Vilkka Sarantila

Kotihoitoa avustava mobiilisovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

25.8.2014

Tekijä Otsikko Sivumäärä Aika	Vilkka Sarantila Kotihoitoa avustava mobiilisovellus 41 sivua 25.8.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Yliopettaja Jaana Holvikivi Lehtori Olli Alm

Insinöörityön tavoite oli kehittää sovelluksen prototyyppi kotihoidon työntekijöiden tueksi. Sovelluksen tuli avustaa hoidettavan suun kunnon tutkimisessa ja opastaa tekemään tarvittavat toimenpiteet sen hoitamiseksi. Haasteena oli löytää sopiva alustariippumaton tekniikka toteutettavaksi mobiililaitteiden verkkoselaimissa ja siitä oli myöhemmin kyettävä kokoamaan natiivi, laitekohtainen sovellus. Kehittäjän kannalta keskeisenä ongelmana olivat myös näyttöresoluutioiden kirjon aiheuttamat vaatimukset. Haasteena oli myös käyttäjien vaihteleva tekninen osaamistaso.

Prototyypin pääsisältönä olivat suun tarkastuksen ja hoidon opetusvideot sekä tarkastuslomake yhteenvetoineen.

Työssä käytetyt sovellustekniikat olivat HTML5, CSS, DOM, JavaScript, jQuery-ohjelmointikehys, jQuery Mobile -mobiiliohjelmointikehys ja PhoneGap-ohjelmistokehys. Käyttöliittymäosiossa huomioitiin yksisivuisen sovelluksen toimintatapa ja sovelluksen sisältämään lomake sekä videot, sillä ne vaativat muuhun verrattuna erityistä teknistä suunnittelua. Testausosiossa tutkittiin natiivin ja selainpohjaisen sovelluksen virrankulutuksen eroja ja käyttöliittymän toimivuutta.

Tuloksena syntyi toimiva, helppokäyttöinen ja alustariippumaton prototyyppisovellus kotihoidon avuksi.

Avainsanat	alustariippumattomuus, monialustakehitys, responsiivisuus, Mobile first, HTML5, mobiilisovellus, hoitosovellus

Author Title Number of Pages Date	Vilkka Sarantila CareApp for elderly people 41 pages 25 August 2014
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Jaana Holvikivi, Project Manager Olli Alm, Lecturer

The goal of this thesis was to develop an application prototype to help homecare workers. The purpose of the application was to assist when examining of mouth-related health issues and to guide with the possible procedures to take care of these issues. The problem was to find a suitable technique to be executed in a web browser of a mobile device and later on, it had to have the possibility to be developed to a native, platform specific application. For a developer, the essential problem was to serve the variety of screen resolutions and diversity of end users' technical skill level.

The core content of the prototype was to create educational videos about how the examination and treatment of mouth area are implemented.

The technical implementations were HTML5, CSS, DOM, JavaScript, jQuery Mobile –mobile programming frame and PhoneGap –software frame. In user interface –section special attention was paid on one page application's mode of action, and the form and videos included in the application, since they required specific technical design compared to the other fields within the work. Testing took a closer look of the differences in power consumption between native and browser –based applications. Testing also took into consideration the functionality of the user interface.

As a result a user-friendly, platform independent prototype of an application was created to aid homecare workers.

Keywords	platform independence, multiplatform development, responsiveness, Mobile first, HTML5, mobile app, careapp

Sisällys

Lyhenteet

HTML	Hypertext Markup Language. Internetin sivunkuvauskieli, jonka avulla sivujen sisältö esitetään.
CSS	Cascading Style Cheets. World Wide Webin tyyliohjeet määrittelevä kieli. Määrittelee www-sivun ulkoasun.
XML	Extensible Markup Language. Rakenteinen merkintäkieli, jolla annetaan tiedolle merkitys.
AJAX	Asynchronous JavaScript And XML. Dynaaminen JavaScript. Tapa jolla informaatiota liikutetaan asiakassovelluksen ja palvelimen välillä. Mahdollistaa www-sivun osittaisen päivittämisen.
DOM	Document Object Model. Standardin mukainen malli rakenteisten dokumenttien esittämiseen olioina.
SDK	Software Development Kit. Yleinen lyhenne eri työkalupaketeista, joilla voi luoda sovelluksia tietyille alustoille.
IOS	Applen kehittämä käyttöjärjestelmä.
Flash	Tarkemmin Adobe Flash. Sovellusalusta multimediaesitysten luomiseen.
W3C	World Wide Web Consortium. Kansainvälinen yhteisö, joka kehittää www:n standardeja.
WAI-ARIA	Web Accessibility Initiative – Accessible Rich Internet Applications. Yhteisö, jonka pyrkimys on tukea esteettömien verkkosivujen leviämistä yleisiksi käytännöiksi.
NFC	Near Field Communication. Joukko standarderja, jotka mahdollistavat lyhyen kantaman kommunikaation laitteiden välillä. Hyödyntää RFID-tekniikka.

SSL	Secure Socets Layer. Turvallisuusprotokolla, joka luo salatun yhteyden asiakassovelluksen ja palvelimen välille tiedonsiirtoon.
UI	User Interface. Käyttöliittymän englanninkielinen lyhenne.
NPM	Node Packaged Modules. Node Js:n pakettienhallintajärjestelmä, joka mahdollistaa suuren osan Node Js:n toiminnallisuuksista.
CMS	Content Management System. Sisällönhallintajärjestelmien yleinen lyhenne.
PHP	Hypertext Preprocessor. Erityisesti web-ohjelmointiin soveltuva skriptauskieli.
ANT	Another Neat Tool. Apachen kehittämä Java-kirjasto ja komentorivityökalu, jonka avulla saadaan automatisoitua sovelluskehitystä.

1 Johdanto

Insinöörityön tavoitteena on kehittää ikäihmisten kotihoitoa avustavan sovelluksen prototyyppi. Sovelluksen tulee parantaa hoidon laatua ja nopeuttaa sitä opastamalla kotihoidon työntekijöitä videoiden, kuvien ja tekstin avulla sekä helpottaa hoitoon liittyvää raportointia sähköisillä lomakkeilla. Prototyyppi keskittyy pelkästään ikäihmisen suun hygieniaan. Työ on tehty yhteistyössä Lahden kaupungin suun terveydenhuollon, kotihoidon ja Metropolia Ammattikorkeakoulun suun terveydenhuollon ja tietotekniikan tutkinto-ohjelmien kanssa. Ajatus avustavan sovelluksen kehittämisestä lähti Lahden kaupungin hammashoidon aloitteesta.

Vanhusten laitoshoidon ongelmana ovat suuret kustannukset niin yhteiskunnalle kuin hoidettavalle tai/ja hänen omaisilleen. Tutkimusten mukaan hoitoa tarvitseva ihminen pysyy keskimäärin terveempänä ja toimintakykyisempänä sitä pidempään, mitä kauemmin hän pystyy asumaan kodissaan. Kotona asuvan vanhuksen suunhygienian omatoiminen huolehtiminen on tutkimuksen mukaan huonoa. Tästä syystä kotihoitajien antamaa hoitoa pyritään tehostamaan ja sen laatua parantamaan nykytekniikan avulla. Lahden kaupunki halusi kehittää mobiilisovelluksen tehostamaan kotihoidon asiakkaiden suun kunnon valvontaa ja hoitoa. Tarkoituksena on myös pystyä erottamaan sovelluksen lomakkeen avulla raja, milloin vanhus tarvitsee ammattilaisen apua ja milloin selvitään kodinhoitajan osaamisella. [1; 35.]

Ohjelmistomäärityksen mukaan tuotetaan mobiilisovelluksen prototyyppi, joka opastaa kotihoidon työntekijöitä arvioimaan ikäihmisen suun terveydentilaa ja tekemään tarvittavia hoitotoimenpiteitä. Sovelluksen avulla työntekijän on myös pystyttävä raportoimaan suun kuntoarvion tulokset sekä ottamaan epäilyttävistä löydöksistä valokuva mobiililaitteen kameralla. Teknisinä vaatimuksina sovellukselle on alustariippumattomuus ja responsiivisuus. Sovelluksesta on oltava mahdollista tuottaa sekä selain- että natiiviversio eri laitteille. Sovelluksen käytettävyys ja sen intuitiivisuus ovat tärkeimpiä näkökulmia ohjelmistomäärityksessä.

Lahtelaisten yhteistyökumppanien toivomuksena on selainpohjainen ratkaisu, sillä kaupungin työntekijöiden laitekanta koostuu vanhemmista Nokian N-sarjan puhelimista. Vuoden 2014 kuluessa kanta uudistuu Samsungin Android-puhelimiksi. Metropolian

tietotekniikan puolelta toivomuksena on kehittää natiivi Android-sovellus, sillä tällöin sovellusta voisi käyttää myös ilman internetyhteyttä ja se olisi levitettävissä myös alueille, joissa yhteydet ovat heikompia. Tämä on tärkeää varsinkin järjestelmää laajennettaessa. Kehitysvaiheessa pitää myös kyetä testaamaan sovelluksen hyödyllisyyttä ja toimivuutta käytännössä. Mikäli sovellus ei käytännön työhön sovi, on jatkokehitystä syytä punnita uudestaan. Vaihtoehtoina voi olla muuttaa sovelluksen toimintaa tai lopettaa kehitys. Testiryhmänä on Lahden kaupungin kotihoito.

Sovelluksen toivotaan parantavan ikäihmisen hoidon laatua hänen omassa kodissaan. Tämä myös epäsuorasti pidentää aikaa, jonka hän pystyy omassa kodissaan asumaan sekä vähentää esimerkiksi lääkäri- ja hammaslääkärikäyntejä. Näin ollen hoidosta on tarkoitus tulla kustannustehokkaampaa.

Kehittäjän näkökulmasta tarkoituksena on myös löytää toimiva ratkaisu palvelemaan yhteistyökumppanin tarpeita aiemmin määritettyjen vaatimusten mukaisesti. Jotta sovellus palvelisi tämänhetkisiä opetus- ja raportointikäyttötarkoituksia mahdollisimman hyvin, sen tavoitteena on toimia eri näyttöko'oilla ja laitteilla.

Sovelluksen kehityksessä tulee ottaa huomioon käyttäjien mahdollinen tietoteknisen osaamisen puute, joten helppokäyttöisyyteen ja intuitiivisuuteen on teknisten ratkaisujen ohella kiinnitettävä erityisen tarkkaa huomiota. Responsiivisuuden ja helppokäyttöisyyden takia sovellus toteutetaan HTML5:llä, CSS3:lla sekä jQuery- ja jQuery Mobile-kirjastoilla. Nämä tekniikat mahdollistavat mobiilisovelluksille geneerisen ja riittävän helppokäyttöisen ulkoasun ja sivurakenteen. Tekniikoiden avulla sovellus saadaan mukautumaan eri näytöille huomattavan pienellä työmäärällä.

2 Kotihoitoa avustavan sovelluksen tausta

2.1 Mobiilisovellukset kotihoidon tukena

Suomessa kotihoitoa avustavia työkaluja tarjoaa Isosta-Britanniasta lähtöisin oleva Tunnstall Oy. Sen CareApp -sovellusperheen avulla kerätään ja tallennetaan tiedot hoidettavasta, hoitajista, käynneistä, tehdyistä toimenpiteistä ja hoitosuunnitelmista sekä hälytysten ja poikkeamien syistä. Sovellus siis käytännössä kattaa jo suuren osan

suunnitelluista toiminnallisuuksista, mutta ei opasta niillä hoidon alueilla, joilla hoitajilla ei ole riittävää tietotaitoa. Sovellusperheen osat ja niiden toiminnallisuudet ovat seuraavat:

- CareApp Plan: Henkilöstö näkee työvuorolistansa, suunnitellut käynnit sekä tiedot käyttäjistä ja toimenpiteistä. Sovelluksella voi tehdä muistiinpanoja ja nähdä kulkutavan päivän suunnitelmassa.
- CareApp Activity: Sovelluksen avulla henkilöstö raportoi hoitokäyntien ajat ja tehdyt toimenpiteet, poikkeamat ja suunnittelemattomat käynnit.
- CareApp Lock: Lockin avulla hoitaja pystyy avaamaan ja lukitsemaan hoitokohteeseen asennetun CareLock-lukitusyksikön lukkoja bluetooth-yhteyden avulla. Tiedot tapahtumista tallentuvat.
- CareApp Alarm: Sovellus hälytysten tekoon. Hälytyksen sattuessa hoitajat saavat suoraan kohteen yhteystiedot. [3.]

Ulkomaisista yrityksistä Yhdysvalloissa toimivalla Addus HomeCare -kotihoitoyrityksellä on yhteistyössä T-Mobilen kanssa vastaava sovellus. Addus HomeCarella on noin 14 500 hoitajaa 26 000 asiakkaalle kahdessakymmenessä osavaltiossa. Tällä mittakaavalla etenkin pienistäkin päivittäisistä hoidon nopeutumisesta johtuvista kustannussäästöistä kertyy vuositasolla suuria säästöjä. Tutkimuksen mukaan siitä, kun avustaja huomaa muutoksia kotihoidon asiakkaan kunnossa, saattaa kestää jopa 45 päivää, ennen kuin paperityö saadaan hoidettua ja asianmukainen hoitohenkilö hälytettyä. Sovelluksen avulla yritys saa automatisoitua muun muassa raportoinnin, joka hoituu paikanpäällä, palkanlaskentaa (voidaan seurata käyntien ja työpäivien kestoja) ja aikataulutusta. Toisin sanoen sovellus vähentää paperityötä huomattavasti sekä parantaa asiakkaan hoitoa nopeuttamalla reagointiaikaa. [34.]

2.2 Kotihoitoa avustavan mobiilisovelluksen edellytykset

Responsiivisuus ja Mobile First

Pohjimmiltaan responsiivinen eli mukautuva suunnittelu tarkoittaa sovellusta tai verkkosivua, joka on kehitetty toimimaan millä tahansa laitteella riippumatta resoluutiosta ja etenkin näytön koosta. Responsiivinen suunnittelu sisältää asiakaspään ohjelmoinnissa kolme pääelementtiä.

1. Suhteellisiin mittoihin perustuva sivutaitto. Pikselimittojen sijaan käytetään koon määrittelyssä prosentteja suhteessa ympäröivään elementtiin. Tällä tavalla elementit säilyttävät oikeat mittasuhteet toisiinsa sivua skaalatessa.

