Iida Rinne

KÄYTTÖLIITTYMÄN SUUNNITTELU MOBIILIHENKILÖREKISTERI-SOVELLUKSELLE

Tietojenkäsittelyn koulutusohjelma

2017

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KÄYTTÖLIITTYMÄN SUUNNITTELU MOBIILIHENKILÖREKISTERI-SOVELLUKSELLE

Rinne, Iida
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toukokuu 2017
Ohjaaja: Hentunen, Ilmari
Sivumäärä: 29
Liitteitä: 0

Asiasanat: mobiilisovellukset, käyttöliittymät, NFC-teknologia, Apache Cordova

---

Opinnäytetyöni aiheena on suunnitella uusi käyttöliittymä mobiilihenkilörekisteri sovellukselle. Mobiilihenkilörekisteri sovelluksella tarkoitetaan tässä työssä henkilön terveystietoja sisältävää mobiilisovellusta, joka pystyy lukemaan sekä tallentamaan henkilön terveystietoja NFC-teknologiaa hyödyntäen.

Työni sisältää tietoa käytetyistä tekniikoista, työn suunnitelmat ja työn toteutus sekä loppuratkaisu. Tavoitteenani on kehittää toimiva prototyyppi käyttöliittymästä Apache Cordova teknologian sekä muiden perinteisten web-teknologioiden avulla. Sovellus suunnitellaan terveysalan ammattilaisten käyttöön ja työn toimeksiantajana toimii SWOcean Oy.

DESIGNING A USER INTERFACE FOR THE MOBILE PERSONAL REGISTRY APPLICATION

Rinne, Iida
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in data processing
May 2017
Supervisor: Hentunen, Ilmari
Number of pages: 29
Appendices: 0

Keywords: mobile applications, user interfaces, NFC technology, Apache Cordova

---

.

The subject of my thesis is to design a new user interface for the mobile personal registry application. In this work mobile personal registry application means, a mobile application that includes a person's health data which can read and record the person's health data by using NFC technology.

My work contains information of used techniques, the work plans and the implementation of the work as well as the final solution. My goal is to develop a functional prototype from the interface with help of Apache Cordova technology and other standard web technologies. The application is designed for professionals who work in health sector and the work is commissioned by SWOcean Oy.

SISÄLLYS

1 JOHDANTO

Työni aiheena on suunnitella uusi käyttöliittymä mobhere sovellukselle eli mobiilihenkilörekisteri-sovellus, joka pitää sisällään henkilön terveysrekisterin, jota pystyy lukemaan ja päivittämään NFC-teknologiaa hyödyntäen mobiililaitteella. Työn toimeksiantaja on porilainen yritys SWOcean Oy ja mobiilisovellus teeteään terveysalan ammattilaiseten käyttöön. Idea tämän sovelluksen kehittämiseen lähti siitä, että nykyään monissa terveyspalveluissa on vaikea saada päivitettyä tietoa asiakkaiden terveydestä tai asiakkaiden terveyshistoriasta, kuten esimerkiksi rokotuksista ja allergioista. Joten helpottaakseen terveysalan ammattilaisia lähti liikkelle idea NFC-tageistä, jotka sisältäisivät potilaan kaikki terveystiedot, kuten sairaudet, lääkitykset, rokotukset ja allergiat, jolloin potilaan ei tarvitse näitä tietoja muistaa ja tieto olisi selvempää myös terveysalan työntekijälle, joka tietää esimerkiksi lääkkeiden nimet ja käyttötarkoituksen.

Mobiilisovellus on siis monivaiheinen ja oma osuuteni tämän sovelluksen tekemisessä on kehittää uusi käyttöliittymä, joka on myös tämän opinnäytetyön pääaiheena. Käyttöliittymän kehityksestä ja käytetyistä tekniikoista aion kertoa tarkemmin tässä kirjoitelmassa. Käyttöliittymän tulen toteuttamaan on Apache Cordova -teknologialla, joka on avoimenlähdekoodin sovelluskehys, jolla pystyy kehittämään hybridisovelluksia mobiilialustoille Javascript, HTML ja CSS -kielien avulla.

Toimeksiantaja on SWOcean Oy, joka on vuonna 2013 perustettu porilainen yritys. Toimialana yrityksellä on atk-laitteisto- ja ohjelmistokonsultointi, yritys toteuttaa myös erilaisia mobiili- ja verkkosovelluksia. SWOcean tuli minulle tutuksi, kun pääsin heille työharjoitteluun ja he antoivat minulle myös tämän toimeksiannon, koska olin silloin aiemmissa työtehtävissäni tehnyt tutkimustyötä tähän opinnäytetyöhön liittyen.

2 TEKNIIKAT

Mobiilisovelluksen käyttöliittymä on toteutettu Cordova-teknologialla sekä muilla perinteisillä web-teknologioilla. Käyttöliittymän toiminnallisuus on toteutettu javascriptin avulla, joten esittelen tässä luvussa myös muutamia Javascript-kirjastoja, joita käytän käyttöliittymän prototyypin toteutuksessa. Kehiteltävän mobiilisovelluksen toiminnallisuuksiin ollaan suunniteltu kuuluvan, että henkilön terveystiedot ja terveyshistorian pystyisi lukemaan sekä tallentamaan NFC-teknologiaa hyödyntäen. Joten esittelen tässä luvussa lyhyesti myös NFC-teknologiaa.

2.1 NFC

NFC (Near Field Communication) eli lähiviestintä tekniikka, joka hyödyntää RFID-teknologiaa (Radio Frequency Identification) lyhyen matkan langattomaan viestintään. (Top Tunniste). RFID:llä tarkoitetaan radiotaajuista tunnistusta, joka hyödyntää radioaaltoja lukeakseen esimerkiksi tagin (tunnisteen) tietoja muutamankin metrin päästä. (Bar Code Graphics 2017). Viestinnän toteuttamiseen tämän tekniikan avulla, tarvitaan tietoa siirtävää laitetta ja tietoa vastaanottavaa laitetta. NFC-standardeja toteuttavat laitteet voidaan jakaa pasiivisiin ja aktiivisiin laitteisiin.

Passiiviset laitteet esimerkiksi NFC-tagit tai muut vastaavat pienet lähettimet pystyvät lähettämään tietoa NFC-laitteisiin käyttäen sähkömagneettikenttiä, jotka aktiivinen laite tuottaa, jolloin ne eivät tarvitse erillistä virtalähdettä. Pasiiviset laitteet eivät kuitenkaan käsittele tiedon lähettämistä muista lähteistä eivätkä pysty yhdistämään muihin passiivisiin komponenteihin. Aktiiviset laitteet pystyvät vastaanottamaan ja lähettämään dataa. Aktiiviset laitteet esimerkiksi älypuhelimet, pystyvät keskustelemaan sekä aktiivisten että pasiivisten laitteiden kanssa. (Triggs 2017.)

NFC-tekniikan toiminto perustuu infromaation lähettämiseen radioaaltojen avulla ja NFC-strandardia pystyy datan käsittelyssä hyödyntämään kolmella eri toiminnolla.

Tunnetuin toiminto peer-to-peer -käytäntö (suomeksi vertaisverkko), jota käytetään paljon älypuhelimissa. Toiminnossa kaksi NFC-tekniikkaa toteuttavaa laittetta pystyvät keskenään vaihtamaan erilaista informaatiota olemalla, joko aktiivisena eli dataa lähettävänä osapuolena tai pasiivisena eli dataa vastaanottavana osapuolena. Lue ja kirjoita toiminto on sen sijaan yksisuuntaista datan siirtoa, jossa aktiivinen laite linkittyy toiseen laitteeseen, josta aktiivinen osapuoli lukee informaatiota. Tätä toiminnallisuutta käytetään kun ollaan vuorovaikutuksessa NFC tagin kanssa. Viimeinen mahdollisesti käytettävä toiminnallisuus NFC-tekniikassa on kortti emulointi, jolloin NFC-laite on käyttäytyy luottokortin tavoin, millä pystyy maksamaan. (Triggs 2017.)

