Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

NIKO STENFORS

Ionic Framework mobiilikäytössä

TIETOJENKÄSITTELYN TUTKINTO-OHJELMA
2023

Tekijä(t)
Stenfors, Niko

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
04 2023

Sivumäärä
30

Julkaisun kieli
Suomi

Julkaisun nimi
Ionic Framework mobiilikäytössä

Tutkinto-ohjelma
Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Työssä kerrotaan Ionic Framework -teknologiasta ja tehtiin tällä alustalla sovellus salilla käymiseen ja tulosten seurantaan. Sovelluksessa käytettiin myös JavaScipt-pohjaista Vue.js-ohjelmointikehystä. Työn tarkoitus on selittää, mikä on Ionic ja saada sovellus omaan käyttöön sellaisessa muodossa, että sitä voi käyttää

Opinnäytetyössä käsitellään ensin Ionic-alusta yleisesti, kuten myös sen historiaa. Läpi käydään myös, millaisia työkaluja sen kanssa voi käyttää ja miten sen kanssa pääsee alkuun, jos haluaa itse tehdä sovelluksen. Muita läpikäytäviä asioita ovat käyttöliittymä sekä Ionic-alustan hyödyt ja haitat.

Sovelluksesta tuli käyttökelpoinen.

Avainsanat:
Mobiilisovellus, Ionic Framework, Vue.js

Author(s)
Stenfors, Niko

Type of Publication
Bachelor's thesis

Date
04 2023

Number of pages
30

Language of publication:
Finnish

Title of publication
Ionic Framework in mobile use

Degree programme
Business Information Systems

Abstract

In the work, the technology of Ionic Framework was introduced and an application for gym workouts and tracking was developed on this platform. The application also used Vue.js language in JavaScript. The purpose of the work was to explain what Ionic is and to create an application for personal use in a usable form.

The thesis first covers Ionic platform in general, including its history. It also goes through the tools that can be used with it and how to get started with creating an application on it. Other topics covered include the user interface, as well as its advantages and disadvantages.

The work successfully explained what Ionic Framework is and the application was created in a way that it could be used

Keywords:
Mobile app, Ionic Framework, Vue.js

SISÄLLYS

1 JOHDANTO

Aiheenani tässä opinnäytetyössä toimii Ionic Framework. Työssä käydään lävitse yleisesti Ionic alustana ja käytettävyyden kannalta. Ensin käydään yleisesti läpi kieltä, sitten siirrytään siitä sen historiaan. Käydään lävitse hyviä ja huonoja puolia kielestä. Työssä myös koodataan Ionic-alustalla pieni sovellus, jolla voi seurata kuntosalilla etenemistä ja tuloksiaan. Työn tavoitteena on myös näyttää, ettei oman sovelluksen tekemisen tarvitse olla niin vaikeaa miltä se kuulostaa. Ionic on hyvä alusta sovelluksen tekemiseen, jos näkee oman tarpeen mihin voisi kehittää sovelluksen. Pääsin käyttämään tätä alustaa monta kuukautta töissäni, ja pääasiassa minun työkuvaani kuului mobiilisovellusten kehittäminen ja tekeminen. Itse ennen kuin aloin työskennellä Ionicin kanssa luulin, että mobiilisovellusten tekeminen olisi todella hankalaa ja työlästä, koska tarvitsee tehdä molemmille alustoille omat koodit ja muutenkin ajatus siirtymisestä web-kehittäjästä mobiilisovellusten kehittäjäksi kuulosti isolta taakalta. Ionic teki tästä siirtymästä paljon mukavampaa sen pohjautuessa web-kehittäjien kieliin ja siihen, ettei tarvitse kirjoittaa enempää kuin yksi koodi, koska se toimii molemmilla alustoilla.

Ionic on käytössä jo monella suurella yrityksellä, enkä tästä ole yhtään yllättynyt. Yrityksien ei välttämättä tarvitse palkata monta tekijää sovellusta varten, koska sitä ei tarvitse tehdä kahdelle alustalle erikseen. Yrityksiä, joilla on käytössä Ionic, ovat esimerkiksi McDonald’s Turkki ja McLaren Automotive (Wiredelta 2020). Sovelluksista huomaa silti sen, että ne ovat hyvin samanlaisia konseptinsa pohjalta, koska Ionicissa on todella yleinen kaava, jota käytetään monissa Ionic-sovelluksissa.

2 IONIC FRAMEWORK

2.1 Yleisesti

Ionic on avoimen lähdekoodin SDK eli ohjelmiston kehityspaketti. Se on rakennettu siten, että sovellusten tekeminen olisi mahdollista hybrid-tyylillä. Sen loivat Drifty-nimisessä yrityksessä toimineet Max Lynch, Ben Sperry ja Adam Bradley vuonna 2013. Ionic on rakennettu AngularJs ja Apache Cordova -pohjalle. AngularJs on JavaScriptiin pohjautuva käyttöliittymäkehys ja Apache Cordova on mobiilisovellusten kehitystekniikka. Näistä kahdesta enemmän myöhemmin tässä työssä. Nykyään AngularJs ei ole ainoa valittava vaihtoehto. Vaihtoehtoja nykyään ovat myös React.js ja Vue.js.

Ionic on myös moniulotteinen, koska sillä ei ole pakko tehdä pelkästään mobiilille progressiivisia verkkosovelluksia. Ionicilla voi tehdä myös sovelluksia suoraan tietokoneelle tai verkkoselaimille. JavaScriptin lisäksi käytetään normaaleja web-kehitys kieliä, joita ovat HTML5- ja CSS-kielet. Näiden kielten ollessa tuttuja sovelluksien tekeminen on todella helppoa. Perusidea on siis, että näillä web-kehityksen työkaluilla voidaan rakentaa mobiilisovellus, joka jaetaan sitten sovelluskauppoihin asennettavaksi laitteisiin käyttämällä Apache Cordovaa.

2.2 Historia

Ensimmäinen alfa-versio julkaistiin marraskuussa vuonna 2013. Ensimmäinen beta julkaistiinkin jo maaliskuussa 2014. Beta-version jälkeen julkaistiin versio 1.0 vuonna 2015 toukokuussa. Ionic käyttää versionimenään yksinkertaisesti version päänumeroa, joten versiota 1.0 kutsutaan nimellä Ionic 1 ja versiota 2.0 kutsutaan nimellä Ionic 2 ja niin edespäin. Ionic 4 oli isoin harppaus eteenpäin ilman vain uutta versiota Angular-työkalusta. Tällä hetkellä ollaan versiossa Ionic 7, joka julkaistiin maaliskuussa vuonna 2023.

Ionic 1 oli ensimmäinen versio ja siinä käytettiin sovellusten luomiseksi Angular 1:tä. Tämä versio ei sisältänyt verkkokomponentteja. Ionic 2 oli laajennus versioon yksi ja siinä uutta oli Angular 2. Sen toiminnallisuudet eroavat täysin sen edellisestä versiosta. Tässä versiossa ei ollut vieläkään verkkokomponentteja. Ionic 3 oli vielä todella samanlainen kuin edeltäjänsä. Tässä päivityksessä oli myös vain päivitetty versio Angular 4. Tämän version jälkeen perustajat tajusivat, että Angular on rajoitettu, mitä tulee esimerkiksi siihen, ettei sillä voi luoda mobiilisovellusta niin kuin esimerkiksi Vue.js:n ja React.js:n avulla. Ionic 4 julkaistiin vuonna 2019 ja siinä tuli vihdoinkin käyttöön web-komponentit. Sillä pystyi luomaan mobiilisovelluksia ja verkkosovelluksia käyttämällä verkkoteknologioita, kuten HTML, CSS ja JavaScript. Se tuki lähes kaikkia selaimia, joissa voitiin ajaa HTML-komponentteja. Tämän jälkeen versiot ovat olleet enimmäkseen pieniä päivityksiä edelliseen versioon. Uusi versio julkaistaan kuuden kuukauden välein. (Sharma 2021.)