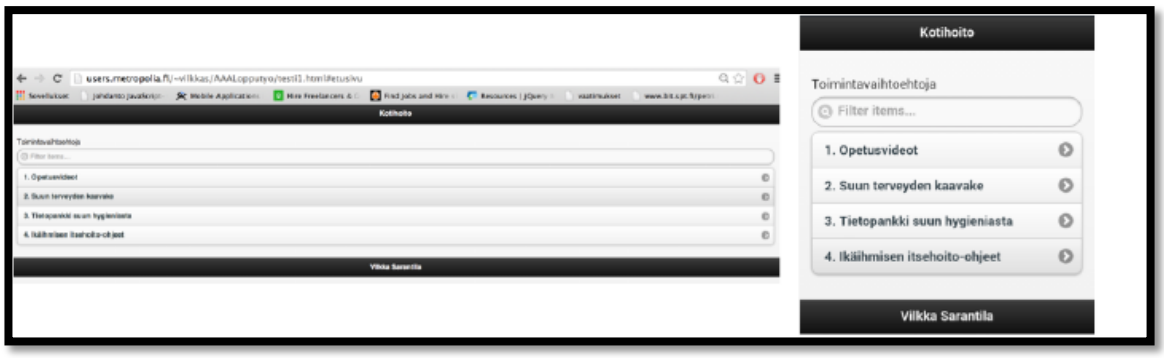
2. Skaalautuvat kuvat ja suhteelliset tekstikoot. Kuville on mahdollista asettaa max-width- tai max-height-tyylimääritys, joka rajoittaa kuvan enimmäisleveyden ympäröivän elementin leveyteen, oli elementin koko mikä hyvänsä. Suhteellinen koko lasketaan kaavalla nykyisen elementin koko jaettuna ympäröivän elementin koolla.

3. CSS3:n moduuli mediakyselyt ja -tyypit. Mediatyyppien avulla voidaan määrittää tietyt tyylimääritykset mediakohtaisesti. Erilaisia ennalta määritettyjä mediatyyppejä on laaja kirjo aina pistekirjoituksesta puhesyntetisaattoreihin. Nämä antavat sovelluskehittäjän määrittää vielä tarkemmin eri tyylimäärityksiä riippuen päätelaitteen näytön koosta ja tyypistä. Tärkeimpiä määritystekijöitä ovat

- näyttöalueen koko (leveys ja korkeus)
- tulostuspaperin koko
- median, esimerkiksi kuvan tai videon suunta (pysty tai vaaka)
- median kuvasuhde
- median värimäärä
- median resoluutio.

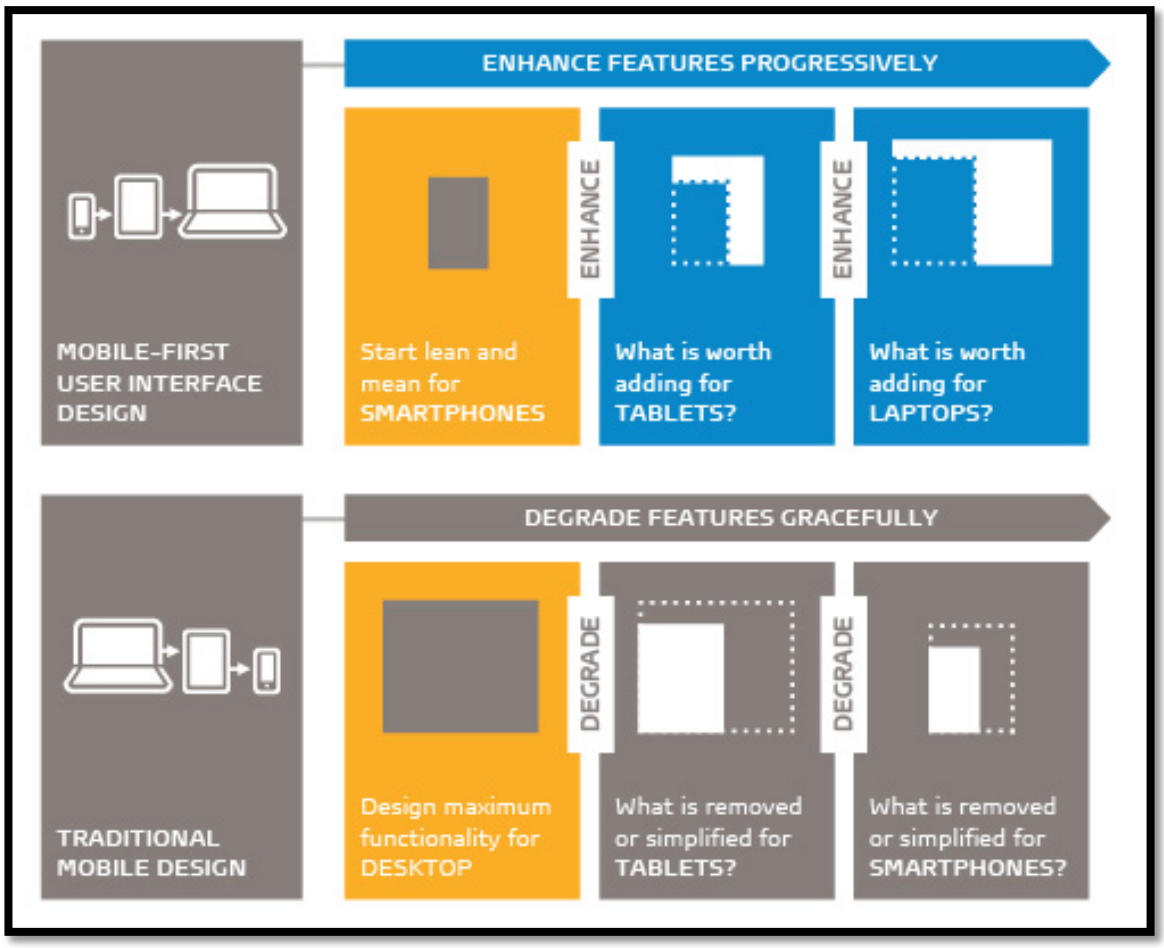
Mediakyselyitä käytettäessä on hyvä käyttää viewport-metatagia, sillä mobiililaitteiden selainten resoluutio on usein pienempi kuin työpöytäkoneiden. Viewport-tagilla voidaan siis määrittää, miten leveänä selaimen tulisi sivua tulkita.

Kuvassa 3 on esimerkki siitä, kuinka sovellus mukautuu 13 tuuman Mac Book Pron ja 4,8 tuuman Samsung Galaxy S III:n näytöille. Koska sovellusta on tarkoitus käyttää ensisijaisesti mobiililaitteissa, kuvalla on tarkoitus havainnollistaa käyttöliittymän mukautuvuutta, ei näyttää kauniilta. Yksinkertaisimmillaan responsiivisen sovelluksen näkymä venyy näytön koon mittaiseksi. Edistyneemmissä sovelluksissa komponenttien esiintyminen, ulkoasu ja käyttäytyminen saattaa muuttua.



Kuva 1. Responsiivinen sovellus kahdella eri päätelaitteella.

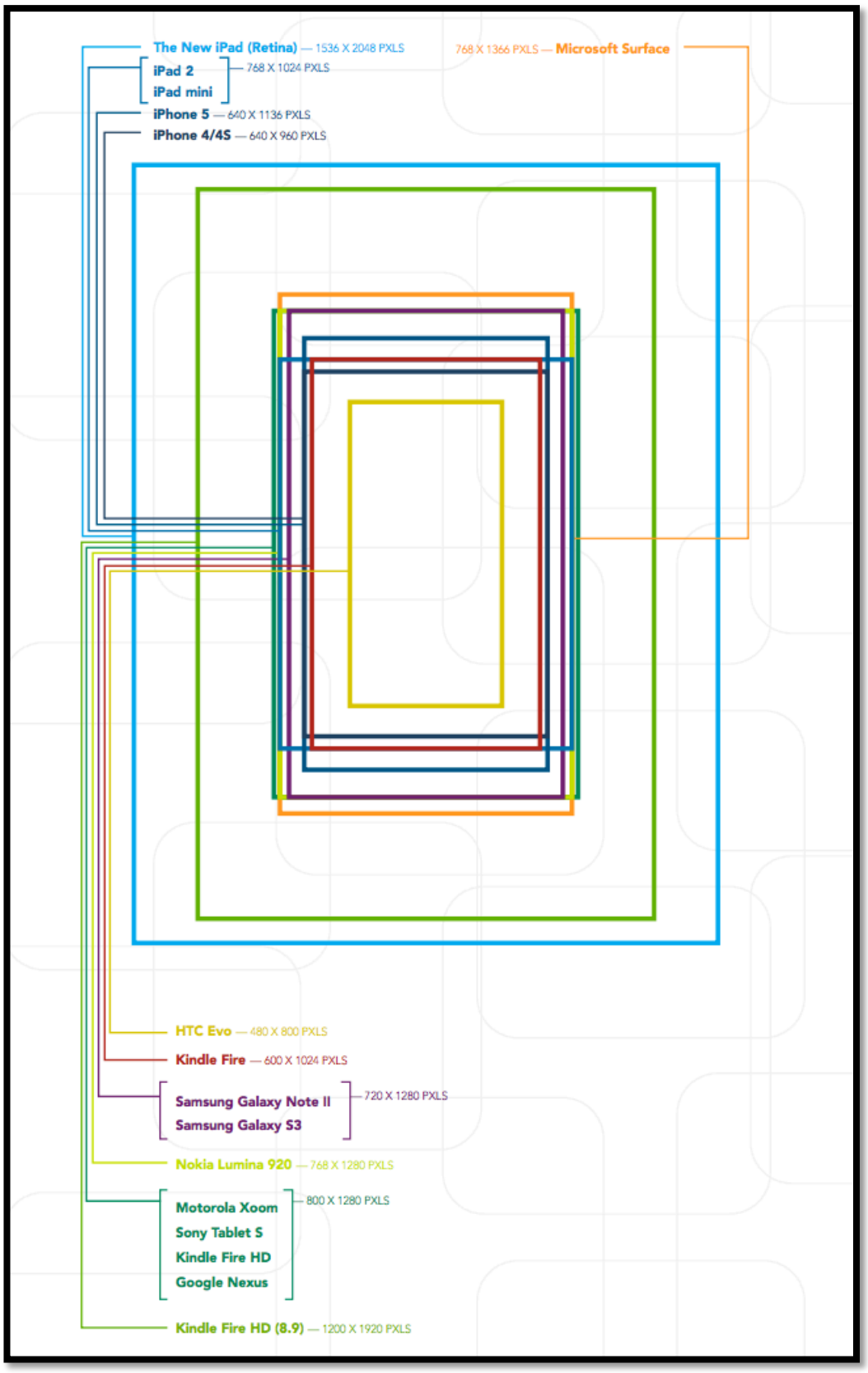
Mobile first -paradigma on hyvin varteenotettava vaihtoehto perinteiselle sovellus- tai www-sivusuunnittelulle. Tavallisesti kehitys alkaa työpöytälaitteille suunnittelusta, mutta Mobile First -lähtökohta on alkaa pohtia ohjelmiston perimmäistä tarkoitusta eli tärkeintä yhteistä tekijää, jonka on näyttävä jokaisella näyttöpäätekoolla. Aluksi varmistetaan mobiilinäkymän toimivuus ja sitten lähdetään laajentamaan ominaisuuksia työpöytänäkymään. Kuvassa 4 on havainnollistettu perinteisen ja Mobile First -menetelmien eroja. Ylemmällä rivillä Mobile First, jossa progressiivisesti mietitään ja lisätään ominaisuuksia suuremmille resoluutioille. [4.]



Kuva 2. Mobile first- ja perinteinen sovelluskehitysmenetelmä [4].

Aiemmin mobiililaitteille on luotu omat sivustonsa ja käyttäjä on ohjattu niille automaattisesti, mikäli kehittäjän luoma ohjelmisto on tehty tunnistamaan asiakaslaitteen näyttökoko. Tällöin myös ylläpidettävänä on kaksi erillistä sivustoa, ja pelkästään perustamiseen liittyvät kustannukset ja työmäärä kasvavat. Toinen vaihtoehto on rakentaa jokaisen valmistajan päätelaitteelle omat sovelluksensa. Tämä adaptiivinen lähestymistapa takaa varmemman toimivuuden, mutta myös kasvattaa työmäärää moninkertaiseksi.

Kuvassa 3 on kuvattu yleisimpien mobiililaitteiden resoluutiot antamaan karkeaa kuvaa responsiivisen suunnittelun tärkeydestä ja jokaiselle laitteelle oman sovelluksen kehittämisen työmäärästä. Pelkästään laitevalmistajien laaja kirjo puhumattakaan jokaisen laitevalmistajan eriresoluutoisista näytöistä tekevät työmäärästä suuren suunniteltaessa omaa, ei-responsiivista sovellusta näille.



Kuva 3. Yleisimpien mobiililaitteiden resoluutiot [5].

Käytettävyys

Koska suunniteltavan sovelluksen käyttäjäryhmän eli kotihoitajien ikäjakauma on laaja, vaihtelee käyttäjien tekninen osaamistasokin. Tästä syystä on keskityttävä luomaan sovelluksesta mahdollisimman yksinkertainen. Tämä tarkoittaa kuitenkin pienempää määrää navigointivaihtoehtoja yhdellä sivulla, ja täten sovelluksen niin kutsuttu sivusyvyys kasvaa. Yksinkertaistaminen tarkoittaa hyvin usein vähemmän sisältöä sivulla, mikä taas tarkoittaa sisällön jakamista useammalle sivulle. Jotta sivujen määrä ei kasvaisi liiaksi ja käytettävyys pysyisi silti helppona, oli löydettävä ratkaisu, joka on riittävän yksinkertainen mutta navigaation eli sivujen syvyys ei pääse kasvamaan liian suureksi. Käyttäjän on koko ajan tiedettävä missä kohtaa sovellusta hän on ja miten päästä toivotulle sivulle. Suunnitteluperiaatteena olikin tunnettu KISS eli Keep It Simple Stupid. Toisin sanoen suunniteltaessa pyrittiin miettimään tarkkaan, mikä sisältö sovelluksessa on ehdottoman olennaista ja minkä voi jättää pois. [36.]

Tarvittava ohjelmakoodin jäsentely, kommentointi ja loogiset kokonaisuudet on otettava huomioon, jotta sovelluksen jatkokehityksestä saataisiin mahdollisimman helppoa. Tämän prototyypin tapauksessa edellä mainituista ongelmista etenkin kehittäjän kannalta selvittiin helposti, sillä ohjelmakoodi koostuu pääasiassa staattisesta HTML:stä ja pienehköstä määrästä JavaScriptiä. jQuery Mobilen rakenne itsessään pakottaa jakamaan koodin hyvin selkeisiin sivukokonaisuuksiin, jotka ovat aina tietyn ohjelmakoodikokonaisuuden sisällä. jQuery Mobilea käsitellään lisää myöhemmin. Riittävä kommentointi tukee jo ennestään selkeää rakennetta.

