Joni Finne

UUTISVIRTA-MOBIILISOVELLUS TIETOTEKNIIKAN OPISKELIJOILLE

Opinnäytetyö

Tietotekniikan koulutusohjelma

Marraskuu 2016

KYAMK
University of Applied Sciences

Tekijä/Tekijät	Tutkinto	Aika
Joni Finne	Insinööri (AMK)	Marraskuu 2016
Opinnäytetyön nimi Uutisvirta-mobiilisovellus tietotekniikan opiskelijoille		40 sivua
Toimeksiantaja Kymenlaakson ammattikorkeakoulu / GameLab		
Ohjaaja Niina Salmi, lehtori		

Tiivistelmä

Opinnäytetyön tarkoitus on esitellä uutisvirta-mobiilisovellus, joka tehtiin työharjoittelun yhteydessä. Sovelluksen tarkoituksena oli helpottaa kommunikaatiota tietotekniikan laboratorion henkilökunnan ja opiskelijoiden välillä käyttämällä lyhyitä ja helppolukuisia postauksia, joista käyttäjä saa ilmoituksen laitteelleen. Työssä esitellään sovelluksen olennaiset ominaisuudet ja niiden toiminnallisuus.

Sovellusta ohjelmoitiin Visual Studio 2015 kehitysympäristössä Cordova-lisäosan kanssa, koska sovellus suunniteltiin monialustaiseksi. Ohjelmointi perustui HTML5-merkintäkieleen, joka sisältää mahdollisuuden käyttää JavaScript-komentosarjoja. Näillä komentosarjoilla ohjelmoitiin sovelluksen ulkonäkö sekä toiminnallisuus käyttäjän ja palvelimen välillä. Sovelluksessa käytettiin palvelimella ollutta tietokantaa, johon tallennettiin käyttäjän tiedot, tagit sekä käyttäjän luomat postaukset.

Alkuperäisen suunnitelman mukaista monialustaista sovellusta ei saatu luotua, vaan luotiin ainoastaan Android-versio, joka testattiin tietotekniikan laboratoriossa paikalla olleiden henkilöiden toimesta. Tätä versiota korjattiin saadun palautteen perusteella, mutta testaus jäi varsin lyhyeksi, sillä kesäaikaan laboratoriossa ei ollut monia käyttäjiä.

Toteutettu sovellus on kehitysversio, jota täytyy vielä jatkokehittää. Mikäli sovellusta haluttaisiin jatkokehittää, tarvittaisiin sovellukseen parempi käyttölittymä ja graafinen ulkoasu. Myös monialustaisuutta täytyy miettiä uudelleen kysynnän mukaan. Testausvaiheessa tehdyn kyselyn mukaan suurimmalla osalla opiskelijoista oli käytössään vain Android-laite.

Asiasanat
sovelluskehitys, Visual Studio, html5, JavaScript, mobiilisovellukset

Author (authors)	Degree	Time
Joni Finne	Bachelor of Engineering	November 2016

Thesis Title

Newsfeed Mobile Application for Students of Computer Science

40 pages

Commissioned by

Kymenlaakso University of Applied Sciences / GameLab

Supervisor

Niina Salmi, Senior Lecturer

Abstract

The objective of this thesis was to present newsfeed mobile application which was made during practical training. Application was meant to help with communication between the computer science laboratory staff and students with small and easy to read posts which user would get notified on their mobile device. Thesis presents the main features of the application as well as their functionality.

Application was programmed with Visual Studio 2015 development environment using Cordova tools which allowed cross-platform development. Programming was based on HTML5 markup language which included possibility to use JavaScript programming language. Applications layout and functionality between user and server were written with JavaScript. Application used database which stores user data, tags and posts made by users.

Originally planned cross-platform application wasn't possible to create which led application to be only for Android devices. This version of the application was tested at the computer science laboratory by the students and staff present at the time. Application was fixed according to the feedback received from the testers. Testing period was cut short because of the summer period as there were no more testers for the application.

This version of the application still needs more developing. If there was going to be any more developing, the application would need better interface and graphics layout. Cross-platform support needs to be thought again as according to an enquiry made for the laboratory users there were mainly Android devices in use.

Keywords

program development, Visual Studio, html5, JavaScript, mobile applications

SISÄLLYS

AVAINSANAT JA SELITYKSET

AJAX	Asynchronous JavaScript and XML on usean eri web-tekniikan mm. JavaScriptin ja XML:n yhdistelmätermi.
Backend	Sovelluksen taustaohjelmisto, palvelin osuus.
Cross-platform	Alustariippumattomuus eli sovellus toimii monella eri käyttöjärjestelmällä.
CSS	Cascading Style Sheets ovat verkkosivujen dokumenttien tyyliohjeet, joiden avulla verkkosivun ulkonäköä säännöstellään.
MIME	Multipurpose Internet Mail Extensions on tiedoston sisällön kertova standardi.
Postaus	Käyttäjän lisäämä viesti, juttu tai uutinen.
Push-notifikaatio	Palvelimen lähettämä datapaketti sovellukselta toiselle.
SDK	Software development kit on kehitystyökalu tietylle sovellukselle.
SFTP	Secure File Transfer Protocol on Secure Shellin päälle rakennettu salatun tiedonsiirron protokolla.
SQL	Scructured Query Language on IBM:n kehittämä, standardoitu tietokantojen kyselykieli.
Tagi	Tag, tagi eli tunniste, avainsana.
URI/URL	Uniform Resource Identifiers / Locator on tiedostojen ja verkkosivun sijainnin tunnistamiseen tarvittava merkkijono.
UUID	Universally Unique Identifier on uniikki, 128 bittiä pitkä tunnistin.

1 JOHDANTO

Työn aihe saatiin Kymenlaakson ammattikorkeakoulun tietotekniikan lehtori Niina Salmelta. Tarkoituksena oli toteuttaa uutisvirran tapainen, HTML5-mobiilisovelluksen luonti, johon suositeltiin Microsoftin Visual Studio 2015:tä. Vaatimuksena sovellukselle oli cross-platform eli monialusta tuki. Tähän kuuluivat Apple iOS, Google Android ja Microsoft Windows Phone -käyttöjärjestelmät. Sovelluksesta haluttiin myös versio, joka tulisi tietotekniikan laboratorioon infonäytölle, josta opiskelijat voisivat tarkistaa mm. ajankohtaisten tapahtumien tiedot.

Kommunikaatio tietotekniikan eri opiskelijaryhmien ja opettajien välillä on osoittautunut ongelmaksi, jonka ratkaisemiseksi sovellus haluttiin luoda. Tieto esimerkiksi tuntien peruuntumisesta ei välittynyt opiskelijoille, tai johonkin tapahtumaan ei saatu tarpeeksi osallistujia. Sovelluksen avulla opiskelijat saisivat tiedon tuntien peruuntumisesta suoraan mobiililaitteilleen.

Varsinainen toteutus tapahtui Kymenlaakson ammattikorkeakoululla tietotekniikan pelilaboratoriossa työharjoittelun aikana. Aluksi työharjoittelussa oli mukana kolme henkilöä, joista yhden harjoittelu päättyi ennen kuin sovellus saatiin valmiiksi.

Työtä jaettiin harjoittelijoiden kesken kahteen kokonaisuuteen: käyttäjälle näkyvään visuaaliseen osaan sekä taustatoimintoihin kuuluvaan palvelin- ja tietokantapuoleen.

2 TYÖKALUT

Sovelluksen kehittämiseen tarvittiin joukko erilaisia työkaluja, jotta päästiin työskentelemään halutuilla alustoilla. Työkaluja käytettiin niin sovelluksen varsinaiseen ohjelmointiin sekä taustatoimintojen toiminnan varmistamiseen.

2.1 Kehitysympäristöt

Kehitysympäristöksi valikoitui Visual Studio 2015 ja sen käyttämä lisäosa Apache Cordova. Sovellus haluttiin saada toimimaan monilla eri alustoilla, jolloin myös työkaluista täytyi löytyä *cross-platform* eli monialustaista tukea.

2.1.1 Visual Studio

Suosituksena olleen Visual Studio 2015:n myötä etsittiin monialustaista kehitystukea halutuille alustoille. Vaihtoehtoina haluttuun sovellukseen olivat Xamarin-lisäosa, joka tuki monialustaista sovelluskehitystä vain Android- ja iOS-käyttöjärjestelmille (Xamarin 2013), sekä Apache Cordova työkalu Visual Studiolle.

Xamarin-lisäosan käyttöä harkittiin alustavasti, mutta lisäosan hinnoittelu ei sopinut projektiin. Apache Cordova, avoimen lähdekoodin työkalu (Visual Studio 2016), soveltui paremmin tämän projektin budjettiin ja kehitykseen, sillä se tuki suoraan kaikkia haluttuja mobiilikäyttöjärjestelmiä.

Apache Cordova toimii Visual Studion yhteydessä ja irrallisena lisäosana komentoriviltä, josta pystyttiin kääntämään halutut sovelluksen versiot tai lisäämään komponentteja Cordovaan.

2.1.2 Android SDK

Sovellusta lähdettiin kehittämään Visual Studio 2015 Community -versiolla pelilaboratorion Windows 10 -käyttöjärjestelmän tietokoneilla. Kehitysympäristöön tarvittiin Android-version kääntämiseen Androidin SDK-manageri sekä tarvittavat muut SDK:t. Sovellusta varten asennettiin Androidin API 23–API 19 SDK-alustat ja SDK:n kääntämistyökaluja.