2.3 Käyttöliittymä

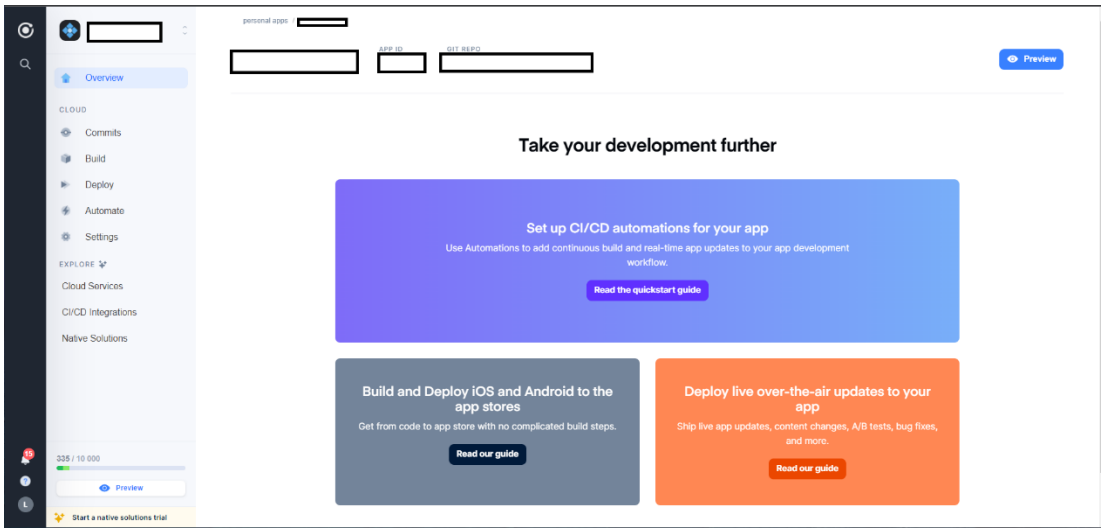
2.3.1 Apps

Ionicin käyttöliittymä on hyvin yksinkertainen ja toimii verkkoselaimella. Etusivulta kirjaudutaan sisään hallintaan. Hallinnassa ensin avautuu sovellukset-näkymä, josta näkee viimeiseksi päivitetyt sovellukset. Peitetyssä kohdassa näkyy sovelluksen nimi, id ja tieto siitä, koska viimeksi sitä on päivitetty. Tästä näkymästä päästään tekemään myös uusi sovellus. Sovelluksen nimeä painamalla pääsee oman sovelluksen näkymään, josta pääsee tekemään säätöjä, päivityksiä ja tiedon muutoksia. Kuvassa 1 on esitetty käyttöliittymä, kun valitsee sovelluksen, jota haluaa työstää.

Sovelluksen omassa käyttöliittymässä voikin sitten tehdä jo monia asioita. Perustietoja, jotka olen peittänyt ovat sovelluksen nimi, id ja git-arkiston sijainti (kuva 1). Cloud-valikon alta löytyy ensin kohta Commits. Täältä löytää viimeisimmät vahvistetut muutokset koodiin ja linkit niihin tehtyjen muutoksien tarkastelemiseen. Build-valikossa tehdään itse sovelluksen rakentaminen, nähdään kaikki edelliset rakennelmat ja niiden tilanne. Painamalla tästä näkymästä painiketta pääsee tekemään uuden rakennelman. Ensin valitaan mitä versiota haluaa käyttää arkistosta ja sen jälkeen kohde.

Kohteita on kolme: verkko, Ios ja Android. Verkko on nopein näistä vaihtoehdoista ja rakentaa verkkoselainversion, joka on todella kätevä nopeissa päivityksissä. Itse käytän tätä töissä todella useasti, kun haluan nopeasti päivityksen asiakkaalle, koska tämä tarjoaa live-päivitysvaihtoehdon. Tämä on minun lempiominaisuuteni, koska kun rakentaa Ios- tai Android-versiota täytyy tämä sen jälkeen siirtää sovelluskauppaan tehdä siellä tarvittavat toimenpiteet ja tietolisäykset, minkä jälkeen versio menee tarkastukseen. Tämä on aikaa vievä prosessi, eivätkä asiakkaat saa tämän kautta päivityksiä nopeasti. Tässä kohtaa live-päivitysominaisuus on todella hieno, koska ei tarvitse kuin valita live-päivitys ja kanava, jolle sen haluaa. Tämän jälkeen Ionic rakentaa siitä uuden version ja laittaa sen päivityksen puhelimiin, joihin sovellus on ladattu. Töissäni sovellukset antoivat sovelluksen avaamisen yhteydessä ilmoituksen, kun uusi päivitys on saatavilla. Live-päivitys toimii molemmilla alustoilla Iosilla ja Androidilla, joten siinäkin vielä yksi positiivinen puoli, ettei näitä eri alustojen päivityksiä tarvitse tehdä erikseen.

Ios ja Android ovat hyvin samantyylisiä. Nämä täytyy tehdä silti molemmat erikseen, koska silloin kun julkaisee sovellusta vasta sovelluskauppoihin, vaaditaan molempiin alustoihin omat versiot. App Store ei ota vastaan rakennelmia yhtä helposti kuin Play Kauppa. Android-versiot voi ladata ja lisätä alustoille suoraan tiedostona. Ennen tähän oli kaksi vaihtoehtoa tiedostomuodossa Apk ja Aab. Nykyään kelpo versio kauppoihin on Aab, jonka onneksi Ionic tekee myös. App Storeen ei voi lisätä suoraan tiedostoa, jonka julkaista vaan se tarvitsee tulla jostain lähteestä. Ioniciin tarvitsee siis konfiguroida App Store -lähde, jotta sovellus saadaan julkaistua sinne. Onneksi Ionic onnistuneen rakennelman jälkeen lähettää tämän noin viiden minuutin sisällä App Storen konsolin sivulle, jotta sen pääsee julkaisemaan. Molemmat versiot tarvitsevat myös sertfikaatit, jotka voi lisätä build-kohdan alavalikosta.



Kuva 1. Sovelluksen sisällä oleva käyttöliittymä

2.3.2 Usage

Usage-valikon alta löytyy meneillään olevan kuukauden tilanne siitä, kuinka monta live-päivitystä on laittanut jo ja kuinka monta voi vielä pistää. Tätä on hyvä pitää silmällä, ettei se pääse yllättämään. Mikäli päivitykset pääsevät loppumaan ja kuukautta on vielä jäljellä pitää joko ottaa yhteys tukeen tai ostaa kalliimpi lisenssi.

