KARELIA-AMMATTIKORKEAKOULU
Media-alan koulutusohjelma

Jarno Parkkinen

MOBIILISOVELLUKSEN KEHITTÄMINEN VERKKOTEKNIIKOILLA

Opinnäytetyö
Kesäkuu 2015

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Kesäkuu 2015
Media-alan koulutusohjelma

Länsikatu 15
80110 JOENSUU
013 260 600

Tekijä
Jarno Parkkinen

Nimeke
Mobiilisovelluksen kehittäminen verkkotekniikoilla

Tiivistelmä

Opinnäytetyön tavoitteena on kehittää mobiilisovellus ja ratkaista mobiilisovelluksen kehityksen vaiheet Android- ja iOS-laitteille siten, että opinnäytetyöraportissa käsitellään vaihtoehdot erilaisille toteutustavoille.

Opinnäytetyöraportissa esitellään yleisimmät verkkotekniikat, kuten HTML5, CSS, JavaScript sekä jQuery. Tekniikoiden tarkastelussa käytetään apuna opinnäytetyön toiminnallisen osuuden tulosta eli Hygipassi-mobiilisovellusta. Opinnäytetyössä mobiilisovellus luodaan käyttämällä verkkotekniikoita.

Opinnäytetyö avaa lukijalle vaihtoehtoiset ratkaisut mobiilikehitykseen ohjelmistokehysten ulkopuolelta ja esittelee hybridisovelluksen teknisen kehittämisen vaiheet. Työssä esitellään myös sovelluksessa käytetyt jQuery-lisäosat siten, että sovelluksesta otetuilla esimerkeillä lukijalle avataan, kuinka koodi vaikuttaa rakenteeseen ja tulostuu sovelluksen graafiseen käyttöliittymään.

Opinnäytetyö on rajattu siten, ettei tekstissä käsitellä sovelluksen lisäämistä mobiililaitteelle tai sen julkaisua sovelluskauppoihin.

Opinnäytetyön tulos on mobiililaitteessa toimiva verkkotekniikoilla kehitetty sovellus, eli hybridisovellus.

Kieli
suomi

Sivuja 37
Liitteet 1
Liitesivumäärä 1

Asiasanat
HTML5, JavaScript, Ohjelmistokehys, jQuery, Hybridisovellus

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
June 2015
Degree Programme in Communication
Länsikatu 15
FI 80110 JOENSUU
FINLAND
013 260 600

Author
Jarno Parkkinen

Title
Mobile developing with web technologies

Abstract

The objective of this thesis was to develop a mobile application and to solve the different stages of development for Android and iOS platforms. To that the report introduces different alternatives for implementation.

The thesis demonstrated the most common web technologies such as HTML5, CSS, JavaScript and jQuery. To demonstrate these technologies, the thesis included a functional part, which is a mobile application called Hygipassi. This mobile application was developed using web technologies.

The thesis provides the reader with alternative solutions to mobile development outside the frameworks and presents the technical stages of developing a hybrid application. The thesis also demonstrates the jQuery plugins used in the application by including examples from the application showing the reader how the code effects the structure and is printed on the graphic interface of the application.

This thesis has been outlined not to include a preview on installing or uploading the application into a mobile device, or an application marketplace.

The result of this thesis is a hybrid application, which is developed with web technologies and can be used in mobile device.

Language	Pages 37
Finnish	Appendices 1
	Pages of Appendices 1

Keywords
HTML5, JavaScript, Framework, jQuery, Hybrid-app

Sisältö

1 Johdanto 5

2 Mobiilisovellusprojektin pohja 6
2.1 Aiheen synty 6
2.2 Konseptointi ja aikataulu 6

3 Verkkotekniikat 8
3.1 Verkkotekniikat mobiilikehityksessä 8
3.2 HTML5 8
3.2.1 Rakenne ja dokumenttityyppi 8
3.2.2 Metatiedot 9
3.3 CSS 11
3.3.1 CSS mobiilikehityksessä 11
3.3.2 Tyylittelyt ja tyylitiedostot 12
3.4. JavaScript ja jQuery 13
3.4.1 JavaScript ja JavaScript-kirjastot 13
3.4.2 jQuery 15

4 Ohjelmistokehykset 16
4.1 Mobiiliohjelmistokehysten käyttötarkoitus 17
4.2 jQuery Mobile 17
4.3 Sencha Touch 20

5 jQuery-lisäosat 22
5.1 Yleistä jQuery-lisäosista 22
5.2 iScroll 22
5.3 SlickQuiz 23

6 Hybridisovellus 25
6.1 Yleistä 25
6.2 Hybridi vai verkkosovellus 26
6.3 Hybridi vastaan ohjelmistokehys sovelluksemme tapauksessa 27

7 Lopputulos 27
7.1.Hygipassi-sovellus 27
7.2 Rakenne ja työtiedostot 28
7.3 Ulkoasu ja sisältö 31

8 Pohdinta 34

Lähteet 36

Liitteet

Liite 1 Rautalankasuunnitelma

1 Johdanto

Opinnäytetyöraportissa käyn läpi vuonna 2013 valmistuneen mobiilisovelluksen beta-version teknisiin ratkaisuihin vaikuttaneet valinnat, sekä avaan mobiilisovellus projektin aikana käytettyjä tekniikoita. Opinnäytetyöraportissa käsitellään mobiilisovellus projektin lähtökohtien ja toiminnan lisäksi pääsääntöisesti HTML5:n ja JavaScriptin yhteistyötä mobiilisovellusta kehittäessä. Lisäksi opinnäytetyöraportissa perustellaan valinnat miksi lopullinen sovellus ei tule käyttämään alun perin suunniteltua mobiiliohjelmistokehystä.

Opinnäytetyön aiheen sain hahmoteltua vuoden 2013 lopulla todettuani mobiilisovellus projektin sisältävän paneutumisen arvoisia teknisiä ratkaisuja, jotka haastoivat siihen aikaan perinteiset mobiilikehitys käytännöt. Sovellus rakennettiin lähtökohdista huolimatta ilman suoraa ohjelmistokehystä verkkotekniikoita käyttäen. Esimerkiksi verkossa tunnettu Smashing Magazine julkaisi tästä niin sanotusta hybriditekniikasta esittelevän artikkelin vasta vuoden 2014 loppupuolella (Rudolph 2014).

Opinnäytetyöraporttini tulee olemaan hyödyllinen lukijoille, jotka ovat kiinnostuneita mobiilikehityksestä. Myös lukijat, joilta löytyy kokemusta webkehityksestä saavat näkökulman mobiilikehitykseen heille tutuilla tekniikoilla. Kohderyhmänä työlle ovat aloittelevat mobiilikehityksestä kiinnostuneet henkilöt, joilla on käsitys verkkotekniikoista. Opinnäytetyöraportissa mobiilisovellusprojektia avataan lähtökohdista alkaen. Työssä esitellään projektissa vaadittuja tekniikoita ja yksittäisiä esimerkkejä kuinka näitä tekniikoita sovelluksen kehityksessä käytettiin. Raportissa esitellään yleisimpiä verkkotekniikoita ja niitä avataan sovelluksesta otetuin esimerkein. Perusteiden pohjalta sovelluksen rakennetta tarkastellaan syvemmin raportin loppupuolella, jolloin lukijalle on selvinnyt yllä mainittujen verkkotekniikoiden tarkoitus. Vaikkakin opinnäytetyöraportissa käydään läpi ratkaisut sovelluksen beta-version valmiiksi saattamiseksi, on se rajattu siten, ettei raporttiin sisälly sovelluksen siirtäminen mobiililaitteeseen tai sen lisäämistä sovelluskauppaan.

2 Mobiilisovellusprojektin pohja

2.1 Aiheen synty

Syksyllä 2013 aloitimme kanssaopiskelija E. Horttanaisen kanssa mobiilisovelluksen kehittämisen. Mobiilisovelluksen tarkoitus oli toimia hygienia oppimista tukevana työkaluna opiskeltaessa hygieniaosaamistestiin. Tämä sovellus nimeltään Hygipassi-sovellus oli projektina luonteva tapa kehittää omaa ammattitaitoa sekä rakentaa konkreettista näytettävää työelämään siirtymistä ajatellen. Esimerkiksi verkkosivuprojektista poiketen mobiilisovellus antoi mahdollisuuden tarkastella jo omatun verkko-osaamisen siirtämistä pienempiin laitteisiin. Projektin aikaiset roolit jakautuivat yhteisen suunnittelun ja konseptoinnin jälkeen siten, että Horttanainen paneutui mobiilisovelluksen graafiseen puoleen, kun minä taas käsittelin teknistä puolta.

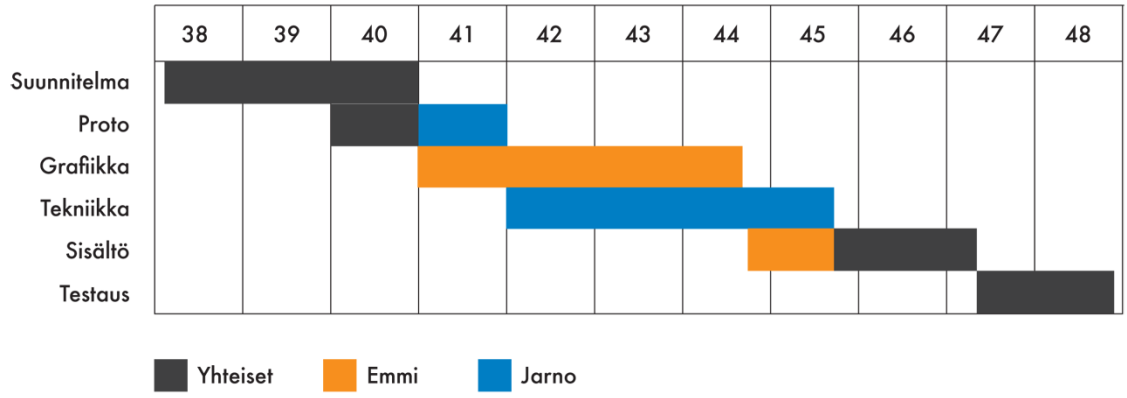
Koska taustaosaaminen verkkotekniikoista, kuten HTML5 ja JavaScript olivat hallussa, oli lähtökohtana käyttää jQuery Mobile -ohjelmistokehystä sovelluksen selkärankana. Jos HTML5 ja JavaScript ovat ennestään tuttuja, on jQuery Mobilen käyttö helppoa (Jain 2012, 8). Lähtökohdista poiketen totesimme projektin aikana että hygieniapassisovelluksen sisältö ja toimintatapa huomioon ottaen pystyimme rakentamaan sovelluksen nykyaikaisilla verkkotekniikoilla.