2.2 Apache Cordova

Apache Cordova, lyhyesti Cordova, on avoimen lähdekoodin ohjelmistokehys, jonka avulla kehittäjä voi muuntaa Javascript-, HTML- ja CSS-kielillä tehdyn sovelluksen natiiviksi sovellukseksi useammalle eri mobiilialustalle (Camden 2016, 3). Cordova-projekti sisältää kolme päätoimintoa: komentokehotetyökalun, pääsyn laitteiden toimintoihin ja kyvyn tukea tulevia toimintoja. Komentokehotetta käytetään projektien luontiin ja koodin kokoamiseen mobiilialustalle. Cordova tarjoaa myös pääsyn useimpiin mobiililaitteen toimintoihin rajapintojen kautta, kuten kameraan, media-tiedotoihin ja yhteystietoihin. Tulevia toimintoja tukeakseen Cordovalla on plugin rajapinta, jolla se pystyy tukemaan mitä tahansa toiminnallisuuksia, joita mobiililaite tukee. Tuettavan toiminnallisuuden lisäämiseksi täytyy hiukan kirjoittaa natiivikoodia ja lisätä se vastaavaan Javascript-kirjastoon, minkä jälkeen tuki on muidenkin kehittäjien saatavilla. Vaikka Cordovalla tehdäänkin sovelluksia mobiililaitteille, niin se ei automaattisesti optimoi koodia mobiililaitteeseen sopivaksi, jolloin tulisi kehittäjän itse keksiä sopiva ratkaisu, jotta sovelluksesta tulisi responsiivisempi, kuten esimerkiksi käyttää projektissa Bootstrap-kirjastoa. (Camden 2016, 7.)

Cordova on aiemmin tunnettu nimellä PhoneGap, jonka on luonut Nitobin vuonna 2008. Se on Cordovan tavoin julkaistu avoimeksi lähdekoodiksi mobiilisovelluksien kehittämiseen ja käyttää Javascript-rajapintoja saadakseen yhteyden laitteen

natiiveihin toimintoihin. PhoneGapista tuli hyvin suosittu kehittäjien keskuudessa, koska mobiilialustoille pystyi kehittämään mobiilisovelluksia samoilla perjaatteilla kuin web-sivuja, joten ei tarvinnut osata natiiveja kieliä. Lokakuun 4. vuonna 2011 Adobe osti Nitobin ja samoihin aikoihin PhoneGab-projekti siirtyi Apache Software Foundationille, jolloin PhoneGap muutti nimensä Apache Cordovaksi. (Camden 2016, 4.)

2.2.1 Cordovan sovellusarkkitehtuuri

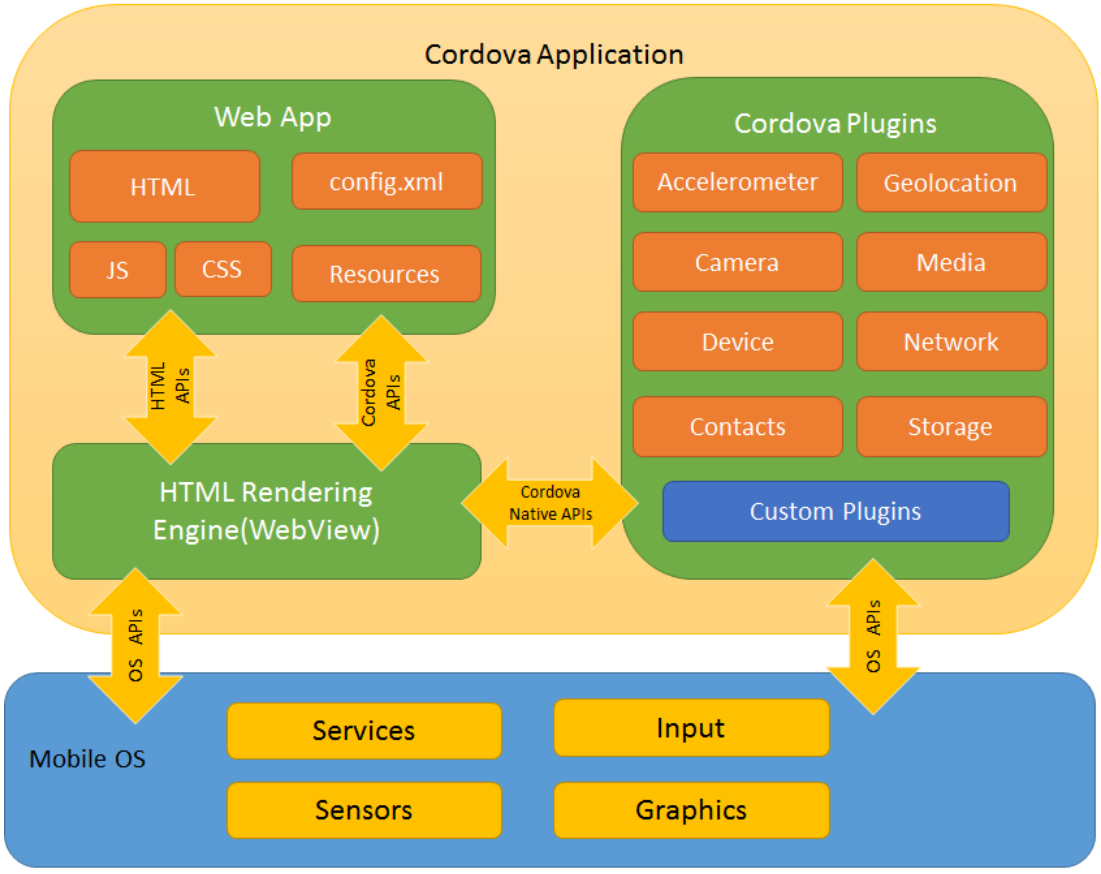
Apache Cordovalla kehitetään siis hybridisovelluksia perinteisten webteknologioiden avulla. Sovellukset paketoidaan natiiveiksi ja toteutetaan kohdistaen ne omille alustoilleen, jossa sovellukset käyttävät alustan ohjelmointirajapintoja (kuvattu kuvassa 1) ja pääsevät näin käsiksi laitteiden kapasiteettiin. (The Apache Software Foundation 2015a). Cordovalla on tuki useammalle alustalle, kuten Android, iOS, Windows, BlackBerry10, Ubuntu, Windows Phone 8 ja OSx. (The Apache Software Foundation 2015b.)

2.2.1.1 Pluginit

Cordovassa käyttetään plugineja eli liitännäisiä tai lisäosia. Ne ovat paketteja, joita käytetään natiivin alustan kanssa kommunikointiin. Pluginit mahdollistavat pääsyn laitteiden ja alustojen toiminnallisuuksiin, kuten kameraan ja GPS:än, mikä yleensä ei ole mahdollista web-pohjaisille sovelluksille. Pluginit tarjoavat rajapinnat, joiden avulla Cordova ja natiivit komponentit pystyvät keskustelemaan keskenään sekä pluginit tarjoavat liitännän laitteen rajapintoihin. Cordova sisältää myös muutaman ydinpluginin (löytyy kuvasta 1 kohdasta Cordova Plugins) joiden avulla sovellus pääsee käsiksi laitteen kapasiteettiin. Myös omien pluginien tekeminen on mahdollistettu Cordovassa. Uusissa Cordova-projekteissa oletuksena on, että projekti ei sisällä valmiita plugineja, ei edes ydinplugineja, jolloin niitä käyttääkseen tulee pluginit erikseen lisätä projektiin. (The Apache Software Foundation 2015a.)

2.2.1.2 Web-sovellus

Kuvassa 1 kohdassa Web App kuvataan sovelluksen kohta, jossa koodi sijaitsee. Sovellus toteutuu aivan kuin web-sivuna oletustiedostossa index.html. Sovellus toteutuu kuvan 1 WebView-kohdassa eli web-näkymässä sisältäen paketin, joka tekee sovelluksesta natiivin, web-näkymä sisältää myös sovelluksen käyttöliittymän. Web App -kohdasta löytyy myös tärkeä config.xml -tiedosto, joka sisältää kaiken tiedon ja parametrit sovelluksesta. (The Apache Software Foundation 2015a.)



Kuva 1. Näkymä Cordovan sovellusarkkitehtuurista. (The Apache Software Foundation).