2.1.3 iOS-kehitys

Apple iOS -version kääntämiseksi Cordovalla tarvittiin lisäksi Applen Mac -tietokone, laitteen käyttöjärjestelmä sekä Xcode-kehitysympäristö. Kehittäminen

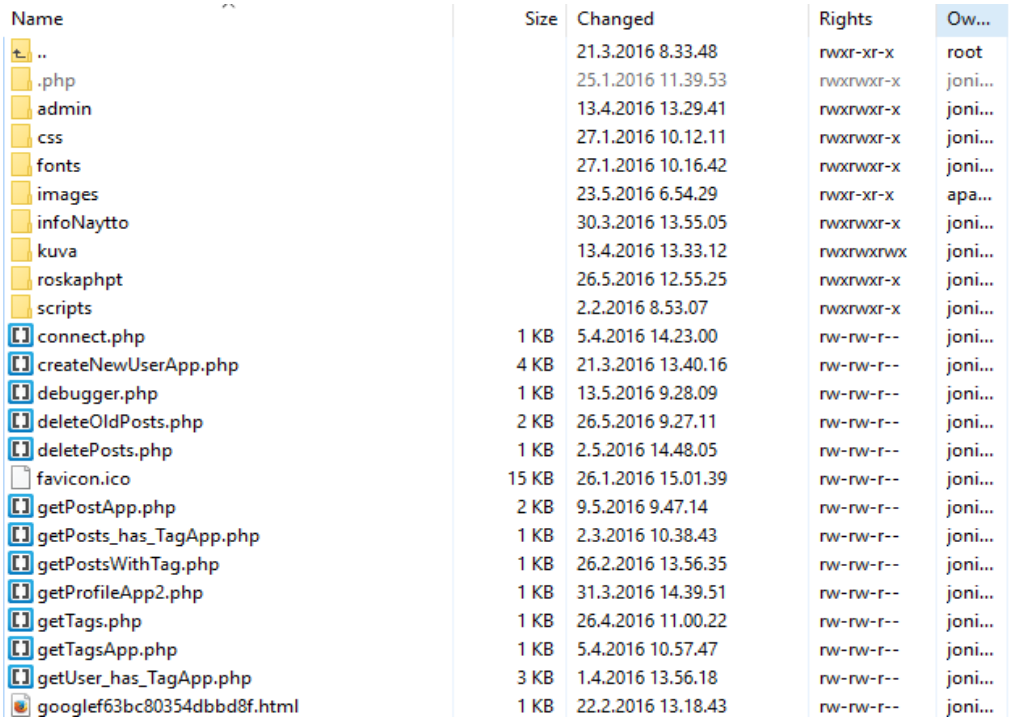
vaati myös Apple Developer Accountin, joka saatiin Kyamkin henkilökunnalta. Tässä työssä käytettiin pelilaboratoriossa ollutta Mac-tietokonetta, jossa oli OSX 10.11 El Capitan -käyttöjärjestelmä.

2.2 Backend-ohjelmistot

Backend-ohjelmistoa eli taustaohjelmistoa tarvittiin, jotta sovelluksen vaatimat SQL-tietokannat saatiin toimimaan. Tätä varten pyydettiin käytettäväksi tietotekniikan laboratoriosta palvelinta, jossa olisi asennettuna SQL-tietokanta. Tarkoituksiin sopiva Papaya-palvelin löytyi, jossa oli asennettuna MariaDB-tietokantajärjestelmä. MariaDB on tehty MySQL-järjestelmän pohjalle alkuperäisten tekijöiden toimesta (GitHub 2016a).

MariaDB-tietokanta toimii SQL-kielellä, jolle kirjoitettiin tietokantahakuja PHP-tiedostoihin. Näitä PHP-tiedostoja kutsuttiin applikaation sisältä AJAX-kutsuja hyödyntäen. AJAX helpotti käyttöä siten, että hakukutsut eivät vaikuttaneet käyttäjän tekemiseen, vaan ne toteutetaan taustalla sovelluksen normaalin toiminnan yhteydessä.

Palvelimelle tarvittiin myös HTML-kansio, jota pystyimme muokkaamaan ja kehittämään. Tämä tuli käyttöön sekä infonäyttöversiota että käyttäjien lähettämiä kuvia varten. Infonäyttöversio sekä kuvat tallennettiin HTML-kansion alle omiin kansioihin.



Name	Size	Changed	Rights	Ow...
..		21.3.2016 8.33.48	rwxr-xr-x	root
.php		25.1.2016 11.39.53	rwxrwxr-x	joni...
admin		13.4.2016 13.29.41	rwxrwxr-x	joni...
css		27.1.2016 10.12.11	rwxrwxr-x	joni...
fonts		27.1.2016 10.16.42	rwxrwxr-x	joni...
images		23.5.2016 6.54.29	rwxr-xr-x	apa...
infoNaytto		30.3.2016 13.55.05	rwxrwxr-x	joni...
kuva		13.4.2016 13.33.12	rwxrwxrwx	joni...
roskaphpt		26.5.2016 12.55.25	rwxrwxr-x	joni...
scripts		2.2.2016 8.53.07	rwxrwxr-x	joni...
connect.php	1 KB	5.4.2016 14.23.00	rw-rw-r--	joni...
createNewUserApp.php	4 KB	21.3.2016 13.40.16	rw-rw-r--	joni...
debugger.php	1 KB	13.5.2016 9.28.09	rw-rw-r--	joni...
deleteOldPosts.php	2 KB	26.5.2016 9.27.11	rw-rw-r--	joni...
deletePosts.php	1 KB	2.5.2016 14.48.05	rw-rw-r--	joni...
favicon.ico	15 KB	26.1.2016 15.01.39	rw-rw-r--	joni...
getPostApp.php	2 KB	9.5.2016 9.47.14	rw-rw-r--	joni...
getPosts_has_TagApp.php	1 KB	2.3.2016 10.38.43	rw-rw-r--	joni...
getPostsWithTag.php	1 KB	26.2.2016 13.56.35	rw-rw-r--	joni...
getProfileApp2.php	1 KB	31.3.2016 14.39.51	rw-rw-r--	joni...
getTags.php	1 KB	26.4.2016 11.00.22	rw-rw-r--	joni...
getTagsApp.php	1 KB	5.4.2016 10.57.47	rw-rw-r--	joni...
getUser_has_TagApp.php	3 KB	1.4.2016 13.56.18	rw-rw-r--	joni...
googlef63bc80354dbbd8f.html	1 KB	22.2.2016 13.18.43	rw-rw-r--	joni...

Kuva 1. Palvelimen tiedostonäkymä, kuvakaappaus WinSCP ohjelmasta

Kuvassa 1 esitetyssä polussa säilytettiin mobiiliapplikaation käyttämiä PHP-tiedostoja sekä joitain ylläpitoa varten tehtyjä PHP-tiedostoja. Ratkaisulla haluttiin myös varmistaa tiedostojen tietoturva estämällä ulkopuolisten pääsy tiedostoihin. Tietoturvaa lisättiin käyttämällä .htaccess-tiedostoa, jonka avulla asetettiin .php-kansio käyttäjätunnuksen ja salasanan taakse.

2.3 Muut työkalut

Kehitysympäristön ja backend-ohjelmistojen lisäksi tarvittiin useampia erilaisia taustaohjelmia, joiden avulla päästiin toteuttamaan haluttuja tuloksia. Muiden työkalujen avulla voitiin myös muokata tiedostoja, jotka eivät olleet mahdollista muokata muualla.

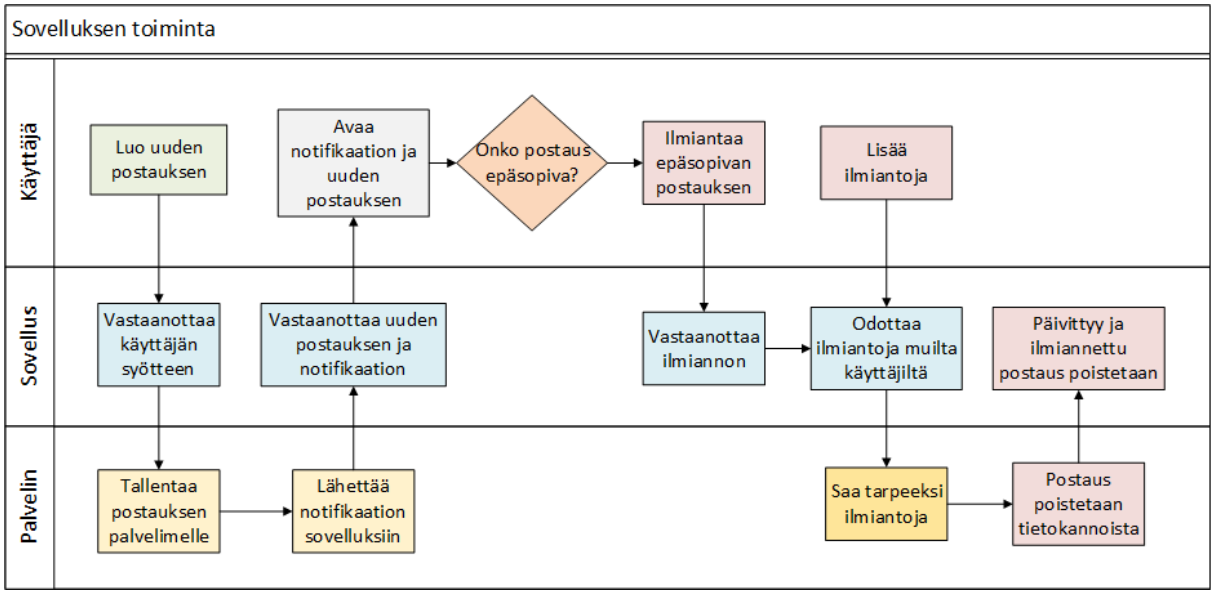
Notepad++ on avoimen lähdekoodin ohjelmointikieliä osaava tekstinkäsittely-ohjelma (Notepad++ 2016). Tällä ohjelmalla voitiin muokata tiedostoja, jotka olivat edellisessä luvussa mainitun palvelimen polussa. Notepad++:ssa muokattavat tiedostot haettiin WinSCP-tiedostonsiirto-ohjelmalla (WinSCP 2014).