2.2 Konseptointi ja aikataulu

Sovelluksen pääsääntöinen tarkoitus käyttäjän näkökulmasta on valmentaa käyttäjä hygieniaosaamistestiin. Tämä tapahtui rakenteella, joka sisälsi ulkoisesti linkitetyn oppimateriaalin, harjoitteluosion ja hygieniaosaamistestiä simuloivan testiosion. Teknisesti sovelluksen tarkoitus oli toimia nopeasti iOS- ja Android -päätelaitteilla. Projektin alkuvaiheilla meille syntyi epäilyksiä omista teknisistä taidois-

tamme, joten päätimme aloittaa sovelluksen kehityksen niin sanotulla *proto-versiolla,* joka eteni tekniikan säännöillä. Proto-version tärkein tarkoitus hyvän teknisen pohjan lisäksi oli varmistaa tekninen tietotaito itsenäisen mobiilisovelluksen toteuttamiseen.

Projektin aikataulua suunnitellessa päädyimme kuuteen osioon, joiden valmistuttua saisimme sovelluksen ensimmäisen version mobiililaitteeseen. Suunnittelemiselle oli varattu noin kolmen viikon pituinen osio, joka proto-version aloituksen yhteydessä yhdistyi graafiselle suunnittelulle ja toteutukselle varatulle ajalle. Suunnitelmien valmistuessa grafiikka ja tekniikka tulisivat kulkemaan rinnakkain muutaman viikon ajan, jonka jälkeen tuotanto siirtyi sisällöntuottovaiheeseen. Testaukselle ja korjauksille oli varattu projektin loppuun sille tarvittava aika. Työvaiheet, eli osiot olivat jaettu siten, että vastuualueet olivat selkeät, mutta pääsääntöisesti mobiilisovellus tultaisiin toteuttamaan yhteistyössä (kuva 1). Sovelluksemme ensimmäinen rautalanka toimi suunnitelmien lähtökohtana (liite 1).



Kuva 1. Projektin aikataulu (Käyttölupa 25.5.2015).

Opinnäytetyöraportin loppupuolella avataan sovelluksen tarkoitusta ja toimintaa tarkemmin sekä esitellään kuvakaappauksia valmiista beta-versiosta. Loppupuoli sisältää myös tulevaisuuden näkymät sovelluksen osalta. Ennen kuin sovellus pystytään teknisesti esittämään, on käsiteltävä perus verkko- ja mobiilitekniikat.

3 Verkkotekniikat

3.1 Verkkotekniikat mobiilikehityksessä

Oli kyseessä verkkosivujen rakentaminen tai mobiilisovelluksen kasaaminen tulee tietty sarja työkaluja väistämättä vastaan. Näistä työkaluista HTML5 toimii merkkikielenä rakenteelle, CSS tyyliohjeena ulkoasun esittämiselle sekä JavaScript interaktioiden ja toimintojen komentajana. (ks. esim. Harrel 2011.) Pelkällä HTML5:n ja CSS-koodin yhteiskäytöllä saadaan luoduksi toimivat verkkosivut (Duckett 2011, 8). JavaScript avaa mahdollisuudet interaktiivisimmille toiminnoille, joiden avulla voidaan luoda reaaliaikaisia toimintoja, kuten esimerkiksi suoraan klikkaukseen reagoivia nappeja (Sawyer McFarland 2014, 13). Pelkkä JavaScriptin tuonti kehitykseen ei takaa verkkotekniikoiden toimimista mobiilissa. Suuremmassa roolissa verkkotekniikoiden käyttöön mobiilikehityksessä oli mobiililaitteiden tuen lisääntyminen HTML5-merkkikielelle. (Leenheer 2015.)

3.2 HTML5

3.2.1 Rakenne ja dokumenttityyppi

HTML on yleisesti käytetty merkintäkieli. Merkintäkielen avulla luodaan rakenteita, joista muodostuu esimerkiksi verkkosivu tai sovellus. HTML-koodi perustuu elementteihin. Elementti muodostuu hakasulkujen sisällä olevista merkeistä tai sanoista. Jokainen elementti pitää aloittaa ja sulkea. Elementin sitominen onnistuu käyttämällä avaavaa ja sulkevaa *tagia* (Duckett 2011, 20). Alla olevassa koodissa esitetään HTML-rakenne joka sisältää otsikon ja sisältötekstin. Otsikko ja sisältöteksti on sidottu HTML-dokumentin sisään.

```
<html>
    <body>
        <h1>Otsikko</h1>
        <p>Sisältöteksti</p>
```

```
        </body>
</html>
```

HTML5, jota käytimme mobiilisovellustamme kehittäessä edustaa tämän hetkisen HTML-merkkikielen viimeisintä versiota. W3C:n vuonna 2014 suorittama suositus johti HTML5:n standardisoitumiseen ja viralliseen julkaisuun. (W3C 2014.) Ennen virallista julkaisua HTML5 oli kuitenkin kehittäjien käytössä niin sanottua julkisena luonnoksena vuodesta 2008 alkaen.

HTML5 toi HTML-merkkikieleen lisää ominaisuuksia eikä se poistanut ominaisuuksia edeltäjästään HTML4:stä. Kaikki HTML4:llä työstetyt rivit toimivat HTML5:n kanssa työskennellessä. (Pilgrim 2015, 12.) Mobiilisovelluksen kehittämisen ajankohdan huomioon ottaen oli loogista käyttää HTML5:tä. Vuonna 2013 kaikki mobiilisovellukseemme tarvittavat HTML5:n ominaisuudet olivat jo laajasti tuettuja (Irish & Manian 2015).

Ensimmäinen konkreettinen huomio verrattuna HTML4:n kanssa työskentelyyn tuli vastaan heti työtiedostojen aloitusvaiheessa, jossa verkkoselaimelle ilmoitetaan dokumentin tyyppi. Tyyppi ilmoitetaan vanhemmassa HTML4 standardissa pitkällä rivillä tietoja, kun taas HTML5 tiedosto aloitetaan yksinkertaisemmalla rivillä (kuvat 2 ja 3).

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2 <html>
3 <head>
```

Kuva 2. Esimerkki vanhemmasta HTML4 versiosta.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
```

Kuva 3. Lyhyempi HTML5:en mukana tuoma dokumentti tyypin esitystapa.

3.2.2 Metatiedot

Kehittämämme sovelluksen oli tarkoitus pyöriä suoraan mahdollisimman monella

eri mobiililaitteella, eikä sovelluksen verkkoversiota ollut suunnitelmissa. HTML-sivut esitetään oletuksena isoille päätelaitteille, kuten tietokoneille. Tällöin se tuo mukanaan epäkäytännöllisyyksiä mobiililaitteita ajatellen. HTML-sivu mobiililaitteessa antaa oletuksena käyttäjän loitontaa kokonaisnäkymää eleillä, mikä on vastoin meidän ajatustamme sovelluksen staattisuudesta ja täten heikentää käytettävyyttä. Käsin loitontamisen estetään asettamalla HTML-dokumenttiin alla oleva rivi.

```
<meta name=”viewport” content=”width=device-width, initial-scale=”1.0”>
```

Koodissa asetus initial-scale=”1.0” asettaa esimerkiksi Safarin mobiiliversiolle rajoituksen loitontamisen käyttöön. Toinen yllä esitetty komento width=device-width antaa mobiiliselaimen ymmärtää, että sivusto on yhtä leveä kuin käytettävä laite (Chuan 2012, 18). Yllä esitetty rivi on *meta-tagi,* joka ei suoraan tulostu sivulle näkyviin, mutta on esimerkiksi selaimen luettavissa. Meta-tagi kuuluu metatietoihin ja tagiin voidaan sisällyttää yllämainitun lisäksi muun muassa sivuston kuvaus, avainsanat tai dokumentin tekijä. (W3Schools 2015a.)

HTML-tiedoston rivit tulostuvat ylhäältä alas, ja siksi metatiedot kannattaa sijoittaa tiedoston alkuriveille. HTML-dokumentteihin onkin hyvä sijoittaa *head-osio*, jonka sisään syötetään esimerkiksi linkitykset dokumentin ulkoisiin tiedostoihin. (W3CSchools 2015b.)

```
<head>
        <!-- Otsikko -->
        <title>Hygieniasovellus</title>
        <!-- Viewport asetukset -->
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- Tyylitiedostot -->
        <link href="css/maintyyli.css" media="screen" rel="stylesheet"
type="text/css">
</head>
```

Yllä olevissa riveissä sovelluksen työversion otsikko lisätään tagilla <title>. Käyttämällä <link> -tagia saadaan dokumenttiin lisättyä ulkoinen projektin tyylejä määrittelevä CSS-tiedosto. Koska projektissa käytettiin useita ulkoisia paketteja, niin JavaScriptin kuin CSS-tiedostojen osalta, ovat metatiedot oleellinen osa HTML-rakennetta.

3.3 CSS

3.3.1 CSS mobiilikehityksessä

CSS toimii säännöstönä merkkikielelle, jonka perusteella esimerkiksi verkkosivut muotoutuvat. Näitä sääntöjä kutsutaan tyyleiksi, jotka ohjaavat muun muassa elementtien kokoa, värejä tai sijoittelua. (Harrel 2011, 27.) Toisin sanoen CSS tuo grafiikan rakenteen päälle. Kuten aiemmin mainitut verkkotekniikat myös CSS kuuluu jatkuvasti kehittyvien standardien piiriin. Tällä hetkellä CSS-versioita löytyy kolme, jotka ovat nimetty lineaarisesti CSS1, CSS2 sekä CSS3. Kuten HTML-merkkikieltä, niin myös CSS-koodikieltä hallinnoi W3C, joka suosittelee uusimmat versiot aika ajoin julkaistaviksi. (Harrel 2011, 35-36.)