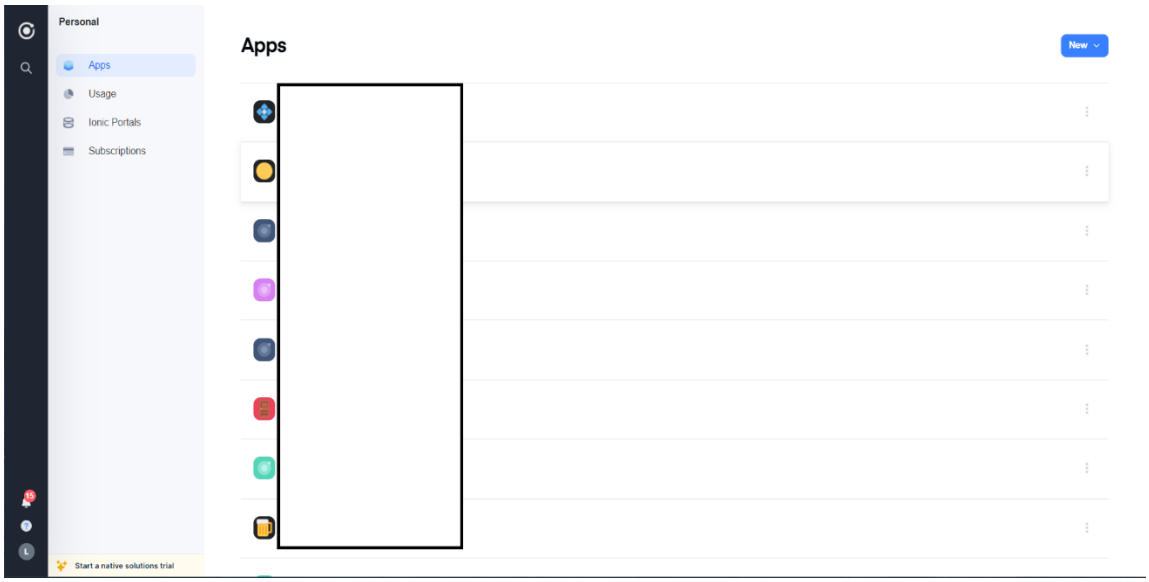
2.3.3 Ionic Portals

Ionic Portals on ominaisuus, jonka avulla monta tiimiä voi turvallisesti luoda sovellusta, koska tässä projekti on hajotettu moneen portaaliin, jotka rakentavat yhdessä koko sovelluksen. Tämä helpottaa työtä, koska voidaan rakentaa, testata ja toimittaa näitä eri portaaleja vaikuttamatta muiden töihin.

2.3.4 Subsciptions hinta- ja lasku näkymä

Subscriptions-välilehden alla on normaalit tilaustiedot. Lisenssejä on saatavilla eri hintoihin, mutta lähtökohta on community, joka maksaa 49 $/kk. Tämän jälkeen on basic ja tämän hinta on 499 $/kk. Standard maksaa 2499 $/kk ja enterprise-lisenssin hinta arvioidaan Ionicilla. Lisenssien eroina on, että joissakin on enemmän ominaisuuksia, päivitysmääriä ja voi lisätä enemmän jäseniä tiimiin.

Kuvassa 2 on esitetty käyttöliittymä, kun avaa verkkosivun ja ylempänä on kohdat selitetty erikseen mihin ne johtavat ja mitä siellä tehdään.



Kuva 2. Käyttöliittymän etusivu

2.4 Hyödyt ja haitat

Olen työn aikana kertonut paljon positiivisia asioita Ionicista, mutta tässä luvussa kerrotaan yhteenveto hyvistä ja huonoista asioista.

Hyötyjä ovat seuraavat:

- helppo oppia ja ymmärtää
  - Mikäli kehittäjä on valmiiksi jo perehtynyt verkkoteknologioihin, kuten esimerkiksi HTML ja JavaScript, ja käyttänyt niitä on Ionicin käyttö suhteellisen helppo oppia. (Sharma 2021.)
- hyvä dokumentointi
  - Ionicin dokumentit ovat hyviä ja ne käyvät läpi melkein kaikki kohdat, joita kehittäjän tarvitsee ymmärtää Ionicin käytössä. (Sharma 2021.)
- mahdollisuus tehdä monille alustoille
  - Ionic tarjoaa toiminnot sovellusten käyttöönottoon useilla laitteilla. Myös idea siitä, että koodi kirjoitetaan vain kerran ja että se suoritetaan millä tahansa laitteella, on vallan mainio. (Sharma 2021.)
- parannettu käyttöliittymä
  - Käyttöliittymää on parannettu ja se näyttää nykyään jo paljon houkuttelevammalta. Se tarjoaa monia räätälöityjä teemoja ja komponentteja. (Sharma 2021.)
- valinnanvapaus
  - Nykyään, kun Ionic sisältää AngularJs lisäksi myös Vue.js ja React.js, on kehittäjällä mahdollisuus valita tekniikka omien mieltymystensä mukaan.

Ionicin haittoja ovat seuraavat:

- suorituskyky
  - Natiivien mobiilisovellusten suorituskyky on parempi verrattuna Ionic-sovelluksiin. Useimmat alustan käyttäjät eivät ole silti kohdanneet tätä ongelmaa, mikä on tietysti positiivista. Tämä ei kuitenkaan tarkoita sitä, etteikö ongelmaa olisi olemassa. (Sharma 2021.)
- turvallisuus
  - Verratessa taas natiiveihin mobiilisovelluksiin Ionic-sovellukset ovat vähemmän turvallisia. Esimerkkinä Ioniciin ei voi luoda maksuyhdyskäytäväsovellusta. Ionic tarjoaa tähän silti kalleimmissa paketeissaan itse tekemäänsä rajapintaa. (Sharma 2021.)
- rajoitetut ominaisuudet
  - Lähes kaikki toiminnot on saatavilla Ionic-sovelluksissa, mutta niitä puuttuu silti. Käyttääkseen näitä toimintoja, joita ei ole mukana, täytyy kehittäjien luoda laajennuksia, joilla saadaan haluttu toiminto. (Sharma 2021.)
- videopelit
  - Ionic ei ole suositeltu videopelien tekemiseen. Yksi syy tähän on juuri Ionic-sovellusten heikko suorituskyky. Esimerkiksi grafiikat eivät näyttäisi ollenkaan hyvältä Ionic-sovelluksissa ja tästä syystä niitä ei edes suositella tekemään. (Sharma 2021.)

3 PROJEKTIN ALOITUS

3.1 Projekti käyntiin

Projektin aloittaminen on todella helppoa ja nopeaa. Laitteelle tarvitsee asentaa ensin Node.js. Asennuksen jälkeen voidaan asentaa sitten itse Ionic, joka onnistuu komentokehotteen kautta. Itse käytän tähän Git Bash -sovellusta, mutta mikä tahansa muukin toimii tähän.

Kun nämä on asennettu onnistuneesti, voidaan aloittaa kehittäminen. Sovelluksen aloittamisessa toimii yksinkertainen komento ”ionic start” (kuva 3). Tämän jälkeen annetaan projektille nimi. Aloittaminen on käytännössä näin helppoa.

Kuva 3. Näkymä komentokehotteessa ”ionic start” komennon jälkeen

Aloitettaessa sovelluksen tekemistä voidaan myös valita, mitä projektipohjaa haluaa käyttää. Vaihtoehtoja ovat tabs, sidemenu tai blank. Tabs-pohjassa valintapalkit ovat näytön alareunassa, ja sidemenu-pohjassa ne ovat sivussa. Blank-pohja on yksi tyhjä sivu ja antaa mahdollisuuden lähteä tyhjältä pöydältä liikkeelle.

Todella kätevä ominaisuus Ionicissa on komento ”ionic serve”. Käytän tätä itse oikeastaan aina, kun kehitän jotain sovelluksiin tai korjaan niitä. Kyseinen komento avaa sovelluksen verkkoselaimelle paikallisesti. Tämä on todella nopea ja helppo tapa kehittää ja korjata sovellusta, koska se päivittyy automaattisesti, kun koodin tallentaa. Virheiden korjaamiseen tämä vasta onkin kätevä, kun sen yhdistää verkkokonsoliin. Tästä näkee kaikki virheet ja myös tietoliikenteen ja niissä olevat mahdolliset virheet. Tämä on todella yksi mielestäni parhaista ominaisuuksista ja on mahdollistettu juuri siksi, koska se käyttää verkkoteknologioita kehittämiseen.