2.3 Mobiililaite kotihoidon apuna

2.3.1 Opetusvideot

Yleisesti asioiden opettaminen opetusvideoiden avulla eroaa perinteisestä opettamisesta huomattavasti. Haasteita tuottaa pelkästään opettajan ja opiskelijan välisen interaktion puuttuminen, jolloin vastuu oppimisesta painottuu videon tekijälle ja loppukädessä oppijalle. Opiskelijan vastuu ennalta taltioidun videon avulla korostuu huomattavasti. Opettaja niin sanotusti saa yhden mahdollisuuden innostaa opiskelijansa oppimaan opetettava aihe, sillä lisäargumentteja ei välttämättä ole mahdollista liitää jälkikäteen. Hoitotyössä opiskelijoilla oletetaan tietenkin olevan motivaatio jota kuitenkin saat-

taa laskea teknisen osaamisen puute, jolloin käyttöliittymäsuunnittelun tärkeys korostuu entisestään.

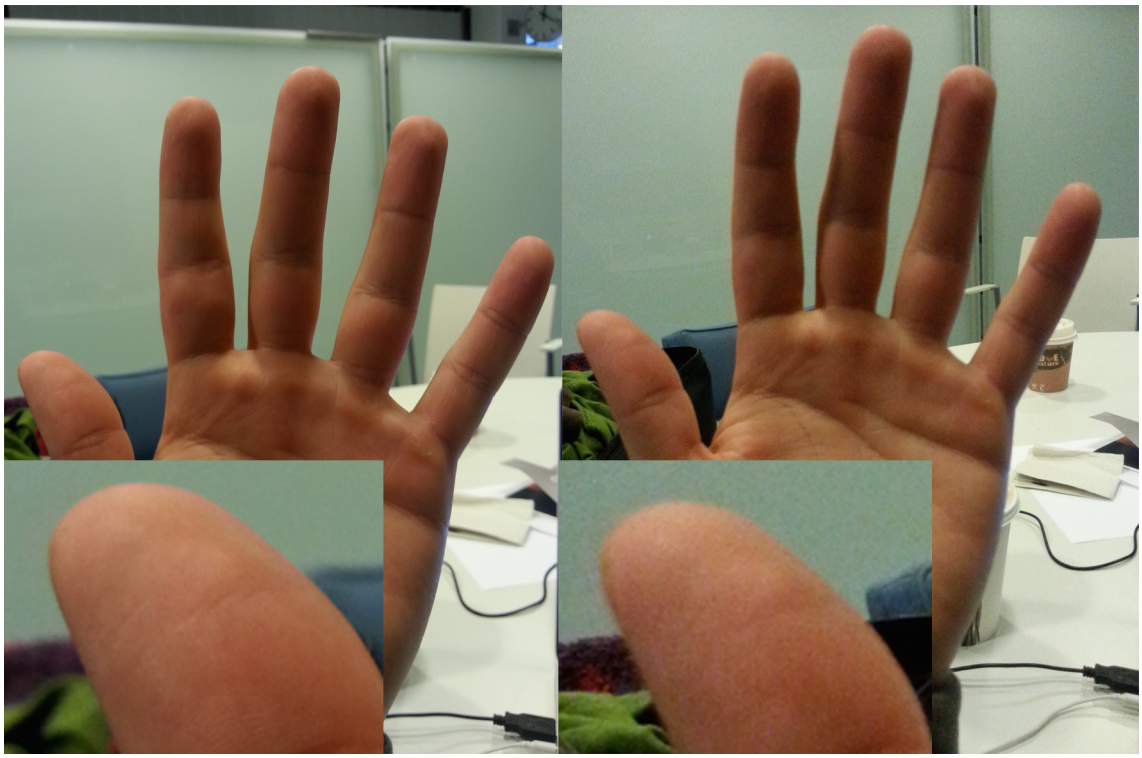
Sovelluksen opetusvideoissa onkin otettu huomioon sen käyttäjien aiempi osaaminen opeteltavalta alueelta ja orientaatioperusta on luotu sisällysluettelon ominaisuudessa toimivilla navigaationapeilla. Tavanomaiseen oppimisympäristöön verrattuna videoiden etuna on etenkin uudelleen toistamisen mahdollisuus tarvittaessa. Käyttäjä voi myös toistaa videon haluamallaan nopeudella, joten oppimisen mahdollisuus kasvaa. Etuna on myös se, että opetus ei ole sidottuna tiettyyn henkilöön, paikkaan tai aikaan, jolloin opettamisesta tulee kustannustehokkaampaa. [37.]

Videoissa on käytetty kertojaa täydentämään liikkuvan kuvan sanomaa. Kertojan selostaessa erilaisia luetelmia on käytetty tekstiä auttamaan katsojaa painamaan kerrotun asian mieleen. Opetusvideoiden sisältö on suunniteltava erityisen tarkkaan, sillä turha sisältö vain hämmentää katsojaa ja kuluttaa turhaan oppijan aikaa.

2.3.2 Kuvan ottaminen

Yleisesti ottaen mobiililaitteiden kamerat eivät vedä vertoja järjestelmäkameroiden kuvanlaadulle. Puhelimien kameroiden kennot eivät ole yhtä herkkiä valolle, ja erilaiset tarkennus- ja polttovälin säätömahdollisuudet ovat hyvin rajalliset. Yksi tärkeimmistä kuvan onnistumiseen vaikuttavista tekijöistä on riittävä valo.

Kotihoitoa avustavassa sovelluksessa on mahdollisuus tarvittaessa kuvan ottamiseen ja sen lähettämiseen hammaslääkäriin konsultointia varten. Selvää on, että tämä tuottaa haasteita etenkin, kun tulee tarvetta saada kuva suun sisäisistä alueista. Kotihoidon työntekijällä tulisikin olla varustuksessaan otsalamppu, jolla olosuhteita saa parannettua. Yksi mahdollisuus hämärässä kuvaamiseen on nostaa kameran herkkyyttä eli ISO-arvoa. Herkkyyden lisääminen kasvattaa kuitenkin hyvin nopeasti kuvassa esiintyvää häiriötä, sillä sähköinen herkkyyden nostaminen pyrkii pelkästään keinotekoisesti vahvistamaan kennon vastaanottamaa signaalia, jolloin myös signaalissa esiintyvät pienet häiriöt kasvavat. Tästä syystä kannattaa herkkyyden nostamisen sijasta aina yrittää lisätä kohteeseen kohdistuvaa valoa. Kuvassa 4 ovat Samsung Galaxy SIII -puhelimella otetut kaksi kuvaa. Kuvien tarkoitus havainnollistaa suurimman ja pienimmän herkkyysarvon välistä eroa.



Kuva 4. Vasemmalla pienellä herkkyydellä ja oikealla suurella herkkyydellä otettu kuva.

2.3.3 Paikannus

Mobiililaitteen paikannukseen pystytään sitomaan lukemattomia hyödyllisiä toiminnallisuuksia. Esimerkiksi hoitajan saapuessa hoitokohteen läheisyyteen voidaan hoidettavan tietojen lataus automatisoida, jolloin toiminta nopeutuu.

Tulevaisuudessa paikannus voi olla mahdollista kytkeä erilaisiin kulunhallintajärjestelmiin, jolloin hoitajan ei tarvitse kantaa avaimia mukanaan vaan hoitokohteen ovi aukeaa lähelle tultaessa. Paikannus voidaan myös sitoa hoitajan päiväohjelman seurantaan. Näin hoitaja voi saada jatkuvaa informaatiota aikatauluista, seuraavasta kohteesta ja aikataulussa pysymisestään. Se voi parhaimmillaan auttaa automatisoimaan monia päivittäisiä prosesseja, joihin muutoin kuluisi työntekijän aikaa.

Ongelmana saattaa olla esimerkiksi paikannuksen epätarkkuudesta johtuva väärän informaation vastaanottaminen tai lähettäminen. Tämä voi pahimmassa tapauksessa johtaa hoitovirheisiin tai kuluttaa enemmän aikaa kuin ei-automatisoitu käytäntö. Tiedon oikeellisuuden varmistamiseen on siis kiinnitettävä erityistä huomiota.

2.4 Käytetyt sovellustekniikat

2.4.1 HTML5 ja CSS

HTML5-kieli on nimensä mukaisesti uusin kehitysaste HTML-kielestä, ja sen oli tarkoitus pyrkiä vastaamaan päätelaitteiden lisääntyvään mobilisoitumiseen. Sen avulla ja hyvin toteutettuna sovellus pystyy toimimaan lähes kaikilla laitteilla, jotka sisältävät selainmoottorin. Sovelluksen toimivuus riippuu selaimen käyttämän moottorin kyvystä suorittaa ohjelmakoodia. [6.]

Verrattuna aiempiin kehitysversioihin HTML5-kieli sisältää useita uusia ohjelmointia helpottavia elementtivaihtoehtoja, kuten canvas-, video- ja audio-elementit sekä mahdollisuuden paikalliseen tiedon tallentamiseen käyttäjän selaimessa. [6; 7; 8.]

HTML5-sovelluksella tarkoitetaan yleisesti sovellusta, joka käyttää webin avoimia tekniikoita, kuten HTML, CSS ja JavaScript. Nämä tekniikat vaativat toimiakseen selainmoottorin, joita ovat esimerkiksi Chromessa WebKit ja Firefoxissa Gecko. Selainmoottori tulkitsee edellä mainittuja kieliä ja tulkkaa halutun sisällön näkyviin selainikkunaan sekä toteuttaa niillä määritetyt toiminnallisuudet. Selainmoottori voi myös toimia itsenäisesti tai osana toista sovellusta, jolloin tavanomaista navigointia ynnä muita sellaisia painikkeita ei ole esillä. Edellisistä versioista poiketen HTML5-sovelluksessa on yleensä vain yksi sisällöltään muuttuva dokumentti eli se on niin sanottu yksisivuinen sovellus. [6; 8.]

Vaihtoehtoisia tekniikoita HTML5:lle ovat eri natiivit sovellustekniikat, kuten Android-laitteille Javan avulla toteutetut sovellukset sekä iOS:lle Objective C:llä toteutetut ohjelmistot. WWW-ympäristössä HTML5 on CSS3:n ja eri JavaScript-kirjastojen kanssa syrjäyttämässä Flash-tekniikkaa. [6.]

2.4.2 JavaScript ja AJAX

JavaScript on alun perin Netscape Communicationsin kehittämä tulkattava kevyt oliopohjainen ohjelmointikieli, jonka perimmäisenä tarkoituksena on määritellä dynaamista toimintaa www-sivuilla. [11; 12.]

Kuten HTML, myös JavaScript suoritetaan selaimessa, joten sen toimintaan ei välttämättä tarvita internetyhteyttä. JavaScriptin avulla puurakenteen solmuja (elementtejä) ja niiden tyylimäärityksiä voidaan muokata dynaamisesti ja lisätä sekä poistaa. [11; 12.]

Kaikki selainmoottorit eivät tue kaikkia JavaScriptin ominaisuuksia, joten kielellä operoitaessa on otettava huomioon kohdeselain, jolla sivuston halutaan ensisijaisesti toimivan. Paljon JavaScriptiä sisältävä sivusto on suuritöinen toteuttaa kaikille selaimille sopivaksi selainmoottorien eroavaisuuksien takia.

AJAX tulee sanoista Asynchronous JavaScript and XML. Kiteytettynä AJAX on kokoelma tekniikoita, joiden tarkoituksena on asiakkassovelluksen ja palvelimen välinen asynkroninen tiedonsiirto. Tämä tarkoittaa sitä, että JavaScript-koodin eri osia ja funktioita voidaan ladata eri aikoihin eikä sivuston koko koodia tarvitse ladata uudestaan. Esimerkiksi käyttäjän painaessa sivustolla linkkiä toiseen välilehteen, välilehti ladataan sivun tiettyyn osaan, joka on ainoa sivuston päivittyvä osa. [11; 12.]

2.4.3 jQuery ja jQuery Mobile

jQuery on jQuery Foundationin kehittämä, maailman käytetyin avoimen lähdekoodin JavaScript-kielen funktiokirjasto, ja sen ensimmäinen versio julkaistiin vuonna 2006. Se on tehty helpottamaan JavaScriptillä ohjelmointia. [13; 14.]

W3C-organisaation tilastoimista www-sivuista 58,2 % käyttää kirjastoa, ja sen markkinaosuus on 93,5 % [13]. Hiljattain on ollut paljon keskustelua kirjaston vanhentumassa olevasta tekniikasta. Alan töitä etsiessäni törmäsin useaan työpaikkailmoitukseen, joissa mainittiin jQueryn olevan aikaansa jäljessä. Se tarjoaa silti erinomaisen kokoelman funktioita tämän prototyypin kehittämiseen. [13.]

Kirjaston voi joko ladata samalle palvelimelle kuin sitä käyttävä HTML/JavaScript-koodi ja viitata siihen URL-osoitteella tai voi viitata URL-osoitteella jonkun muun tahon palvelimella sijaitsevaan tiedostoon. Esimerkiksi Google tarjoaa tämän linkin. Toimintavarmuuden vuoksi on kuitenkin suositeltavaa ladata kirjasto omalle palvelimelle, jolloin sitä käyttäviä funktioita voi testata myös yhteydettömässä tilassa eikä kirjaston toimivuus ole riippuvainen muiden palvelimien toimintakyvystä. [14; 13.]

Yleinen käytäntö on viitata kirjastoon HTML-koodin HEAD- tai BODY-osan alussa, minkä jälkeen voidaan käyttää kirjaston funktioita. On myös muistettava lisätä viittaus omaan tai muihin erillisiin JavaScript-tiedostoihin vasta jQuery-viittauksen jälkeen, mikäli ne käyttävät kehyksen funktioita.

JavaScriptin heikkoutena on, että sen käyttämiä dynaamisia funktioita voidaan käyttää vasta, kun koko sivu on renderöity lähdekoodista. Tämä tarkoittaa sitä, että selainmoottorin täytyy ladata kaikki sivuston osat, kuten kuvat ja videot ynnä muut, ennen kuin se antaa dynaamiset toiminnot asiakkaan käytettäväksi. jQuery-kirjasto tarjoaa ratkaisuksi tähän ”.ready()”-funktion, joka sallii omien skriptien käytön useimmissa tapauksissa välittömästi, kun DOM on ladattu. CSS-tyylimääritykset on tärkeä ladata ennen skriptien lataamista, jotta niihin viittaavien funktioiden toiminnallisuus varmistetaan. [14; 16.]

jQuery Mobile on HTML5-pohjainen käyttöliittymien ohjelmointikehys ja sovelluskirjasto. Se perustuu semanttiseen HTML:ään, eli elementeille annetaan rooli, jonka pohjalta samat elementit saavat eri ulkoasuja, merkityksiä ja toiminnallisuuksia. WAI-ARIA mahdollistaa esimerkiksi saman elementin eri tilat ja näppäimistönavigoinnin. Esimerkkejä eri roolituksista on esillä koodiesimerkeissä. [15.]