2.2.1.3 Kehityspolut

Cordovalla on kaksi mahdollista työtapaa, joita voi käyttää sovellusten kehittämiseen. Yleinsä sovellusten tekemisessä tulee käytettyä molempia työtapoja, joista ensimmäinen on laiteriippumaton työtapa. Sitä käytetään, kun halutaan

sovelluksen toimivan useammalla eri mobiilikäyttöjärjestelmällä vähäisellä alusta kohtaisella kehittämisellä. Laiteriippumattomassa kehityksessä käytetään CLI:tä (Command Line Interface) eli komentokehotetta keskeisenä työkaluna, jonka avulla pystyy rakentamaan kerralla projektin useammalle alustalle. CLI myös kopioi alihakemistoihin kokoelman yleisiä web-arvoja jokaisellle mobiilialustalle, tekee konfguraation muutokset ja ajaa koodit luodakseen sovelluksen binäärit sekä tarjoaa yleisiä rajapintoja pluginien hakemiseen sovelluksille. (The Apache Software Foundation 2015a.) Toinen työtapa on alusta-keskeinen, jolloin halutaan tehdä sovellus vain yhdelle alustalle ja tahdotaan kehittää sovellusta alemmillakin tasoilla. Tätä työtapaa kannattaa käyttää, jos projektia pitää muokata SDK:lla (Software Development Kit) eli ohjelmankehityspaketilla. Työtapa luottaa kokoelmaan alempitasoisiin koodien komentoriveihin, jotka ovat räätälöity jokaiselle tuetulle alustalle ja erillinen Plugman-apuväline, jonka avulla pystyy asentamaan plugineja. (The Apache Software Foundation 2015a.)

2.3 Web-teknologiat

Tämä osio sisältää lyhyen esittelyn käyttöliittymän prototyypissä käytetyistä web-teknologioista eli HTML-, CSS- ja Javascript-kielistä. HTML (Hypertext Markup Language) on yleinen merkintäkieli verkkosivujen luomiseen. HTML käyttää elementtejä, jotka esitetään tagien (tunnisteiden) muodossa verkkosivun rakentamiseen. Tagit kertovat selaimelle, miten sivun rakenne tulee muodostumaan ja tagit esitetään pareittain avaavalla tagilla ja sulkevalla tagilla, esimerkiksi tähän väliin upotetaan teksti, joka halutaan näkyvän selaimessa. . (Refsnes Data 2017a.) HTML-kieltä pystyy tuottamaan tavallisen tekstieditorin avulla ja sille on kehitelty jo useampia versioita, mutta tässä työssä käytettään HTML5:sta eli viimeisintä HTML versiota.

CSS (Cascading Style Sheets) on tyylikieli, jonka avulla pystytään manipuloimaan verkkosivujen ulkoasua haluttuun muotoon. CSS:n avulla voidaan määritellä esimerkiksi tekstin väri, fonttien tyyli, taustavärit tai -kuvat, sarakkeiden mitoitus ja paljon muuta. (Tutorialspoint 2017a.) CSS koostuu tyylisäännöistä, jotka koostuvat

kolmesta osasta: valitsin (selector), ominaisuus (property) ja arvo (value). Valitsin on HTML-tagi, jossa määriteltyä tyyliä halutaan käyttää. Ominaisuus on HTML-tagin attribuutti tyyppi, jotka muunnetaan CSS ominaisuuksiksi. Attribuutteja voi esimerkiksi olla väri tai reunus. Arvo on se asia, joka määritellään omainaisuuksille, esimerkiksi väri-ominaisuudelle voidaan antaa arvo punainen. CSS tyylisäännön syntaksi, voidaan ilmaista seuraavasti: selector{ property: value}. (Tutorialspoint 2017b.) Yleinsä CSS yhdistetään HTML:n sekä XHTML:n kanssa ja CSS:stä on olemassa kolme versiota, joista käytän tässä työssä viimeisintä versiota eli CSS3:sta

Javascript on dynaaminen ohjelmointikieli, jolla pyritään tekemään interaktiivisia (vuorovaikutteisia) sivuja. Tämän kielen ohjelmia kutsutaan skripteiksi, jotka voidaan kirjoittaa suoraan HTML-dokumenttiin script-tagien sisälle tai skriptit voidaan sisällyttää erilliseen tiedostoon, johon HTML-dokumentti viittaa. Selaimen lisäksi Javascriptiä pystyy ajamaan myös palvelimilla ja millä tahansa muulla laitteella, jossa on ohjelma Javascript engine. Selaimessa Javascript voi tehdä kaikkea, mikä liittyy verkkosivun manipulointiin tai vuorovaikutukseen käyttäjän ja verkkopalvelimen kanssa. Esimerkiksi Javascript pystyy luomaan uutta sisältöä HTML-dokumenttiin, muokkaamaan tyylejä, kysyä kysmyksiä käyttäjältä ja näyttää viestejä. (Kantor 2017.)

2.4 Kirjastot

Käyn tässä osiossa lyhyesti läpi Onsen UI ja D3- sekä jQuery-kirjastot, jotka ovat tärkeimpiä kirjastoja käyttöliittymän toteutuksessa. Onsen UI on avoimen lähdekoodin Javascript-kirjasto, joka on suunniteltu mobiili sovellusten toetuttamiseen. Onsen UI koostuu kolmesta kerroksesta, jotka tarjoavat useita komponentteja käyttöliittymän toteutukseen. Onsen UI:n ensimmäinen kerros css-komponentit, jotka on kirjoitettu cssnext-kielellä, seuraava kerros on web-komponentit, jotka on kirjoitettu natiivi Javascriptillä ja viimeisen kerroksen avulla Onsen UI pystyy sitoutumaan muihin suosittuihin kehyksiin, kuten Vue.j, AngularJS 1, AngularJS 2+ ja React. Onsen UI sopii hyvin hybridi sovellusten kehittämiseen Cordovan kanssa, jonka vuoksi käytän kyseistä kirjastoa myös nykyisessä sovelluksessa. Onsen UI mallintaa myös Android ja IOS standardeja, jotka

helpottavat mobiilisovelluksen tekemistä ja sovittamista Android- ja IOS-alustoille. (Monaca x Onsen UI Team 2017.)

D3 (Data-Driven Documents tai D3.js) on Javascript-kirjastoa, jota voidaan käyttää datan visualisointiin sekä visuaalisten toiminnallisuuksien toteuttamiseen web-standardien avulla, kuten SVG:n, CSS:n ja HTML:n. D3-kirjaston hyötyjä on, että se on erittäin nopea, tukee suuria tietomääriä ja dynaamista käyttäytymistä. Datan avulla D3 mahdollistaa myös DOM:in (Document Object Model) manipuloinnin, jonka avulla saadaan sovellukseen joustavuutta ja dynaamisia ominaisuuksia. (Bostock 2017.)

JQuery on hyvin tunnettu ja paljon käytetty ilmainen Javascript-kirjasto. JQueryn on luonut John Regins vuonna 2006, joka suunnitteli jQueryn yksinkertaistamaan HTML-asiakaspuolen komentosarjoja eli skriptejä. JQueryn avulla pystytään esimerkiksi manipuloimaan HTML-dokumentteja tehokkaasti sekä toteuttamaan tapahtuman käsittelyjä ja animaatioita myös AJAX:in (Asynchronous Javascript And XML) käyttö on yksinkertaisempaa kirjaston tarjoaman heppokäyttöisen rajapinnan avulla, mikä toimii useimmissa selaimissa. (Bibeault, Katz & De Rosa 2015, 3.) AJAX on tekniikka, jolla kyetään luomaan nopeita ja dynaamisia verkkosivuja. AJAX mahdollistaa verkkosivujen päivittämisen asynkronisesti lähettämällä pienen määrän dataa palvelimelle, jolloin verkkosivun osia kyetään päivittämään ilman, että sivu uudelleen ladataan. (Refsnes Data 2017b.) JQuery-kirjaston pieni koko, nopeus ja monipuolisuus ovat sen vahvuuksia, jonka vuoksi myös itse suosin kirjaston käyttämistä. JQuery koodin tunnistaa $-merkistä, joka mahdollistaa jQuery-koodin määrittämisen ja käyttämisen.