3.2 Cordova

Apache Cordova on avoimen lähdekoodin tekniikka, joka on vastuussa siitä, miten yhdestä koodista saadaan toimiva versio eri alustoille. Cordova ottaa verkkosovelluksen ja hahmottelee siitä verkkonäkymän. Tämä verkkonäkymä on oma

sovelluskomponenttinsa, jota käytetään verkkosisällön näyttämisessä natiivissa sovelluksessa. Tämä on helpompi käytännössä ymmärtää niin, että tämä komponentti toimii niin, että se näyttää normaalin verkkoselaimen ilman käyttöliittymäelementtejä, kuten URL-kenttää tai tilapalkkia. Tämä säiliössä toimiva verkkosovellus toimii kuten muutkin verkkosovellukset, jotka toimivat mobiiliselaimissa. Se voi avata lisää HTML-sivuja ja suorittaa JavaScript-koodia. Tämän tyylisiä mobiilisovelluksia kutsutaan usein hybridisovelluksiksi.

Normaalisti verkkopohjaiset sovellukset ajetaan hiekkalaatikoissa. Tällä tarkoitetaan sitä, että ei ole suoraa pääsyä laitteen erilaisiin laitteisto- ja ohjelmisto-ominaisuuksiin. Esimerkkinä tästä toimisi yhteystiedot. Tiedot numeroista, nimistä ja sähköposteista eivät ole verkkosovelluksen käytössä. Cordovalla on olemassa tähän rajapintoja, jotka mahdollistavat pääsyn moniin laitteen ominaisuuksiin, kuten juuri esimerkiksi yhteystietoihin. Näitä ominaisuuksia pääsee käyttämään, kun lisää laajennukset projektiin.

3.3 Capacitor

Capacitor on Ionicin itse kehittämä avoimen lähdekoodin tekniikka, jolla voi myös tehdä yhdestä koodista natiivia eri alustoille. Capacitor ja Cordova ovat kilpailevia tekniikoita, joilla on molemmilla sama tavoite. Capacitor on käytännössä uudempi ja se käyttää hyväkseen viimeisimpiä verkkorajapintoja. Capacitor avaa täyden natiivin funktionaalisuuden riippumatta siitä, millä alustalla sovellusta käyttää. Tällä tavalla ei tarvitse monta rajapintaa eri alustoille, vaan ominaisuuksia voi tehdä kätevästi yhdellä rajapinnalla. Esimerkiksi kameran käyttämiseen ei tarvitse tehdä iOS- ja Android-ympäristöihin eri rajapintoja, vaikka näiden kameroiden logiikat ovat eri avaamisen ja käyttämisen kannalta.

Capacitor on näistä kahdesta vaihtoehdosta parempi oikeastaan siksi, että se on uudempi ja kehittyneempi. Capacitor on myös aktiivisesti ja laadukkaasti ylläpidetty, joka helpottaa kehittämistä, sillä korjaavia päivityksiä tulee varmasti ajallaan. Capacitor puolella on kokonainen tiimi kehittäjiä, markkinoijia ja asiakaspalvelua keskittynyt täysin tämän tuotteen kehittämiseen eteenpäin.

3.4 JavaScript

JavaScript on todella yleinen ohjelmointikieli verkkokehittämisessä ja tätä pääsee myös Ionicissa käyttämään. Sen päätarkoitus on selaimilla luoda dynaamisia toiminnallisuuksia. Tätä ohjelmointikieltä pääsee myös käyttämään Ionicissa, mutta ne ovat AngularJs, Vue.js ja React.js. Jokaista näistä eroaa jollain tavalla toisistaan lähinnä syntaksissa.

3.4.1 AngularJs

AngularJs on JavaScript-teknologia, jota käytetään dynaamisten verkkosovellusten luomiseksi (kuva 4). Alun perin AngularJs oli Googlen aloittama projekti, mutta nykyään se on avointa lähdekoodia. Se pohjautuu täysin HTML- ja JavaScript-ohjelmointikieliin, joten uutta kieltä tai syntaksia ei tarvitse opetella.

AngularJs muuttaa staattisen HTML:n dynaamiseksi ja se laajentaa myös HTML:n kykyjä lisäämällä sisäänrakennettuja komponentteja. AngularJs sallii myös mahdollisuuden luoda omia attribuutteja käyttämällä normaalia JavaScriptiä.

```
<!DOCTYPE html>

<html>
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app>
    Enter Your Name: <input type="text" ng-model="name" /> <br />
    Hello <label ng-bind="name"></label>
</body>
</html>
```

Kuva 4. Esimerkki AngularJs-koodista

3.4.2 Vue.js

Vue.js on itselle näistä kaikista tutuin, koska päällisin puolen teen itse työt tällä kielellä (kuva 5). Vue.js on progressiivinen teknologia käyttöliittymien rakentamiseen. Vue.js:n ero muihin vaihtoehtoihin on se, että se on suunniteltu käyttöönotettavaksi vaiheittain. Pääkirjasto keskittyy vain näkymätasoon ja on helppo oppia ja integroida muihin käytössä oleviin kirjastoihin tai projekteihin. Vue.js on myös helposti ylläpidettävissä ja testattavissa.

Vue.js on mahdollista lisätä vain yhteen komponenttiin tai sitten sen pohjalta voi rakentaa koko projektin. Omassa työpaikassa suurin osa koodista on kirjoitettu AngularJs-kielellä, mutta siirtyminen Vue.js:ään on aloitettu ja uudet ominaisuudet koodataan tällä. Hyvä puoli on juuri se, että Vue.js toimii aivan normaalisti, eikä siinä näe mitään eroa käyttäjän näkökulmasta AngularJs-kieleen. Omasta mielestä tämä kieli on helpompi sisäistää ja sen oppiminen on helpompaa kuin muiden mainitsemieni kielien. Vue.js on myös omalla tavallaan oliopohjaisen ohjelmoinnin tyylinen kieli.

HTML

```
<div id="app">
  {{ message }}
</div>
```

JS

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Hello Vue!

Kuva 5. Esimerkki Vue.js-koodista

3.4.3 React.js

React.js (kuva 6.) on näistä kielistä itselle vähiten käytetyin, enkä ole siitä läheskään yhtä perillä kuin muissa mainitsemissani kielissä. React.js on myös suunniteltu asteittaista käyttöönottoa varten, joten se perustuu samaan tyyliin kuin Vue.js.

React.js on myös avoimen lähdekoodin kirjasto, jota käytetään käyttöliittymien rakentamiseen. Sitä käytetään erityisesti yksisivuisille sovelluksille. Sen avulla voidaan käyttää uusiksi valmiiksi tehtyjä käyttöliittymäkomponentteja. Sen on luonut Jordan Walke, joka on Facebookissa työskentelevä ohjelmistokehittäjä. React on siis myös käytössä Facebookissa ja se otettiin käyttöön siellä jo vuonna 2011 ja sen jälkeen myös Instagramissa vuonna 2012. (Pandit 2021.)