Tutkiessani eri yritysten www-sivujen lähdekoodeja huomasin, että useat suurista yrityksistä käyttävät kehystä sivujen mobiiliversioissaan. Tunnetuimpia niistä ovat ehkäpä Ikea, Disney World ja suomalaisista yrityksistä ainakin Valtion rautatiet. Sen avulla voidaan rakentaa responsiivisia käyttöliittymiä kaikille suosituimmille mobiili- ja työpöytälaitteille, kuten Androidin, iOS:n ja Nokian eri käyttöjärjestelmille. [15; 17.]

Kehys on rakennettu erityisesti mobiilia käyttöä ajatellen ja se sisältääkin hyvän tuen eri kosketustoiminnoille sekä AJAXille, jonka avulla sivut ladataan asynkronisesti DOM:iin. AJAXin avulla sivujen väliset siirtymät ja muut toiminnot sekä ulkoasu voidaan ohjelmoida näyttämään natiivin tai muuten geneerisen mobiilisovelluksen kaltaisilta, vaikka sovelluksen kaikki osat on kuvattu yhdessä dokumentissa. [15.]

Käyttöliittymäkehyksellä kehitettäviä ohjelmia pystyy kehittämään koko kirjaston voimin, mutta lopulliseen versioon kannattaa jQuery Mobilen sivuilla olevan builderin avulla ladata vain siinä käytettävät toiminnallisuudet ja komponentit sovelluksen keventämi-

seksi ja nopeuttamiseksi. Sivut tarjoavat myös työkalun omien teemojen luomiseen sekä laajan valikoiman kolmansien osapuolien luomiin lisäosiin. [18.]

Koska monet vanhemmista mobiililaitteista eivät edelleenkään tue JavaScriptiä on jQuery Foundation luonut 3-asteisen luokituksen laitteista, jotka tukevat sitä täysin tai osin:

- A-grade Täysin tuettu kokemus AJAX-pohjaisilla animoiduilla sivun siirtymillä.
- B-grade Osin tuettu kokemus ilman AJAX-pohjaisia navigaatio-ominaisuuksia.
- C-grade Toimiva HTML-kokemus.

  [19.]

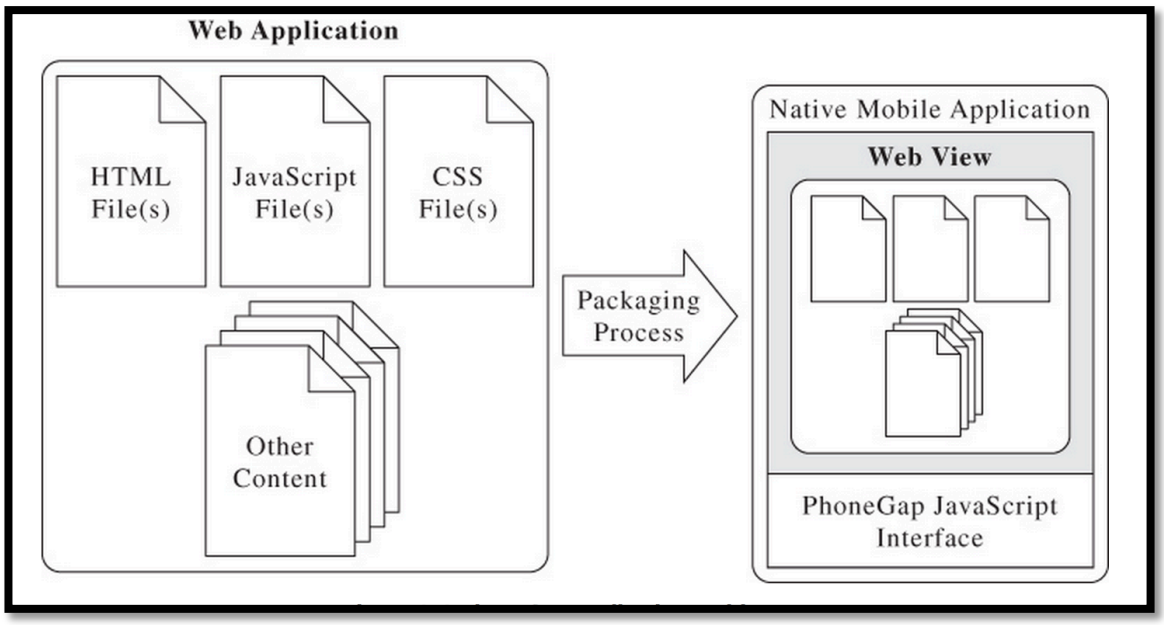
2.4.4 PhoneGap

PhoneGap yleisesti

PhoneGap on mobiiliohjelmointikehys, joka mahdollistaa natiivien sovellusten luomisen eri valmistajien laitteisiin niille tarkoitettuja ohjelmointikieliä käyttämättä. PhoneGapia käytettäessä kehitys tehdään luvussa 2.4.3 esiteltyjen HTML:n, CSS:n ja JavaScriptin avulla. Se käyttää hyödykseen laitevalmistajien selainkomponenttia natiivin sovelluksen sisällä. JavaScriptillä päästään käyttämään laitteiden natiivien ominaisuuksien rajapintoja, kuten kameraa, tallennustilaa, kiihtyvyysanturia ynnä muihita. PhoneGapin tarkoituksena on nopeuttaa ja helpottaa mobiilisovelluskehitystä käyttämällä edellä mainittuja kieliä ja paketoimalla valmis sovellus halutulle laitteelle.

Toimintaperiaate

PhoneGap käyttää kohdelaitteen WebWiew-komponenttia aiemmin mainituilla tekniikoilla toteutetun sisällön näyttämiseen. Kuvassa 5 on kuvattu PhoneGapin toimintaperiaate. HTML-, CSS- ja JavaScript-koodi suoritetaan edellä mainitun komponentin sisällä, jolloin puhelimen on mahdollista käsitellä sitä. Komponentin nimi on laitevalmistajakohtainen, mutta periaate on sama. Android-laitteissa nimi on WebView, iOS:ssä UIWebView ja Microsoftin alustoilla WebBrowser. [25; 29.]



Kuva 5. PhoneGap-mobiiliohjelmointikehyksen toimintaperiaate [25].

Kehys perustuu avoimen lähdekoodin Apache Cordova-projektiin joka on kokoelma laiterajapintoja, joiden avulla päästään käyttämään eri laitevalmistajien laitteiden natiiveja funktioita kutsumalla niitä JavaScriptillä. [20.] Taulukko 1 havainnollistaa PhoneGapin tarjoamia rajapintoja laitekohtaisiin toiminnallisuuksiin eri laitteilla.

Taulukko 1. PhoneGapin rajapinnat laitekohtaisille toiminnallisuuksille [20].

	iPhone	**Android**	**Black-berry**	**WebOs**	**Win-dows Phone 7 / 8**	**Sym-bian**	**Bada**
Kiihtyvyysanturi	x	x	x	x	x	x	x
Kamera	x	x	x	x	x	x	x
Kompassi	x	x	-	x	x	-	x
Yhteystiedot	x	x	x	-	x	x	x
Tiedostot	x	x	x	-	x	-	-
Paikannus	x	x	x	x	x	x	x
Media	x	x	-	-	x	-	-
Verkko	x	x	x	x	x	x	x
Hälytysilmoi-tukset	x	x	x	x	x	x	x
Ääni-ilmoitukset	x	x	x	x	x	x	x
Värinä-ilmoitukset	x	x	x	x	x	x	x
Muisti/Tallennus	x	x	x	x	x	-	-

Rajapinnat eivät rajoitu pelkästään Cordovan rajapintoihin, vaan monet kolmannen osapuolen kehittäjät ovat antaneet panoksensa. Kolmannen osapuolen kehittäjät eivät ole pelkästään yksityishenkilöitä vaan myös yrityksiä ja korkeakouluja, kuten Massachusetts Institute of Technology, MIT. Hyötysovellusten kannalta oleellisia liitäntöjä, joita prototyypin jatkokehityksessä voi hyödyntää, ovat

- NFC
- EmailComposer with attachments
- Bluetooth serial
- SSL Certificate Checker
- American Bible Society Biblesearch
- InApp Purchase
- WebView Debug
- Android InAppBilling
- Facebook Plug-in
- Instagram Plug-in.

[26.]

Vaikka kehitys tehdään web-tekniikoilla, on sovelluskehittäjän osattava julkaista lopullinen sovellus alkuperäisillä sovelluskehitystyökaluilla tai on käytettävä PhoneGapin tarjoamaa palvelua. [26.]

Eri mobiilikehitystyökalut suosituimmille alustoille ovat

- Android - Eclipse Android SDK
- iOS - iOS SDK , XCode IDE
- Microsoft - Visual Studio 2005/2008
- Symbian- The Java Platform, Standard Edition 8 Development Kit (JDK 8)

[21; 22; 23; 24].

PhoneGapin avulla kehittämisen helppous ei kuitenkaan ole itsestäänselvyys. Jokaisella alustalla JavaScriptin avulla kutsutut natiivit funktiot, kuten kuvan ottaminen, saatta-

vat toimia alustasta riippuen hieman eri tavoilla, joten ei ole taattua, että sovellus toimii jokaisella alustalla, mikäli se toimii yhdellä. Tästä syystä sitä on testattava jokaisella alustalla erikseen, mikä voi olla työlästä. Siksi itse lähtisin liikkeelle ensisijaisen alustan valitsemisesta, jolle sovellus kehitetään loppuun saakka. Tämän jälkeen voidaan jatkaa kehittämistä muille alustoille yksi kerrallaan. Alussa vaikeuksia saattaa tuottaa jokaiselle laitteelle sopivan kehitysalustan opettelu.

Mikäli jokaista kehitystyökalua ei halua tai ole aikaa tai mahdollisuutta opetella, tarjoaa PhoneGap työkalun nimeltä PhoneGap Build helpottamaan kokoamisprosessia. Tämä pilvipalvelu hoitaa julkaisemisen kehittäjän puolesta, mikä saattaa säästää aikaa ja vaivaa huomattavasti. PhoneGap Buildin avulla kehittäjä ei pelkästään saa koottua ohjelmistoaan jokaiselle alustalle, vaan saa sen myös kohdealustojen viimeisimmälle versiolle yhteensopivaksi. Palvelu tarjoaa myös synkronoinnin julkiseen tai yksityiseen Github-tiliin. Tämä helpottaa tuoreimman version kokoamista ja testaamista. Myös muihin versionhallintajärjestelmiin yhdistäminen on mahdollista. Kehittäjän ei palvelun avulla tarvitse tehdä muuta kuin ladata projektiin kuuluvat tiedostot zip-tiedostona tai sellaisenaan siihen ja muutamissa minuuteissa vastaanottaa latausosoitteet jokaiselle alustalle erikseen. PhoneGap Buildin käyttäminen avoimen lähdekoodin ja yhteisöllisten ja julkisten sovellusten luomiseen on ilmaista. [26.]

3 Kotihoitoa avustava mobiilisovellus

Insinöörityönä tehdyn kotihoitoa avustavan mobiilisovelluksen on tarkoitus opastaa hoitotilanteessa asiakkaan kotona. Se auttaa hoitajaa etupäässä videoiden, mutta myös kuvien ja tekstin avulla tutkimaan asiakkaan suun terveydentilaa sekä hoitamaan sitä. Terveydentilan seurantaa helpottamaan on sovelluksessa sähköinen seurantalomake, jonka yhteenvedossa on mahdollista ottaa kuvia epäilyttävistä löydöistä.

Suun terveydentilan kartoitustiheys riippuu tarpeesta ja suun kunnosta. Oletuksena on kuitenkin, ettei sitä suoriteta jokaisella käyntikerralla. Hoitaja saattaa ensimmäisillä kerroilla joutua seuraamaan tarkkaan videoiden ohjeistusta, mutta kokemuksen karttuessa videoita ei tarvitse kuin muistin virkistämiseen. Lomaketta sen sijaan on käytettävä joka

kerta suun kuntotutkimusta suoritettaessa. Hoitaja täyttää lomakkeen ja lähettää yhteenvedon taustajärjestelmään. Kotihoitoa avustavan sovelluksen sisältö on seuraava:

- Etusivu.
- Opetusvideot.
  (a) Suun terveyden tutkimuksen lomake. 6-kohtainen lomake.
  (b) Yhteenveto, joka täyttyy lomakkeen täytön perusteella suoritetuista toimenpiteistä. Yhteenveto sisältää myös lähetä- ja tulosta-mahdollisuudet.
- Tietopankki suun hygieniasta.
- Ikäihmisen itsehoito-ohjeet.

3.1 Vaatimukset

Ikäihmisten kotihoitoa avustavan mobiilisovelluksen perimmäisenä vaatimuksena oli parantaa hoidettavan suun ja sen ympäristön terveyttä. Sovelluksen on oltava helposti laajennettavissa ja sen on toimittava mobiililaitteen selaimessa. Avustavan sovelluksen on myös toimittava yleisimpien mobiililaitteiden resoluutioiden puitteissa, eivätkä sen toiminnallisuudet saa kärsiä resoluution tai näytön koon muuttuessa. Sovelluksesta on pystyttävä tuottamaan natiivi laitekohtainen sovellus aluksi Android-laitteille, ja sen on toimittava offline-tilassa. Ilman internetyhteyttä sovelluksen tulee pystyä toistamaan siinä oleva sisältö ja tallentamaan siinä oleva täytetty lomake siihen asti, kunnes laite saa yhteyden. Kun yhteys on mahdollinen, sovelluksen on pystyttävä lähettämään lomakkeen dataa hammashoitolaan. Myöhemmässä kehitysvaiheessa sovellukseen saattaa tulla tarvetta sisällönhallintajärjestelmän avulla datan lisäämiseen. Tämä saattaa olla esimerkiksi uusia ohjeita, kuvia tai videoita vanhustenhoitoon liittyen.

Sovelluksen on oltava riittävän intuitiivinen, jotta sen käytön oppivat nopeasti sekä enemmän että vähemmän tietoteknisiä taitoja omaavat henkilöt. Tämän takia virheiden mahdollisuudet on minimoitava ja painallus- ynnä muusta hämmennyksestä johtuvien virheiden sattuessa onkin tärkeää, että siitä pystytään palaamaan tai sen pystyy perumaan mahdollisimman helposti.

3.2 Kehitystyön kulku

Kotihoitoa avustavan sovelluksen prototyypin kehityksen työnkulku pääpiirteittäin Android-alustalle PhoneGapin avulla kehitettäessä oli seuraava:

HTML5:n, jQueryn ja jQuery Moblien avulla sovelluksen käyttöliittymän ja ei natiivien toiminnallisuuksien, kuten erilaisten siirtymäanimaatioiden, kehittäminen ja testaaminen tietokoneen selaimella.