CSS-säännöt luetaan järjestyksessä, mikä tarkoittaa lähtökohtaisesti sitä, että viimeisin rivi tiedostorakenteessa kumoaa aiemmin määritellyt tyylit (Harrel 2011, 27). Käytännössä, jos CSS-tyylitiedoston ensimmäinen rivi osoittaa fonttikooksi 12 pistettä ajautuu se yli, jos tiedostoissa myöhemmin määritellään fontille kooksi 14 pistettä.

Mobiilisovellus projektissa CSS-koodia käytettiin sen omaan tarkoitukseensa, eli grafiikan esittämiseen. Käyttämämme JavaScript lisäosat sisälsivät omia tyylitiedostoja. Jotta saadaan täysi hallinta koko sovelluksen tyyleihin, päädyttiin tekemään niin sanottu *override-tyylitiedosto*. Kyseisen tyylitiedoston on tarkoitus yliajaa esimerkiksi HTML:n sisäänrakennetut määritteet (Harrel 2011, 32). Sisäänrakennettujen tyylien lisäksi mobiilisovelluksessa käytettiin pääsääntöisesti yhtä päätyylitiedostoa, sekä kahta erillistä tyylitiedostoa lisäosia varten.

3.3.2 Tyylittelyt ja tyylitiedostot

Kun halutaan tyylitellä tietty HTML-dokumentin elementti, annetaan sille CSS-sääntö. Säännön voi sisällyttää suoraan HTML-dokumenttiin joko elementin sisään tai <style> -tagin sisään. (W3Schools 2015c.)

```
<div class="quizResults">
        <h3 class="quizScore">Sait pisteet: <span></span></h3>
        <h3 class="quizLevel"><strong>Tasosi:</strong> <span></span></h3>
        <div class="quizResultsCopy" style=”font-size: 24px;”>
        </div>
</div>

<style>
.quizScore {
        font-size: 24px;
}
</style>
```

Projektista otetussa esimerkissä elementille luokalla quizResultsCopy on määritelty tyyli suoraan elementtiin, kun taas luokalla quizScore oleva elementti saa tyylimäärittelyt <style> -tagin sisästä. Tagien sisään tehdyt tyylit pitää osoittaa elementille *luokilla* tai *id:llä*.

Luokka voi osoittaa moneen elementtiin yhtä aikaa ja täten vaihtaa kaikki samalla luokalla nimetyt laatikot taustaväriltään sinisiksi. Id sen sijaan kannattaa pitää sivukohtaisesti uniikkina. (W3Schools 2015d.) Vaikkakin id:lle voidaan määrittää tyyli samalla tavalla kuin luokalle, määritetään id:lle myös esimerkiksi JavaScriptin puolella toimintoja. Toimintojen toimivuuden kannalta id:t kannattaa pitää erillään. Projektissa käytettiin pääsääntöisesti tyylittelyyn luokkia, mutta tiettyihin id:n

sisältäviin elementteihin tyylit määriteltiin id:n mukaan.

Alla käytännön CSS-esimerkki projektin päätyylisäännöstöstä, eli maintyyli.css -tiedostosta. Esimerkistä ilmenee niin luokan, kuin id:n käyttö tyylittelyssä.

```
.teksti {
        float: left;
        margin-left: 20px;
        vertical-align: middle;
        line-height: 40px;
}
#kategoria_linkki {
        color: #1a1a1a;
        margin-top:12px;
        height: 40px;
        border: solid #e6e6e6 1px;
        border-radius: 3px;
        overflow: hidden;
}
```

3.4. JavaScript ja jQuery

3.4.1 JavaScript ja JavaScript-kirjastot

HTML5:n ja CSS:n avulla rakennettuun graafiseen näkymään saadaan eloa ja interaktiivisuutta lisäämällä siihen JavaScriptiä. JavaScriptillä mahdollistetaan niin pienet, kuin suuretkin dynaamiset ominaisuudet. Näitä ominaisuuksia ovat esimerkiksi ponnahdusikkunat, sekä kokonaiset interaktiiviset käyttöliittymät. (Sawyer McFarland 2014, 13.)

JavaScriptin monipuolisuuden ansiosta sitä käytetään myös erillään verkkopalveluista. Adoben ohjelmistoihin voidaan JavaScriptillä kirjoittaa toimintoja, sekä uusimman Mac OS X -käyttöjärjestelmän käyttäjät voivat kirjoittaa omia automaatioitaan JavaScriptillä. (Sawyer McFarland 2014, 15.) JavaScriptin tuomat lisät ovat mobiilisovelluksemme kannalta välttämättömiä, varsinkin ottaen huomioon sen tuomat mahdollisuudet rakentaa käyttöliittymille oleellisia toimintoja.

Pelkällä JavaScriptillä voidaan kirjoittaa suurin osa toiminnoista ja se on täten soveltuva mobiilikehitykseen, mutta lähtökohtaisesti JavaScriptin kirjoittaminen on vaativaa. (Sawyer McFarland 2014, 15.) JavaScriptin käyttöä helpottavat JavaScript-kirjastot. JavaScript-kirjastot ovat kokoelmia monista toimiviksi todetuista esikirjoitetuista JavaScript-toiminnoista (Sawyer McFarland 2014, 106). Käytännössä tämä tarkoittaa sitä, että projektin aikana pystyimme kutsumaan kirjastoista tarvittavia toimintoja, ilman että kirjoittaisimme ja testaisimme monta riviä omaa ohjelmointia.

JavaScriptiä voidaan syöttää dokumenttiin samaan tapaan kuin aiemmin esiteltyä CSS:ä. JavaScript voidaan sisällyttää suoraan HTML-dokumenttiin <script> -tagin sisään tai linkittämällä ulkoinen tiedosto dokumentin metatietoihin. Projektissa käytettiin sivukohtaisia toiminnallisuuksia dokumentin sisäisesti, sekä kirjastot ja lisäosat linkitettiin ulkoisista tiedostoista.

```
<script type="text/javascript" src="js/jquery-1.9.1.js"></script>
<script type="text/javascript">
        $(function(){
                $('ul li a').click(function(){
                        var item=$(this).parent();
                        $('ul li').removeClass('active2');
                        item.addClass("active2")
                });
```

```
    });
</script>
```

Koodiesimerkissä yllä lisätään jQuery-kirjasto työtiedostojen käyttöön, sekä dokumentin sisäinen JavaScript asettaa jQuerya käyttäen tietyille HTML-elementeille CSS-luokan painalluksesta. Mobiilisovelluksessa käytettiin pääsääntöisenä JavaScript-tiedostona yllämainittua jQuery-kirjastoa, joka on yksi suosituimmista avoimen lähdekoodin JavaScript-kirjastoista. jQueryn suosiosta kertoo se, että sitä käytetään esimerkiksi Drupalin ja WordPressin tasoisissa sisällönhallintajärjestelmissä. (Sawyer McFarland 2014, 15.)

3.4.2 jQuery

jQueryn suosio oli yksi pääsyistä miksi päädyimme sitä käyttämään. Verkkokehitysyhteisöissä jQueryn suosio näkyy myös siten, että on olemassa työpaikkoja, joihin haetaan yksinomaan jQuery-ohjelmoijia, yleisen JavaScript-ohjelmoijan sijaan. (Sawyer McFarland 2014, 15.) Kaikki kokemus jQuerystä kehittää ammatillista kasvua ja käytän jQuerya työtehtävissäni päivittäin.

jQuery-kirjaston lisääminen työtiedostoihin onnistuu kahdella eri tavalla. Kirjaston voi linkittää tiedostoihin suoraan verkosta, jolloin lopullinen tuotos joutuu olemaan yhteydessä verkkoon (Sawyer McFarland 2014, 109). Meidän sovelluksemme käytön ollessa mahdollinen ilman verkkoyhteyttä, päädyimme sisällyttämään jQuery-tiedoston suoraan kansiorakenteeseen. Alla esitetyssä koodissa alempi rivi lisää jQuery-kirjaston paikallisena työtiedostoihin, kun taas ylempi rivi vaatii verkkoyhteyden.

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js">
</script>

<script src="js/jquery-1.11.0.min.js"></script>
```

JavaScriptillä rakennetut rivit toimivat suurimmaksi osaksi sovelluksemme sisältöjen komentamiseen. Näitä komentoja ovat muun muassa lisäosien asetusten määritteleminen, sekä valikon elävöittäminen (Sawyer McFarland 2014, 112). JavaScript vaatii näitä komentoja varten HTML-elementit joihin kohdentua. Onkin ongelmallista, jos komento käydään läpi ennen kuin kaikki elementit ovat ladattuina. Jos dokumentin latausta ei odoteta, komento ei toimi halutulla tavalla tai ei ollenkaan. On siis hyvä, että jQuery tuo tähän ongelmaan ratkaisun alla esitetyillä riveillä.

```
<script>
	$(document).ready(function() {
		// komento tulee tähän
	});
</script>
```

Riveissä jQuery tarkastaa, että kaikki dokumentin HTML-elementit ovat ladattuna ennen komennon läpikäyntiä (Sawyer McFarland 2014, 112). Projektissa käytettiin jQueryn sisältämää lyhennettä, joka on pelkistetty versio yllä olevasta koodista.