WinSCP yhdistää paikallisen koneen palvelimelle halutulla protokolalla. Tässä toteutuksessa yhdistettiin SFTP-protokolalla ja palvelimelle annetulla käyttäjätunnuksella suoraan koulun laboratorion palvelimelle. (R. Heinisuo & I. Rauta 2007, 58-59.)

Node.js on JavaScriptin ajamiseen tarkoitettu moottori, joka sisältää NPM:n eli suurimman avoimen lähdekoodin JavaScript-kirjaston (Node.js 2016). Tämän avulla pystyttiin asentamaan Cordova CLI eli komentorivityökalu, jonka jälkeen voitiin luoda uusia Cordova-projekteja ja muokata niitä komentorivillä.

3 TOIMEKSIANTO JA SUUNNITTELU

Toimeksianto oli uutisvirtasovellus tietotekniikan laboratorion käyttöön. Sovelluksen vaatimuksena oli, että käyttäjät voivat lisätä postauksia, ilmiantaa huonoja postauksia, lisätä tageja lisäämiinsä postauksiin sekä seurata kyseisiä tageja. Tageja haluttiin erittelemään postauksia, jotka olivat tarkoitettu eri ryhmille ja eri asioita koskeviksi. Esimerkiksi tietotekniikan vuoden 2013 ryhmälle luotiin tagi ”TI13”. Vastaavia tageja luotiin kaikille tietotekniikan ryhmille, tapahtumille sekä tärkeille uutisille.

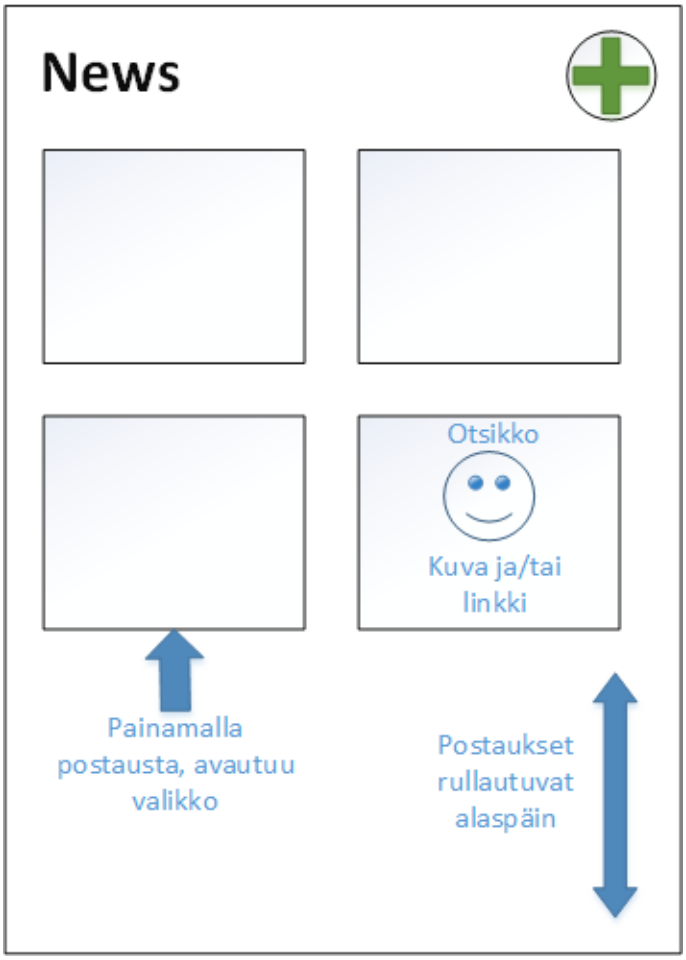


Kuva 2. Uimaratakaavio sovelluksen toiminnasta

Kuvassa 2 on alustavasti suunniteltu sovelluksen toiminta, jossa toiminta alkaa postauksen luomisesta. Sovelluksessa luotu postaus tallentuu palvelimella olevaan tietokantaan ja postauksesta lähetetään notifikaatio eli ilmoitus.

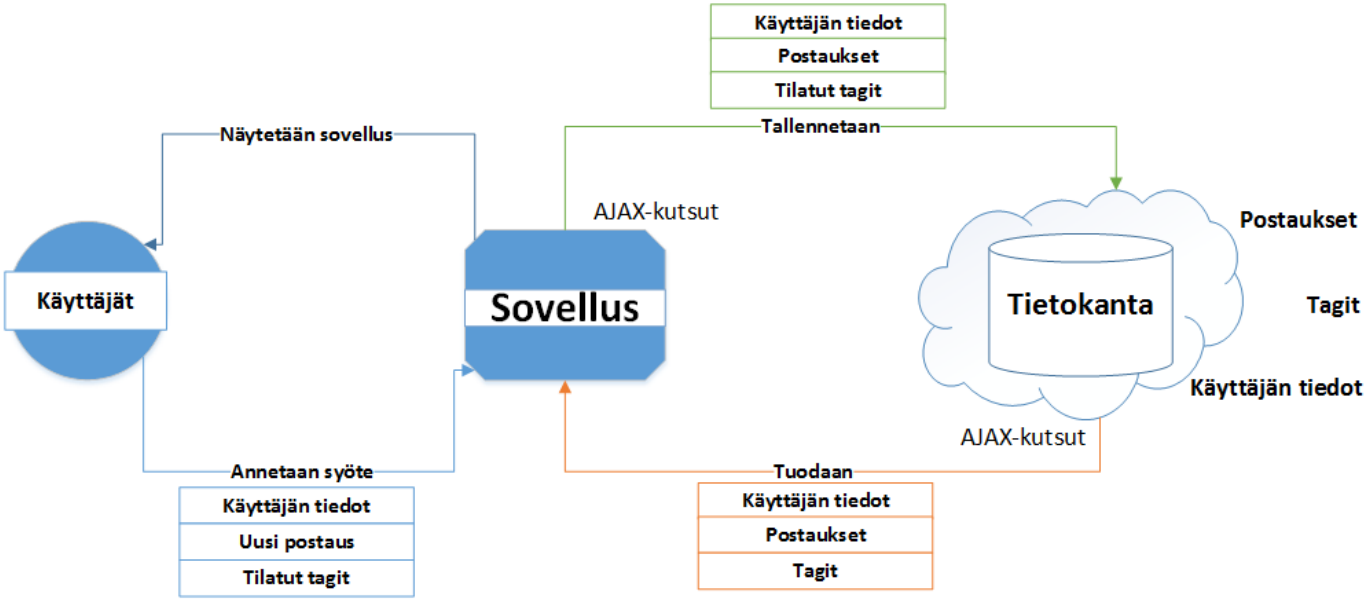
Vain ne käyttäjät, jotka ovat tilanneet kyseisen postauksen tagit, saavat notifikaation.

Avattuaan notifikaation, sovellus näyttää kyseisen postauksen käyttäjälle. Mikäli postaus on käyttäjän mielestä sopimaton tai muuten huono, voi käyttäjä ilmiantaa kyseisen postauksen. Jos postaus ilmiannetaan tarpeeksi monta kertaa, pyydetään palvelinta poistamaan se ja siihen liitetyt tiedot.



Kuva 3. Suunniteltu näkymä sovelluksesta

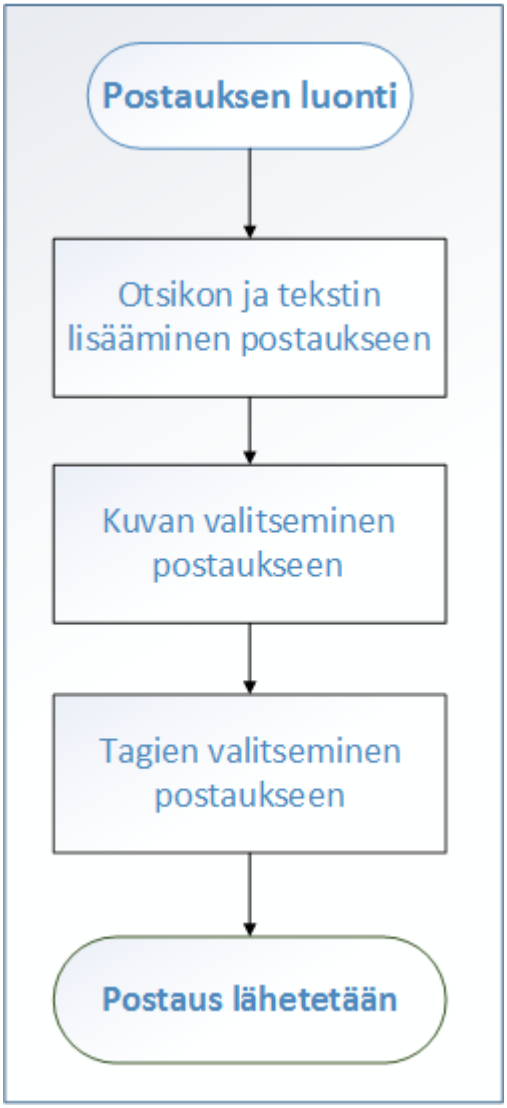
Kuvassa 3 on esitetty suunniteltu näkymä, kun sovellus avataan. Tässä näkymässä on auki *News*-välilehti. Näkymä sisältää postauksia ja uuden postauksen lisäysnapin. Postauksia saisi lisää liu'uttamalla ruutua alaspäin ja postauksen voi avata isommaksi painamalla sitä.