Eclipsen, Android SDK:n (Software Developement Kit), ja ADT:n (Android Development Tools) asennus.

Apache Cordova ja PhoneGap asennetaan Node Js:n pakettien hallintaan tarkoitetun NPM:n avulla, joten seuraavaksi asennetaan tämä ohjelmisto. Node Js on palvelimella toimiva JavaScriptin suoritusympäristö. Sen vaatimukset suorituskyvyltä ovat pienet, joten sillä voidaan suorittaa palveluja kevyelläkin alustalla. [25; 26.]

NPM on Node Js:n pakettien hallintaan tarkoitettu järjestelmä. Node Js:n toiminnallisuus perustuukin suurimmilta osin ulkoisiin paketteihin. [28.]

Apache Cordovan lataaminen ja asentaminen. PhoneGapin ja Cordovan voi asentaa globaalisti komennolla

```
$ sudo npm install -g phonegap tai -cordova. -g
```

-g-merkintä käskee asentamaan PhoneGapin globaalisti. Sudo sallii ohjelmien asentamisen koneelle pääkäyttäjänä. Yleisesti tätä komentoa ei suositella. Sudo-komentoa käytettäessä pääte vaatii salasanavahvistusta. [28.]

PhoneGapin viimeisimmän version lataaminen päätteen avulla.

PATH-ympäristömuuttujan asettaminen viittaamaan Eclipsen Android-kehitysympäristön platform-tools- ja tools-hakemistoon, jotta ladatut paketit löytävät oikeaan osoitteeseen. Muuttuja on komentotulkissa (shell) määriteltyjä muuttujia, jotka

määräävät polun, josta jokin tietty käynnistettävä ohjelma haetaan. PATH on käytännössä merkkijono, joka koostuu kaksoispisteellä erotetuista hakemistonimistä. [27.]

Uuden projektin aloittaminen päätteeltä komennolla $phonegap create com.vilkka.sovellus Hoitosovellus, jossa ensin kerrotaan, että halutaan luoda phonegapilla uusi sovellus, minkä jälkeen annetaan paketin nimi ja sen jälkeen sovelluksen nimi.

Kuten listauksen alussa on mainittu, sovellus kehitettiin selainympäristössä. Kun sovellus toimi täysin pöytäkoneen selaimessa, sen lähdekoodi siirrettiin tarvittavine tiedostoineen edellisessä kohdassa luotuun PhoneGap-projektiin oikeille paikoilleen. Tämän jälkeen annettiin tarvittavat oikeudet ja sovellus olikin valmis koottavaksi PhoneGapin avulla. Tätä menetelmää pystytään käyttämään parhaiten, mikäli sovellus ei vaadi natiiveja toiminnallisuuksia, sillä niiden toimintaa ei selaimessa pystytä varmistamaan. Toisaalta jQuery Mobilen osuutta kehitettäessä voidaan hyvällä suunnittelulla rakentaa sovellus niin, että siihen lisätään PhoneGapin toiminnallisuudet jälkeen päin, eikä näitä kahta tarvitse tehdä samanaikaisesti. Tämä helpottaa virheiden sattuessa niiden etsimistä.

Huomioitava on, että tämä on vain yksi tapa luoda sovellus edellä mainittuja komponentteja käyttäen. Itse koin menetelmän hyväksi sen helpon omaksuttavuuden takia, mutta se ei missään nimessä ole ainoa oikea.

Natiivien toiminnallisuuksien kutsuminen JavaScriptin avulla tehdään kutsumalla navigator-oliota, minkä jälkeen kutsutaan halutun toiminnallisuuden oliot ja annetaan niille vaadittavat parametrit. Esimerkiksi yhteystiedot-objektin kutsuminen ja yhteystiedoista etsiminen tehdään koodiesimerkin 1 mukaisesti.

Koodiesimerkissä 1 määritellään ensin options-muuttujalle suodatin haettavan henkilön nimellä, eli tässä tapauksessa haetaan henkilöä nimeltä Bob. Sen jälkeen fields-muuttujalle määritellään, mitä tietoja haetulta henkilöltä halutaan, minkä jälkeen suoritetaan haku. Haussa määritellään edellä mainittujen parametrien lisäksi palautusfunktiot onSuccess ja onError. [30.]

```
var options = new ContactFindOptions();
```

```
options.filter="Bob";
var fields = ["displayName", "name"];
navigator.contacts.find(fields, onSuccess, onError, options);

function onSuccess(contacts)
 {
           for (var i=0; i<contacts.length; i++)
           {
           console.log("Display Name = " + contacts[i].displayName);
           }
}

function onError(contactError)
{
           alert(onError!');
}
```

Koodiesimerkki 1. Yhteystiedoista etsiminen [30].

Mikäli kysely onnistuu, haetaan hakuparametreissa määritetyn suodattimen mukaiset tiedot silmukan avulla, koska niitä voi olla useampia kuin yksi. Tämän jälkeen tehdään halutut toimenpiteet. Esimerkissä ne vain tulostaan selaimen lokiin.

Mikäli haku ei onnistu, suoritetaan haluttu toimenpide. Tässä esimerkissä ilmoitetaan käyttäjälle haun epäonnistumisesta.

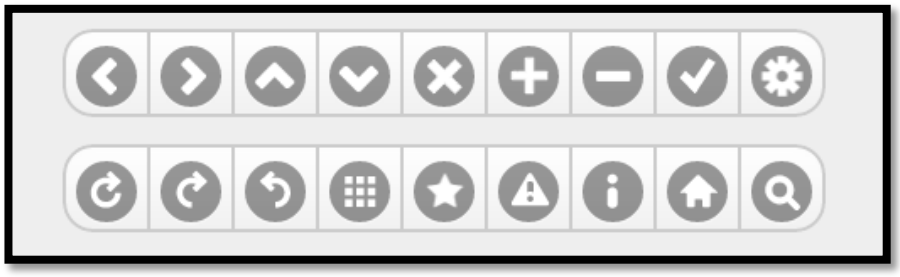
Lähes kaikkien rajapintojen perusperiaate on sama. Nyt toteutetussa sovelluksen prototyypissä ei ole ollut tarvetta toistaiseksi käyttää yhtäkään PhoneGapin tarjoamista rajapinnoista, joten tässä dokumentissa niitä ei käsitellä enempää.

3.3 Käyttöliittymä

jQuery Mobile vastaa käyttöliittymäkomponenttien sovittamisesta eri alustoille. Sen toiminta perustuu pääasiassa HTML-elementtien ja CSS:n toimintaan, ja lopullista käyttökokemusta on parannettu omalla JavaScript-koodilla, joka tekee omat muutoksensa lopulliseen ohjelmakoodiin HTML-elementtien ja CSS:n lataamisen jälkeen. Toisin sanoen HTML-elementit ladataan joka tapauksessa, vaikka selain ei tukisikaan JavaSc-

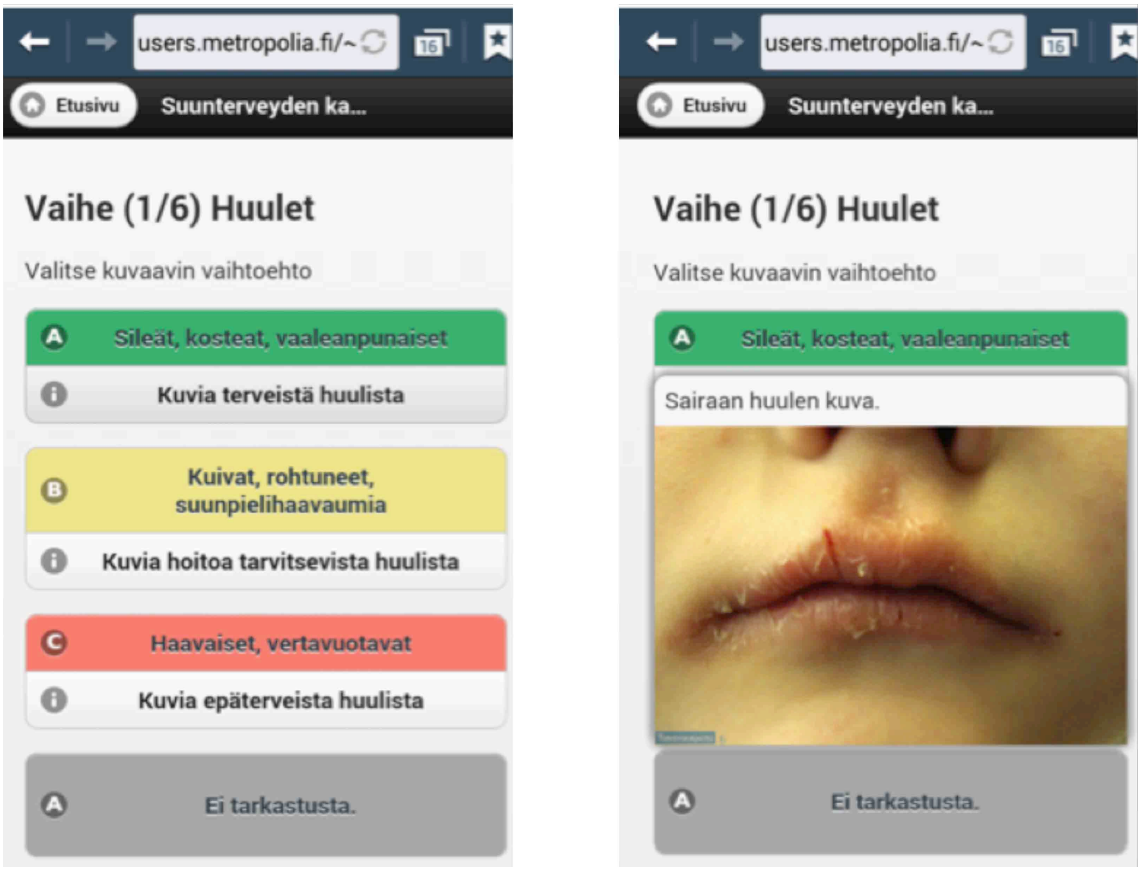
riptiä. Tällöin näkymään tulee koko yksisivuisen sivuston sisältö eivätkä JavaScriptillä luodut toiminnallisuudet toimi. Mikäli jokin osa ladataan ruudulle vasta jQuery Mobilen lataamisen jälkeen, se saattaa näyttää poikkeavalta muuhun ulkoasuun nähden, sillä kehyksen toiminnallisuudet eivät ehdin vaikuttaa siihen. Tämän ongelman kanssa jouduin painimaan luodessani oman JavaScript-koodin avulla dynaamisesti sisältöä perustuen sovelluksessa olevan lomakkeen täyttöön. Ongelma ilmenee, kun lomakkeen yhteenvedon sisältöä tuotetaan lomakkeen täytön mukaan.

Kuten aiemmin mainittu, jQuery Mobile tekee käyttöliittymästä esteettisesti miellyttävämmän. Se pyöristää automaattisesti terävät kulmat esimerkiksi napeissa ja tarjoaa valmiiksi oivan valikoiman navigointia avustavia nappeihin upotettavia kuvakkeita. Yleisimmät esimerkit näistä lienevät x poistamiseen ja i informaatioon sekä rattaan kuva asetusnappia kuvaamaan. Kuva 6 näyttää kaikki kehyksen tarjoamat kuvakkeet. [18.]



Kuva 6. jQuery Mobilen peruskuvakkeet [18].

Kuvakkeita ei ole kuitenkaan sidottu mihinkään tiettyyn toimintoon, joten niiden käyttökohde on täysin kehittäjästä kiinni. Estetiikka on hyvin pieni osa jQuery Mobilen hyödyllisyyttä. Se tarjoaa useimmille mobiilikäyttöliittymille tyypillisiä kuvan 7 mukaisia ponnahdusikkunoita, vetovalikoita, ynnä muita käyttöä helpottavia toimintoja.



Kuva 7. Lomakkeen näkymä ilman ponnahdusikkunaa ja sen kanssa.

Tärkeää on huomata, että moni sovelluksen ydintoiminnoista tarvitsee toimiakseen JavaScriptiä. Tämä tarkoittaa, että mikäli toimintavarmuus halutaan säilyttää, on sovellusta pystyttävä käyttämään selaimella, jossa tämä on mahdollista. Sovellus toimii auttavasti ilman sitä, mutta esimerkiksi lomakkeen täyttö tai videoiden navigaationapit eivät toimi tarkoituksenmukaisesti. Videon pystyy silti katsomaan ja sovelluksen koko sisällön näkemään.

3.4 Sovelluksen rakenne

Arkkitehtuuriltaan sovelluksen prototyyppi vastaa hyvin suurelta osin niin kutsuttua Fat-Client-toteutustapaa, jonka määritelmän mukaan suuri osa tai koko sovelluslogiikka on asiakaspäässä. Tällä tavalla toteutettua asiakassovellusta on mahdollista käyttää täysin tai lähes toimivana ilman toimivaa verkkoyhteyttä. Mikäli sovelluksen tiedot saadaan ladattua selaimesta verkkoyhteyden kanssa, yhteyden katketessa kaikki ladatut

tiedot ovat vielä käytettävissä. Näin ollen esimerkiksi sovelluksen lomakkeen pystyy täyttämään. [31; 32.]

Sovelluksen selainversio on periaatteessa sekoitus staattista ja dynaamista www-sivumallia, eli sovelluksen sisältö on tuotettu jo valmiiksi. Kun selain lähettää palvelupyynnön palvelimelle, sen ei tarvitse prosessoida pyyntöä, vaan se voi lähettää saman tien vastauksena sivuston sisältöineen. Sovellus ei kuitenkaan pysy muuttumattomana JavaScriptin ansiosta, vaan reagoi käyttäjän toimintoihin muuttamalla jo valmista sisältöä.

Suurin osa jQuery Mobilen tarjoamista layout-elementeistä ovat DIV-elementtejä, joiden ulkomuoto ja rooli määritellään attribuutilla ”data-role” . Koodiesimerkissä 2 on luotu yksinkertainen sivu, jossa ylä- ja alatunniste. DIV-elementtien sisäinen sisällön muotoilu tehdään standardien HTML-elementtien avulla, kuten otsikointi h1–h5 ja kappaleet p-elementin avulla. DIV-elementtien sisälle voidaan rakentaa sisältö ja navigointi helposti käyttämällä listoja ja nappeja. Listaelementit ovat tietysti UL-elementtejä, joille voidaan myös määrittää rooli ja tyyli. Esimerkiksi nappeja sisältävän elementit voi järjestää allekkain tai vierekkäin. Napit ovat kehyksen avulla muotoiltuja linkkejä lähes jokaisessa tapauksessa.

```
<body>
<div data-role="page">
           <div data-role="header">
                      <h1>My Header</h1>
           </div><!-- /header-->

           <div data-role="content">
                      <p>Hello world</p>
           </div><!-- /content-->

           <div data-role="footer">
                      <h1>My Footer</h1>
           </div><!-- /footer-->
</div><!-- /page-->
</body>
```

Koodiesimerkki 2. Yksinkertaisen jQuery Mobilella luodun sivun lähdekoodi.