```
<script>
	$(function() {
		// komento tulee tähän
	});
</script>
```

4 Ohjelmistokehykset

4.1 Mobiiliohjelmistokehysten käyttötarkoitus

Lähtökohtaisena tarkoituksena sovelluksemme työstöön oli lähteä rakentamaan sovellusta valmiin ohjelmistokehyksen ympärille. Aiempi mobiilikehityskokemus kehysten, kuten jQuery Mobilen ja Sencha Touch:in käytöstä rajasi kehys vaihtoehdot kahden välille.

Ohjelmistokehysten tarkoitus on helpottaa ohjelman, sovelluksen tai verkkopalvelun kehitystä. Ohjelmistokehykset tuovat mukanaan muun muassa useita kirjastoja, mahdollisuudet tietokantoihin sekä useissa tapauksissa myös käyttöliittymä- ja grafiikkasapluunoita. (Wikipedia 2015.) Verrattuna esimerkiksi aiemmin raportissa käsiteltyihin JavaScript-kirjastoihin, kokonaiset ohjelmistokehykset sisältävät laajempia kokonaisuuksia yksittäisten esikirjotettujen rivien sijaan. Kokonaisuudet ovat kasattu siten, että kehittäjä välttyy ylimääräiseltä työltä jonka vuoksi esimerkiksi mobiiliohjelmistokehys tehostaa mobiilisovellusten käytännön kehitystä ja säästää tuotantoon kuluvaa aikaa. (DocForge 2013.)

Mobiilikehitykseen soveltuvia ohjelmistokehyksiä löytyy useita ja niiden välisiä eroja voidaan vertailla esimerkiksi tuen laajuuden kannalta. Tukea voidaan tarkastella aihealueittain muun muassa käyttöjärjestelmätuen, mobiililaitteen ominaisuuksien tai lisenssivaihtoehtojen kannalta. (Falk 2015.) Meille tärkeimmät vertailtavat aiheet olivat hyvin pitkälti kehysten pääohjelmointikieli, hyvä dokumentointi sekä aikaisempi kokemus. Myös avoin tai ilmainen lisenssi oli projektimme kannalta tärkeää ottaa huomioon. Sovelluksen kehityksen alkuvaiheessa päädyimme vertailussa jQuery Mobilen käyttöön.

4.2 jQuery Mobile

jQuery Mobile kuuluu avoimenlähdekoodin *UI-ohjelmistokehyksiin*. Kehys perustuu vahvasti HTML5-, CSS3- ja jQuery -tekniikoihin ja tuo mukanaan mobiilisovelluksille olennaisen kosketuskäyttöisen käyttöliittymän. (Jain 2012, 7.) Käyttö-

liittymän rakentamista jQuery Mobilelle on helppoa siihen kuuluvan teematyökalun avulla, joka sisältää valmiit sapluunat niin napeille kuin lomakkeille (Themeroller 2015). Teematyökalut ovat monipuoliset, mutta sitovat käyttöliittymän oletuksena sapluunaan, jonka muokkaaminen sovelluksemme graafisten suunnitelmien mukaiseksi vaatisi paljon CSS:n korvaamista ja yliajamista. Ilman teematyökaluja jQuery Mobile sisältää silti oletusteeman. Yksi syy jQuery Mobilen projektista pois jättämiselle oli toteamus siitä, että graafinen käyttöliittymä oli sovelluksemme tapauksessa tehokkaampaa luoda alusta alkaen itse. Näin välttyisimme sapluunoiden tuomien tyylisäännösten ristiriidoilta ja täten ylimääräiseltä tyylien uudelleen kirjoittamiselta. jQuery Mobilella monisivuisen sovelluksen voi luoda yhteen HTML-tiedostoon siten, että sovelluksen sisäiset näkymät ovat samassa tiedostossa.

```
<body>
    <!-- KOTISIVU ALKAA -->
    <div data-role="page" id="kotisivu" >
        <div data-iscroll="" data-role="content">

            <h1>Kotisivu otsikko</h1>


        </div>

        <div data-role="content" id="alaosa">
            <!-- navbar -->
            <div data-role="navbar" id="alamenu">
                <ul>
                    <li><a href="#kotisivu" data-role="button">Kotisivu</a></li>
                    <li><a href="#oppimateriaali" data-role="button">Oppimateriaali</a></li>
                    <li><a href="#kysymykset" data-role="button">Kysymykset</a></li>
                    <li><a href="#lisatietoa" data-role="button">Lisätietoa</a></li>
                </ul>
            </div>
            <!-- /navbar -->
        </div>
    </div>
    <!-- KOTISIVU PÄÄTTYY-->

    <!-- OPPIMATERIAALI ALKAA -->
    <div data-role="page" id="oppimateriaali" >
        <div data-iscroll="" data-role="content">

            <h1>Oppimateriaali otsikko</h1>

        </div>

        <div data-role="content" id="alaosa">
            <!-- navbar -->
            <div data-role="navbar" id="alamenu">
                <ul>
                    <li><a href="#kotisivu" data-role="button">Kotisivu</a></li>
                    <li><a href="#oppimateriaali" data-role="button">Oppimateriaali</a></li>
                    <li><a href="#kysymykset" data-role="button">Kysymykset</a></li>
                    <li><a href="#lisatietoa" data-role="button">Lisätietoa</a></li>
                </ul>
            </div>
            <!-- /navbar -->
        </div>
    </div>
    <!-- OPPIMATERIAALI PÄÄTTYY-->
</body>
```

Kuva 4. HTML5-rakenne jQuery Mobile -kehyksen komennoilla.

Kuvassa yllä (kuva 4) on varhaisessa sovelluksemme kehitysvaiheessa oleva jQuery Mobile koodi, joka sisältää kaksi sisältösivua ja navigaation. Näkymien ollessa samassa tiedostossa ne latautuvat yhtä aikaa lisäten sisällöistä riippuen latausaikaa. Esitetystä koodista voi todeta, että koodi rakentuu HTML:n ympärille ja toiminnot mitkä jQuery Mobile tuo mukanaan tapahtuvat hyvin pitkälti sen sisäisistä JavaScript-tiedostoista käsin.

Navigointi on jQuery Mobilessa tehty helpoksi kehittää ja hyväksi käyttää. Sovelluksen erilliset näkymät sidotaan HTML-elementtien sisään ja niille osoitetaan *sivu ID*. HTML:n sekaan kirjoitetut rivit lähettävät pyynnön jQuery Mobilen JavaScript-tiedostoille, jotka siten ohjaavat esimerkiksi linkkiä painaessa CSS3 siirtymäanimaatiot sivujen välille (Jain 2012, 22). jQuery Mobile hoitaa siis näkymästä toiselle siirtymisen ja lisää mukaan valmiiksi määritellyn animaation. Useimmissa tapauksissa siirtymäanimaatio sopii sovellusten ilmeeseen ja käyttöliittymiin, mutta meidän sovelluksemme tapauksessa animaatio ei tuonut lisäarvoa ja olisi täten vaatinut ylimääräistä työtä poistaa käytöstä.

Vaikka syitä jQuery Mobilesta luopumiseen löytyi muun muassa tyylittelystä, niin suurimmaksi ongelmaksi jQuery Mobilen kanssa osoittautui latausajat. Sovelluksemme käynnistyminen kesti kohtuuttoman kauan, vaikka sisältö oli vielä alkuvaiheessa ja vähäistä. Sovelluksemme oli tarkoitus toimia paikallisesti ilman verkkoyhteyttä, joten latausajoissa ei tarvinnut kiinnittää huomiota verkkoliittymätyyppeihin. Paikallisen sovelluksen latausaikoihin vaikuttavat muun muassa JavaScript-tiedoston kilotavu koko. Jokainen kilotavu JavaScriptiä lisää latausaikaa noin yhden millisekuntin (Roveb 2012). Ottaen huomioon haluamamme lisäosat, sekä jQuery Mobile -kehyksen tuomat ominaisuudet, päädyimme jättämään jQuery Mobilen projektin ulkopuolelle.

Vaikkakin jQuery Mobile ei soveltunut sovelluksemme kehitykseen, voin sitä suositella erinäisiin projekteihin. jQuery Mobilen selkeästi hyviä puolia ovat muun muassa laaja selain- ja laitetuki, valmiit lomake sapluunat ja tapauskohtaisesti teematyökalut (Creative bloq 2013). Hyvien puolien lisäksi jQuery Mobile ei rajoitu pelkästään mobiililaitteille. Suosittelisin kehystä kustannustehokkaaseen

työskentelyyn, jossa kehitysaikataulut ovat kireitä ja lopputuotoksien ei tarvitse olla täysin kustomoituja graafisesti.

4.3 Sencha Touch

Ennen varsinaisen projektin käynnistymistä jQuery Mobilen lisäksi kehyskokemusta löytyi Sencha Touch -ohjelmistokehyksestä. Sencha Touch sisältää natiiviin mobiilisovellukseen sopivia käyttöliittymä osia, valmiita animaatioita sekä hipaisuominaisuuksia (Sencha 2015). Nykyaikaisilla mobiililaitteilla Sencha Touch -sovellukset näyttävät ja tuntuvat natiiveihin sovelluksiin verrattuna samalta (Mehtonen 2013, 44). jQuery Mobile ja Sencha Touch ovat toisiinsa verrattuna hyvin samankaltaisia. Kummatkin kehyksistä perustuvat HTML5- ja JavaScript-kieliin, sekä toimivat iOS- ja Android -laitteilla (Falk 2015).

Sencha Touch on jQuery Mobilea huomattavasti tuotteistetumpi ratkaisu, jolle voi halutessaan ostaa tukipalveluita yrityskäyttöön (Sencha 2015). Sencha Touch nojautuu vahvasti JavaScriptillä kirjoittamiseen, jonka rakenne kirjoitetaan JavaScriptillä. Täten se myös eroaa koodiltaan muista esiteltävistä kehyksistä (kuva 5).