3 KÄYTTÖLIITTYMÄN SUUNNITTELU JA LÄHTÖKOHDAT

Työn toimeksiantaja oli jo aiemmin kehitellyt sovellukselle käyttöliittymän, jota käytin esimerkkinä tehdessäni uutta käyttöliittymää sovellukselle. Aiempi versio käyttöliittymästä antoi myös hyvää tietoa siitä, mitä henkilön tietoja uudenkin käyttöliittymän tulisi sisältää. Olin aiemmin tehnyt työhön liittyviä tietojenhakua esimerkiksi rokotteista ja allergioista, jotka koottiin excel-taulukkoon ja ne tullaan toteuttamaan sovelluksessa JSON-muodossa. Työlle oli annettu asiakkaitten puolesta hyvin vapaat kädet ja aikataulua ei työlle ollut. Työn valmistuminen oli siis minusta kiinni, koska työssä käytetyt teknologiat eivät olleet minulle tuttuja, niin jouduin aluksi käyttämään aikaani tekniikoiden opetteluun ja tutustumiseen.

3.1 Työn kuvaus ja suunnitelmat

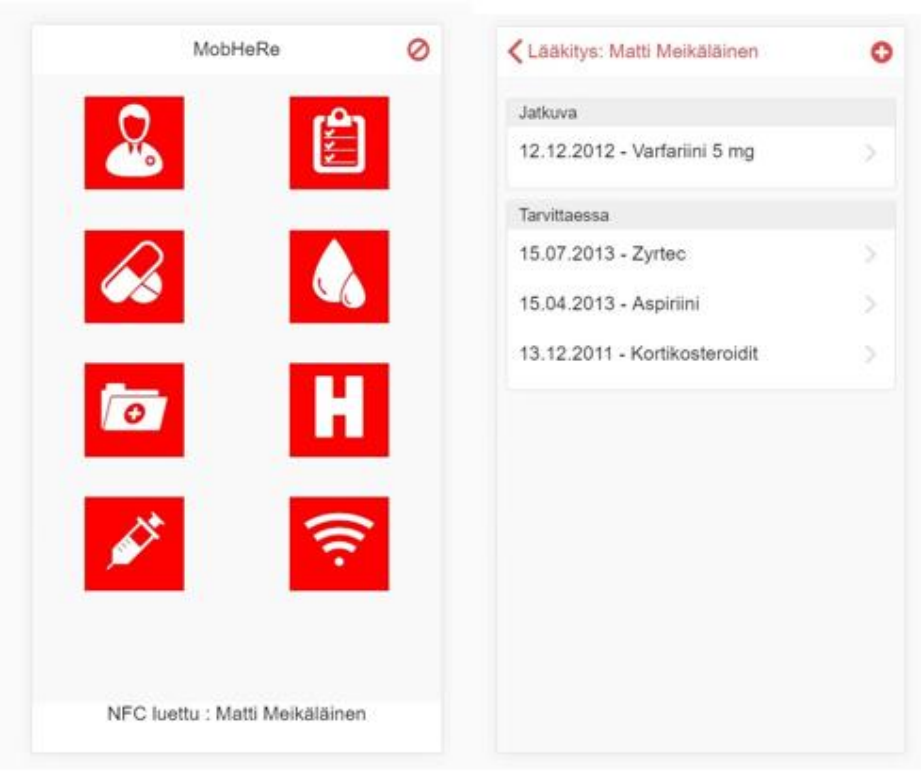
Suunnitelmissa oli, että käyttöliittymä edelleen suunniteltaisiin vain mobiililaitteille ja siinä tulisi näkyä selkeästi henkilön terveystiedot. Käyttöliittymän ei myöskään tulisi sisältää paljoa ylimääräistä tekstiä tai selitystä, koska sovellus tulee ammattilaisten käyttöön niin oletettiin, että he ymmärtävät oman ammattisanastonsa, esimerkiksi lääkenimet. Asiakkaiden toiveiden mukaan sovellukseen suunniteltiin aloitusnäytölle aikajanoja, joihin pystyy sisältämään merkintöjä. Suunnitelmissa myös oli, että merkintöjä klikkaamalla pystyy nopeasti näkemään, esimerkiksi myönnetyt lääkkeet sekä todetut sairaudet. Suunnitelmisssa oli siis sijoittaa tärkeimmät terveystiedot sovelluksen aloitusnäyttöön, kuten sairaus-, lääke-, allergia- ja rokotustiedot, jotka olisi helppo tarkastaa myös hätätilanteissa, ilman turhia klikkauksia tai sivun vaihtoja. Kaikkien aikajanojen tuli myös mahtua yhdelle sivulle, niin että käyttäjän ei tarvitsisi liikuttaa sivua.

Aikajanojen suunnittelussa oletettiin, että aikajanoihin merkityt kohdat ilmenisivät ympyräkuvioina ja aikajana olisi henkilön elinkaaren pituinen, joten aikajanoista

pitäisi tehdä zoomattavia. Suunniteltut ympyräkuviot puolestaan olisivat väriltään erilasia riippuen esimerkiksi sairauden tai allergian vakavuudesta, jolloin ympyrän väriksi voisi asettaa esimerkiksi kirkkaan punainen, jolloin se olisi helppo löytää muista merkinnöistä, joille suunniteltaisiin neutraalimpi väri. Sairauden ja allergian vakavuuden määrittelyyn suunnittelin toteuttaa janan, johon toteuttetaisiin yksinkertainen asteikko, jonka avulla terveydenalan ammattilainen kykenisi antamaan oman arvionsa sairauden tai allergian vakavuudesta.

3.2 Aiempi versio

Mobiilihenkilörekisteri-sovellukselle on toimeksiantaja jo aiemmin kehittänyt käyttöliittymän, jonka uudistaminen on päätavoitteeni työtä tehdessäni. Aiempi mobiilisovelluksen käyttöliittymä on myös toteutettu Apache Cordova -teknologialla ja sen toiminnallisuudet vastasivat perinteistä mobiilisovellusta. Kuvassa 2 vasemalla ollaan kuvattu, sovelluksen aloitusnäkymä, jossa liikutaan eri sivujen välillä koskettamalla ikoneita (kuvakkeita). Ikoneiden takaa löytyy tietoa esimerkiksi henkilön lääkityksestä, rokotuksista ja allergioista. Kuvassa 2 oikealla ollaan kuvattu henkilön lääketietoja sisältävä sivu, joka löytyy lääke-ikonin takaa. Aiemman version toteutuksessa ollaan käytetty Onsen UI sekä Angular JS -kirjastoa, joista ainakin Onsen UI -kirjastoa tulen käyttämään uuden käyttöliittymän prototyypissä. Suurimpia tulevia muutoksia tulee käyttöliittymän aloitusnäkymään, sillä vaikka vanhakin käyttöliittymän toiminnot olivat hyvin toteutettuja ja yksinkertaisia, niin sovellusta pyytänneet asiakkaat halusivat toteuttaa käyttöliittymän, jossa heille kriittinen tieto näkyisi käyttöliittymän pääsivulla.

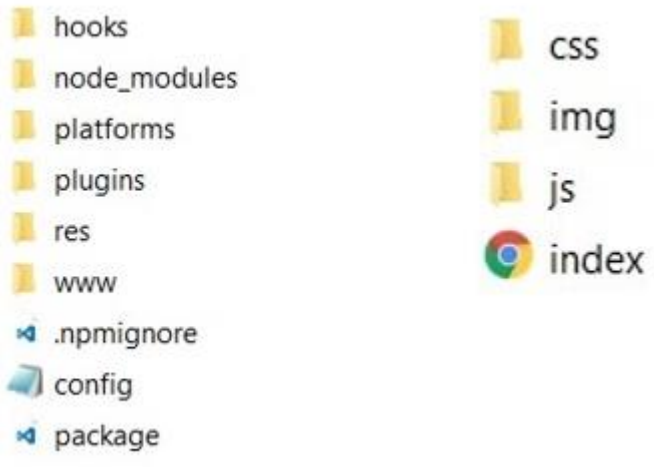


Kuva 2. Vasemalla Mobhere-sovelluksen aloitusnäkymä ja oikealla kuvattuna lääkitystiedot.