Kuva 4. Sovelluksen suunniteltu toimintakaavio

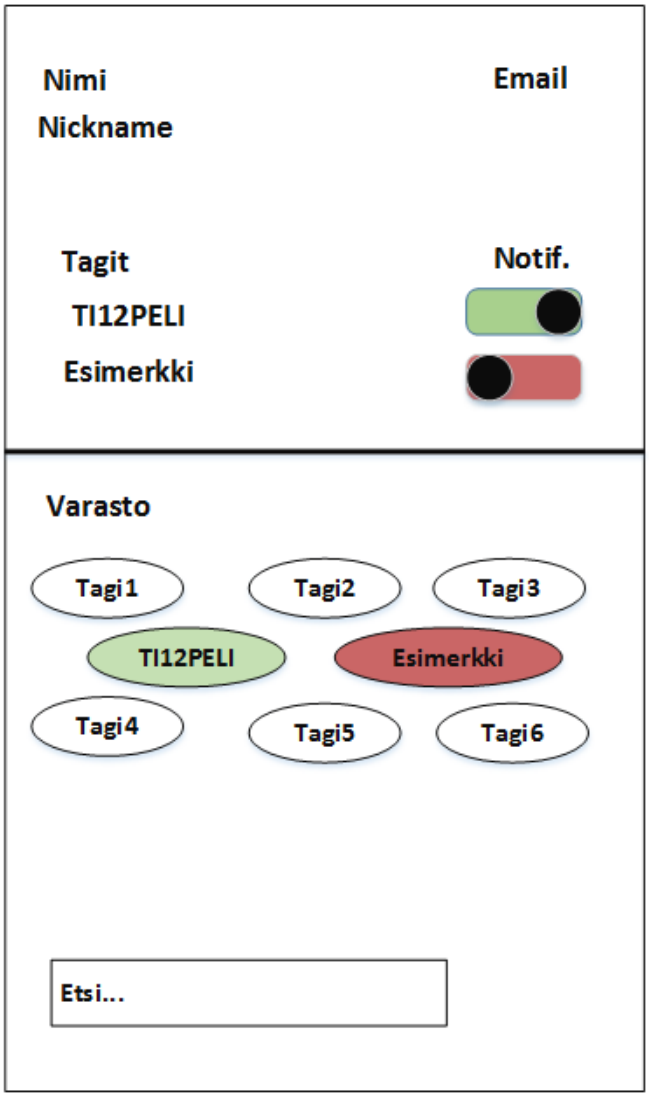
Kuvassa 4 nähdään sovelluksen käyttämät tiedot, jotka haluttiin tallettaa laboratorion Papaya-palvelimelle. Käyttäjä antaa syötteenä omat tietonsa, tilaamansa tagit ja mahdollisen postauksen. Nämä tiedot välitetään sovelluksesta palvelimelle, jonne ne tallennetaan tietokantaan. AJAX-kutsujen avulla saadaan tiedot haettua palvelimelta ja vietyä sinne.

Postauksiksi haluttiin muun muassa ajankohtaisia tapahtumia, mielenkiintoisia linkkejä tai opettajien ilmoituksia kursseista. Lisäksi näkyviin tulisivat postauksien lisäämisajankohta ja tagit. Postaukset listattaisiin neljään eri välilehteen: *Recent* – viimeisimmät, *Events* – tapahtumat, *News* – uutiset sekä *All* – kaikki. Näiden välillä liikutaan sovelluksen yläpalkin nappien avulla tai sovellusikkunaa sivuille liu’uttamalla.



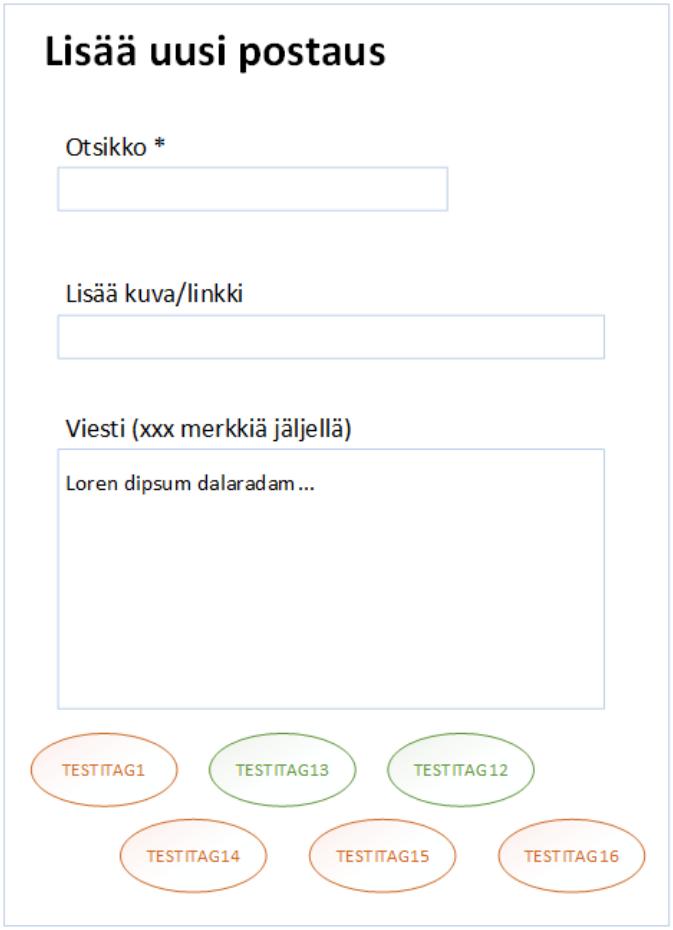
Kuva 5. Postauksen luomisen vuokaavio

Kuvasta 5 nähdään suunniteltu prosessi postauksien luomiseen ja lähettämiseen. Aluksi käyttäjä lisää haluamansa otsikon ja tekstin postaukseensa, mutta vain otsikko on pakollinen. Seuraavaksi käyttäjä voi valita haluamansa kuvan laitteensa kuvagalleriasta tai linkkaamalla kuvan internetistä. Ennen lähettämistä käyttäjä voi valita haluamansa tagit, jotta muut käyttäjät saavat notifkaation uudesta postauksesta. Lopuksi postaus lähetetään, jolloin se tallennetaan tietokantaan.



Kuva 6. Suunniteltu profiilisivu

Kuvassa 6 nähtävissä suunniteltu versio profiilisivusta, josta käyttäjä voisi muokata omia tietojaan ja tilata haluamiaan tageja. Kuvassa nähtävissä tilattu tagi TI12PELI ja Esimerkki, josta TI12PELI-tagi on tilattu. Tarvittaessa käyttäjä voisi hakea haluamiaan tageja, mikäli ne eivät näy tässä listassa.



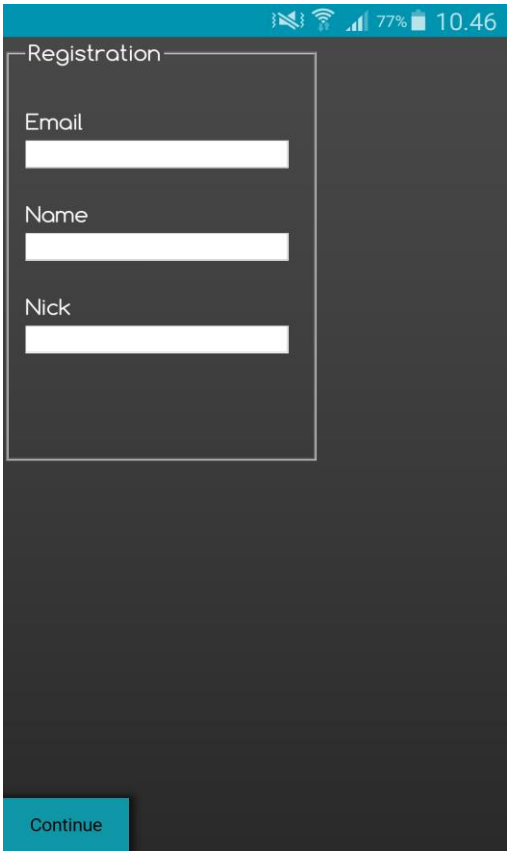
Kuva 7. Suunniteltu postaussivu näkymä

Kuvassa 7 nähdään suunniteltu näkymä postaussivusta, josta käyttäjä voi lisätä postauksia. Sivulla nähdään tekstilaatikot käyttäjän syötteelle. Otsikkoon on annettu tähtimerkki, joka tarkoittaa kyseisen tekstilaatikon olevan pakollinen. Lisäksi käyttäjä voi lisätä haluaman linkin ja viestin. Viestillä on määritelty maksimipituus. Kuten kuvassa 5 esitettiin, voi käyttäjä valita haluamansa tagit, jolloin muut käyttäjät saavat notifikaation uudesta postauksesta.

4 TOTEUTUS

Sovellus jaettiin useampaan osaan, jotta pystyttiin keskittymään yhteen osa-alueeseen kerrallaan. Näiden avulla voitiin myös seurata sovelluksen toimivuutta ja rakennetta. Sovelluksen toimivuutta pystyttiin testaamaan ja seuraamaan, kun rajattiin mahdolliset virhetilanteet tiettyyn osa-alueeseen eikä koko sovellukseen. Rakennetta pystyttiin hienosäätämään, kun nähtiin sovelluksen kokonaisuus, mistä se rakentui ja mitä osa-alueita tarvittiin.