```
Ext.application({
    name: 'Hygipassi',

    launch: function() {
        Ext.create("Ext.tab.Panel", {
            fullscreen: true,
            items: [
                {
                    title: 'Kotisivu',
                    iconCls: 'home',
                    html: 'Kotisivu otsikko'
                },
                {
                    title: 'Oppimateriaali',
                    iconCls: 'star',
                    html: 'Oppimateriaali otsikko'
                },
            ]
        });
    }
});
```

Kuva 5. Sencha Touch ja JSON.

Esimerkkikoodi on JavaScriptillä toteutettu *JSON listaus*, joka käyttää Sencha Touch -kehystä. Riveillä luodaan näkymä, jonka sisään rakennetaan navigaatio. Navigaatiolle osoitetaan otsikko, ikoni ja esimerkiksi HTML-sisältö, joilla saadaan perus runko sovellukselle. Sencha Touch kehys onkin siis JavaScriptiä osaavalle henkilölle luonteva ratkaisu ja isoissa kokonaisuuksissa, joissa tarvitaan tietokantakäsittelyä, on Sencha Touch esitellyistä kehyksistä paras. Tietokantoja sovelluksemme tapauksessa olisi voinut käyttää hyväksi esimerkiksi tietovisamaista osiota kehitettäessä. Tietokannat olisivat mahdollistaneet tilastojen rakentamisen ja käyttäjien tietojen tallentamisen paikallisesti. Myöhemmin tietovisaosio rakennettiin kevyemmin jQuery-lisäosaa hyödyntäen.

Sencha Touch kuuluu Sencha tuoteperheeseen ja sille on toteutettu virallinen kehitysohjelmisto Sencha Architect. Sencha Architect tuo mukanaan kehittämistä helpottavia ominaisuuksia kuten *WYSIWYG –editorin* ja *drag-and-drop* –ominaisuudet. Aloittelevalle mobiilikehittäjälle ohjelmisto on houkutteleva, mutta maksullinen. Sencha Architectistä löytyy ilmainen kokeilu versio monelle eri käyttöjärjestelmälle. (Sencha 2015.)

Projektin resurssit huomioon ottaen ja JavaScript osaamisen puute kehitysajankohtana vaikuttivat päätökseen jättää Sencha Touch odottamaan seuraavia projekteja. Suosittelen Sencha tuoteperhettä sisällöltään suurien sovellusten kehittämiseen. Hyvänä esimerkkinä mobiilisovellus Ilosaarirock 2013 iOS- ja Android -versiot on toteutettu vahvasti Sencha tuoteperheeseen painottuen.

Vaikka kaikki ohjelmistokehykset tuovat helpotuksia yleiseen mobiilikehitykseen, niin halusimme kokeilla vaihtoehtoista ratkaisua sovelluksen kehitykseen. Sovelluksemme tarkka määrittely ja sisällön laajuus soveltui myös yksinomaan verkkotekniikoilla toteutettavaan ratkaisuun. HTML5 on hyvin pitkälle kehittynyt ja JavaScript yhteisöt ovat luoneet jQuerya käyttäen laajan valikoiman työkaluja valmiin sovelluksen loppuunsaattamiseksi.

5 jQuery-lisäosat

5.1 Yleistä jQuery-lisäosista

jQuerylle on rakennettu lukemattomia työkaluja eli *lisäosia*. Lisäosien tarkoitus on tuoda valtava määrä ominaisuuksia kehittäjien käyttöön, ilman että kehittäjät joutuisivat ne yksi kerrallaan itse kirjoittamaan. Lisäosia löytyy laidasta laitaan ja aloitteleva kehittäjä pystyy luomaan täydellisiä kokonaisuuksia yhdistelemällä eri ominaisuuksia lisääviä lisäosia. Yksinkertaisemmillaan jQuery-lisäosat voivat olla JavaScriptillä kirjoitettuja animaatioita, mutta lisäosat laajentuvat aina lomakkeen käsittelijöistä tiedostonsiirtoon asti (Angelov 2013). Kuka tahansa jQueryä hallitseva kehittäjä voi luoda ja julkaista oman lisäosan (Rotton 2013). Lisäosat ovat useimmissa tapauksissa ilmaisia ja niiden käyttöönotto ja hallinta ovat usein alkuperäisen lisäosan tekijästä kiinni. Harkitusti valitulla työkalupakilla aloittelevat kehittäjät voivat rakentaa toimivia sovelluskokonaisuuksia.

Lähtökohtaisesti halusimme paikata ohjelmistokehyksien pois jättämisestä syntyneitä aukkoja, kuten natiivin myötäisen käyttöliittymän ja mobiililaitteiden käytölle välttämättömät pyyhkäisy toiminnot. Käyttöliittymä sapluunan pystyimme toteuttamaan alusta asti HTML:n ja CSS:än yhteistyöllä. Mobiilisovelluksemme rakentuu vahvasti kahden jQuery-lisäosan ympärille. Lisäosista iScroll ansiosta sovellukselle saadaan mobiilille olennainen pyyhkäisy toimimaan, siten että sitä voitaisiin verrata natiivisovellukseen. Toinen käytössä oleva lisäosa on SlickQuiz, jonka avulla voidaan rakentaa tietokilpailumaisia osioita, jotka toimivat sovelluksen sisällöllisesti tärkeimpänä ominaisuutena.

5.2 iScroll

Vuonna 2013 verkkosovellusten sivunvieritys mobiililaitteilla toimi omien kokemustemme perusteella hyvin vaihtelevasti. Pehmeä, natiivisovelluksista tuttu si-

sältöä ylös ja alas selaava vieritys toimi verkkosovelluksissa ilman ohjelmistokehystä hyvin heikosti, jos ollenkaan. iScroll on Matteo Spinellin kehittämä jQuery-lisäosa, jonka natiivimaiset vieritys ominaisuudet toimivat muun muassa tietokoneissa, mobiililaitteissa ja jopa älytelevision puolella. iScrollin hyviä puolia laajan laitetuen lisäksi on sen pieni tiedostoko verrattuna ominaisuuksiin (Spinelli 2014). Tiedoston suorituskykyyn nähden lisäosa ei käytännössä lisännyt sovelluksen latausaikaa ja sopi suunniteltuun kompaktiin toteutukseen.

iScrollin käyttöönotto onnistui linkittämällä se halutun näkymän metatietoihin paikallisena, jonka jälkeen lisäosan komentaminen onnistui JavaScript-riveillä (kuva 6). iScroll-lisäosan oletusasetukset toimivat sovelluksemme tapauksessa ilman muokkauksia.

```
14      <!-- iScroll 4 -->
15      <script type="text/javascript" src="js/iscroll.js"></script>
16
17      <script type="text/javascript">
18
19          var myScroll;
20          function loaded() {
21              myScroll = new iScroll('wrapper', { scrollbarClass: 'myScrollbar' });
22          }
23
24          document.addEventListener('touchmove', function (e) { e.preventDefault(); }, false);
25
26          document.addEventListener('DOMContentLoaded', function () { setTimeout(loaded, 200); }, false);
27
28      </script>
```

Kuva 6. Harjoitteluosiossa alustettu iScroll-lisäosa.

5.3 SlickQuiz

Sovelluksemme tarkoituksena on valmentaa käyttäjä hygieniaosaamistestiin ja päädyimme rakentamaan sovelluksen sisällön kysymyksiin ja suoraan palautteeseen. Sovelluksen harjoitteluosio sisältää kysymyksiä, joihin käyttäjä vastaa hänelle annetuin vaihtoehdoin. Vastauksen jälkeen sovellus antaa suoran palautteen siitä, oliko vastaus oikein ja perustelut oikealla vastaukselle. Kysymysten määrä nousi sovelluksen beta-versiossa yli 150 kysymyksen.

Kysymysten määrä, sekä sovellukselle olennainen suora palaute, vaativat paljon omaa JavaScript-koodia tai tarkoitukseen sopivan lisäosan. Julie Cameronin ke-

hittämä SlickQuiz –lisäosa sopi tarkoitukseemme täydellisesti. Lisäosa sisällytetään projektiin metatietoihin samaan tapaan kuin muutkin JavaScript-tiedostot. SlickQuiz –lisäosan käyttöönotossa oli otettava huomioon lisäosan asetukset. Asetukset pystyy vaihtamaan JavaScript-tiedostoon niille varatuille kohdille (Kuva 7).

```
/*!
 * SlickQuiz jQuery Plugin
 * http://github.com/QuickenLoans/SlickQuiz
 *
 * @updated August 2, 2013
 *
 * @author Julie Cameron - http://www.jewlofthelotus.com
 * @copyright (c) 2013 Quicken Loans - http://www.quickenloans.com
 * @license MIT
 */

(function($){
    $.slickQuiz = function(element, options) {
        var plugin   = this,
            $element = $(element),
            _element = '#' + $element.attr('id'),

            defaults = {
                checkAnswerText:  'Tarkista vastaus!',
                nextQuestionText: 'Seuraava &raquo;',
                backButtonText: '',
                tryAgainText: 'Kokeile uudestaan',
                skipStartButton: true,
                numberOfQuestions: '50',
                randomSort: false,
                randomSortQuestions: true,
                randomSortAnswers: false,
                preventUnanswered: false,
                completionResponseMessaging: false,
                disableResponseMessaging: false
            },
```

Kuva 7. SlickQuiz tekijä- ja lisenssitiedot, sekä asetukset.

Kysymysosiot jaettiin erillisiin JavaScript-tiedostoihin siten, että jokaiselle osiolle oli omat kysymykset omassa tiedostossaan (kuva 8). Tämä ratkaisu edesauttaa sisältötyötä siten, että eri henkilö pystyi työstämään eri osion kysymyksiä eikä tiedostoiden päivittämiseen tullut päällekkäisyyksiä.