4 TOTEUTUS

Tässä luvussa käyn läpi käyttöliittymän prototyypin toteutusta. Esittelen pääasiassa Javascript-koodia, koska sillä toteutetaan toiminnallisuudet käyttöliittymässä ja vaativat sen vuoksi eniten läpikäyntiä. Uuden käyttöliittymän toteutus lähti liikkeelle asentamalla Cordova ja luomalla uusi Cordova projekti komentokehotteessa, jonka päälle käyttöliittymää aletaan rakentamaan. Kuvassa 3 ollaan kuvattu Cordova-projektin luomat kansiot, kun projekti ollaan luotu komentokehotteessa. Kuvan 3 oikeallapuolella ollaan kuvattu avattu www-kansio, johon tullaan sisällyttämään kaikki projektin CSS-, Javascript ja HTML-tiedostot eli tämä kansio on prototyypin ydin. Kuvan index.html-tiedostoon tullaan, sisällyttämään kaikki prototyypin HTML-koodi ja css-kansioo sisällytetään kaikki CSS-tiedostot sekä js-kansioon luonnolliseti sisällytetään kaikki Javascript-tiedostot. Projektiin ollaan valmiiksi tallennettu android- ja iso-alustat, koska lopullinen sovellus tullaan ajamaan ainakin

Android ja iOS -mobiililaitteilla, joten projektiin on hyvä asettaa kyseisille alustoille tuet. Alustat ollaan sijoitettu kuvan platforms-kansioon. Projektiin ajettiin valmiiksi myös NFC-plugini plugins-kansioon, NFC-toimintojen toteuttamista varten, joita ei nyt toteuteta tässä prototyypissä. Prototyypin rakentamiseen aion käyttää Visual Studio Code -koodieditoria. Käyttöliittymässä käytetään Onsen UI -kirjastoa, joka helpottaa toiminnallisuuksien toteuttamista ja käyttöliittymän ulkoasun määrittämisessä. Onsen UI -kirjaston elementit pystyy tunnistamaan ons-etuliitteestä. Keskityn toteutuksessa enimmäkseni käyttöliittymän pääsivun esittelyyn, koska pääsivu sisältää suurimmat toiminnallisuudet ja on muutenkin tärkeimpiä kohtia prototyypin kehittelyssä.



Kuva 3. Vasemalla projektin kansiot yleisnäkymässä ja oikealla avattu www-kansio.

Käyttöliittymän kehittelyn alussa, ensimmäisiä vastaan tulevia ongelmia oli, että miten toteuttaisin käyttöliittymän aikajanat ja kuinka tekisin niistä zoomattavat? Aikajanat päätettiin toteuttaa D3 Javascript-kirjastoa käyttämällä, koska kirjasto sopii hyvin visuaalisen datan tuottamiseen, joten netistä löytyvien esimerkkien ja ohjeiden avulla, saatiin aikajanat toteutettua käyttöliittymään. Kuvassa 4 on näytetty koodi, jolla aikajanat kehiteltiin. Aikajanojen kehitys lähti siis siitä, että luotiin D3-kisrjastoa hyödyntäen SVG-elementti (Scalable Vector Graphics) eli elementti, jota käytetään vektoripohjaisen grafiikan määrittelyyn. SVG määrittelee grafiikan XML-muodossa ja SVG:n sisältämää grafiikkaa pystyy zoomaamaan ja animaatioida. (Refsnes Data 2017c). SVG-elementtiin määriteltiin skaalautuvuus (scale) henkilön syntymäpäivästä tähän päivään. Samassa koodissa määritellään tulevan janan ja

janan tekstin zoomattavuus D3-kirjaston zoom-komentoa käyttäe. Kuvasta 4 näkee, että zoomaus-käytöstä koskevassa koodin kohdassa kutsutaan circleZoom-funktiota, jonka avulla myös merkinnöistä saadaan zoomattavia. Seuraavaksi toteutetaan koodissa aikajana, johon upotetaan edellä luotu zoom-tapahtuma. Lopuksi lisätään tekstit luotuun aikajanaan, jotta saadaan käyttäjälle päivämäärät näkyviin.

```
var svg = d3.select(timeline)
            .append("svg")
            .attr("width", w)
            .attr("height", h);

var scale = d3.time.scale()
              .domain([mindate, maxdate])
              .range([0, w])
              .nice();

var xaxis = d3.svg.axis().scale(scale)
                .orient("bottom")
                .tickFormat(finFormat)
                .ticks(5);

var zoom = d3.behavior.zoom()
            .on("zoom", function(){
            svg.select("g").call(xaxis).selectAll("text").style("font-size", "14px");
            circleZoom();
            }).x(scale);

var rect = svg.append("rect")
  .attr("x", 0)
  .attr("y", 0)
  .attr("width",w - 100)
  .attr("height", h)
  .attr("opacity", 0)
  .call(zoom);

  svg.append("g")
    .attr("class", "xaxis")
    .call(xaxis)
    .selectAll("text")
    .style("font-size", "14px");
```

Kuva 4. Koodi aikajanojen toteutuksesta D3-kirjastoa käyttäen.

Aikajanojen toteutuksen jälkeen alettiin toteuttamaan muita käyttöliittymän kohtia, kuten alavalikko, joka toteutettiin Onsen UI -kirjaston ons-tabbar-elementtiä käyttäen. Alavalikon avulla päästään muihin henkilön tietoihin esimerkiksi yhteystietoihin. Siirryttävät sivut ollaan sisälletty ons-template-elementin sisälle, joka mahdollistaa sivun vaihtumisen käyttöliittymässä, joten alavalikon pystyi tekemään puhtaasti Onsen UI -kirjaston avulla ilman, että Javascriptillä olisi pitänyt tuottaa joitain toiminnallisuuksia. Kuvassa 5 ollaan kuvattu alavalikon toteuttava koodi. Alavalikon kuvakkeet ovat peräisin Font Awesome -kirjastosta, joka tarjoaa paljon eri ikoneita moneen käyttöön. Font Awesome -ikonit tunnistaa fa-etuliitteestä.

```
<ons-tabbar>
  <ons-tab page="index.html"  icon="fa-hospital-o" active="true"></ons-tab>
  <ons-tab page="info.html" icon="fa-medkit" id="blood_info"></ons-tab>
  <ons-tab page="about.html" icon="fa-clipboard"></ons-tab>
  <ons-tab page="access.html" icon="fa-wifi"></ons-tab>
</ons-tabbar>
```

Kuva 5. Alavalikon toteuttava koodi.

Tuli myös ratkaista ongelma, miten merkintöjä lisättäisiin aikajanoihin ja kuinka saataisiin aikajanojen päivämäärät standardisoitua Suomen standardeihin? Päivämäärien standardisoinnin toteuttava koodi on kuvattuna kuvassa 6. Kuvan 6 koodista näkee, että aikajanoihin määritellään suomenkielinen formaatti ja päivämäärän muoto määritellän suomalainen päivämäärä standardi, sillä muuten päivämäärät esitettäisiin oletusmuodossaan eli ISO 8601 standardin mukaisesti, jolloin päivämäärät esitetään vuosi-kuukausi-päivä mukaisessa järjestyksessä. Päivämäärät ja ajat saadaan haettua käyttämällä Javascriptin metodeja: getHours(), getDate(), getMonth() jne. D3-kirjaston timeformat-metodia käyttämällä pystytään muokkaamaan päivämäärät haluttuun muotoon, kuten kuvasta 6 näkee finformat-kohdassa, että olen asettanut päivämäärien esittämisen suomalaisiin standardeihin. Päivämäärien muoto määritellään aikajanoihin samalla, kun aikajanoihin asetettiin niiden skaalattavuus.

```
var Fin = d3.locale({
  "dateTime": "%A, %-d. %Bta %Y klo %X",
  "date": "%-d.%-m.%Y",
  "time": "%H:%M:%S",
  "periods": ["a.m.", "p.m."],
  "days": ["Sunnuntai", "Maanantai", "Tiistai", "Keskiviikko", "Torstai", "Perjantai", "Lauantai"],
  "shortDays": ["Su", "Ma", "Ti", "Ke", "To", "Pe", "La"],
  "months": ["Tammikuu", "Helmikuu", "Maaliskuu", "Huhtikuu", "Toukokuu", "Kesäkuu", "Heinäkuu", "Elokuu", "Syys
  "shortMonths": ["Tammi", "Helmi", "Maalis", "Huhti", "Touko", "Kesä", "Heinä", "Elo", "Syys", "Loka", "Marras"
});

var finFormat = Fin.timeFormat.multi([
  ["%H:%M", function(d) { return d.getMinutes(); }],
  ["%H:%M", function(d) { return d.getHours(); }],
  ["%a %d", function(d) { return d.getDay() && d.getDate() != 1; }],
  ["%d %b", function(d) { return d.getDate() != 1; }],
  ["%B", function(d) { return d.getMonth(); }],
  ["%Y", function() { return true; }]
]);
```

Kuva 6. Päivämäärien muokkaus Suomen standardeihin

Datan lisäys aikajanoihin on toteutettu parilla eri tavalla. Ensimmäinen datan lisäys tapahtuu valmiiksi määritellyistä päivämääristä, jotka on asetettu JSON-objektiin.