4.1 Rekisteröintisivu



Kuva 8. Ensimmäisen käynnistyksen yhteydessä avautuva rekisteröityminen

Kuvassa 8 nähdään tapa saada käyttäjän olennaiset tiedot talteen sovelluksen ensimmäisen käynnistyksen jälkeen. Ensirekisteröitymissivulla varmistettiin, että jokainen sovelluksen avannut käyttäjä luo itselleen profiilin ja antaa siihen tarvittavat tiedot. Käyttäjää pyydetään antamaan sähköposti, oma nimi sekä nimimerkki postauksia varten.

```
<body onload="onLoad()"  id="bootyWform">
    <script>
                function onLoad() {
                    document.getElementById("LaunchCount").value = window.localStorage.getItem('launchCount');
                    document.getElementById("AndroidID").value = window.localStorage.getItem('AndroID');
                    document.getElementById("DeviceID").value = window.localStorage.getItem('device_id');

                    document.getElementById("aID").innerHTML = "Android id: " + window.localStorage.getItem('AndroID');
                    document.getElementById("dID").innerHTML = "Device id: " + window.localStorage.getItem('device_id');

                }
    </script>
    <form id="inputForm" name="insert" method="post" style="height: 200vh;"><!--action pois?-->
        <fieldset class="fieldset">
            <legend>Registration</legend>

            <br />
            Email<br id="emailbr" />
            <input type="text" name="Email" id="emailField" maxlength="255" style="width: 95%;"
                    onfocus="scrollTo('emailbr')" onblur="scrollUp()" required><br>

            <br />
            Name<br id="namebr" />
            <input type="text" name="Name" id="nameField" value="" style="width: 95%;"
                    onfocus="scrollTo('namebr')" onblur="scrollUp()" required><br>

            <br />
            Nick<br id="nickbr" />
            <input type="text" name="Nickname" id="nickField" value="" style="width: 95%;"
                    onfocus="scrollTo('nickbr')" onblur="scrollUp()"><br>

            <input type="hidden" name="LaunchCount" id="LaunchCount" value="default">
            <input type="hidden" name="AndroID" id="AndroidID" value="default">
            <input type="hidden" name="DeviceID" id="DeviceID" value="default">

            <p hidden id="aID">androidID_</p><br /> <!--debug-->
            <p hidden id="dID">deviceID</p><br /><!--debug-->

            <br />
            <input class="singleButton" id="saveButton" type="submit" value="">

            <img id="loadingIcon" src="images/ring-alt.gif" alt="uploading">

        </fieldset>
    </form>
```

Ohjelmalistaus 1. Rekisteröintisivun lomakkeen lähdekoodi

Ohjelmalistauksessa 1 nähtävissä olevassa lähdekoodissa on rekisteröitymissivun HTML body:n alku, jossa kutsutaan tätä seuraavassa script-osiossa luotua onLoad-funktiota. Tässä funktiossa haetaan laitteen välimuistista sovelluksen tallentamat arvot launchCount, AndroID ja device_id. Näiden arvot asetetaan niitä vastaaviin taulukon piilotettuihin syötteisiin LaunchCount, AndroID ja DeviceID.

LaunchCount on sovelluksen laskema kokonaisluku, kuinka monta kertaa käyttäjä on avannut sovelluksen. AndroID:tä käytetään push-notifikaatioiden

lähettämiseen Googlen Notifikaatio-palvelimelle ja sitä kautta takaisin muille sovelluksen käyttäjille heidän AndroID-tunnisteidensa avulla. AndroID saadaan Googlen Cloud Messaging -palvelimelta, kun push-notifikaatiot alustetaan sovelluksessa (Google Cloud Messaging 2016). DeviceID on laitteiden yksilöllinen tunniste, joka saadaan Cordovan device-lisäosan avulla device.uuid-ominaisuudella (GitHub 2016b). Tämä saatu merkkijono tallennetaan laitteen ja selaimen pysyvään localStorage-muistiin (P. Lehdonvirta & J. K. Korpela 2013, 143).

OnLoad-funktion jälkeen luodaan lomake, johon käyttäjä lisää haluamansa arvot. Lomakkeelle annettiin tunniste *inputForm*. Lomakkeen syötekentille annettiin niitä vastaavat nimet: Email, Name, Nickname, LaunchCount, AndroID ja DeviceID. Vaadittuja kohtia lomakkeessa olivat *email* ja *name*. Vaatiminen määriteltiin *required*-asetuksella syötekenttää luodessa. Email-kentässä asetettiin maksimipituudeksi 255 merkkiä. Lomakkeessa oleva tallennuspainike aloitti submit-toiminnon. Sivulla oli myös latauskuva *loadingIcon*, joka pyöri näytöllä sivua ladatessa.

```
    <script>
            // this is the id of the form
        $("#inputForm").submit(function (e) {

            document.getElementById("loadingIcon").style.display = "block";
            // the script where you handle the form input.
            var url = "http://papaya.ictlab.kyamk.fi/                    .php";

                $.ajax({
                    type: "POST",
                    url: url,
                    data: $("#inputForm").serialize(), // serializes the form's elements.
                    success: function(data)
                    {
                        //alert(data + "-"); // show response from the php script.

                        if (data.indexOf("success") > -1) {
                            window.localStorage.setItem('firstTimeSetupCompleted', true);
                            window.location = "index.html";

                        }
                        else {
                            alert("error while creating user: " + data);
                        }
                        document.getElementById("loadingIcon").style.display = "none";
                    }
                });

                e.preventDefault(); // avoid to execute the actual submit of the form.
            });

    </script>
</body>
```

Ohjelmalistaus 2. Loppuosa rekisteröitymissivun lähdekoodista

Ohjelmalistauksessa 2 nähdään lähdekoodin loppuosa eli rekisteröimisen tallentaminen. Lomakkeen tunnistimella #inputForm löydetään haluttu lomake ja sen submit-toiminto. Tälle toiminnolle annettiin funktio e, joka asetti latauskuvan näkymään. Funktio toimitti lomakkeen tiedot AJAX:n avulla palvelimella olevalle PHP-tiedostolle, joka tallensi saadut arvot palvelimen tietokantaan.

Lomakkeelta saadut tiedot sarjallistettiin jQueryn .serialize-metodilla merkkijonoksi, joka voitiin hakea PHP-tiedostossa. Tiedot lähetettiin POST-metodilla, joka on http-metodeista turvallisin vaihtoehto (W3schools 2016a). AJAX-kutsusta saatiin onnistuessa ulos data objekti, josta voitiin hakea merkkijono success. Jos dataobjektista löydettiin kyseinen merkkijono, tallennettiin laitteen välimuistiin totuusarvomuuttuja firstTimeSetupCompleted ja tälle arvo true.

Tämän jälkeen siirryttiin sovelluksen perusnäkymään index.html. Mikäli merkkijonohaku ei tuottanut tulosta, annetaan virheilmoitus ja latauskuva piilotetaan. Lomakkeen alkuperäisen lähetyspainikkeen toiminto estetään jQueryn preventDefault-metodilla.

4.2 Perusnäkymä

Kuva 9. Ohjelman perusnäkymä puhelimessa

Kuvassa 9 on nähtävissä All-välilehden ruudukkonäkymä esimerkki postauksien kanssa. Kuvasta nähdään myös kuusi kokonaista postausta. Nämä sisältävät otsikon, tekstikentän, postauksen tehneen käyttäjän nimimerkin ja postauksen lähettämisestä kuluneen ajan. Lisäksi kuvasta nähdään käyttöliittymän osia: postauksen lisäyspainike ja siirtymäpalkin painikkeet. Nämä painikkeet ovat vasemmalta katsottuna *Recent*, *Events*, *News*, *All* sekä profiilisivu. Uusi käyttäjä näkee rekisteröitymisen jälkeen Recent-välilehden, joka on oletuksena tyhjä. Käyttäjän tarvitsee tilata tageja nähdäkseen postauksia tällä välilehdellä.

```
var recentContainer = new newsContainer(window.innerWidth * 0, "#2a597d");
var eventsContainer = new newsContainer(window.innerWidth * 1, "#7e257e");
var _newsContainer = new newsContainer(window.innerWidth * 2, "#a5bc36");
var allContainer = new newsContainer(window.innerWidth * 3, "#c28738");

var containers = [recentContainer, eventsContainer, _newsContainer, allContainer];
```

Ohjelmalistaus 3. Välilehtien säilöjen luonti ja listaus

Ohjelmalistauksessa 3 luodaan sovelluksen käyttämät välilehdet ja ne listataan *containers*-taulukkoon. Välilehdet ovat *newsContainer*-funktion luomia objekteja, jotka tarvitsevat aloitussijainnin sekä värin. Väriä käytettiin alkuperäisissä kehitysversioissa esittämään eri välilehtien eri taustaväriä, mutta tästä luovuttiin. Sijainti saatiin, kun laskettiin ikkunan leveys ja annettiin tälle kerroin, jolloin välilehdet saatiin omiksi sivuiksi.

```
function onLoad() {
    //var masterdiv = document.getElementById("masterDiv");
    //masterdiv.style.height = "100%";// - 2;//border 1px*2

    storageOnLoad();
    closePopup();
    closeVotemenu();



    //canvas.addEventListener("deviceready", onLoad(), false);
    document.addEventListener("deviceready", onDeviceReady, false);
    fillNeebsWithStuff();
    upperButtonClick(currentPage);//
    draw();//first draw
    input.onLoad_();


    //_deviceready = true;
    cubeOnLoad();

    for (var i = 0; i < containers.length; i++) {

        document.getElementById("slaveDiv" + i).style.width = window.innerWidth;
        document.getElementById("slaveDiv" + i).style.height = window.innerHeight - 40;
    }
};
```