Reactin avulla voidaan luoda suuria verkkosovelluksia, jotka voivat muuttaa tietoja lataamatta sivua uudelleen. Tällaisten tekemisen perustana on nopea, skaalautuva ja yksinkertainen kieli. Sitä voidaan käyttää yhdessä edellä mainittujen kielien kanssa.

```
ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```

Kuva 6. Esimerkkiä React.js-koodista

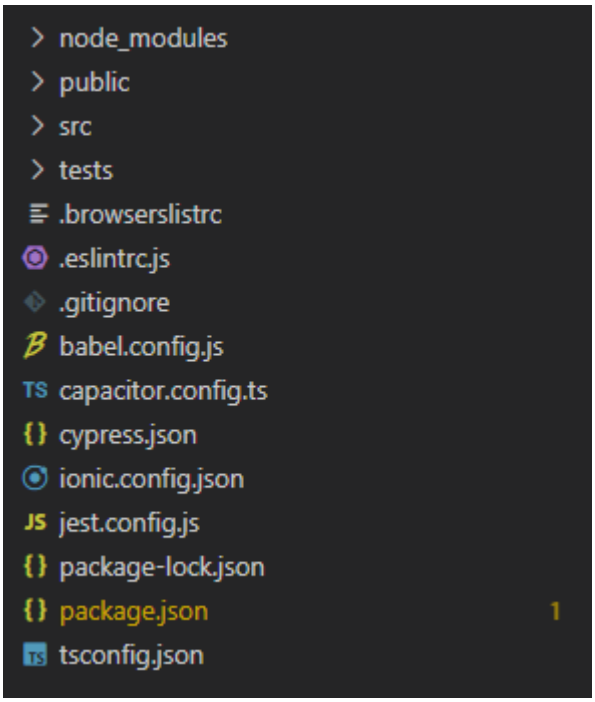
4 MOBIILISOVELLUS

4.1 Aloitus

Työhön kuuluu myös mobiilisovellus. Mobiilisovellus on salilla käymistä ja oman kehityksen seuraamista varten. Aloitin suunnittelemalla näkymiä. Miltä ne voisivat näyttää ja mitä ominaisuuksia tulee mihinkin näkymään.

Näkymät ajattelin pitää yksinkertaisena ja alustavasti suunnitelma on tehdä aloitusnäkymä, jossa voi aloittaa harjoittelun. Aloituksen jälkeen voi laittaa liikkeet, painot ja toistot ylös, jotka tallentuvat, kun lopettaa treenin. Toinen välilehti on kalenteri näkymä, josta pääsee katsomaan omia harjoituksia ja niiden tuloksia. Tämä näkymä on hyvä yleisnäkymä, jotta näkee kuinka paljon kuluvan kuukauden aikana, on harjoitellut. Kolmannessa näkymässä voi merkata omaa painoa ja seurata graafista käyrää mihin suuntaan menee.

Ionic tämän työn tekemiseen on hyvä alusta, koska siinä voi käyttää tuttuja verkkoteknologioita, joita olen käyttänyt töissä jo paljon. Tämä työ tehdään Ionic 7 -versiolla, koska se on uusin ja varustettu parhailla ominaisuuksilla. Tämä helpottaa varsinkin kalenterin kanssa, koska Ionic-pohjalla on jo valmiiksi hyvä valmis kalenteriominaisuus.
Kun näkymien suunnitelmat olivat valmiita, aloin rakentaa projektia ensimmäiseksi tekemällä Git-projektin ja yhdistämällä tämän omaan koneeseen helpottamaan versiohallintaa. Tämän jälkeen aloitin projekti kirjoittamalla terminaaliin komento ”ionic start trainingDiary tabs --type vue”. Komento aloittaa projektin ja parametrit tähän kutsuun on nimi, aloituspohja ja JavaScript-kielen tyyppi. Tämä komento luo pohjan työlle valmiiksi (kuva 7.), josta on helppo lähteä tekemään. Komento luo esimerkiksi src-kansion, jossa on pitkälti kaikki näkyvä koodi ja ohjaukset mikä näkymä vie mihinkin.



Kuva 7. Ionic-pohjarakenne

Ensimmäisenä kirjoitetaan komentokehotteeseen komento ”ionic serve”. Komennolla saadaan käynnistymään selaimelle sovellus, jotta sitä on helpompi lähteä testaamaan ja rakentamaan.

Suunnitelma on tehdä ensin JSON-lista harjoituksista, jotta niitä on helppo käsitellä ja aloittaa tekeminen. Listaan tulee treeni ensimmäiseksi arvoksi, jonka alle luodaan harjoitustuloksia. Esimerkiksi treeni voi olla veto, jonka alle merkitään harjoitus, esimerkiksi alatalja. Tämän alla on toistojen määrä, paino ja päivämäärä. Tällä formaatilla on helppo lähteä työstämään eteenpäin ja käsitellä arvoja listatyylillä.

Ensimmäinen sivu on siis vain sivu, josta pääsee aloittamaan harjoituksen ja tulosten merkkaamisen. Tämän painikkeen painamisen jälkeen avautuu valintaruutu, jossa on näiden edellä mainittujen JSON-listojen sisältämien harjoitukset ja nimet. Tästä pitää valita, mitä treeniä haluaa tehdä. Valinnan jälkeen avautuu näkymä liikkeistä, josta myös valitaan yksittäinen liike. Tähän oli tärkeää laittaa myös valinta, koska jos liikkeet tulisivat samassa järjestyksessä voisi tulla ongelma laitteen ollessa varattu. Tällöin tarvitsisi joko odottaa tai siirtyä seuraavaan liikkeeseen. Tähän oli kaksi ideaa:

joko esittää tyylit yksi kerrallaan tai kaikki tyylit kerralla. Ensimmäisessä vaihtoehdossa pääsisi navigoimaan yksi kerrallaan eteen- ja taaksepäin. Treenin voi alareunasta myös keskeyttää vapaasti eikä niitä tarvitse mennä loppuun asti, jotta treenin voi lopettaa.

Toinen sivu on kalenterinäkymä, jossa on normaali kalenteri, jossa päiväkohtaisesti näkyy, mitä on harjoitellut. Kalenteri on kuukausinäkymällä ja päivän kohdalla näkyy palkki, jos sille päivälle on merkinnyt harjoituksen. Tässä sinisessä palkissa lukee harjoituksen nimi, jotta heti aloitusnäkymässä on helppo ymmärtää, mistä on kyse ja mitä on treenattu ja milloin. Näkymästä pystyy myös tällöin seuraamaan hyvin lepopäiviä, koska näissä kohdissa ei ole mitään merkkausta. Klikatessa päivää aukeaa ikkuna, joka näyttää treenin tiedot eli toistot ja painot. Tähän näkymään suunnittelin ottavani valmiin Ionicin komponentin (kuva 8.), jota muokkaan niin, että siinä näkyy nuo edellä mainitut treenit. Tähän löytyy monta esimerkkiä, joten sen toteuttaminen ei ole niinkään vaikeaa. Tästä näkymästä pääset myös korjaamaan tietoja, jos huomaa että joku niistä on väärä.

```
<template>
  <ion-datetime></ion-datetime>
</template>

<script lang="ts">
  import { IonDatetime } from '@ionic/vue';
  import { defineComponent } from 'vue';

  export default defineComponent({
    components: { IonDatetime },
  });
</script>
```

Kuva 8. Ionicin valmiskomponentti kalenterista

Kolmas näkymä on graafinen näkymä, joka alkaa valitsemalla, mitä haluaa katsoa. Valintaruudusta voi taas valita JSON-listalta treenin. Graafi on viivakaavio, joka näyttää edistymisen harjoituksen valinnan jälkeen. Tälle näkymälle oli vaikea miettiä muita ominaisuuksia, joten päädyin pitämään sen näin yksinkertaisena. Näillä

ominaisuuksilla pärjää tässä näkymässä jo pitkälle, eikä se muuta tarvinnutkaan. Viivakaaviota varten on valmiita kirjastoja, joita aion käyttää ja tähän oli myös hyviä esimerkkejä, joiden avulla pääsee alkuun. Tätäkin näkymää tarvitsee myös vähän muokata, jotta siitä voisi valita aina eri harjoituksen.

Saatuani suunnitelman valmiiksi oli aika lähteä ohjelmoimaan sovellusta. Suunnitelma on edetä siinä järjestyksessä näkymiä, kuin ne on ylempänä lueteltu. Tällä tavalla prosessi pysyy kasassa ja järjestelmällisenä. Tärkeimmät ominaisuudet ensin ja loput perässä.

4.2 Ohjelmointi

Ensimmäisenä lähdetään luomaan välilehtipalkki. Tämän tekeminen on yksinkertaista. Luodaan mallipohja, johon tehdään välilehtiä. Ionic kielenä toimii helposti tässä, kun voidaan käyttää valmiita elementtejä. Tässä tapauksessa rakenne on helppo luoda ion-tabs-elementillä. Tämä näkymä ei tarvitse paljoakaan JavaScript-koodia ja tässä pärjää pitkälti pelkällä Ionicilla (kuva 9). Välilehdet toimivat Ion-tab-button-elementin href-attribuutilla, joka ohjaa oikealle sivulle.