Datan lisäyksen koodi on kuvattuna kuvassa 7. Kuvan 7 koodissa luodaan uusi circle-elementti valitsemalla koko aikajana ja asettamalla circle-elementti valmiiksi määriteltyn datan antamiin paikkoihin. Valmiin datan lisäys aikajanaan tapahtuu D3-kirjaston data-metodin avulla, joka suosii datan lisäystä taulukko tai objektina muodossa. Valmiiksi asetettuihin merkintöihin lisätään kaikkiin myös klikkaus-funktio D3-kirjaston avulla, jonka koodi on kuvattuna kuvassa 8. Klikkaus-funktion toteutus lähtee liikkeelle siitä, että haetaan svg-elementistä div-elementti, jonka id:n mukaan haetaan kaikki tietyt circle-elementit, jolloin saadaan tietyt Onsen UI -kirjaston ons-dialog-elementit avautumaan tiettyjä merkintöjä klikkaamalla. Prototyypissä Klikkaus-funktio on upotettu valmiiksi määriteltyihin merkintöihin, vain havainnollistamaan merkintöjen toimintoja. Avautuvat ons-dialog-elementit sisältävät esimerkkidataa infosta, jolla havainnollistetaan suunniteltua rakennetta ons-dialog-elementin sisällä.

```
function drawCircles(d) {
  var events = svg.selectAll("rect.item").data(d);
  events.enter()
    .append("circle")
    .attr("class", "item")
    .attr("cx", function(d) {
      return scale(d);
    })
    .attr("r", 9);

  events.exit()
    .remove();
}
```

Kuva 7. Valmiiksi määriteltyjen merkintöjen asettaminen aikajanaan.

```
var clicker = svg.selectAll("#timeline circle");
clicker.on("tap", function(i){ showDialog("sickness-info");
var self = d3.select(this);
});

clicker = svg.selectAll("#timeline2 circle");
clicker.on("tap", function(i){ showDialog("medicine-info");
var self = d3.select(this);
});

var clicker = svg.selectAll("#timeline3 circle");
clicker.on("tap", function(i){ showDialog("allergy-info");
var self = d3.select(this);
});

var clicker = svg.selectAll("#timeline4 circle");
clicker.on("tap", function(i){ showDialog("vaccine-info");
var self = d3.select(this);
});
```

Kuva 8. Klikkaus-funktion koodi.

Uuden datan lisäys on puolestaan kuvattu kuvassa 9, jossa datan lisäys tapahtuu D3-kirjaston datum-komentoa käyttäen. D3-datum sopii hyvin prototyyppiin käytettäväksi, sillä se sitoo datan suoraan elementtiin, tässä tapauksessa aikajanaan, ja sillä on helpompi lisätä yksittäistä dataa. Merkinnän lisäys tapahtuu taas luomalla circle-elementti ja asettamalla se annetun datan mukaan aikajanaan. Data, jonka mukaan merkintä asetettaan on päivämäärä, joka annetaan samalla kun käyttäjä lisää merkinnän. Lisäämällä merkintöjä tekemään funktioon erilaisia if-else-ehtoja, saadaan merkintöihin määriteltyä eri värejä ehdosta riippuen. Kuvan 9 koodista pystyy näkemään merkintöihin sisälletyn klikkaus-funktion, jonka avulla saadaan klikkaus-tapahtuma upotettua uuteen merkintää, jolloin se kykenee tekemään muutoksia sekä näyttämään ons-dialog-elementin käyttöliittymässä. Osassa merkintöjen lisäyksessä käytetään jQuery-kirjaston each-metodia, jolla käydään läpi lisättyjä elementtejä, sillä esimerkiksi lääkeitä lisätessä saattaa ollaa merkattuna ylös jokin lääkitys, joka lisätään mukaan uuden lääkemerkinnän kanssa.

```
svg.append("circle")
.datum(pd)
.attr("class", "item")
.on("click", function(){
  if(timeline === "#timeline"){
    $("#inf").empty();
    str = "";
    $('.add-sickness').each(function(){
      var text = $(this).text();
      str += "<ons-list-item><ons-col>"+text+"</ons-col><ons-col>"+formatdate+"</ons-col></ons-
    });
  $("#inf").append(str);
  showDialog("sickness-info");
  }
  if(timeline === "#timeline2"){
      $("#inf2").empty();
      str = "";
      if(document.getElementById("ongoing").checked){
      $('.add-medicine').each(function(){
        var text = $(this).text();
        str += "<ons-list-item><ons-col>"+text+"</ons-col><ons-col class='right'>"+formatdate+"
      });
      $("#inf2").append(str);
    }
     else{
      $("#inf2").append(dialog);
     }
      showDialog("medicine-info");
  }
  if(timeline === "#timeline3"){
      $("#inf3").empty();
      $("#inf3").append(dialog);
      showDialog("allergy-info");
  }
  if(timeline === "#timeline4"){
      $("#inf4").empty();
      $("#inf4").append(dialog);
      showDialog("vaccine-info");
  }
})
.attr("cx", function(pd) {
  return scale(pd);
```

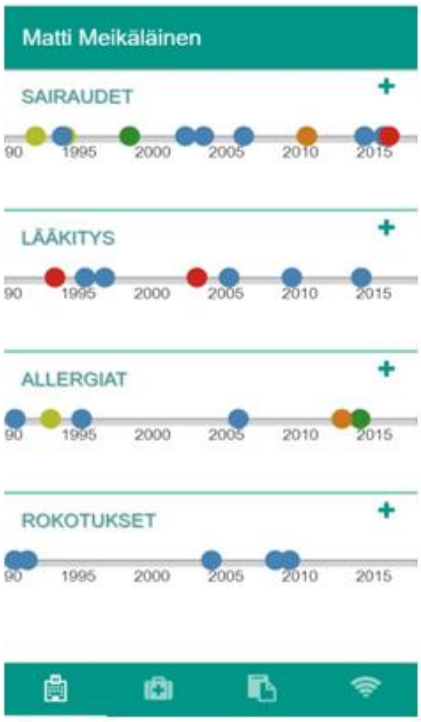
Kuva 9. Koodi uuden merkinnän määrittelystä aikajanaan.

5 LOPPURATKAISU

Tässä luvussa käyn läpi prototyypin lopputulosta. Käyn myös läpi joitakin kohtia, jotka jäivät vielä toteuttamatta tai vaatisivat tulevaisuusessa parantelua. Prototyypin lopullinen pääsivu on kuvattuna kuvassa 10, käyttöliittymä on kuvassa Android-näkymässä. Kuvassa 10 näkyvät eri merkintöjen värit kertovat, kuinka tärkeitä merkinnät ovat, esimerkiksi sairauksien vakavuus määritellään värit väliltä sinisestä punaiseen ja lääkke-merkintöjen väri kertoo niiden jatkuvuudesta. Kuvassa 8 näkyvää plus-ikonia klikkaamalla saadaan esiin ons-dialog-elementti, jonka avulla

voidaan lisätä aikajanaan uusi merkintä. Merkinnän paikka määräytyy aina päivämäärän mukaan, kun merkintä on lisätty

Prototyypissä ei olla vielä määritelty, miten merkinnän väri määräytyy, jos lisätään useampi esimerkiksi allergia kerralla. Kuvittelisin kuitenkin, että merkinnän väri määräytyy suurimman arvion saaneen allergian tai sairauden mukaan. Lääkkeiden merkinnässä, mikäli lääkitys on jatkuva niin merkintä on silloin punainen (kuvattuna kuvassa 10 lääke kohdassa), joten pitäsi vielä selvittää, että tulisiko silloin kaikkien merkintöjen olla punaisia, niin kauan kuin lääkitys jatkuu. Mahdollista olisi myös muuttaa koko värien määrittelyä merkintöihin, mutta siihen tarvittaisiin, minusta ammattilaisen mielipide, jolloin saataisiin tieto siitä, miten he haluaisivat korostaa merkintöjä.