Mikäli kehittäjä haluaa muuttaa jQuery Mobilen CSS-tyylimääritystiedostosta automaattisesti tulevia muotoiluja, tämä onnistuu yksinkertaisesti lataamalla ohjelmakoodissa omat määritykset kehyksen määritysten latauksen jälkeen, jolloin kehittäjän määritykset korvaavat kehyksen määritykset. Tällä tavoin kehittäjä voi korvata jo tehdyt määritykset. Tämäkin on suhteellisen tavallinen käytäntö määrittäessä kaikkien web-sovellusten tyyliä.

jQuery Mobile tarjoaa myös niin kutsuttuja elinkaarifunktioita, joiden avulla voi suorittaa toimintoja tietyn näkymän latautuessa ilman käyttäjän interaktiota. Tämä toiminto on erityisen kätevä varsinkin, kun koko sovelluksen käyttöliittymä on kuvattu yhdessä dokumentissa. Elinkaarifunktiot ovat hyvin tyypillisiä mobiileissa käyttöliittymissä. Esimerkiksi kun Androidille ohjelmoidaan Java-kielellä, tiettyä näkymää ladattaessa tapahtuu tietty funktio, ja mikäli sama näkymä peittyy toisella näkymällä osin tai kokonaan, suoritetaan automaattisesti joko edellisen näkymän tauotus tai tuhoaminen. Näissä on hyvä muistaa käyttäjän lisäämien tietojen mahdollinen tallentaminen tai turhan tiedon hävittäminen muistin vapauttamiseksi tietyissä sovelluksen elinkaaren vaiheissa. jQuery Mobilen elinkaarielementit ovat sidottuna moniin kosketus-, hiiri- ja työpöytäelementteihin, joten samat funktiot toimivat niin mobiili- kuin työpöytäympäristössä. Toisin kuin jQuery-kehyksessä, jQuery Mobilen funktiot voidaan valjastaa käyttöön $(document).bind('pageinit'){ }-funktiolla $(document).ready()-funktion sijaan. Bind-funktiolle annetaan halutun tapahtumaobjektin nimi, kuten pageinit, pagecreate, pagebeforeload tai pageloadfailed. Elinkaarifunktioita on niin sanotusti jokaiseen tarpeeseen. Bind-funktiota kotihoitoa avustavan sovelluksen lomakkeessa on käytetty useaan otteeseen. Jokainen lomakkeeseen sidotun taulukon solu tyhjennetään näkymää ladattaessa. Näin varmistetaan, ettei lomakkeeseen tule kaksinkertaisia merkintöjä. [33.]

On hyvä muistaa, että myös PhoneGap tarjoaa elinkaarifunktioita, jotka toimivat todennäköisesti jQueryn elinkaarifunktioita varmemmin sillä luodussa sovelluksessa. Näillä funktioilla voi hallita kuitenkin esimerkiksi useampaa sivutiedostoa ja saada ohjelmasta näin enemmän natiivin ohjelman kaltaisen. PhoneGapin elinkaarifunktiot tunnistavat myös, mikäli internetyhteys on saatavilla tai katkeaa. Sovellukselle voidaan tehdä yhteyden katketessa senhetkisen tilan tallennus ja jatkaa myöhemmin samasta kohdasta, mikäli sovellus vaatii toimiakseen yhteyden.

Prototyypissä keskityttiin pelkästään perustoimintojen kehittämiseen, joten sovelluksen suunnittelussa ei tarvinnut varautua käyttäjämäärien vaihteluun. Todennäköistä on, että käyttäjämäärät ovat helposti ennustettavissa, sillä jokainen asiakasyritys lataa sovelluksen omalle palvelimellensa ja se on pelkästään sen työntekijöiden käytettävissä. Sovellus sisältää mahdollisuuden lähettää hoidettavan henkilön suun tutkimustuloksen, nimen ja tutkimuspäivämäärän sähköpostitoiminnolla eteenpäin, mutta toistaiseksi ei ole määritetty osoitetta, mihin posti lähtee eteenpäin. PhoneGap sisältää itsessään mahdollisuuden listata osoitteita tai resursseja, joita sovellus saa käyttää. Muut kuin kehittäjän listaamat resurssit ovat automaattisesti estettyjä. Tästä menetelmästä käytetään nimitystä whitelisting. Sovellusta kehitettäessä on pääpaino asetettu käyttöliittymän ja teknisen toteutuksen suunnittelulle enemmän kuin itse tekniselle toteutukselle.

3.4.1 jQuery Mobilen avulla luodun prototyypin malli

Ohjelmakoodin alussa head-elementtien sisällä on haettu jQuery Mobilen CSS-tiedosto, jQuery-kehys ja jQuery Mobilen kehys koodiesimerkin 3 mukaisesti.

```
<link rel="stylesheet"
href="http://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.css">

<script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
<script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.js"></script>
```

Koodiesimerkki 3. Ulkoisen tyylimäärityksen ja JavaScriptin lisääminen.

Head-elementissä on myös muutamia olennaisia tyylimäärityksiä CSS:llä.
Body-elementissä on ladattu koko sovelluksen ennen toimenpiteitä näkyvillä oleva sisältö.

3.4.2 Lomake

Kotihoitoa avustavan sovelluksen suun terveyden lomake on tarkoitettu hoitajien muistin avuksi ja varmistamaan tutkimuksen laatua. Lomake pyrittiin toteuttamaan mahdollisimman yksinkertaiseksi käytettävyydeltään ja toimintavarmuudeltaan.

Toiminta toteutettiin JavaScriptillä niin, että jokainen tutkittava alue on omassa näkymässään. Tutkittavat osa-alueet ovat seuraavat:

- huulet
- suun limakalvot, ikenet
- kieli
- hampaat
- proteesit
- sylki.

Kuten mainittu, jokainen osa-alue sisältää neljä valintavaihtoehtoa, joista valitaan tutkimuksen perusteella hoidettavan suun kuntoa lähinnä vastaava vaihtoehto. Vaihtoehdot merkittiin liikennevaloista tutulla värikoodilla ja erotettiin toisistaan kirjaimen sisältävällä kuvakkeilla selkeyden vuoksi.

Mikäli tutkittava alue on kunnossa, valitaan A-vaihtoehto, jolla on vihreä värikoodi. Jos keltaisella värikoodilla merkitty vaihtoehto valitaan, se tarkoittaa, että kotihoitajan tekemä toimenpide riittää. Punaisen värikoodin valitseminen tarkoittaa, että tutkittava alue on niin huonossa kunnossa, että se vaatii ammattilaisen konsultointia. Viimeinen vaihtoehto on ”ei tutkimusta”, jolla on harmaa värikoodi.

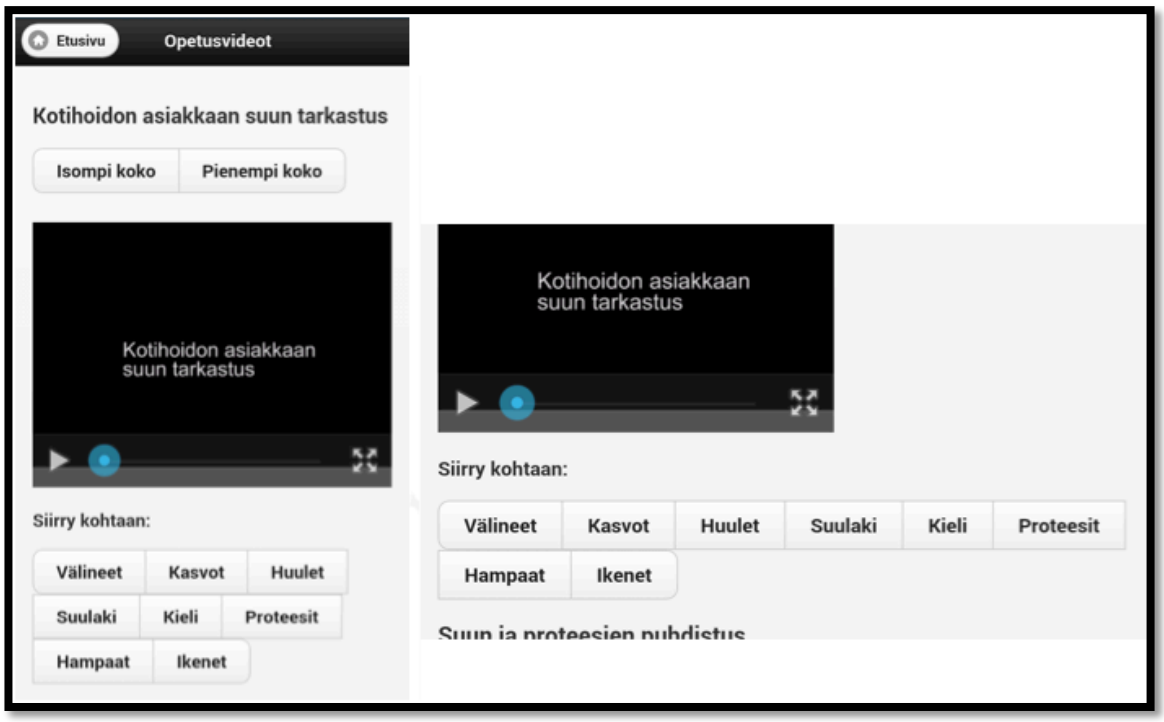
Lomakkeen vaihtoehtojen painaminen laukaisee funktion, joka täyttää JavaScriptillä taulukko-oliota yhteenvetoa varten käyttäjän valintojen mukaan. Yksisivuinen sovellusrakenne mahdollistaa taulukon käytön tiedon väliaikaiseen varastointiin, sillä koko sivua ei missään vaiheessa päivitetä uudelleen. Tärkeää on kuitenkin muistaa tyhjentää taulukko tai sen solu tarvittaessa. Mikäli käyttäjä siirtyy kesken lomakkeen täytön etusivulle, tyhjennetään taulukko. Jokaiselle lomakkeen näkymälle on määritetty oma solunsa taulukossa, joten aina kun näkymä ladataan, solu tyhjennetään varmuuden vuoksi. Näin vältytään päällekkäisiltä merkinnöiltä. Taulukon tyhjentäminen suoritetaan varmuuden vuoksi kahdella eri funktiolla virhetilanteiden varalta. Toinen funktioista on sidottuna jQuery Mobilen näkymälatauksen elinkaarifunktioon, ja toinen käynnistetään napin painalluksesta.

3.4.3 Videot

Sovellukseen lisättiin tässä kehitysvaiheessa kaksi videota. Ensimmäinen opastaa kotihoidon työntekijää tutkimaan ikäihmisen kasvojen alueen ja suun oikeaoppisesti, toinen suorittamaan suun ja proteesien puhdistuksen.

Videoiden käyttäminen tuottaa haasteita suhteessa muuhun sisältöön suuren koon takia. Tästä syystä sovelluksen käyttötarkoitus on pidettävä muistissa videon formaattia ja muita laatuun ja kokoon vaikuttavia tekijöitä määritettäessä. Näitä tekijöitä ovat muun muassa näytteenottotaajuus, bittisyvyys, bittinopeus (prosessoitujen bittien määrä sekunnissa), videoiden lukumäärä ja resoluutio. Koska käyttölaite on pääasiassa mobiili, voi videon resoluutio olla suhteellisen pieni, mutta samasta syystä videoiden muistista viemän tilan täytyy olla myös pieni suhteessa videon pituuteen, käytettiin videota sitten natiivissa sovelluksessa tai selaimesta. Ensimmäisessä tapauksessa videot saattavat viedä laitteen omasta muistista liikaa tilaa, ja toisessa tapauksessa videon lataaminen palvelimelta on mahdollisesti liian hidasta ja aikaa vievää. Vaikka kyseessä olisi streamaava palvelin, saattaa hitaalla internetyhteydellä videon lataaminen vaatia liikaa aikaa kotihoitajan kiireisestä aikataulusta. Toisaalta selaimella käytettäessä voidaan ladata samasta videosta useita erikokoisia ja -laatuisia versioita, joita voidaan ladata riippuen asiakaslaitteen verkkoyhteyden nopeudesta.

Videoiden fyysistä kokoa näytöllä voi käyttäjä itse halutessaan muuttaa itselleen sopivaksi. Koon suurentaminen ja pienentäminen on toteutettu yksinkertaisesti napeilla kuvan 8 mukaisesti. Sovittaminen tehtiin tavallisella JavaScriptin funktiolla, joka muuttaa videoikkunan tyylimääritysattribuutteja. Videoiden minimikoko määritettiin 20 pikseliin. Maksimikokoa ei ole.



Kuva 8. Videosivu pysty- ja vaaka-asennossa.

3.5 Testaus ja virheenkorjaus

3.5.1 Virrankulutus

Hoitosovelluksen virrankulutus on tärkeä minimoida, sillä kotihoitajan päivä saattaa olla pitkä ja akun olisi hyvä riittää työpäivän ajan. Jatkuva internetyhteys mobiililaitteessa kuluttaa akkua nopeasti, minkä takia oli syytä tutkia kotihoitoa avustavan sovelluksen selain- ja natiiviversion virrankulutuksen eroja. Kotihoitajan aika on hyvin rajallista käyntiä kohden, joten sovelluksen käytön on oltava myös mahdollisimman nopeaa. Tutkimuksen tarkoituksena oli vertailla akun kulutusta ja käytön nopeutta molemmissa ympäristöissä.

Taulukossa 2 on vertailtu natiivin sovelluksen ja selainpohjaisen sovelluksen suhteellista akun kulutusta ja kestoa ennalta määrätyn toimenpiteen suorittamiseen. Tutkimusasetelmana oli suorittaa ennalta määritetyt toimenpiteet Samsung Galaxy S III -puhelimella. Tehtävänä oli navigoida etusivulta kolmeen eri valikkoon vuorollaan ja jokaisen jälkeen takaisin etusivulle ja lopuksi sulkea sovellus. Käytettyä aikaa ja akunkäyttöä mitattiin GSam Battery Monitor -mobiilisovelluksella, joka antaa tarkat tiedot

sovelluksen käyttöajasta ja akunkäytöstä suhteessa kokonaisuudessaan käytettyyn aikaan mittausaikana. Mittaustilastot nollattiin joka mittauksen jälkeen, ja ohjelmat pakotettiin sulkeutumaan asetuksista mittauskertojen välissä. Mittaukset suoritettiin puhelimen ollessa samassa paikassa, joten paikasta johtuva web-yhteyden signaalin siirtonopeus ei päässyt vaikuttamaan mittaustulosten luotettavuuteen. Puhelimen muita sovelluksia ei avattu eikä suljettu testauksen aikana, joten niiltäkin osin testiympäristö pysyi muuttumattomana.