```
<template>
  <ion-page>
    <ion-tabs>
      <ion-router-outlet></ion-router-outlet>
      <ion-tab-bar slot="bottom">
        <ion-tab-button tab="tab1" href="/tabs/tab1">
          <ion-icon :icon="triangle" />
          <ion-label>Aloita treeni</ion-label>
        </ion-tab-button>

        <ion-tab-button tab="tab2" href="/tabs/tab2">
          <ion-icon :icon="ellipse" />
          <ion-label>Kalenteri</ion-label>
        </ion-tab-button>

        <ion-tab-button tab="tab3" href="/tabs/tab3">
          <ion-icon :icon="square" />
          <ion-label>Kehittyminen</ion-label>
        </ion-tab-button>
      </ion-tab-bar>
    </ion-tabs>
  </ion-page>
</template>
```

Kuva 9. Ionicilla kirjoitettu välilehtipalkki

Ensimmäisen välilehden tekeminen vaatii ensin sivun, joka toimii vain yksinkertaisena linkkinä eteenpäin sovelluksessa. Tässä sivussa on painike, joka ohjaa aloittamaan harjoituksen ja toimii niin sanotusti kotisivuna. Sovelluksen päätavoite on pystyä merkitsemään ylös harjoitustietoja. Näin ollen tämä sivu oli loogisinta olla etusivuna, jotta pääsee heti aloittamaan treenin tulosten merkitsemisen. Painike toimii myös HTML:stä tutulla href-attribuutilla. Sivulla on myös painike, jolla pääsee lisäämään päivän painon, jotta sitä pääsee seuraamaan graafi-välilehdellä. Painiketta painamalla aukeaa yksinkertainen laatikko, johon kirjataan paino, minkä jälkeen painetaan tallenna. Tämä suorittaa funktion, joka tallentaa annetun luvun ja kirjaushetken päivämäärän. Tämä on tarkoitus olla aamupaino, joka lisätään vain kerran päivässä. Funktiossa on myös virheentarkastus. Jos datasta löytyy jo kyseiselle päivälle tehty merkintä, tallennus esittää virheen käyttäjälle siitä, että ei voi tehdä kahta tallennusta samalle päivälle.

Aloita treeni -painikkeen painamisen jälkeen sovellus ohjautuu sivulle, josta pääsee valitsemaan halutun harjoituksen. Tällä sivulla valitaan listasta harjoitus, joka tulee Vuen data-attribuuttista, jossa on lista harjoituksia. Vuessa for-toistorakenne toimii myös HTML-komponenttien luomiseen. Se kirjoitetaan elementin v-for-attribuutin kuvan 10 esittämällä tavalla. Tässä voidaan toistaa Vue data-attribuutin sisällä olevia listoja ja tehdä niillä komponentteja. Vuen dataan kirjoitetun listat ovat automaattisesti natiiveja komponentteja ja päivittyvät natiivisti, kun niihin tulee muutoksia. Toistorakenne on todella voimakas ja helppo tapa luoda esimerkiksi listoja vaihtoehdoista ja myös esimerkiksi isoja taulukkoja, joissa on paljon tietoa. Näkyviin saadaan tiedot niin, että ne pistetään neljän hakasulkeen sisään.

```
<template>
<ion-page>
    <ion-header>
      <ion-toolbar>
        <ion-title class="center">Valitse Treeni</ion-title>
      </ion-toolbar>
    </ion-header>
    <ion-content :fullscreen="true">
      <ion-header collapse="condense">
        <ion-toolbar>
          <ion-title size="large" class="center">Valitse Treeni</ion-title>
        </ion-toolbar>
      </ion-header>

      <ion-list class="centerContent">
        <ion-item>
          <ion-select placeholder="Valitse treeni">
            <ion-select-option v-for="exercise in exercises" :key="exercise">{{ exercise }}</ion-select-option>
          </ion-select>
        </ion-item>
      </ion-list>
      <ion-button class="viewExercise">Näytä liikkeet!</ion-button>
    </ion-content>
  </ion-page>
</template>
```

Kuva 10. Esimerkki v-for rakenteesta

Sen jälkeen, kun valitaan liike, avautuu lista liikkeistä ja tuloksia voi alkaa lisäämään niille tarkoitetulle kohdalle. Sivunvaihto tapahtuu siten, että painikkeen painaminen laukaisee Vue-funktion, joka vaihtaa Vuen datamuuttujan arvon true-arvosta false-arvoon. Muuttujan vaihtuessa tarkastetaan v-if-attribuutilla, onko arvo true vai false. Tällä perusteella vaihdetaan näkyviä komponentteja. Ensin siis näkyy näkymä, jossa valitaan listasta yksinkertaisesti, minkä harjoituksen haluaa tehdä ja sen jälkeen tulee heti liikkeet ja niihin merkittävät toistot ja painot. Toiseen Ionic-komponenttiin kirjoitetaan v-if ja toiseen v-else, jolloin natiivisti sivu ymmärtää vaihtaa komponenttia

oikeaksi. Muuttujan vaihtuessa ja uuden näkymän latautuessa tehdään taas uusi v-for toistorakenne. Tämä toistorakenne on sisäkkäinen toisto (kuva 11.), jossa ensin toistetaan koko datalista computed-funktion kautta, joka palauttaa listan harjoituksesta ja sen jälkeen tehdään toinen toistorakenne, joka toistaa niin monta kertaa harjoituksen, kun datassa tulleita liikesarja määriä on. Esimerkiksi, jos datan mukana tulee harjoite, jossa lukee kolme settiä, toistorakenne luo kolme komponenttia, joihin voi laittaa tuloksen ylös. Kun on merkannut tulokset, voi painaa tallenna painiketta. Tallenna-painikkeen napsauttaminen aiheuttaa Vuen funktion saveExercise suorittamisen.

```
<ion-list class="centerContent">
  <div v-for="exercise in exerciseData" :key="exercise">
    <div v-for="exe in exercise.sets" :key="exe">
      <p style="font-size: 12px;">{{ exercise.name }}</p>
      <input type="text" placeholder="Paino">
      <input type="text" placeholder="Toistot">
    </div>
  </div>
</ion-list>
```

Kuva 11. Tuplatoistorakenne

Data kulkee JSON-muodossa harjoitus näkymässä ja se on luotu niin että harjoitukset tulee muodossa, jossa on harjoituksen nimi ja settien määrä. Tallennuksesta treenit tallentuvat myös JSON-muodossa harjoituksen nimi, setin numero, paino sillä setillä, toistot ja päivä. Kuvassa myös esimerkkinä miltä sovellukseen syötetty harjoitus näyttää ja mitä se sisältää. Tämä rakenne on todella yksinkertainen, eikä tarvitse muuta kuin treenin nimen ja settien määrän, jotta tiedetään, kuinka monta kertaa toistorakenne ajetaan lävitse (kuva 12).