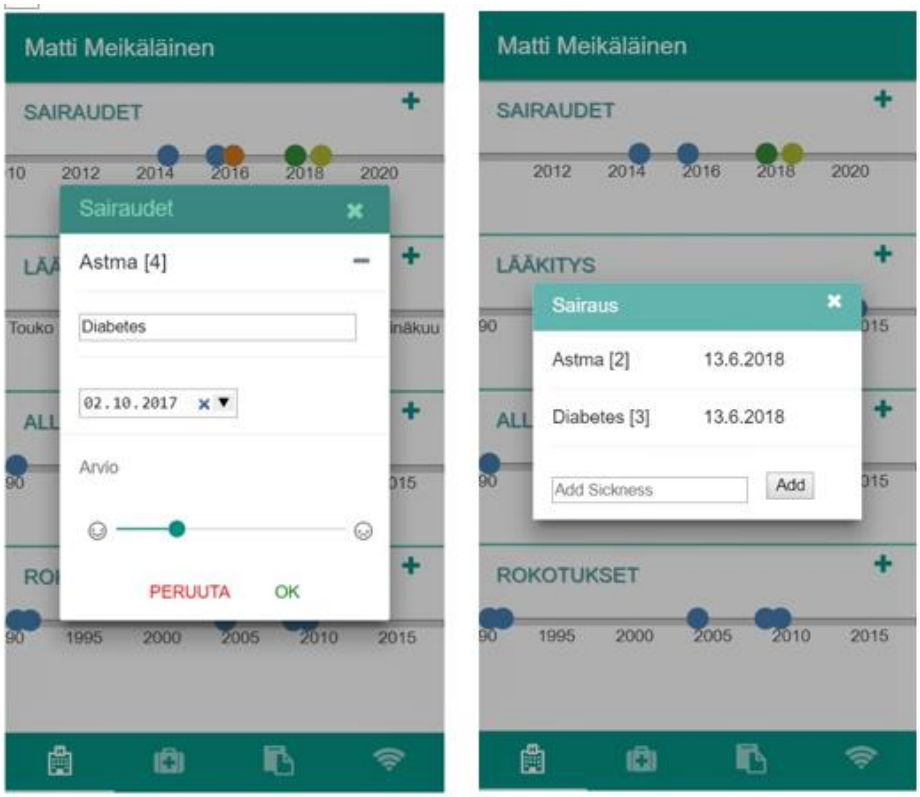


Kuva 10. Kuvassa on käyttöliittymän pääsivu.

Uusia merkintöjä lisäävä ons-dialog-elementti on kuvattuna kuvassa 11 vasemmalla. Vasemman puoleisen kuvan Ons-dialogissa lisätään uusi sairaus. Ons-dialogin yläreunaan on lisätty aiemmin todetun sairauden nimi ja sairauden arvio hakasulkeiden sisällä. Sairauden pystyy poistamaa nimen vieressä olevasta miinus-ikonista esimerkiksi, silloin kun sairautta ei enää ole, muuten tämä sairaus lisätään

mukaan uuden todetun sairauden lisäksi. Tekstikenttään määritellään sairauden nimi ja päivämäärä-tekstikentään määritellään päivämäärä, johon merkintä sijoittuu aikajanalla. Mikäli sairaus-tekstikenttä on tyhjä ja aiempaakaan sairautta ei ole, niin merkintää ei silloin lisätä aikajanaan. Oletuksena kaikissa päivämäärä tekstikentissä on kuluvan päivän päivämäärä. Kuvan arvio kohdassa pystyy liukusäätimen avulla määrittämään merkinnän värin. Liukusäädintä käytetään merkinnän värin määrittelyyn sairauksia ja allergioita lisätessä, koska niissä on mahdollista antaa arvio vakavuudesta. Liukusäätimen arvot ovat väliltä 1-5, jossa 1 on postiivinen arvio ja 5 negatiivinen arvio. Lääkkeissä merkinnän värin määrittelyyn käytetään vain jatkuva valintaruutua (checkbox), jolloin merkintä saa värikseen punaisen. Muuten sairauden merkintä värittyy oletusväriksi, jolloin oletetaan, että lääkitys on kertaluonteinen. Rokotteille ei olla kehitelty mitään muuta väri variaatiota kuin merkintöjen oletusväri.

Kuvassa 11 oikealla näkyy ons-dialog-elemetti, joka ilmestyy kun klikataan merkintää. Ons-dialogit sisältävät esimerkiksi kuvassa 11:n oikeanpuoleisessa näkymässä näkyvän sairauden nimen, arvio vakavuudesta ja päivämäärän jolloin merkintä ollaan tehty. Mikäli kyseessä on sairaus, joka on jo aiemmin todettu niin merkintää lisätessä se saa saman päivämäärän kuin uusi lisätty sairaus. Kuvassa näkyvä dialogin alalaidassa näkyy tekstikenttä, jonka tarkoituksena on, että merkintää pystyisi lisäämään esimerkiksi sairaus kun merkintä ollaan jo tehty. Muidenkin osioiden merkintöjen sisältö noudattaa samaa mallia.

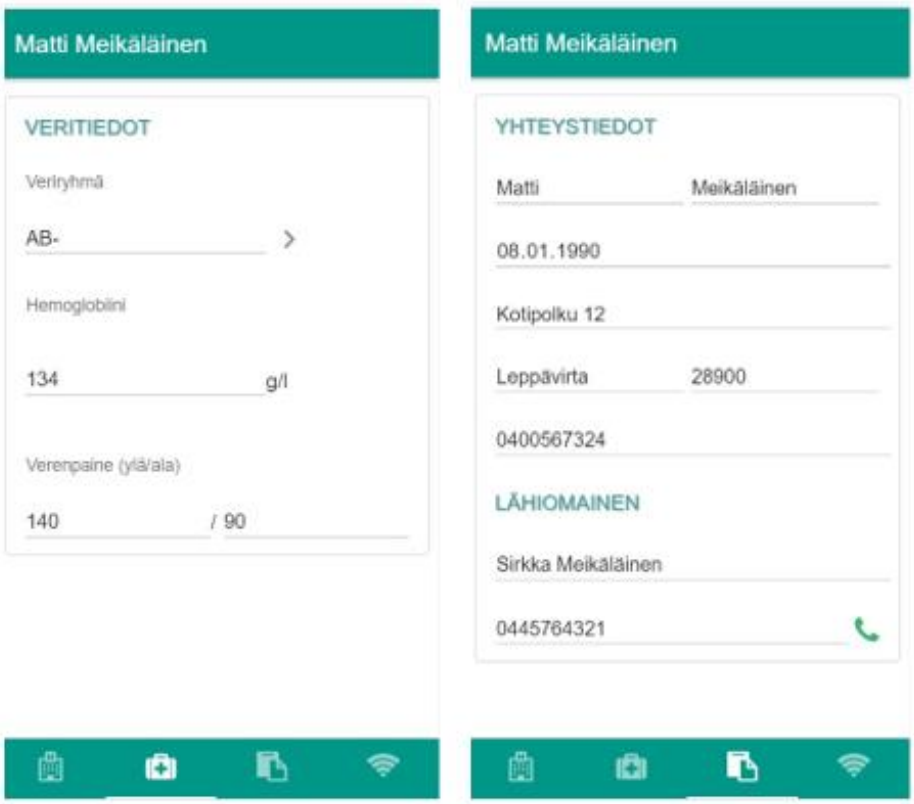


Kuva 11. Vasemalla uuden sairauden lisääminen ja oikealla sairaus-merkinnän sisältö.