Ohjelmalistaus 4. Perusnäkymän käyttämä onLoad-funktion lähdekoodi

Ohjelmalistauksessa 4 nähdään perusnäkymän aloituksessa käytettävä onLoad-funktio. Tämä funktio mm. lataa välimuistin tiedot, sulkee ilmianto- ja poistovalikot, piirtää postausruudukon ja siirtyy välilehden *Recent* näkymään. Sivun siirtyminen sisältää päivitysfunktion, joka tyhjentää ruudukon, hakee siihen uudet postaukset ja piirtää postaukset ruudukkoon.

```
function refreshAll() {

    for (var i = 0; i < containers.length; i++) {
        containers[i].clear();
    }
    pullData(allContainer.news);

    input.onLoad_();
}
```

Ohjelmalistaus 5. refreshAll-funktio, joka päivittää newsContainer-objektit

RefreshAll-funktion lähdekoodi on nähtävissä ohjelmalistauksessa 5. Funktiota kutsutaan upperButtonClick-funktiossa, joka löytyy ohjelmalistauksessa 4 esitetystä onLoad-funktiosta. RefreshAll käy läpi kaikki containers-taulukon alkiot ja tyhjentää ne clear-funktiolla. Tämän jälkeen kutsutaan pullData-funktiota, joka käyttää parametrinään allContainer-objektin sisältämää news-taulukkoa.

```
function pullData(neebs) {

    var data = new Array();

    $.ajax({
        url: "http://papaya.ictlab.kyamk.fi/                .php",
        type: "GET",
        datatype: "json",
        crossDomain: true,
        success: function (html) {

            data = JSON.parse(html);
            //neebs = new Array();
            for (var i = 0; i < data.length; i++) {//title, text, url, timeStamp) {
                ////////////////////////array($row['PostsID'],$row['Title'],$row['Textbox'],$row['Picture'],$row['Timestamp'],$Nickname);
                neebs.push(new newsObject("" + data[i][0], "" + data[i][1], "" + data[i][2], "" + data[i][3], "" + data[i][4], "" + data[i][5]));
            }
            neebs.reverse();
            pullTags(neebs);
            updatePostDivs();
        },
        error: function (result) {

            console.log("error loading posts: " + result);

            containers[0].news.push(new newsObject("818888888", "no connection", "error loading posts: " + result, "images/no_img.png", "1961-08-04"));
            containers[1].news.push(new newsObject("828888888", "no connection", "error loading posts: " + result, "images/no_img.png", "1961-08-04"));
            containers[2].news.push(new newsObject("838888888", "no connection", "error loading posts: " + result, "images/no_img.png", "1961-08-04"));
            containers[3].news.push(new newsObject("848888888", "no connection", "error loading posts: " + result, "images/no_img.png", "1961-08-04"));
            updatePostDivs();

        },
        complete: function () {
            //alert("ajax completed");
        }
    });
};
```

Ohjelmalistaus 6. pullData-funktion lähdekoodi

Ohjelmalistauksessa 6 esitetty pullData-funktio ottaa itseensä *neebs*-parametrin. Tämä parametri on tyhjä tai jo olemassa oleva taulukko, kuten ohjelmalis-

tauksessa 5 mainittu *news*-taulukko. AJAX-kutsu haki palvelimelta kaikki postaukset. Kutsusta saatu JSON-tyyppinen merkkijono jäsennellään JavaScriptin käyttämäksi merkkijonoksi. Tämä jäsennelty merkkijono käydään läpi for-silmukassa, jossa saadut tiedot työnnetään uusiin newsObject-objektin parametrina saatuihin taulukoihin. NewsObject saa itseensä parametreinä postauksen tunnisteen, otsikon, tekstikentän sisällön, linkin kuvaan, aikaleiman ja postauksen tekijän nimimerkin.

Mikäli AJAX-kutsusta ei saada tulosta, luodaan kehityskonsoliin virheilmoitus. Sovellukseen luodaan postausruutujen tilalle tyhjät postaukset, jotka kertovat virheestä. Tämä johtuu heikosta internet-yhteydestä tai palvelimen alasajosta kehitysvaiheessa.

Taulukon alkioiden järjestys käännetään toisinpäin taulukon *reverse*-metodilla (W3schools 2016b), koska postaukset tulevat palvelimelta käänteisessä aikajärjestyksessä. Sovelluksessa haluttiin näyttää uusimmat postaukset ylimpänä.

AJAX-kutsusta saadut postaukset sisälsivät tageja, jolloin tarvittiin hakea kaikki tagit *pullTags*-funktiolla. Tämä funktio teki samankaltaisen AJAX-kutsun kuten *pullData*-funktio, mutta haki vain tagit. Näitä käytettiin heti saman funktion sisällä, kun kutsuttiin *pullPostHasTag*. Tällä haettiin postauksien sisältämät tagit ja asetettiin ne postauksiin. Kun postaukset ovat saaneet omat taginsa, haetaan käyttäjän tilaamat tagit laitteen tunnisteen avulla. Jokaisella käyttäjällä on yksilöllinen laitteentunniste. Tällöin tehdään AJAX-kutsu, jossa lähetetään laitteen tunniste palvelimelle, haetaan käyttäjän tiedot ja SQL:llä haetaan käyttäjän tilaamat tagit.

Kun postaus on saatu luotua, sen tagit asetettua ja käyttäjän tagit haettua, voidaan päivittää *containers*-taulukon osioiden sisältämät postausruudut. Taulukon alkiot käydään läpi for-silmukassa ja välilehtien sisältämät postaukset piirretään newsObjectin createDivs-funktiossa.

```
this.createDivs = function (divID, cubeWidth, cubeGap) {

    var targetDiv = document.getElementById(divID);
    var tdHeight = cubeGap;
    //clear target
    while (targetDiv.lastChild) {
        targetDiv.removeChild(targetDiv.lastChild);
    }

    for (var i = 0; i < this.news.length; i += 2) {
        this.news[i].setXY(Math.round(cubeGap), Math.round(cubeGap + i / 2 * (cubeWidth + cubeGap)));

        var post = document.createElement("div");
        post.setAttribute("class", "smallPost");
        post.setAttribute("style", "position: absolute; left: " + this.news[i].getX() + "px; top: " + this.news[i].getY()
                            + "px; width: " + cubeWidth + "px; height: " + cubeWidth + "px;");

        var url = this.news[i].getUrl();
        if (url == "" || url == null) {
        }
        else {
            post.style.background = "#747474 url('" + url + "') no-repeat center center";
        }

        var titteli = document.createElement("h1");
        titteli.appendChild(document.createTextNode(this.news[i].getTitle()));
        titteli.setAttribute("class", "postTitle");
        post.appendChild(titteli);

        var teksti = document.createElement("p");
        teksti.appendChild(document.createTextNode(this.news[i].getText()));
        teksti.setAttribute("class", "postText");
        post.appendChild(teksti);

        var timestamp = document.createElement("p");
        timestamp.appendChild(document.createTextNode(this.news[i].getSender() + this.news[i].getTimeSpamp()));
        timestamp.setAttribute("class", "postTime");
        post.appendChild(timestamp);

        onClickListeners(post, i, this.news[i].getTitle());
        targetDiv.appendChild(post);

        tdHeight += cubeGap + cubeWidth;
    }
```

Ohjelmalistaus 7. newsObjectin sisältämä createDivs-funktion lähdekoodi

Ohjelmalistauksessa 7 nähtävä newsObject pitää sisällään createDivs-funktion, jota kutsuttiin edellä mainitussa postausten haun yhteydessä. Tälle funktiolle annetaan parametreinä *divID* eli välilehden tunniste, johon kyseinen postaus piirretään. Lisäksi tarvittiin *cubeWidth* eli postauksen ruudun leveys sovellusikkunassa ja ruutujen väli *cubeGap*.

Välilehdelle haettiin *targetDiv*-muuttujalle getElementById-funktiolla, jolloin voitiin tarkistaa, onko välilehdellä olemassa olevia child-objekteja eli postauksia. Nämä jo olemassa olleet postaukset poistettiin, jotta välilehtiin ei tulisi samoja postauksia useita kertoja.

Tämän jälkeen aloitettiin for-silmukka, jossa käytiin läpi puolet *news*-taulukon alkioista. Tällöin voitiin piirtää välilehden vasemman puolen postaukset. Lopuksi asetettiin postauksille uudet sijainnit *setXY*-funktiolla, jolla laskettiin postaus ruudun leveyden ja välin avulla postausten sopivat sijainnit sovelluksen välilehden vasemmalta puolelta.

Post-muuttujassa aloitettiin uusien postausruutujen luonti, jonka jälkeen muuttujalla voitiin asettaa HTML-attribuutteja. Ruuduille annettiin luokaksi *small-post*, jota käytettiin ruudun ulkonäön muokkaamiseksi **CSS**-tiedostossa. Myös ruutujen sijainti asetettiin silmukan alussa laskettujen X- ja Y-arvojen perusteella.

Tämän jälkeen haettiin postauksesta saadut verkkosijainnit eli URL:t ja tarkistettiin, oliko saatu arvo tyhjä. Mikäli arvo ei ollut tyhjä, annettiin postaukselle kyseinen URL, jonka käyttäjä pystyi avaamaan sovelluksesta.