```
{"Push":[
    {"name":"Flat Dumbbell Bench Press", "set":1, "weight":20, "reps":12, "date":"2023-04-16"},
    {"name":"Flat Dumbbell Bench Press", "set":2, "weight":20, "reps":12, "date":"2023-04-16"},
    {"name":"Flat Dumbbell Bench Press", "set":3, "weight":20, "reps":12, "date":"2023-04-16"}
]}

{"Push":[
    {"name":"Flat Dumbbell Bench Press", "sets":3},
    {"name":"Incline Bench Press", "sets":3},
    {"name":"Chest Flyes", "sets":3},
    {"name":"Close Grip Bench Press", "sets":3},
    {"name":"Tricep Pushdown", "sets":3},
    {"name":"Dumbbell Shoulder Press", "sets":3},
    {"name":"Dumbell Lateral Raise", "sets":3}
]}
```

Kuva 12. Esimerkki muodosta missä JSON-data tallentuu

Kalenterinäkymä luodaan Ionicin omalla ion-datetime-komponentilla. Kalenteriin haetaan JSON-lista ja tästä listasta katsotaan päivä. Tämä päivä, joka tulee datasta sidotaan kalenterissa päivään. Kalenterissa ratkaisu tähän on se, että luodaan v-model kenttä kalenterille nimellä ”selected”. Tuon avulla päästään käsiksi siihen päivään, jonka käyttäjä on valinnut ja voidaan verrata listaan treenejä ja hakea sieltä samalla päivällä oleva harjoitukset. Funktion läpikäytyä tämä lista harjoituksia tehdään uusi lista näistä harjoituksista, jotka ovat kyseisellä päivällä. Uuden listan tullessa luoduksi avautuu klikkaamalla päivää ion-modal-komponentti (kuva 13). Tämä uusi ikkuna pitää sisällään treenin, jonka käyttäjä on tehnyt ja listaa datan. Tästä näkymästä pääsee myös muokkaamaan dataa, jos siinä huomaa virheitä. Lista näkymän pohjalla on painikkeet peruuta, joka ajaa funktion closeModal tämä funktio sulkee komponentin. Tämän lisäksi siinä on tallennus nappula, joka ajaa funktion saveModal. Tämä funktio tarkistaa onko muutoksia tehty. Jos harjoituksia on muutettu, se vaihtaa JSON-datan tallennusta varten uuteen dataan. Listan muodostamisen jälkeen se sulkee komponentin, jolloin päästään takaisin kalenteri näkymään. Kalenterissa pääsee menemään kuukausia yläkulmasta eteenpäin ja taaksepäin. Nämä tulokset tulee samanlaisella toistorakenteella, kuin aiemmatkin vastaavat toistavat rakenteet on tehty, jotta saadaan komponentteja oikea määrä.

Kuva 13. Kalenteri päivää painaessa avautuva näkymä tehdyistä harjoituksista

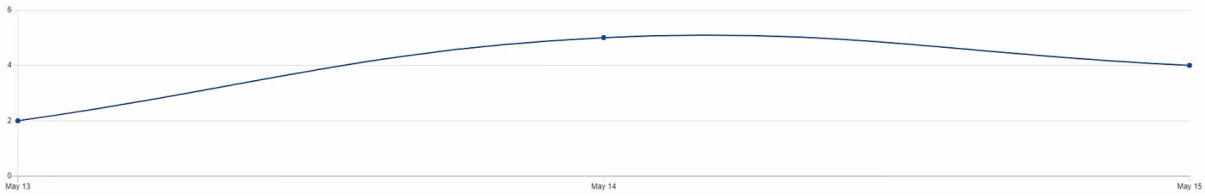
Viimeinen näkymä on tulosten seuraamiselle tarkoitettu viivadiagrammi näkymä. Tässä näkymässä on ikkuna, johon piirtyy kaavio x- ja y-akselin arvojen perusteella. X-akselilla on painomäärä ja y akselilla on päivämäärä. Graafin alla on kaksi valinta laatikkoa, joista toisessa on kaikki harjoitukset, jotka haetaan JSON-datasta ja laitetaan alasvetovalikkoon. Tästä valikosta voi siis valita, minkä treenin graafi piirtää näkyville. Tämän alasvetovalikon alla on toinen vastaavanlainen, mutta tämä vertaa sitä, kuinka pitkältä ajalta halutaan nähdä tulokset. Tässä on vaihtoehtoina yksi kuukausi, kolme kuukautta, kuusi kuukautta tai vuosi. Vaihtoehtoja voisi laittaa paljon enemmänkin, mutta näin, että nämäkin riittävät jo pitkälle ja niillä pääsee seuraamaan edistymistä jo hyvin. Tässä näkymässä siis valitaan ensin treeni ja sitten oletuksena on kolme kuukautta ajallisesti ja tämä piirtää näkyville graafin. Tässä näkymässä käytetään kirjastoja nimeltä Vue Chartjs ja Chartkick. Näillä kirjastoilla sai tehtyä mukavasti tämän graafin, eikä näiden kirjastojen käyttö ollut hankalaa. Käytännössä ei tarvitse kuin asentaa kirjastot konfiguroida ne main.ts tiedostoon ja käyttää niitä yksinkertaisilla HTML-elementeillä. Tämä kirjasto toimii sellaisella tägillä kuin line-chart (kuva 14). Sillä saa luotua yksinkertaisen viivakaavion. Tähän elementtiin laitetaan vielä data, joka tulee suoraan JSON-muodossa tägin sisällä annettavaan data parametriin ja kirjasto ymmärtää tästä piirtää graafin.