```
var quizJSON = {
    "info": {
        "name":    "Treenaus aiheesta mikrobiologia",
        "main":    "",
        "results": "",
        "level1":  "Loistavaa! Kaikki oikein!",
        "level2":  "Hyvä! Melkein täydellinen suoritus.",
        "level3":  "Sait yli puolet oikein! Jatka harjoittelua.",
        "level4":  "Tarvitset lisää treeniä.",
        "level5":  "Aina ei voi voittaa!" // no comma here
    },
    "questions": [
        { // Question 1
            "q": "Lukeutuuko home mikrobeihin?",
            "a": [
                {"option": "Kyllä",      "correct": true},
                {"option": "Ei",     "correct": false} // no comma here
            ],
            "correct": "<p><span>Vastasit oikein!</span> Home on mikrobi. Muita mikrobeita homeen lisäksi ovat mm. bakteerit, virukset, sekä hiivat.",
            "incorrect": "<p><span>Vastasit väärin!</span> Home on mikrobi. Muita mikrobeita homeen lisäksi ovat mm. bakteerit, virukset, sekä hiivat." // no comma here
        },
        { // Question 2
            "q": "Bakteerit voivat lisääntyä tunneissa, jopa miljooniin soluihin.",
            "a": [
                {"option": "Oikein",      "correct": true},
                {"option": "Väärin",     "correct": false} // no comma here
            ],
           "correct": "<p><span>Vastasit oikein!</span>Bakteerit lisääntyvät jakautumalla suvuttomasti kahtia. Kun bakteerille tarjotaan sille otollista ruokaa, voivat bakteerit lisääntyä muutamassakin tunnissa jo miljooniin soluihin grammassa elintarviketta.</p>",
            "incorrect": "<p><span>Vastasit väärin!</span>Bakteerit lisääntyvät jakautumalla suvuttomasti kahtia. Kun bakteerille tarjotaan sille otollista ruokaa, voivat bakteerit lisääntyä muutamassakin tunnissa jo miljooniin soluihin grammassa elintarviketta.</p>" // no comma here
        },
```

Kuva 8. Mikrobit-osion kysymykset JSON-taulukossa. Kysymykset yksi ja kaksi ovat korostettuna.

Hyvillä lisäosilla saadaan korvatuksi ohjelmistokehysten tuomat hyödyt ja täten päästään lähelle natiivia toteutusta. Sovelluksemme laajuus ja toiminnallisuudet huomioon ottaen voidaan todeta, että valmiin sovelluksen voi rakentaa verkkotekniikoilla. Ratkaisu on siis verkkotekniikoiden ja ohjelmistokehysten yhteistyöllä kasattu *hybridisovellus,* tai yksinomaan verkkotekniikoilla toteutettu *verkkosovellus*.

6 Hybridisovellus

6.1 Yleistä

Sovelluksemme kehitysajankohtana mobiilisovellukset voitiin jakaa kolmeen kategoriaan. *Natiivisovellus* on suoraan laitteelle osoitettu, laitteen ymmärtämällä kielellä koodattu ja laitteen ominaisuuksia hyödyntävä sovellus. Natiivisovellus

asentaa tarvittavat tiedostot laitteeseen ja pystyy myös käyttämään laitteen sisäänrakennettuja toimintoja, kuten eleitä ja mobiililaitteen ilmoituskeskusta. (Budiu 2013.) *Verkkosovellus* on kokonaisuudessaan verkkotekniikoilla toteutettu sovellus, joka pyörii mobiililaitteen selaimessa. Verkkosovelluksessa kuljetaan verkosta tuttujen linkkien välityksellä näkymästä toiseen (Budiu 2013). Verkkosovellus onkin nykypäivänä yleisimpiä mobiilissa käytettäviä sovellusmuotoja, koska siihen lasketaan *responsiivisesti* toimivat HTML5-sivustot.

Hybridisovellus on yhdistelmä edellä mainituista sovelluskategorioista. Hybridisovellus rakennetaan verkkotekniikoilla ja sovelluksen esitys nojautuu hyvin pitkälti verkkosovelluksen lailla selaimessa pyörimiseen. Hybridisovelluksen tapauksessa sovellus pyörii mobiililaitteen selaimen sijaan, niin sanotussa *web-näkymässä*. Hybridisovellukset tarvitsevat viimeistään pakkausvaiheessa ohjelmointikehystä, jonka avulla päästään laitteen ominaisuuksiin käsiksi (Budiu 2013). Näihin ohjelmointikehyksiin kuuluu Cordova, jota käytämme projektissa tulevaisuudessa tulevaan viralliseen julkaisuun.

6.2 Hybridi vai verkkosovellus

Verkkosovellus tarvitsee aina internet-yhteyden toimiakseen, jos se pyöritetään mobiililaitteen selaimessa (Mäkinen 2015, 9). Sovelluksemme yksi lähtökohtaisista tavoitteista oli toiminta mahdollisuus ilman verkkoyhteyttä, joten verkkosovellus toteutus olisi jo tämän perusteella ollut ristiriidassa tavoitteiden kanssa. Kehityksessä totesimme verkkosovelluksen eroavan liikaa toivotusta natiivista kokemuksesta myös visuaalisella puolella käytettävyydessä. Esimerkiksi selatessa näkymää ylös ja alas, näkymän rajat liikkuivat kosketuksen mukana paljastaen tyhjää tilaa näkymän ulkopuolelta. Lisäosan iScroll tarkoitus oli poistaa kyseinen ongelma ja siinä se onnistui tietokoneen selaimella sovellusta käytettäessä. Mobiililaitteilla ongelma toistui ja rikkoi tarkoituksen kiinteästä valikosta ja staattisesta sisällöstä. Hybridisovelluksen tapauksessa sitomalla verkkototeutus Cordovalla mobiililaitteeseen saimme iScroll-lisäosan toimimaan haluamallamme tavalla. Verkkosovelluksen kehittäminen on helpoin reitti mobiilisovelluksen luomiseen, jos taustalta löytyy osaamista yleisimpiin verkkotekniikoihin. Yllä mainitut ongelmat verkkosovelluksen kehityksessä ja verkkosovelluksen hitaus verrattuna

hybridiratkaisuun johtivat päätökseen jatkaa kehitystä verkkotekniikoilla ottaen Cordova ohjelmointikehyksen mukaan julkaisua lähentyessä.

6.3 Hybridi vastaan ohjelmistokehys sovelluksemme tapauksessa

Valmiit kehysten sisältämät käyttöliittymäsapluunat tuovat omat haasteensa ottaen huomioon, että suurimmat mobiilialustat eroavat toisistaan huomattavasti esimerkiksi grafiikaltaan ja toiminnoiltaan. Eroavaisuudet eivät jää esimerkiksi Windowsin ja Androidin välillä pelkästään ulkoasullisiin eroihin, mutta myös siirtymiin ja animaatioihin. (Borley 2012.) Kaikki tarkastelemamme kehykset tuovat mukanaan graafiset sapluunat, jotka sovelluksemme kannalta olivat turhia ja niiden ulkoasulliset tyylittelyt toivat ylimääräistä lisätyötä. Tiesimme sovelluksen kohdealustat ja niiden ulkoasulliset standardit, joten puhtaan HTML- ja CSS-rakenteen luominen niin sanotusti alusta alkaen itse oli suoraviivaisempaa ilman valmiita sapluunoita.

On tärkeää tietää kehitettävän sovelluksen laajuus ja käyttötarkoitus. Sisällöiltään laajemmissa ja rakenteellisesti monipuolisimmissa mobiilisovelluksissa HTML5:n latausajat voivat kasvaa ilman virtaviivaista ohjelmistokehystä. Osa sovelluksista voi vaatia esimerkiksi mobiililaitteen fyysisiä antureita hyväkseen, joten tämän kaltaisissa tapauksissa ohjelmistokehykset ovat loogisempi ratkaisu mobiilikehitykseen. (Stark 2011.)

7 Lopputulos

7.1.Hygipassi-sovellus

Vuonna 2013 valmistunut beta-versio mobiilisovelluksestamme täytti kaikki sille asetetut odotukset niin toteutukseltaan, kuin lopputulokseltaan. Sovelluksen kehittäminen ja sen aikana eteen tulleet haasteet ratkesivat siten, että aiemmin opittu tekninen osaaminen tuli hyödynnettyä uudella tavalla. Kehitysprojektin aikana teknisten vaihtoehtojen punnitsemisen lisäksi yleinen projektinhallinta ja

suunnittelu nousivat tärkeään rooliin.

Sovellus on kehitetty iOS ja Android alustoja varten, eikä sisällä tukea muille alustoille kuten Windows Phonelle. Sovellus toimii ilman verkkoyhteyttä paikallisena hybridisovelluksena, eikä käytä suoranaisesti mitään puhelimen antureita hyväkseen. Sovellusta kehitettäessä pääsääntöisinä työvälineinä olivat jQuery-kirjasto sekä JavaScript-lisäosat SlickQuiz ja iScroll. Tekninen kehittäminen tapahtui pääsääntöisesti Adobe Dreamweaver -ohjelmistolla, jonka lisäksi sovelluksen testaukset ja esikatselun suoritimme verkkosovelluksena Google Chrome -selaimella. Kehityksen loppuvaiheilla varsinkin mobiililaitteella testaukseen kehitystä tuki Cordova -ohjelmointikehys, sekä Xcode -ohjelmisto.

7.2 Rakenne ja työtiedostot

Sovelluksen rakennetta tarkastellessa selviää tarkemmin kuinka projektissamme kehitimme teknisesti hyvin natiivisovelluksen kaltaisen toteutuksen. Rakenteeltaan sovellus jakautuu aloitusruudun lisäksi harjoittelu-, testi- sekä oppimateriaaliosioihin. Osiot ovat erillisiä HTML-tiedostoja jotka linkittyvät toisiinsa HTML-elementti iframe sisään. Jotta saimme haluamamme nopean, natiivimaisen navigoinnin näkymien välille, käytimme kyseistä elementtiä pääsääntöisenä näkymien esittäjänä. Sovelluksen indeksitiedosto sisältää navigaation ja elementin, jonka sisään eri näkymät esitetään (kuva 9). Navigoinnin lisäksi yllä mainittu tiedosto sisältää tarvittavat metatiedot, lisätyt ulkoiset kirjastot sekä iScroll-lisäosaan vaadittavat JavaScript komennot (kuva 10).

```
<body>

<div id="navbar">

    <div id="navicons" class="float_right">
        <ul id="naviul">
            <div id="mainicon" class="float_left"><li><a href="etusivu.html#top" target="iframe"><img height="45px" src="images/logobeta.png"/></a></li></div>
            <div id="icon_oppi" class="float_right icons"><li><a href="oppimateriaali.html#top" target="iframe"><img src="images/pienet/oppi.png"/></a></li></div>
            <div id="icon_tentti" class="float_right icons"><li><a href="tentti_intro.html#top" target="iframe"><img src="images/pienet/tentti.png"/></a></li></div>
            <div id="icon_treeni" class="float_right icons"><li><a href="treenaus.html#top" target="iframe"><img src="images/pienet/treeni.png"/></a></li></div>
        </ul>
    </div>
</div>

<iframe id="iframe" name="iframe" src="treenaus.html" frameborder="0" style="overflow:hidden;overflow-x:hidden;overflow-y:hidden;height:100%;width:100%;position:absolute;top:0px;left:0px;right:0px;bottom:0px" height="100%" width="100%"></iframe>


</body>
```

Kuva 9. index.html tiedoston rakenne.