Taulukko 2. Natiivissa ja selainympäristössä toteutetun akkutestin tulokset.

	Natiivi		Selain	
Mittauskerta	Akku (%/stot)	Kesto (s)	Akku (%/stot)	Kesto (s)
1	16,3	10	26,9	17
2	17,7	11	20,1	15
3	9,9	10	20,1	16
4	12,1	9	21,5	14
5	17,3	9	27,7	12
6	16,5	9	24,6	14
7	20,4	9	31,7	15
8	16,1	8	26,8	14
9	15,2	9	25,8	13
10	17,1	9	25,3	13
Yhteensä	158,6	93	250,5	143
Keskiarvo	15,86	9,3	25,05	14,3
Moodi	-	9	20,1	14
Mediaani	16,4	9	25,55	14
Keskihajonta	2,950781441	0,823272602	3,653993249	1,494434118

Ajan moodien ja keskiarvojen erotuksesta voidaan helposti päätellä, että selainpohjainen sovelluksen käyttö vie keskimäärin noin 55,56 prosenttia enemmän aikaa suhteessa natiivin sovelluksen käyttöön. Testissä käytetyn sovelluksen versioissa ei ladattu sisällön osalta muuta kuin teksti, joten pääteltävissä on, että sisällön koon kasvaessa myös latausajat hidastuvat selaimessa, mikä tarkoittaa suhteellisen käyttöajan pitenemistä entisestään.

Keskihajonta kertoo mittaustulosten keskimääräisen poikkeamisen otannan keskiarvosta. Akun kulutuksen ja ajan keskihajonnoista voidaan päätellä, että kun samaa toimintoa toistetaan useita kertoja, toimintoon kulunut aika ei vaihtele kovin merkittävästi.

3.5.2 Käyttöliittymä

Kehitysvaiheessa sovelluksen navigaatiota, lomaketta ja videoiden katsomista testattiin aktiivisesti. Kokeilin itse toiminnot jokaisen muutoksen jälkeen ja annoin tasaisin väliajoin päivitetyn version testattavaksi testihenkilöille. Testauskierroksia testihenkilöiden kanssa oli neljä.

Testihenkilöillä oli käytössään Nokian N8 (Symbian)-, Samsung Galaxy S3 (Android)- ja iPhone 4 (iOs) -puhelimet. Näin saatiin testattua skaalautuvuus ja toimivuus suosituimmilla alustoilla. Natiiviversiota testasin vain itse Android-puhelimella, sillä muille alustoille ei kehitetty omia versioita. Testihenkilöille annettiin erilaisia tehtäviä sovelluksen käytössä, minkä jälkeen henkilöt raportoivat kehittäjälle suoriutumisestaan ja virheistä tehtävän aikana.

3.5.3 Testausasetelma

Testaustilanteessa testihenkilöille annettiin ennalta määrättyjä tehtäviä, joiden suorittamista seurasin vierestä. Testauksen jälkeen testihenkilöt raportoivat kehittäjälle omia huomioitaan ja parannusehdotuksia sovellukseen liittyen.

Kierros 1. Yleinen navigaatio, rakenne ja intuitiivisuus

Testihenkilöt saivat käyttää sovellusta haluamallaan tavalla, ja sen aikana ja jälkeen he antoivat arvioita käytön helppoudesta ja intuitiivisuudesta navigaatiosta ja rakenteesta.

Tulos: Sovelluksen käyttöliittymässä ei ollut testihenkilöiden mielestä muuta korjattavaa, kuin lomakkeen kuvakkeet ja yleinen värimaailma. Lomakkeen kuvakkeet olivat aluksi kaikissa kohdissa samat. Testikierroksen jälkeen ne vaihdettiin selkeämmiksi A-, B-, C-vaihtoehdoiksi tuomaan selkeämmin esille, että napeista valitaan yksi vaihtoehto. Testikäyttäjien toivomuksesta värimaailma vaihdettiin harmaa- ja valkosävyisestä sinertäväksi.

Kierros 2. Videoiden katsominen, videon navigaationapit

Testihenkilöitä ohjattiin navigoimaan videot-sivulle ja katsomaan opastusvideot läpi. Tämän jälkeen he saivat navigoida hyppää kohtaan-napeilla kohtaan, joka jäi heille epäselväksi.

Tulos: Tehtävän suorittaminen onnistui erinomaisesti kaikilta testihenkilöiltä. Ainoana korjausehdotuksena toivottiin hyppää kohtaan-nappien ulkonäön muuttamista ”kauniimmaksi”. Muutosta ei toteutettu, sillä ehdotuksen tyyppinen korjaus olisi vienyt liikaa tilaa näytöltä.

Kierros 3. Lomakkeen täyttäminen, virhepainallus lomakkeessa ja virheen korjaus

Testihenkilöt ohjattiin navigoimaan sovelluksen lomakkeeseen ja täyttämään se. Henkilöt saivat päättää missä kohtaa lomaketta tekevät ”virhepainalluksen” ja navigoivat takaisin virhepainalluskohtaan sekä valitsevat ”oikean” vaihtoehdon.

Tulos: Yksinkertaisuudesta ja selkeydestä huolimatta lomakkeen täyttäminen ei ollut etenkään ikääntyneemmille käyttäjille selvää. Tästä syystä lomakkeen alkuun toivottiin ohjeikkunaa, joka ilmestyy lomakkeen ensimmäisellä sivulla automaattisesti ja opastaa täytössä. Kun lomakkeen täyttöperiaate kerrottiin, onnistui sen täyttäminen ja virhepainalluksen korjaus moitteettomasti.

Kierros 4. Lomakkeen yhteenvetosivun täyttö

Lomake oli lähtötilanteessa täytetty kaikille samalla tavalla. Testihenkilölle avattiin yhteenvetosivu valmiiksi ja henkilö täytti asiakkaan nimen, rastitti toimenpideluettelosta tehdyt toimenpiteen ja otti kuvan oireesta ja lisäsi sen lomakkeeseen.

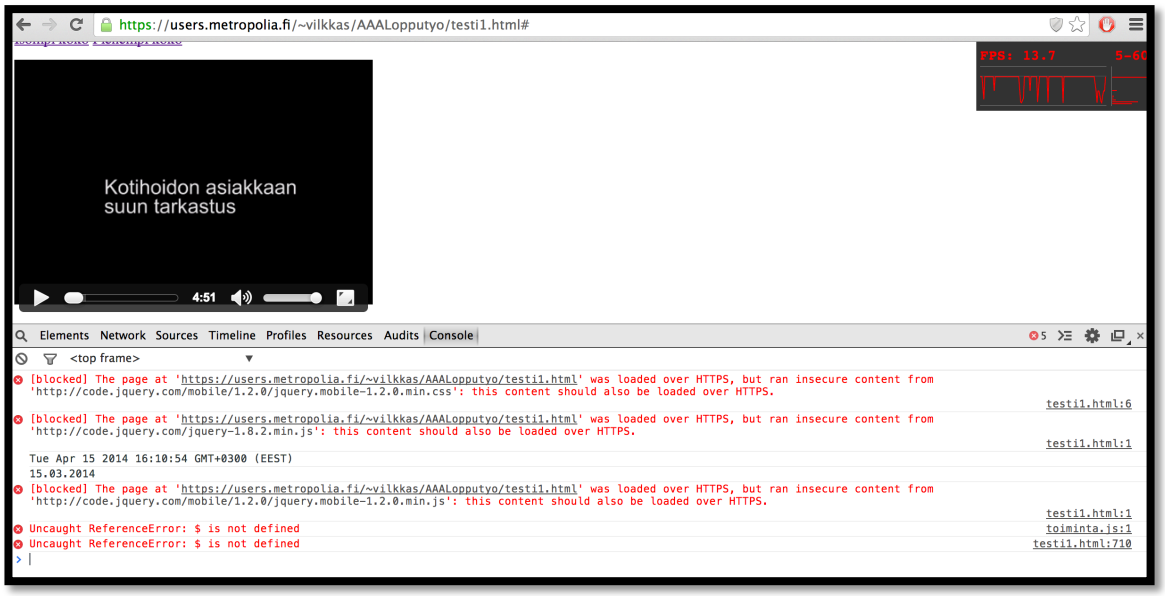
Tulos: Yhteenvetosivun täyttäminen oli niin helppoa, että kaikki testihenkilöt onnistuivat siinä moitteettomasti. Ainoa korjattava asia oli yhteenvedon ”rasti ruutuun” -osion rastituslaatikoiden liian suuri koko iPhonen näytöllä. Tämäkin korjattiin viimeiseen versioon.

Tekniikasta johtuvia virheitä testaustilanteissa ei esiintynyt ensimmäistäkään, todennäköisesti siitä syystä, että sovelluksen prototyypin rakenne oli vielä suhteellisen yksinkertainen. Ulkoasun puolesta testihenkilöt antoivat arvioita lähinnä värimaailman suh-

teen, joka ei ollut heidän mielestään paras mahdollinen. Sovelluksen viimeiseen versioon tätäkin muutettiin.

Kehitettäessä virheitä etsittiin Google Chrome -selaimen erinomaisilla kehitystyökaluilla. Kuvassa 9 esitetty näkymä Chrome -selaimen (Console-ikkunasta alhaalla) ja ruudun päivitysnopeudesta (oikealla yläkulmassa). Päivitysnopeus näytetään muodossa ruutua sekunnissa. Työkalun mukaan jQuery Mobilen CSS-tiedostoa ei voida ladata, sillä sivu on ladattu salatulla yhteydellä, mutta tyylitiedostoa ei. Tähän ratkaisuna voi joko ladata tyylitiedoston samalle palvelimelle sivutiedoston kanssa tai käyttää salaamatonta yhteyttä.

Käytettäessä jQuery Mobilea jQueryn tarjoamalta palvelimelta saadaan Consolella myös ohjeita ja informaatiota tehdyistä päivityksistä. Sitä voi käyttää siis yksisuuntaisena tiedonvälityskanavana jQuery Foundationin kehittäjien ja muiden sovelluskehittäjien välillä.

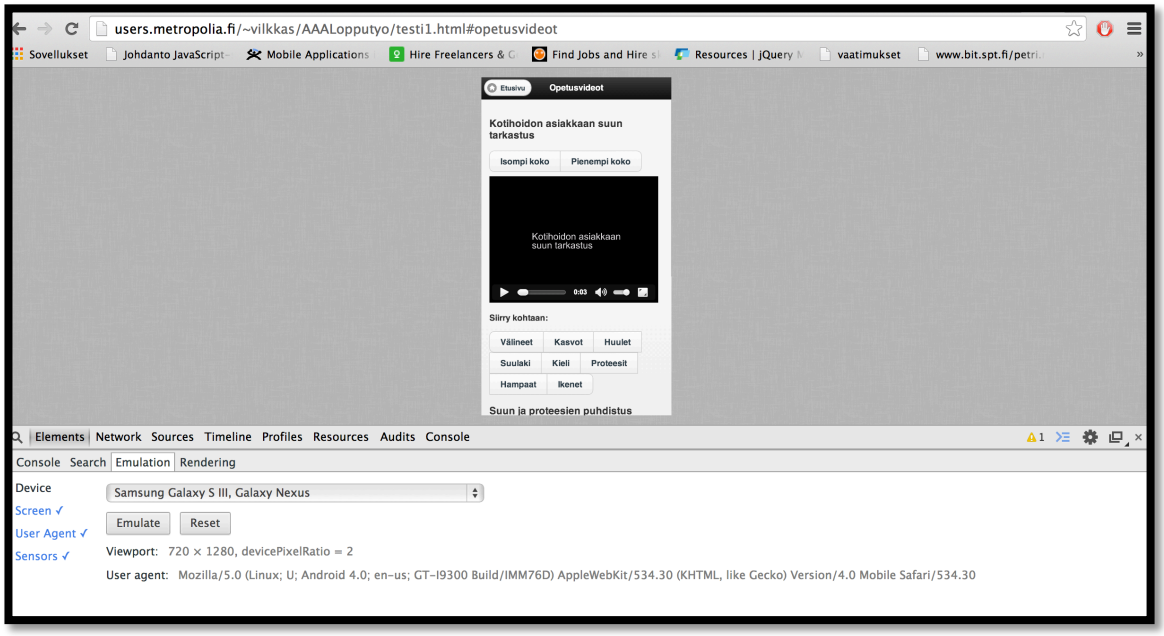


Kuva 9. Chrome-selaimen Console-kehitystyökalu.

Chromen uusimmat kehitystyökalut tarjoavat myös simuloinnin eri päätelaitteiden näyttöjen resoluutiolla. Ominaisuus on uusi eikä toimi kunnolla, joten kun emuloitava laite on valittu ja emulointi käynnistetty, on sivusto päivitettävä vielä, jotta jQuery Mobilen avulla oikeankokoiseksi skaalattu näyttö saadaan näyttämään siltä, miltä sen kuuluukin näyttää. Ilman päivittämistä emulaattori yrittää näyttää web-sivuille tarkoitettua näky-

mää puhelimen resoluutiolla, joten emulaattori antaa virheellisen kuvan. Emulaattori antaa myös kehittäjän käyttöön yksinkertaisten kosketustapahtumien, paikannuksen koordinaattien syöttämisen sekä kiihtyvyysanturin ja orientaatiomuutoksien emuloinnin. Työkalut mahdollistavat testaamisen eri mediatyypeillä, mikäli niitä on CSS-tiedostoissa määritetty.

Yksi merkittävä uudistus on myös mahdollisuus testata sovellusta eri selainmoottoreilla. Mukana on kaikkien yleisimpien selainten simuloinnit. Kuvassa 10 nähdään videoikkuna emuloituna selaimessa Samsung Galaxy S III:n näytölle.



Kuva 10. Chrome-selaimen emulointityökalu.

3.6 Merkittävimmät ongelmat ja niiden ratkaisut

Kotihoitoa avustavan sovelluksen kehityksen aikana kehittämiseen käytetty tietokone rikkoutui ja jouduttiin vaihtamaan uuteen. Tämä tarkoitti OS X -käyttöjärjestelmän päivittämistä Maverics-versioon, mikäaiheutti huomattavasti ongelmia PhoneGapin kanssa. Ongelmaa selvitettäessä kävi ilmi, että Mavericsista puuttui Apache Software Foundationin ANT-sovellustyökalu, jolla automatisoidaan eri ohjelmistojen kokoamisprosesseja. Työkalu käyttää XML-merkintätapaa prosessien ja riippuvuuksien kuvaa-

miseen. Sovelluksen suhteellisen yksinkertaisuuden vuoksi itse kehitystyössä ei mittavia ongelmia esiintynyt.