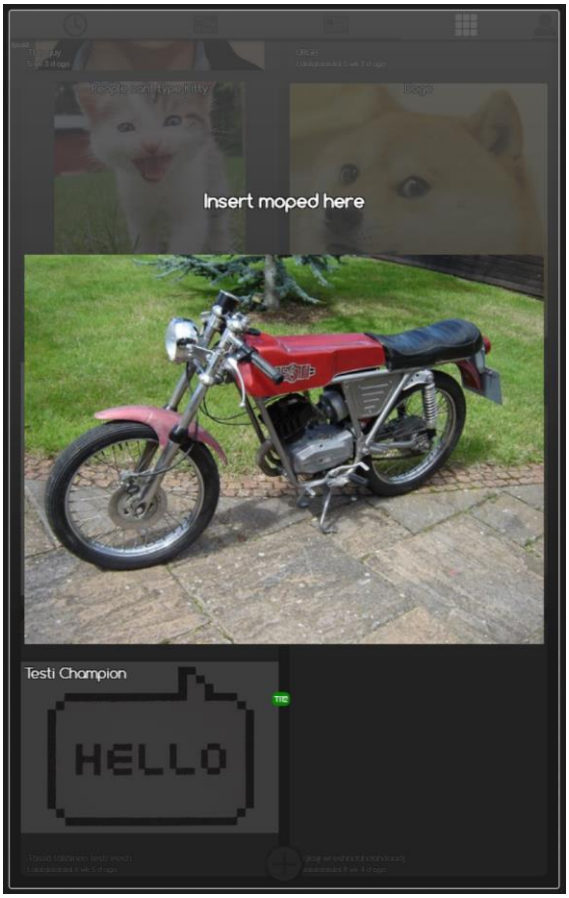
Postaukseen lisättiin myös postauksesta saadut otsikko, postauksen sisältämä teksti, postauksen lähettäjä ja postauksen lähettämisestä kulunut aika. Nämä luodut elementit lisättiin silmukan alussa luotuun post-muuttujaan eli postausruutuun, jonka jälkeen ruudut lisättiin välilehdelle.

Välilehden vasemmasta puolesta silmukan lopuksi laskettiin myös välilehden korkeus, lisäämällä postausruudun leveys ja väli jokaisella silmukan kierroksella. Tällöin saatiin alimman postauksen sijainti. Mikäli alimmalla rivillä olisi vain yksi postaus, tulisi se välilehden vasemmalle puolelle, jolloin tämän sijainti tarvittiin laskea.

Edellä laskettu välilehden korkeus *tdHeight* asetettiin välilehdelle style.height-funktiolla. Korkeuteen lisättiin myös 200 pikseliä, jotta alin postaus ei jäisi postauksen lisäysnapin taakse. Välilehden leveys saatiin sovelluksen ikkunan leveydellä, jolloin jokainen välilehti kattaisi yhdellä kertaa näkyvän osuuden sovelluksesta. Välilehden vasemman puolen piirtämisen jälkeen tehtiin sama silmukka oikealle puolelle, ilman välilehden korkeuden laskemista.

4.3 Käyttöliittymä

Sovelluksen käyttöliittymästä haluttiin yksinkertainen, jotta sitä olisi helppo käyttää. Postaukset haluttiin näkymään ruudukkonäkymään, että niitä saataisiin näkymiin mahdollisimman paljon kerralla. Toisaalta postaus ruutujen koko piti olla tarpeeksi iso, jotta niissä oleva teksti saatiin näkyviin selkeästi. Tällä tavoin saadaan näkyviin aina vähintään kaksi postausta, laitteesta riippumatta.



Kuva 10. Uutiskuutio näkymä tabletti-laitteessa

Kuvassa 10 avattiin postaus uutiskuutioon tabletti-laitteessa. Uutiskuutioon avattu postaus skaalautuu eri laitteilla niiden piirtotarkkuuden mukaan. Uutiskuutio on 4 sivuinen palikka, jossa uutiset näkyvät isompina kuin tavallisessa ruudukkonäkymässä. Palikkaa voidaan vierittää vasemmalle tai oikealle, kunnes päästään uusimpaan tai vanhimpaan postaukseen. Tagit rullaavat tekstin ja kuvan alapuolella oikealta vasemmalle.

Kuva 11. Ilmianto- ja poistovalikko uutiskuutiosta

Kuvassa 11 on nähtävissä valikko, joka saatiin esille painamalla uutiskuution postausta muutaman sekunnin ajan. Tämän jälkeen käyttäjä pystyi valitsemaan kahden painikkeen väliltä. Käyttäjä pystyi ilmiantamaan postauksen lippu-painikkeesta tai poistamaan koko postauksen roskakori-painikkeesta.

Postauksen poistaminen edellytti, että käyttäjä oli luonut kyseisen postauksen. Kuvassa näkyy myös emulaattorissa näkyvä alert-ikkuna, josta kehittäjät näkevät, mitä kyseinen nappi teki. Kuvassa 11 oli painettu roskakori-painiketta, jolloin postaus poistettiin ja kehittäjä sai näkyviin alert-ikkunan. Kyseinen ikkuna ei näy varsinaisessa sovelluksessa.

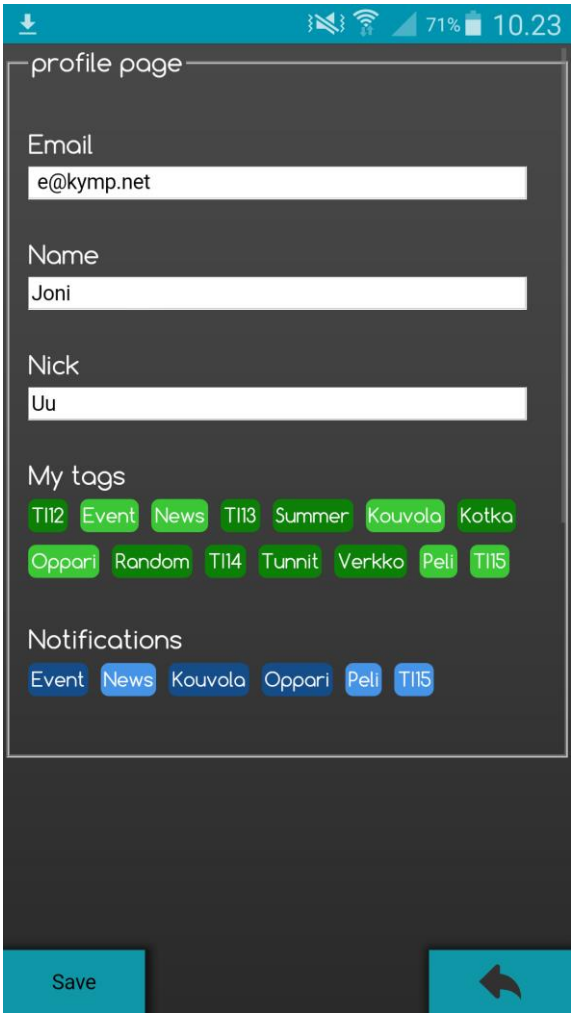
```
.singleButton{
    width: 25%;
    height: 40px;

    border: 0px none #000000;
    border-radius: 0;

    background-repeat: no-repeat !important;
    background-size: contain !important;
    background-position: center center !important;
    background-color: #0f97a7 !important;
    background-size: 35px 35px !important;

    box-shadow: 1px 1px 6px 3px #121212;

    position: fixed;
}
```

Ohjelmalistaus 8. Osa buttons.css-tiedostosta

Käyttöliittymän ulkonäköä muokattiin **CSS**-tiedostojen avulla. Ohjelmalistauksessa 8 on nähtävissä osalle painikkeista luotu *buttons.css*-tiedostosta. Tässä tiedostossa määritellään mm. painikkeiden koot, värit ja miten ne reagoivat käyttäjän toimintoihin. Leveys säädettiin skaalautuvaksi, joten sen arvoksi annettiin 25 % näytön leveydestä. Korkeudella annettiin arvoksi 40 pikseliä.

CSS-tiedostossa voitiin määritellä ulkonäkö haluttuun luokkaan, tässä tapauksessa luokka singleButton saa itseensä sen alla olevat attribuutit. Luokka määritellään pisteellä, tunniste (id) ristikkomerkillä (#) ja halutut elementit elementin nimellä esimerkiksi p tai div.

4.4 Profiilisivu



Kuva 12. Profiilisivu, jossa halutut tagit valittu

Sovellukseen luotiin kuvan 12 mukainen profiilisivu, josta käyttäjät voivat tilata haluamansa tagit näkyviin omaan perusnäkymään. Käyttäjä pystyi valitsemaan myös, mistä kaikista tageista hän halusi saada notifikaatiot. Kuvassa 12 nähdään, että käyttäjä on tilannut tagin *Peli.* Kun toinen käyttäjä lähettää sovellukseen *Peli*-tagin alle postauksen, saa ensimmäinen käyttäjä puhelimeensa push-notifikaation tästä postauksesta.

```php
<?php

define ( 'DB_HOST',
define ( 'DB_USER',
define ( 'DB_PASS',
define ( 'DB_NAME',
$con = new mysqli(DB_HOST,DB_USER,DB_PASS,DB_NAME);

/*include "connect.php";*/

  $query = "SELECT * FROM Tag";
  $results = $con->query($query);
  $return = array();
  if($results) {
    while($row = $results->fetch_assoc()) {
      // $return[] = array((string)$row['Title'],(string)$row['Textbox']);
    $return[] = array($row['TagID'],$row['TagName']);
    }
  }
   echo json_encode($return);
  $con->close();
 ?>
```

Ohjelmalistaus 9. getTagsApp.php -tiedoston lähdekoodi

Tagit haetaan profiili- ja postaussivulle ohjelmalistauksessa 9 näkyvän getTagsApp.php -tiedoston avulla. Tässä tiedostossa tehdään yhteys MariaDB-tietokantaan ja tehdään SQL-kutsuilla haku kaikista *Tag*-taulun arvoista. Haun tulokset asetetaan taulukkoon, joka lähetetään takaisin sovellukselle. Kuvasta on poistettu tietokannan tarkat kirjautumistiedot.