```
<head>
    <!-- OTSIKKO -->
    <title>Hygieniasovellus</title>

    <!-- VIEWPORT ASETUKSET -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <!-- jQuery -->
    <script type="text/javascript" src="js/jquery-1.9.1.js"></script>

    <!-- iScroll 4 -->
    <script type="text/javascript" src="js/iscroll.js"></script>

    <script type="text/javascript">

                    var myScroll;
                    function loaded() {
                        myScroll = new iScroll('wrapper');
                    }

                    document.addEventListener('touchmove', function (e) { e.preventDefault(); }, false);

                    document.addEventListener('DOMContentLoaded', function () { setTimeout(loaded, 200); }, false);

                    </script>
```

Kuva 10. Meta-tiedot, ulkoiset kirjastot ja JavaScript-komento.

Päänavigoinnin lisäksi näkymästä toiseen siirtyminen tapahtui sisäsivuilla ohjaten samaan iframe -elementtiin. Harjoitteluosio on jaettu sovelluksessa kategorioihin, jotka osoittavat kunkin aihealueen omaan HTML-tiedostoon (kuva 11). Jokainen aihealue tiedosto on SlickQuiz-kysymyspakettia lukuun ottamatta samanlaisia (kuva 12).

```
<div id="wrapper">
    <div id="scroller">
        <div id="content">

            <h1>Treenaus</h1>
            <div id="pallo_info" ></div><p>Pallolla merkityissä kysymyksissä voit vastata vain yhdellä vaihtoehdolla.</p>
            <div id="nelio_info" ></div><p>Neliöllä merkityissä kysymyksissä voit valita useita vaihtoehtoja.</p>
                    <div id="kategoria_lista">
                        <a href="treeni_mikrobit.html#top" target="iframe"><div id="kategoria_linkki">
                            <div class="kuva sini"><img class="likuva" src="images/pienet/mikrobiologia.png"/></div><div
class="teksti">Mikrobiologia</div><div class="nuoli"><img  height="25" class="likuva" src="images/nuoli.png"/></div>
                        </div></a>
                        <a href="treeni_terveys.html#top" target="iframe"><div id="kategoria_linkki">
                            <div class="kuva tumma"><img class="likuva" src="images/pienet/vaara.png"/></div><div
class="teksti">Terveydelliset haitat</div><div class="nuoli"><img  height="25" class="likuva" src="images/nuoli.png"/></div>
                        </div></a>
                        <a href="treeni_elintarvike.html#top" target="iframe"><div id="kategoria_linkki">
                            <div class="kuva vihrea"><img class="likuva" src="images/pienet/tyotavat.png"/></div><div
class="teksti">Elintarvikkeiden käsittely</div><div class="nuoli"><img  height="25" class="likuva"  src="images/nuoli.png"/>
</div>
                        </div></a>
                        <a href="treeni_hygienia.html#top" target="iframe"><div id="kategoria_linkki">
                            <div class="kuva sini"><img class="likuva" height="25" src="images/kasi.png"/></div><div
class="teksti">Henkilökohtainen hygienia</div><div class="nuoli"><img class="likuva" height="25" src="images/nuoli.png"/>
</div>
                        </div></a>
                        <a href="treeni_puhtaanapito.html#top" target="iframe"><div id="kategoria_linkki">
                            <div class="kuva tumma"><img class="likuva" src="images/pienet/jatehuolto.png"/></div><div
class="teksti">Puhtaanapito</div><div class="nuoli"><img class="likuva" height="25" src="images/nuoli.png"/></div>
                        </div></a>
                        <a href="treeni_omavalvonta.html#top" target="iframe"><div id="kategoria_linkki">
                            <div class="kuva vihrea"><img class="likuva" height="25" src="images/pienet/omavalvonta.png"/>
</div><div class="teksti">Omavalvonta</div><div class="nuoli"><img class="likuva" height="25" src="images/nuoli.png"/></div>
                        </div></a>
                        <a href="treeni_laki.html#top" target="iframe"><div id="kategoria_linkki">
                            <div class="kuva sini"><img class="likuva" height="25" src="images/pienet/laki.png"/></div><div
class="teksti">Elintarvikelainsäädäntö</div><div class="nuoli"><img class="likuva" height="25" src="images/nuoli.png"/></div>
                        </div></a>

                    </div>
        </div>
    </div>
</div>
```

Kuva 11. Harjoitteluosion navigaatio. Ohjaukset esimerkiksi tiedostoon *treeni_mikrobit.html*.

```
    <!-- SlickQuiz -->
        <link href="css/reset.css" media="screen" rel="stylesheet" type="text/css">
        <link href="css/slickQuiz.css" media="screen" rel="stylesheet" type="text/css">
        <link href="css/master.css" media="screen" rel="stylesheet" type="text/css">

        <script src="js/slickQuiz-treenausmalli.js"></script>
        <script src="js/master.js"></script>
        <script src="js/kysymykset/slickQuiz-treenaus-mikrobit.js"></script>
</head>

<body>
    <div  id="slickQuiz">
        <h1 class="quizName"><!-- where the quiz name goes --></h1>

        <div class="quizArea">
            <div class="quizHeader">
                <!-- where the quiz main copy goes -->

                <a class="button startQuiz" href="#">Aloita treenaus!</a>
            </div>

            <!-- where the quiz gets built -->
        </div>

        <div class="quizResults">
            <h3 class="quizScore">Sait pisteet: <span><!-- where the quiz score goes --></span></h3>

            <h3 class="quizLevel"><strong>Tasosi:</strong> <span><!-- where the quiz ranking level goes --></span></h3>