4 Johtopäätökset ja tulevaisuus

Insinöörityön tavoitteena oli kehittää tiiviissä yhteistyössä johdannossa mainittujen yhteistyökumppanien kanssa toimiva kotihoitoa avustavan mobiilisovelluksen prototyyppi. Prototyypin tuli kyetä aluksi palvelemaan kotihoidon työntekijöitä opastamalla suun ja sen lähialueiden hygienian tutkimisessa ja hoitamisessa.

Sovelluksen teknisinä vaatimuksina oli alustariippumattomuus, responsiivisuus ja mahdollisuus tuottaa sekä selain- että natiivisovellus eri laitteille. Sovelluksen käytettävyys ja sen helppous olivat tärkeimpiä aspekteja prototyypissä, sillä käyttäjäryhmien teknisen osaaminen voi olla mitä hyvänsä ja sovellusta on silti osattava käyttää. Virheet sovelluksen tietyillä osa-alueilla saattavat tarkoittaa myös virheellistä hoitoa tai pahimmissa tapauksissa hoidettavan tilan huononemista tai hoidon puutetta.

Nykypäivän sovelluskehityksen etuna on kehitystyökalujen ja tekniikoiden nopea kehittyminen ja suuri valikoima. Lähes kaikkiin tarkoituksiin löytyy oma työkalu, ja jokaiseen kehittäjän kohtaamaan ongelmaan on yleensä keksitty jo ratkaisu, joka löytyy useimmiten lyhyen tiedonhaun jälkeen. Työkalujen käyttö ja tekniikat helpottuvat koko ajan. Tämä aiheuttaa myös hankaluuksia valittaessa omiin tarkoituksiin parhaiten istuvaa tekniikkaa. Projektin valmistumisen jälkeen törmäsin Zurbin Foundation -nimiseen responsiiviseen ohjelmointikehykseen. Kehykseen tutustuessani huomasin, että se tarjoaa ainoastaan ”rautalankamallin” kehyksen asettelulle ja loppu on paljon laajemmin kehittäjän määriteltävissä verrattuna jQuery Mobileen.

Ongelmia kehityksen aikana aiheutui oikeastaan pelkästään kehityslaitteiston rikkoutumisesta. Haastavinta oli ehdottomasti päätteen käyttö ja siihen liittyvän PhoneGapin asennusprosessi. Tämä johtui aiemman osaamisen puutteesta. Prosessin olisi pystynyt ohjeita seuraamalla suorittamaan tajuamatta mitä tekee ja saamaan silti onnistuneen lopputuloksen, mutta virheiden sattuessa niiden korjaaminen olisi ollut erittäin suuritöistä. Myös sovelluksen testaus ja virheiden korjaus paketoidulla sovelluksella osoittautui haastavaksi. Huolelliset valmistelut ja riittävä dokumentaatioon tutustuminen ennen

kokoamista auttoivat tähän ongelmaan. JavaScript-koodin voi tarkastaa monilla internetistä löytyvillä palveluilla.

Testauksessa sovelluksen käytettävyys osoittautui jo ensimmäisessä versiossa hyväksi. Korjaukset testien perusteella olivat lähinnä kosmeettista viimeistelyä, kuten lomakkeen nappien kuvakkeiden vaihto. Myös teknisen toteutuksen taso oli yhteistyökumppaneiden mukaan riittävä prototyypille. Itse kehittäjänä olisin toivonut hieman haastavampaa projektia vaikka tässäkin oppi huomattavan määrän uutta usealle alustalle samanaikaisesta kehittämisestä ja responsiivisista tekniikoista. Päätelmänä voi myös todeta avoimen lähdekoodin monialustakehittämisen olevan itse vielä kehitysvaiheessa eikä niin sanottua täydellistä ratkaisua ole olemassa. Tähän vaikuttaa osin työkalujen tarjoajien voitontavoittelu. Tulevaisuus vaikuttaa silti lupaavalta.

Mikäli sovellusta aiotaan kehittää eteenpäin jatkossa ja yhdistää se eri palvelinpuolen taustajärjestelmiin, kuten eri potilastietojärjestelmät, tietovarastot ja seuranta- ja tilastointisovellukset, pitää arkkitehtuuria miettiä uudestaan tietoturvan näkökulmasta huomattavasti tarkemmin. Kyseessä on kuitenkin hyvin arkaluontoinen data.

Prototyyppi antaa erinomaisen lähtökohdan laajemman sovelluksen laajemmalle suunnittelulle ja toteutukselle ja sen avulla voidaankin pyrkiä hakemaan EU:lta apurahaa jatkokehitykselle. Tulevaisuuden mahdollisuudet ja sovelluksen lopullisen version mukanaan tuomat hyödyt käytettyjen tekniikoiden kanssa ovat hyvin pitkälle tarpeista kiinni, jolloin on mahdollista suunnitella viimeistelty ulkoasu ja graafinen ohjeistus jokaiseen tarpeeseen erikseen. Ulkoasun puolesta prototyypin tavoitteena olikin tarjota sapluuna, jossa käytettävyys on mietitty tarkkaan ja esteettinen ilme jätetty suhteellisen karkeaksi.

Tulevaisuudessa sovelluksella on mahdollista tilastoida ikäihmisten terveyttä yksityiskohtaisesti henkilötasolla, kaupunginosittain, maakunnittain ja muodostaa datamalleja sekä ennakoida tulevaisuuden kehitystä. Optimistisissa arvioissa tämä johtaa hoidon paranemiseen ja tehokkuuteen niin hoidossa kuin kustannuksissa. Hoito nopeutuu ja tehostuu ja hoitovirheet vähenevät.

Kaiken kaikkiaan projekti onnistui ja alussa asetettuihin tavoitteisiin päästiin. Prototyyppi on toimiva niin selaimessa kuin Android-ympäristössä natiivina sovelluksena. Sen

käytön oppiminen on huomattavan nopeaa ja intuitiivista, ja se antaa suuntaviivoja tulevaisuuden kehityksen kannalta. Sovelluksen elinkaarta ja ylläpitoa ajatellen on syytä punnita tarkkaan, jatkaako kehitystä selaimelle vai käyttöjärjestelmäkohtaiselle ympäristölle. Selaimen elinkaari on pidempi ja päivitys helpompaa.

Lähteet

1 Keskinen, Helinä. 2009. Ikääntyneiden suuhygienia. Verkkodokumentti. Terveyskirjasto. <http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=trs00065>. Päivitetty 15.9.2009. Luettu 20.3.2014.

2 Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018. 2014. Verkkodokumentti. Cisco Mobile Networks. <http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=trs00065>. Päivitetty 15.9.2009. Luettu 20.3.2014.

3 CareApp-mobiilisovellukset. 2014. Verkkodokumentti. Tunstall-tuotteet. Tunstall Oy. <http://www.tunstallnordic.com/fi/tuotteet/tunstallin-tuotteet/mobiili-it-tuki-careapp>. Päivitetty. Luettu 1.4.2014.

4 Murphy, Mike. 2014. Move over mobile - why successful apps start with the consumer, not the device. Verkkodokumentti. Gfk MAgazine. <http://www.gfk.com/magazine/talk/talk-what-is-the-big-next-future-proofed-product-innovation/move-over-mobile-why-successful-apps-start-with-the-consumer-not-the-device>. Luettu 1.4.2014.

5 Tablet dimensions infographic. 2014. Verkkodokumentti. AppStudio. <http://www.appstudio.net/pdfs/tablet_dimensions_infographic.pdf>. Päivitetty 7.2.2014. Luettu 10.4.2014.

6 Korpela, Jukka. 2012. Mitä HTML5-sovellukset ovat. Verkkodokumentti. HTML5-kirja. <http://html5kirja.fi/2012/08/31/mita-html5-sovellukset-ovat/>. Päivitetty 31.8.2012. Luettu 8.4.2014.

7 Mitä HTML5-sovellukset ovat. Verkkodokumentti. 2014. w3schools. <http://www.w3schools.com/html/html5_webstorage.asp>. Luettu 8.4.2014.

8 HTML5. Verkkodokumentti. 2014. W3C. <http://www.w3.org/tr/html5/>. Päivitetty 4.2.2014. Luettu 24.3.2014.

9 The HTML DOM (Document Object Model). Verkkodokumentti. w3schools. The World Wide Web Consortium. <http://www.w3schools.com/js/js_htmldom.asp>. Luettu 24.3.2014.

10 DOM: Document Object Model veppikoodaajan näkökulmasta. Verkkodokumentti. 2008. Ohjelmointiputka. <http://www.w3schools.com/js/js_htmldom.asp>. Luettu 16.3.2014.

11 AJAX Introduction. Verkkodokumentti. w3schools. The World Wide Web Consortium. <http://www.w3schools.com/js/js_htmldom.asp>. Luettu 16.3.2014.

12 JavaScript and the ECMAScript Specification. Verkkodokumentti. 2014. Mozilla Developer Network. <https://developer.mozilla.org/en-us/docs/web/javascript/guide/javascript_overview#javascript_and_the_ecmascript_specification sulkukiinni. Päivitetty 7.3.2014. Luettu 22.3.2014.

13 Usage of JavaScript libraries for websites. Verkkodokumentti. 2014. w3 techs. The World Wide Web Consortium. <http://w3techs.com/technologies/overview/javascript_library/all>. Päivitetty 23.4.2014. Luettu 23.4.2014.

14 How jQuery Works. Verkkodokumentti. 2014. jQuery Foundation. <http://w3techs.com/technologies/overview/javascript_library/all>. Luettu 20.4.2014.

15 jQuery Mobile Overview. Verkkodokumentti. 2014. jQuery Mobile. jQuery Foundation. <http://demos.jquerymobile.com/1.0/docs/about/intro.html>. Luettu 20.4.2014.

16 jQuery. Verkkodokumentti. 2014. jQuery Foundation. <http://api.jquery.com/ready/>. Luettu 15.4.2014.

17 Welcome. 2013. Verkkodokumentti. VR-Yhtymä Oy. <https://shop.vr.fi/vrmobiili/welcome.do>. Luettu 15.4.2014.

18 Theming: Built to be branded. 2014. Verkkodokumentti. The jQuery Foundation. <http://jquerymobile.com/>. Luettu 1.4.2014.

19 jQuery Mobile 1.4 Supported Platforms. 2014. Verkkodokumentti. jQuery Mobile. . jQuery Foundation. <http://jquerymobile.com/gbs/1.4>. Luettu 20.3.2014.

20 About Apache Cordova™. 2014. Verkkodokumentti. Apache Cordova. Apache Software Foundation. <https://cordova.apache.org/>. Päivitetty 9.4.2014. Luettu 21.4.2014.

21 Apache Software Foundation. 2013. Get the Android SDK. Verkkodokumentti. Android Developers. <https://developer.android.com/sdk/index.html?hl=sko>. Luettu 11.4.2014.

22 Access additional resources in the iOS Dev Center. 2014. Verkkodokumentti. Apple Inc. iOS Dev Center. <https://developer.apple.com/devcenter/ios/index.action>. Luettu 11.4.2014.

23 Getting Started with Windows Mobile Application Development. 2014. Verkkodokumentti. Microsoft. <https://developer.apple.com/devcenter/ios/index.action>. Luettu 11.4.2014.

24 Java Platform, Standard Edition. Verkkodokumentti. Oracle Corporation. Oracle Technology Network. <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Luettu 11.4.2014.

25 Allen Sarah, Graupera Vidal, Lundrigan Lee, Apress. 2010. Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows . Verkkodokumentti. PhoneGap. <http://www.google.fi/books?hl=fi&lr=&id=nhjuzouj94gc&oi=fnd&pg=pp1&dq=phone-gap+development+process&ots=rov4hqarur&sig=7dmukikv7zemrrigqsolshxpcyc&redir_esc=y#v=onepage&q=phonegap&f=false>. Luettu 14.4.2014.

26 Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows. 2014. Verkkodokumentti. Adobe Systems Incorporated. Adobe® PhoneGap™ Build. <https://build.phonegap.com/>. Luettu 14.4.2014.

27 Korpela, Jukka. 2003. Ympäristömuuttujat. Verkkodokumentti. Unix-opas. <https://www.cs.tut.fi/~jkorpela/unix/6.6.html>. Päivitetty 18.10.2003. Luettu 14.4.2014.

28 Salonen, Jaakko. 2012. Johdanto JavaScript-sovellusten kehitykseen Node.js:llä. Verkkodokumentti. blite.iki.fi. <http://blite.iki.fi/artikkelit/javascript-nodejs-johdanto/>. Päivitetty 17.9.2012. Luettu 14.4.2014.

29 Wargo, John M. 2012. How PhoneGap Works. Verkkodokumentti. PhoneGap Essentials. <http://my.safaribooksonline.com/book/programming/mobile/9780132928373/1dot-introduction-to-phonegap/ch01lev1sec3>. Päivitetty 11.6.2012. Luettu 1.4.2014.

30 Contacts. 2012. Verkkodokumentti. Adobe Systems Inc. <http://docs.phonegap.com/en/1.0.0/phonegap_contacts_contacts.md.html#contactfindoptions>. Päivitetty 11.6.2012. Luettu 20.3.2014.

31 Nuutinen, Petri. 2014. Arkkitehtuuri. Verkkodokumentti. Satakunnan ammattikorkeakoulu. <http://web.samk.fi/staff/petri.nuutinen/arkkitehtuuri/kalvo3.pdf>. Luettu 20.3.2014.

32 WWW-ohjelmointi. 2008. Verkkodokumentti. Koulutuskeskus Salpaus. <http://edu.phkk.fi/opiskelu/internet-ohjelmointi/www-ohjelmointi/www-ohjelmointi>. Luettu 20.3.2014.

33 Events. 2013. Verkkodokumentti. jQuery Foundation. <http://demos.jquerymobile.com/1.2.1/docs/api/events.html>. Luettu 22.3.2014.

34 TIn-home healthcare leader improves client care and reduces costs with T-Mobile®. 2014. Verkkodokumentti. T-Mobile. <http://how-to.t-mobile.com/addus-case-study/>. Luettu 1.7.2014.

35 Jokela, Merja 2014. Vastaava Hammashoitaja. Lahden Kaupunki. Haastattelu:

36 Ovaska, Päivi. 2002. Ohjelmiston suunnitteluperiaatteita. Verkkodokumentti. Lappeenranta University of Technology. <http://www2.it.lut.fi/kurssit/01-02/010758000/suunnitteluperiaatteet.pdf>. Luettu 1.7.2014.

37 Oppimis- ja ohjauskäsityksiä. Verkkodokumentti. Itä-Suomen yliopisto. <https://www.uef.fi/fi/aducate/oppimis-ja-ohjauskasityksia#behaviorism>. Luettu 30.6.2014.