarin. NativeScript.org-verkkosivulla. Viitattu 17.12.2018. https://www.nativescript.org/blog/nativescript-and-xamarin.

How React Native works. 2018. React Native GitHub -sivulla. Viitattu 29.11.2018. https://github.com/facebook/react-native.

Ionic Native. N.d. Ionic-dokumentaatiosivulla. Viitattu 29.3.2019. https://ionicframework.com/docs/v3/native/.

Kharlampidi, V. 2019. Introduction. Framework7-dokumentaatiosivulla. Viitattu 29.3.2019. https://framework7.io/docs/introduction.html.

Learn the Basics. 2018. React Native -dokumentaatiosivulla. Viitattu 30.11.2018. https://facebook.github.io/react-native/docs/tutorial.

Lucas, E. & Wiegert, C. 2019. What is Ionic Framework?. Ionic-dokumentaatiosivulla. Viitattu 29.3.2019. https://ionicframework.com/docs/intro.

Matthews, B. 2018. An introduction to Framework 7. Viitattu 3.2.2019. Framework 7 GitHub -sivu. https://github.com/framework7io/framework7/wiki.

Petzold, C. 2016. Creating Mobile Apps with Xamarin.Forms. Viitattu 18.12.2018. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/creating-mobile-apps-xamarin-forms/#download-chapters-and-summaries.

The dev-friendly app platform for building cross-platform apps with one codebase, for any device, with the web. 2019. ionicframework-verkkosivulla. Viitattu 29.3.2019. https://ionicframework.com/what-is-ionic.

Who’s using React Native? 2018. facebook.github.io-verkkosivulla. Viitattu 18.12.2018. https://facebook.github.io/react-native/showcase.

Why not Expo? N.d. Expo-dokumentaatiosivulla. Viitattu 2.12.2018. https://docs.expo.io/versions/v31.0.0/introduction/why-not-expo.

Xamarin.Forms. N.d. Xamarin.Forms-dokumentaatiosivulla. Viitattu 29.3.2019. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/.

Xamarin.Forms Quickstart Deep Dive. 2018. Xamarin.Forms-dokumentaatiosivulla. Viitattu 29.3.2019. https://docs.microsoft.com/fi-fi/xamarin/get-started/quickstarts/deepdive?pivots=windows.