            <div class="quizResultsCopy">
                <!-- where the quiz result copy goes -->
            </div>
        </div>
    </div>
</body>
```

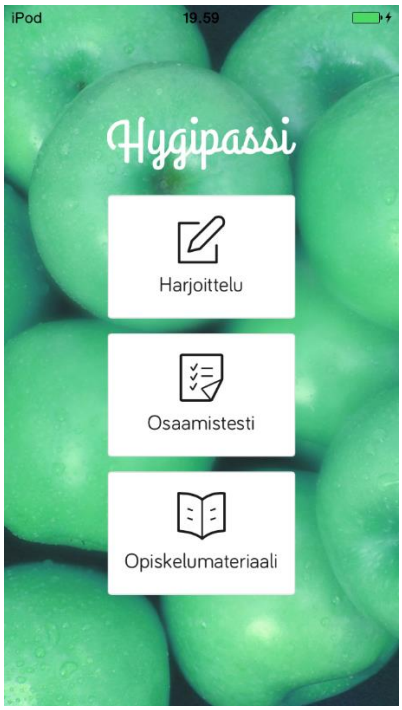
Kuva 12. Mikrobit aihe-alueen SlickQuiz-osio. Koodissa on korostettu oikea kysymyspaketti.

Sovellukseemme toteutettu rakenne eroaa esimerkiksi jQuery Mobilesta ja Sencha Touchista siten, että päädyimme ratkaisuun, jossa jokainen näkymä pyrittiin pitämään eri HTML-tiedostoissa. Nämä erilliset tiedostot sisälsivät samankaltaisuuksia suurimmaksi osaksi muun muassa metatiedoissa.

7.3 Ulkoasu ja sisältö

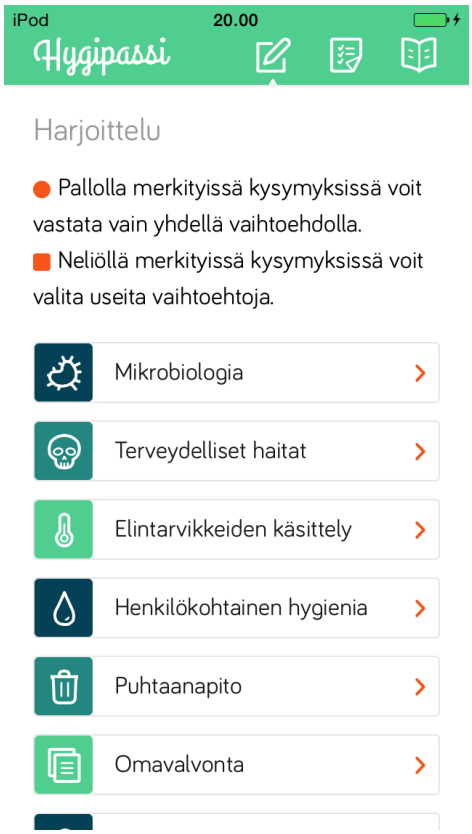
Projektissa roolini vastuualue painottui vahvasti tekniselle puolelle, mutta olin mukana graafisissa toteutuksissa muun muassa antamalla mielipiteitä Horttanaisen ratkaisuihin, sekä seuraamalla vierestä asiantuntevaa graafista suunnittelua. Tässä luvussa esittelen kuvin, miltä aikaisemmin avatut koodit näyttävät sovelluksen käyttöliittymässä.

Sovelluksen etusivu eli aloitusnäkymä sisältää iframe-linkitykset sovelluksen pääsisältöön (kuva 13).



Kuva 13. Hygipassi-mobiilisovelluksen aloitusnäkymä.

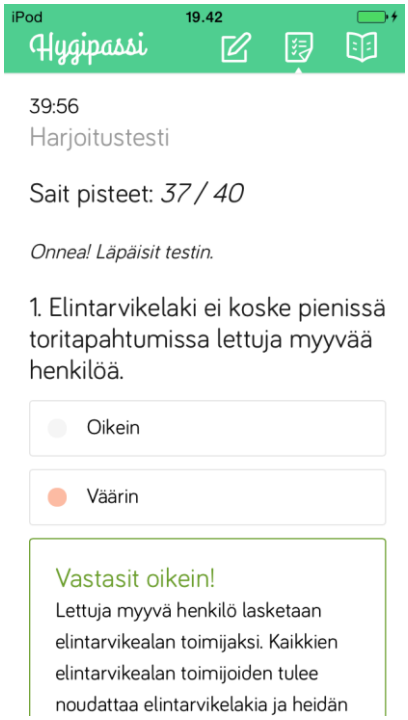
Valitsemalla harjoitteluosioon johtavan linkin, avautuu harjoitusosio. Tässä vaiheessa mukaan tulee myös indeksitiedostossa luotu navigaatio, joka seuraa käyttöliittymässä jokaisen näkymän mukana (kuva 14).



Kuva 14. Harjoitteluosio, joka sisältää ohjeet ja aihealueet.

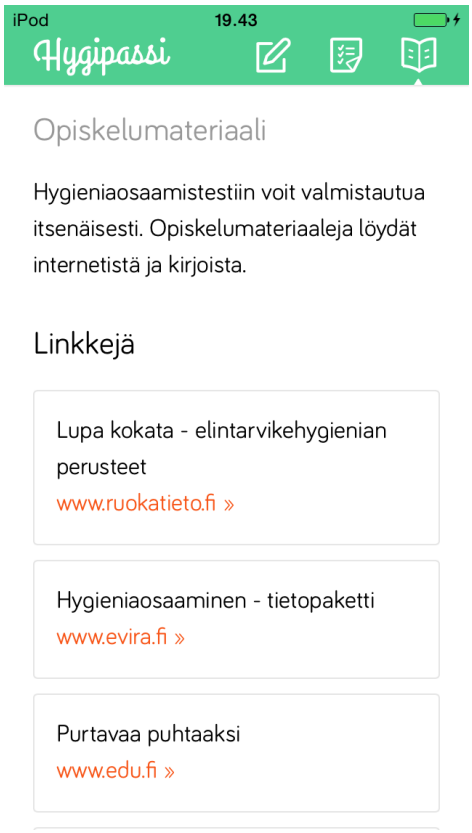
Harjoitteluosiossa listautuu aikaisemmin koodissa rakennettuun kategorianavigaatioon (kuva 11). Osaamistesti ja harjoitteluosion sisällöt toimivat SlickQuiz-lisäosalla rakennettuna ja ovat samankaltaisia (kuva 15 ja kuva 16).

Kuva 15. Harjoitteluosion kysymys ja käyttäjän syöttämä vastaus.



Kuva 16. Käyttäjän läpäisemä osaamistesti.

Sovelluksen viimeinen näkymä sisältää linkit opiskelumateriaaliin, jonka avulla käyttäjä voi valmistautua harjoitteluosioon ja osaamistestiin (kuva 17).



Kuva 17. Opiskelumateriaali, joka sisältää linkit ulkoisiin oppimateriaaleihin.

8 Pohdinta

Pohdin kokonaisuutta työelämälähtöisesti, sekä tarkastelen sovelluksen nykytilaa ja tulevaisuutta. Pyrin pitämään opinnäytetyöraportin sisällöt ajankohtaisina siten, että paneuduin tekniikkoihin, jotka ovat todennäköisesti pitkäikäisiä. HTML5, CSS, JavaScript ja jQuery tulevat todennäköisesti johtamaan verkkotekniikoilla toteutettavia kehittämisprojekteja. Tekstistä on rajattu pois ohjelmistokehyksiä, jotka eivät oleellisesti vaikuttaneet sovelluksemme kehityksen suuntaan. Opinnäytetyöraportissani käyn läpi asioita, jotka tulevat vastaan teknisessä kehittämisessä, oli kyseessä sitten verkkopalvelun tai mobiilisovelluksen kehittäminen. Näitä asioita ovat muun muassa valinnat kehitysalustojen ja tapojen väliltä. Tämän hetkisen tilanteen mukaan tulen työskentelemään paljon projektien parissa, joissa toteutustavat vaikuttavat paljon lopputulokseen. Usein lopputuloksen

pitää olla laadukas, mutta samaan aikaan kustannustehokas toteuttaa. Opinnäytetyön toiminallisessa osuudessa opin mobiilisovelluksen kaikki kehitysvaiheet ja sain tuloksia kuinka rakentaa mobiilisovellus pääosin verkkotekniikoita käyttäen.

Projektille varattu aika oli rajallinen, jonka ansiosta opin myös taidot kustannustehokkaaseen suunnitteluun ja kehittämiseen. Pienessä ryhmässä työskentelyssä on olennaista, että pääsääntöisestä roolituksesta huolimatta kokonaisuus pysyy koko projektin aikana kaikkien jäsenten hallussa. Valmis mobiilisovellus toimii hyvänä esimerkkinä osaamisesta niin tekniikan, konseptoinnin kuin projektinhallinnan puolelta. Konkreettinen valmis mobiilisovellus on jo aikaisemmassa työhaku tilanteessa osoittautunut hyväksi osaksi portfoliota.

Tarkoitus olisi viimeistellä sovellus Horttanaisen kanssa siihen muotoon, että sen voi julkaista Android- ja iOS -laitteille. Tekniseen viimeistelyyn sisältyy muun muassa jQuery–kirjaston päivittäminen uusimpaan versioon, sekä lisäosien versioiden tarkastaminen. Myös Cordova prosessi pitää toteuttaa uudelleen, uusien puhelin versioiden ollessa kehittyneempiä vuoteen 2013 verrattuna. Mobiilisovelluksemme tekninen ja graafinen runko mahdollistavat jatkokehitystä myös muihin sovellustarpeisiin ja alustavat keskustelut Horttanaisen kanssa kehittämisen jatkon suhteen ovat positiivisia.

Vaikka raportissa ilmenee tekniset valinnat joihin vuoden 2013 sovellusversion aikana päädyimme, on mahdollista, että tulevaisuudessa ottaisimme huomioon myös muut mobiiliohjelmointikehykset. Seuraan innolla kehittyvää alaa ja sen tuomia mahdollisuuksia niin mobiili- kuin verkkokehityksessä. Uskon, että tulevaisuudessa esimerkiksi verkkoyhteyksien maailmanlaajuisen yleistymisen ja verkkonopeuksien kasvamisen ansiosta natiivin- ja verkkototeutuksen välillä tulee olemaan enää hyvin pieni ero.

Lähteet

Angelov, M. 2013. 50 Amazing jQuery Plugins That You Should Start Using Right Now. http://tutorialzine.com/2013/04/50-amazing-jquery-plugins/. 10.05.2015.

Borley, R. 2012. The problem with mobile frameworks. https://boag-world.com/mobile-web/the-problem-with-mobile-frameworks/. 23.7.2014.

Budiu, R. 2013. Mobile: Native Apps, Web Apps, and Hybrid Apps. http://www.nngroup.com/articles/mobile-native-apps/. 14.9.2014.

Chuan, S. 2015 HTML5 Mobile Development Cookbook. Birmingham B3 2PB, UK: Packt Publishing Ltd.

Creative Bloq. 2013. Five Reasons To Use jQuery Mobile. http://www.creativeb-loq.com/jquery/five-reasons-use-jquery-mobile-4135691. 20.3.2015.

DocForge. 2013. Framework. http://docforge.com/wiki/Framework. 14.5.2015.

Duckett, J. 2011. HTML & CSS Design and Build Websites. Indianapolis, IN: John Wiley & Sons, Inc.

Falk, M. Mobile frameworks comparison chart. http://mobile-frameworks-com-parison-chart.com/. 20.3.2015.

Harrel, W. 2011. HTML, CSS & JavaScript® Mobile Development For Dummies. Hoboken, NJ: John Wiley & Sons, Inc.

Irish, P. & Manian, D. 2015. HTML5 & CSS3 Readiness. http://html5readi-ness.com/. 19.3.2015.

Jain, C. 2012. jQuery Mobile Cookbook. Birmingham B3 2PB, UK: Packt Pub-lishing Ltd.

Leenheer, N. 2015. HTML5 test. https://html5test.com/results/mobile.htm .19.3.2015.

Mehtonen, M. 2013. Sencha Touch 2 –pluginit. Karelia-Ammattikorkeakoulu. Viestinnän koulutusohjelma. Opinnäytetyö. http://theseus.fi/bitstream/handle/10024/57205/Mehtonen_Matti.pdf?se-quence=1. 29.5.2015.

Mäkinen, A. 2015. HTML5:n integrointi Windows Phone –sovellukseen. Karelia-Ammattikorkeakoulu. Viestinnän koulutusohjelma. Opinnäytetyö. http://theseus.fi/bitstream/handle/10024/86850/Makinen_Anni.pdf?se-quence=1. 29.5.2015.

Pilgrim, M. 2015. Dive Into HTML5. http://www.je-susda.com/docs/ebooks/ebook_manual_en_dive-into-html5.pdf. 11.3.2015.

The Roveb Blog. 2012. Our Expirience With jQuery Mobile and Sencha Touch. http://blog.roveb.com/post/17259708005/our-experience-with-jquery-mo-bile-and-sencha-touch. 25.3.2015.

Rotton, T. 2013. Writing Your Own jQuery Plugins. http://blog.teamtree-house.com/writing-your-own-jquery-plugins. 10.05.2015.

Rudolph, P. 2014. Hybrid Mobile Apps: Providing A Native Experience With Web

Technologies. Smashing Magazine http://www.smashingmagazine.com/2014/10/21/providing-a-native-experience-with-web-technologies/. 21.10.2014.

Sawyer McFarland, D. 2014. JavaScript & jQuery: The Missing Manual. Sebastopol, CA: O'Reilly Media, Inc.

Sencha 2015. Sencha Touch. http://www.sencha.com/products/touch/#overview. 14.5.2015.

Spinelli, M. 2014. ISCROLL 5. http://cubiq.org/iscroll-5. 1.5.2015.

Stark, J. 2011. The developer's guide to mobile frameworks. http://www.creativebloq.com/android/developers-guide-mobile-frameworks-10116682. 25.04.2015.

Themeroller. 2015. ThemeRoller for jQuery Mobile. https://themeroller.jquerymobile.com/. 14.4.2015.

W3C. 2014. W3C Recommendation 28 October 2014. http://www.w3.org/TR/html5/. 10.3.2015.

W3C SCHOOLS. 2015a. HTML <meta> Tag

http://www.w3schools.com/tags/tag_meta.asp. 20.4.2015.

W3C SCHOOLS. 2015b. HTML Head

http://www.w3schools.com/html/html_head.asp. 20.4.2015.

W3C SCHOOLS. 2015c. HTML <style> Tag

http://www.w3schools.com/tags/tag_style.asp. 20.4.2015.

Wikipedia. 2015. Web application framework

http://en.wikipedia.org/wiki/Web_application_framework. 12.4.2015.

Rautalankasuunnitelma