Prototyyppiin jatkokehittelyssä pitäisi määritellä, sellainen toiminto, joka estää käyttäjää lisäämästä duplikaatteja eli samoja esimerkiksi sairauksia samaan merkintään. Jatkossa pitäisi myös kehitellä merkintöjen lisäykseen toiminto, että käyttäjä kykenisi lisäämään useamman uuden arvon yhdellä kerralla samaan merkintää ilman, että käyttäjän tarvitsisi käydä klikkaamassa vasta luotua merkintää lisätäkseen siihen arvoja. Käyttöliittymässä pitäisi myös muokata päivämäärien ja nimien valintaa, koska tällä hetkellä päivämääriä valitaan tavallisesta HTML:n date-tyyppinen tekstikenttä ja merkintöjen arvoja haetaan HTML:n datalist-tyyppisestä tekstikentästä. Datalist on yksi HTML:n tekstikenttä-tyyppi, joka sisältää valmiiksi määriteltyjä arvoja, joita selain ehdottaa kun tekstikentään kirjoitetaan. Nämä tekstikentät eivät sovellu puhelimien näytölle, eivätkä ole kovinkaan käyttöystävällisiä kosketusnäytöllisissä puhelimissa.

Käyttöliittymän alavalikon avulla siirrytään muille sivuille. Sivujen ulkonäön päätin pitää yksinkertaisena ja mahdollisimman informatiivisena. Sivut ovat rakenteiltaan yksinkertaisia, eivätkä poikkea paljoa edellisen käyttöliittymän samankaltaisten

sivujen sisällöstä. Yhteystieto ja veritieto sivujen toteutuksessa käytin Onsen UI -kirjaston omia tekstikenttiä eli ons-input-tageja, koska tietojen muokkaaminen onnistuu käyttäjältä siten helpommin. Veritiedot-sivu on kuvattuna vasemalla kuvassa 12 ja se sisältää tällä hetkellä vain henkilön veritiedot. Kuvan veriryhmä kohdassa näkyvän nuoli-ikonin kautta, pystyy valitsemaan veriryhmän valmiiksi laadituista vaihtoehdoista. Veritiedot-sivun sisältöön en ole tehnyt mitään muutoksia, joten se on tiedoiltaan samanlainen kuin aikaisemmassa käyttöliittymässä. Sivu voisi mielestäni sisältää myös muita terveystietoja esimerkiksi paino ja pituus, mutta en tiedä onko se sovelluksen käyttäjille merkittävää tietoa. Kuvassa 12 oikealla on kuvattu yhteystieto-sivu, joka sisältää henkilön yhteystietojen lisäksi lähiomaisen yhteystiedot. Kuvan 12:n yhteystieto sivun alakulmassa on puhelin-ikoni, jolle ei olla vielä toiminnallisuutta toteutettu, mutta ajatuksena oli toteuttaa ikoni niin, että sitä painamalla pystyisi suoraan soitamaan omaiselle tai sitten kopioimaan omaisen numero ja sitä kautta hänelle soittamaan. Yhteyssivu sisältää tiedon siitä, onko sovellus muodostanut yhteyden nfc-laitteeseen. Tämä sivu ei sisällä mitään kunnon toiminnallisuuksia, joten sivu toimii prototyypissä lähinnä esimerkkinä sivun sisällöstä.



Kuva 12. Vasemalla on kuvattuna veritiedot-sivu ja oikealla yhteystieto-sivu.

6 POHDINTA

Prototyypin toteutus käyttöliittymästä oli opettavainen kokemus. Uusien teknologioiden käyttäminen oli antoisaa ja opin paljon uuttakin prototyyppiä tehdessäni. Olen erityisen tyytyväinen aikajanojen toteutukseen, koska niiden toteutus oli uudessa käyttöliittymässä tärkein kohta sekä suurin uudistus edellisestä käyttöliittymästä. Koska kyseessä on vasta prototyyppi, niin käyttöliittymän ulkoasu tulee todennäköisesti vielä muuttumaan valmiissa sovelluksessa, mutta perusajatus käyttöliittymän toiminnasta tulisi olla sama.

Käyttöliitymän prototyyppi onnistui mielestäni melko hyvin, mutta suoritukseni olisi ollut tehokkaampi, jos minulla olisi ollut aikaisempaa kokemusta Apache Cordova teknologiasta ja D3-kirjastosta, joiden opettelu vei minulta hiukan liikaa aikaa työtä tehdessäni. Kyseisten tekniikoiden opetteluun kuluneen ajan olisin mielummin käyttänyt suunnitelmien ja toteutuksen hiomiseen, jolloin prototyyppikin olisi saanut lisää ominaisuuksia ja toiminnallisuuksia. Olisin myös toivonut enemmän tietoa siitä, mitä käyttöliittymässä tulisi näkyä, sillä prototyyppin sisältö perustuu hyvin paljon aiemman käyttöliittymän sisältöön sekä omaan näkemykseeni, siitä mikä voisi olla tarpeellista tällaisessa sovelluksessa. Joten ammattilaisen näkökulmaa olisin kaivannut käyttöliittymää tehdessäni, jolloin olisin tiennyt, mikä on heille ensisijaista tietoa, jota pitäisi korostaa käyttöliittymässä.

Kaiken kaikkiaan on ollut hienoa olla mukana tällaisessa mielenkiintoisessa projektissa. Tuleva sovellus tuntuu hyvin tarpeelliselta ja ajankohtaiselta, jonka vuoksi toivonkin, että toimeksiantajani vie projektin loppuun ja sovellus saataisiin valmiiksi.

LÄHTEET

Bar Code Graphics, Inc. 2017. What is RFID? Viitattu 01.06.2017. http://www.epc-rfid.info/rfid

Bibeault, B., Katz, Y. & De Rosa, A. 2015. jQuery in Action, Third Edition. Shelter Island: Manning Publications Co. Viitattu 25.08.2017. https://manning-content.s3.amazonaws.com/download/4/7205cce-afdc-4d67-bb92-764e225fbeb2/Sample_Ch_01.pdf

Bostock, M. 2017. Introduction. Viitattu 28.08.2017. https://d3js.org/

Camden, R. 2016. Apache Cordova in Action. Shelter Island: Manning Publications Co. Viitattu 30.06.2017. https://github.com/liubo404/itebooks/blob/master/pdf/Apache%20Cordova%20in%20Action.pdf

GitHub, Inc. 2017. Home. Viitattu 28.08.2017. https://github.com/d3/d3/wiki

Kantor, I. 2017. An Introduction to Javascript. Viitattu 22.09.2017. https://javascript.info/intro#what-is-javascript

Monaca x Onsen UI Team 2017. What is Onsen UI? Viitattu 26.08.2017. https://onsen.io/v2/guide/index.html#what-is-onsen-ui

Refsnes Data 2017a. HTML introduction. Viitattu 22.09.2017. https://www.w3schools.com/html/html_intro.asp

Refsnes Data 2017b. What is AJAX? Viitattu 25.08.2017. https://www.w3schools.com/PhP/php_ajax_intro.asp

Refsnes Data 2017c. SVG Tutorial. Viitattu 29.09.2017. https://www.w3schools.com/graphics/svg_intro.asp

The Apache Software Foundation 2015a. Overview. Viitattu 31.03.2017. https://cordova.apache.org/docs/en/latest/guide/overview/index.html

The Apache Software Foundation 2015b. Platform Support. Viitattu 31.03.2017. https://cordova.apache.org/docs/en/latest/guide/support/index.html

Top Tunniste Oy. RFID/NFC Tekniikka. Viitattu 01.06.2017. https://toptunniste.fi/rfidnfc-tekniikka/

Triggs, R. 15.2.2017. What is NFC & how does it work? Viitattu 01.06.2017. http://www.androidauthority.com/what-is-nfc-270730/

Tutorialspoint 2017a. What is CSS? Viitattu 22.09.2017. https://www.tutorialspoint.com/css/what_is_css.htm

Tutorialspoint 2017b. CSS-Syntax. Viitattu 22.09.2017.
https://www.tutorialspoint.com/css/css_syntax.htm

KUVALÄHTEET

Kuva 1. The Apache Software Foundation. Architecture.
https://cordova.apache.org/docs/en/latest/guide/overview/index.html#architectu