```
<line-chart :data="exerciseData" ></line-chart>
```

Kuva 14. line-chart-tägi ja sen yksinkertaisuus

Tiedot haetaan JSON-listasta muuttujaan exerciseData, joka on tyhjä lista ja se päivittyy aina, kun vaihdetaan mitä harjoitusta halutaan näyttää. Alasvetovalikkoon tehdään kuuntelijafunktio, joka ottaa v-model-arvon siitä, mikä on valittuna valikosta ja hakee tämän harjoituksen nimen avulla listaan treenit, jotka löytyvät datasta samalla nimellä. Tämän voisi tehdä myös niin, että id olisi jokaisella harjoitteella erillisessä taulussa, joita käytettäisiin sitten loogisesti. Tämän aion tehdä tulevaisuudessa helpottamaan varsinkin kantakäsittelyä ja joitakin kyselyjä kantaan. Toisesta alasvetovalikosta otetaan sitten aikarajaus ja otetaan listaan vain ne, jotka näiden päivien rajausten sisällä ovat. JSON-datan valmistuttua Vue huomaa, että yksi datan tägien alla olevista

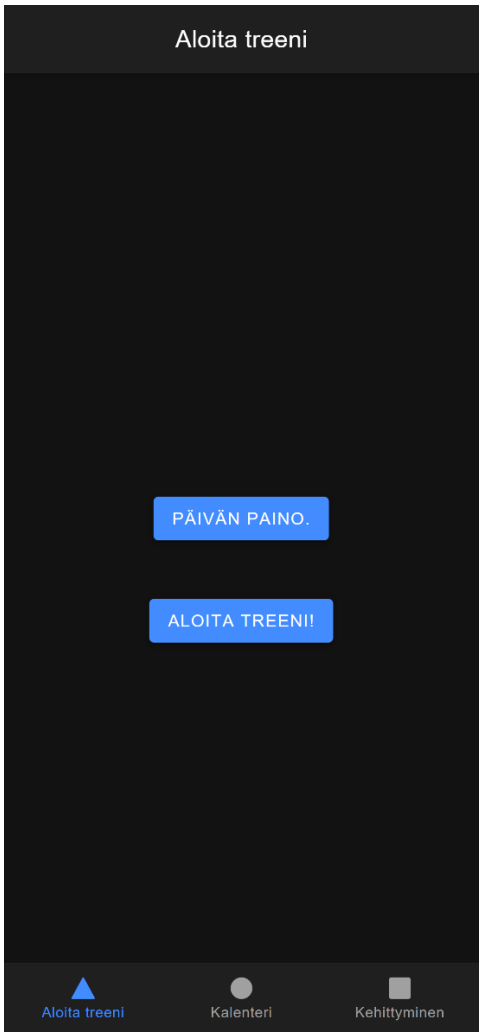
muuttujista on muuttunut ja tajuaa sitten päivittää näkymän ja kirjasto piirtää uuden graafin (kuva 15).



Kuva 15. Esimerkki miltä viivadiagrammi näyttää

4.3 Sovelluksen yhteenveto

Sovelluksesta tuli omaan käyttöön jo toimiva siihen tarkoitukseen, johon sen ensiksi halusin luoda. Voin merkitä tulokset ylös itselleni, jotta pääsen seuraamaan niitä puhelimelta. Pieniä bugeja lukuun ottamatta voin siis mennä salille harjoittelemaan ja kirjata tulokseni ylös ja katsoa niitä kalenteri näkymästä ja muokata jälkeenpäin virheiden varalta. Tyylien puolesta sovellus on vielä vähän raaka, mutta en keskittynyt tässä vielä enempää niihin vaan halusin ne toiminallisuudet, joilla pääsen aloittamaan käyttöä. Graafi näkymä jäi myös vielä hiukan kesken ja muutama bugi jää tähän vielä korjattavaksi. Sovelluksen suorituskyky on hyvä vaikkakin sovelluksessa on silti loppujen lopuksi niin vähän sisältöä, että sen ei pitäisikään kärsiä hitauksista. Tämä oli mainittuna hyödyissä ja haitoissa, mutta kuten tästä voi nähdä, jotta pääsisi siihen pisteeseen, että sovellus alkaisi hidastumaan. Vaikka tyylejä en lähtenyt ihmeellisemmin tekemään sovelluksesta saa silti ihan järkevän näköisen pienelläkin työllä (kuva 16).



Kuva 16. Sovelluksen etusivu

5 YHTEENVETO

Tämä aihe tuli opinnäytetyöksi aika luontevasti, koska tein kyseisen työkalun kanssa töitä päivittäin. Pidän henkilökohtaisesti tästä työkalusta paljon ja se on antanut itselleni helpon lähestymistavan kohti mobiilikehittämistä. Isoja positiivisia puolia on Ionicin käytön helppous, useat hienot ominaisuudet, sen monipuolisuus ja valinnan vara ohjelmointikielen puolesta. Ionic alkaa myös selvästi nousta käytetymmäksi, koska koko ajan löytää enemmän tietoa ja ohjeita verkosta, miten tehdä erilaisia asioita. Ionicissa on myös miinuspuolia, joita varmasti keskitytään korjaamaan ja edistämään

kehittäjien kesken. Ionicin ollessa tarpeeksi kehittynyt ja monikäyttöinen se voisi hyvin korvata todella monta mobiilisovelluksen tekemiseen tarkoitettua teknologiaa.

Työni keskittyi Ionicin historiaan ja miten se suurin piirtein toimii. Katsottiin myös miltä näyttää Ionicin käyttöliittymä ja mitä eri vaihtoehtoja ohjelmointikieliin on tämän työkalun kanssa tekemiseen. Työhön myös kuului oman mobiilisovelluksen tekeminen, jonka aion ottaa omaan käyttöön sen tulessa siihen pisteeseen, että olen itse tyytyväinen ja valmis sitä käyttämään. Oman sovelluksen tekeminen oli todella hyvä projekti oman osaamisen kannalta, koska se lähti aivan nollasta. Töissä olin ennen lähinnä korjannut, päivittänyt ja kehittänyt jo valmista pohjaa, mutta kun pääsi tekemään kokonaisen sovelluksen itse tajusi senkin, kuinka paljon siihen menee työtä, mutta senkin tajusi, ettei se ole mitään ydinfysiikkaa. Opin samalla käyttämään taas uutta kirjastoa graafien tekemiseen, josta voi olla hyötyä tulevaisuudessa, jos joskus tulee mieleen tehdä jokin uusi projekti, jossa voisi käyttää tätä opittua kirjastoa hyödyksi. Ionic on todella sulava ja valmis kieli tehdä monia eriasioita web – ja mobiilisovellusten kanssa, enkä yhtään yllättyisi, jos sitä alettaisiin käyttää useammassakin paikassa, kun sitä on kehitetty lisää ja parannettu asioita, jotka sitä tarvitsevat. Tähänkin ongelmaan varmasti löytyy apua myös ulkoisista kirjastoista, koska nykypäivänä sovelluskehitys on niin avointa, että melkein mikä vaan mikä käy mielessä on mahdollista tehdä

Tästä työstä jää minulle paljon käteen. Yleistä oppia Ionicista ja yksi onnistunut projekti, jota voi varmasti käyttää ansioluettelossa. Sovellukset ovat jo nyt iso asia ja niiden merkitys koko ajan vain nousee. Sovelluksen jatkokehitykseen on muutama idea, jotka voisi toteuttaa esimerkiksi tavoitteiden laatiminen ja harjoitusohjelmien laatiminen. Näihin voisi olla ajatuksena, että tavoitteiden laatimisen kannalta olisi kirjasto eri harjoituksia ja tähän sivuun voisi merkata päivämäärän ja paino tavoitteen, jonka haluaisi saavuttaa. Tämän tavoitteen voisi sitoa jotenkin jo olemassa olevaan viivadiagrammiin ja siitä voisi nähdä, kuinka lähellä on saavuttaa tavoitteen ja kuukauden välein sovellus voisi antaa yhteenvedon siitä saavutettiinko näitä tavoitteita. Toinen ominaisuus olisi harjoitusohjelmien laatiminen. Tämän voisi toteuttaa sillä tavalla, että etsisi jostain valmiin listan eri harjoituksia, joita voisi käyttää myös tavoite näkymässä. Tästä listasta tehtäisiin kirjasto ja kirjastosta voisi tehdä omia harjoituksia ja laatia vapaasti liikkeitä, toistoja ja työmääriä. Tämän ominaisuuden myötä sovellus voisi olla

jo siinä kunnossa, että sitä voisi antaa julkiseen käyttöön, eikä pitää vain omana projektina.

LÄHTEET

Griffith, C n.d. What is Apache Cordova? Viitattu 1.2.2022 https://ionic.io/resources/articles/what-is-apache-cordova

Mittal, A 2017. What is Ionic? Viitattu 1.2.2022 https://hackernoon.com/what-is-ionic-c1da6eab0d8a

Pandit, N 2021. What And Why React.js? Viitattu 1.2.2022 https://www.c-sharpcorner.com/article/what-and-why-reactjs/

Sharma, Y 2021. Ionic framework version history. Viitattu 2.2.2022. https://tutorialslink.com/Articles/Ionic-Framework-Version-History-Ionic-Framework/3051

Sharma, Y 2021. What are the advantages and disadvantages of the Ionic Framework? Viitattu 2.2.2022. https://tutorialslink.com/Articles/What-are-the-Advantages-and-disadvantages-of-Ionic-Framework-Ionic-Framework/3050

Ionic.io n.d. Capacitor vs Cordova: What are the Differences Between Them? Viitattu

18.4.2023 Capacitor vs. Cordova: What is the Difference Between Them? (ionic.io)

TutorialsTeacher n.d. What is AngularJs? Viitattu 1.2.2022 https://www.tutorialsteacher.com/angularjs/what-is-angularjs

Vue.js n.d. What is Vue.js? Viitattu 1.2.2022 https://vuejs.org/v2/guide/

Wiredelta 2020. 10 Most Popular Ionic Apps of 2019. Viitattu 2.2.2022. https://wiredelta.com/10-most-popular-ionic-apps-2019/