Profiilisivulla käyttäjä pystyy päivittämään ja muokkaamaan omia tietojaan. Kun käyttäjä painaa *Save*-painiketta, lomakkeen tunnisteella pystytään antamaan kyseiselle lomakkeelle tallennusfunktio. Tämä JavaScript-funktio käyttää AJAX-kutsua lähettääkseen tiedot lomakkeelta takaisin palvelimelle, josta ne haetaan toisella JavaScript-funktiolla näkymään takaisin sivulle.

4.5 Postaussivu

Kuva 13. Uusi postaus postaussivulla Android-laitteessa

Kuvassa 13 käyttäjä on asettanut otsikon, linkannut kuvan internetistä ja kirjoittanut aiheellisen viestin. Käyttäjä on myös valinnut muutaman tagin, jotka sopivat postaukseen. Nämä tagit tilanneet muut käyttäjät saavat notifikaation sovelluksesta. Postauksessa otsikko on pakollinen, koska lyhyempiin postauksiin ei tarvita kuvaa tai pidempää tekstiä. Tämä on tarkoitettu esimerkiksi tuntien perumisesta kertoviin postauksiin.

Kuvassa 13 on nähtävissä kuvan valitsemispainikkeet, joiden avulla käyttäjä voi hakea kuvan mobiililaitteensa galleriasta. Toinen painike on JPG- ja PNG-kuville ja toinen GIF-kuville, joiden enimmäiskoko on 2 megatavua.

```
function getImage() {

    // Retrieve image file location from specified source


    navigator.camera.getPicture(getMime, function (message) {
        alert('get picture failed: ' + message);
    }, {
        quality: 75,
        destinationType: navigator.camera.DestinationType.FILE_URI,
        sourceType: navigator.camera.PictureSourceType.PHOTOLIBRARY,
        targetWidth: 720,
        targetHeight: 720,
        correctOrientation: true
    }
    );

}
function getGif() {

    // Retrieve image file location from specified source


    navigator.camera.getPicture(getMime, function (message) {
        alert('get picture failed: ' + message);
    }, {
        quality: 50,
        destinationType: navigator.camera.DestinationType.FILE_URI,
        sourceType: navigator.camera.PictureSourceType.PHOTOLIBRARY
    }
    );

}
```

Ohjelmalistaus 10. Kuvien tallentamiseen tarvittavat getImage- ja getGif-metodit

Ohjelmalistauksessa 10 on esitetty kuvan hakeminen suoraan mobiililaitteen muistista Cordovan *camera*-lisäosan avulla. Lisäosalla voitiin ottaa kuva myös suoraan laitteen kamerasta, mutta tämä vaihdettiin muistista hakemiseen sourceTypen parametrillä **navigator.camera.PictureSourceType.PHOTOLIBRARY.** Kuvan hakeminen funktiolla *getPicture* otti vastaan tiettyjä asetuksia, kuten kuvan skaalaus *targetWidth*- ja *targetHeight*-asetuksilla. Asetuksilla määritettiin myös kuvan suuntaus *correctOrientation* avulla. (GitHub 2016c)

Molemmissa funktioissa kuva tallennettiin mobiililaitteen muistista Papaya-palvelimelle, joten kuvan sijainti oli haettava laitteesta destinationTypen parametrillä **navigator.camera.DestinationType.FILE_URI.** Tämä asetus antoi tiedoston sijainnin URI:n (W3C 2001), joka käännettiin hakemistomerkinnäksi.

```
function getMime(imageURI) {
    window.resolveLocalFileSystemURI(imageURI, function (fileEntry) {
        fileEntry.file(function (filee) {

            var mimetype = filee.type.toString().split("/").pop();
            //alert(mimetype); //THIS IS MIME TYPE
            uploadPhoto(imageURI, mimetype);
        }, function () {
            alert('error');
        });

    }, onError);
}
```

Ohjelmalistaus 11. getMime-metodi kuvan tiedostotyypin selvittämiseksi

Ohjelmalistauksessa 11 on nähtävissä getMime-metodi, jonka avulla saadusta URI:sta selvitetään kuvan MIME-tyyppi (Freed & Borenstein 1996). Saatu arvo muutettiin merkkijonoon, josta MIME-tyyppi eli tiedostopääte irrotettiin. Tämän jälkeen kutsuttiin uploadPhoto-funktiota, joka tarvitsi imageURI- ja mimetype-muuttujat.

```
function uploadPhoto(imageURI, mimetype) {



    var options = new FileUploadOptions();
    options.fileKey = "file";
    options.fileName = imageURI.substr(imageURI.lastIndexOf('/') + 1);
    options.mimeType = mimetype;

    var params = new Object();
    params.value1 = mimetype;
    //params.value2 = mimetype;

    options.params = params;
    options.chunkedMode = false;

    var victory = function (r) {

        //console.log("Successful upload...");
        //console.log("Code = " + r.responseCode);
        //di(fileEntry.fullPath +
        if (r.responseCode == 200) {
            //alert(r.response);
            var imgurl = "http://papaya.ictlab.kyamk.fi/        /      /" + r.response;
            document.getElementById("urlField").value = imgurl;
            changeImage();
        }
        else {
            alert("error unknown. Response code: " + r.responseCode);
        }
    }

    var defeat = function (error) {
        alert("An error has occurred: Code: " + error.code);
    }

    var ft = new FileTransfer();
    ft.upload(imageURI, "http://papaya.ictlab.kyamk.fi/                .php", victory, defeat, options);
}
```

Ohjelmalistaus 12. uploadPhoto-funktio, jolla käyttäjän kuvat tallennetaan palvelimelle

Ohjelmalistauksessa 12 nähdään uploadPhoto-funktio, joka käyttää edellä mainituista funktioista saatuja muuttujia imageURI ja mimetype. Näitä muuttujia käytetään File Transfer -lisäosan *FileUploadOptions*-muodostimessa. Tämän muodostimen parametrilla *options* asetettiin fileName eli tiedostonimi imageURI-muuttujasta. Samalla parametrilla asetettiin myös mimeType eli MIME-tyyppi *mimetype*-muuttujasta.

Lisäksi annettiin params-avainkoodille mimetypen arvo, joka lähetetään HTTP-kutsussa. Asetuksena annettiin chunkedModelle arvo false, jolloin palvelimelle tallennus ei tapahdu paloiteltuna suoratoistona. (GitHub 2016d.)

5 JATKOKEHITYS

Sovelluksesta saatiin valmiiksi kehitysversio, joka annettiin tietotekniikan laboratorion opiskelijoille testattavaksi. Testausvaiheessa ilmeni muutamia pieniä virheitä, jotka korjattiin. Mikäli sovellusta jatkokehitettäisiin, tapahtuisi tämä muiden opiskelijoiden toimesta.

Jatkokehitystä varten tulisi projekti luoda uudelleen ja käyttää ajan tasalla olevia lisäosia ja push-notifikaatio-palveluja. Tämän opinnäytetyön kirjoittamisen aikana muun muassa Googlen hallinnoima push-notifikaatio-palvelu on vaihtunut. Tämä aiheuttaa ongelmia myös kehitysversion push-notifikaatioiden toimivuuteen.

Visual Studion lisäosana käytetty Apache Cordova oli päivittynyt useamman kerran jo sovelluksen kehitysvaiheen aikana. Projektia ei riskeerattu päivittämällä jo toimivia osia. Jatkokehityksessä täytyisi myös harkita Xamarin-lisäosan käyttöä, sillä kyseinen lisäosa tuli maksuttomaksi ennen sovelluksen testausvaihetta.

Tietotekniikan laboratorion palvelimella olevat PHP-tiedostot sekä tietokanta ovat toimivia ratkaisuja. Jatkokehitykseen kannattaisi luoda nämä tiedostot uudelleen, jotta mitään yhteensopivuusongelmia ei esiinny.

Edellä mainitut seikat eivät sulje pois mahdollisuutta, että sovellusta kehitettäisiin jo olemassa olevan kehitysversion päälle. Tällöin jatkokehittäjät joutuisivat opiskelemaan jo olemassa olevat funktiot, johon tämä opinnäytetyö auttaa.

6 POHDINTA JA YHTEENVETO

Sovelluksen lähtökohtana oli tilanne, jossa yhteydenpidossa tietotekniikan laboratorion henkilökunnan ja opiskelijoiden välillä oli ongelmia. Tämän ongelman ratkaisu sovelluksen avulla ei vaikuttanut riittävältä, mutta toimeksiannon kehitys aloitettiin. Kehitysvaiheen jälkeen julkaistulla kehitysversiolla oli mahdollista korjata olemassa olevaa kommunikaatio-ongelmaa.

Testauksessa ollutta sovelluksen kehitysversiota ei käytetty kommunikaatio-ongelman ratkaisemiseen, sillä testausvaiheessa ei ollut oppitunteja. Tämän takia testausvaihe alkukesällä 2016 jäi lyhyeksi.

Toimeksiannon perusteella toteutettiin osa halutusta sovelluksesta, ainoastaan monialustatuki jäi vajaaksi. Sovelluksen kääntäminen ja sovittaminen iOS-alustalle osoittautuikin vaikeammaksi kuin kuviteltiin. Myös Windows-mobiililaitteille kehittäminen oli tarpeetonta. Tietotekniikan laboratoriosta ei testiryhmän kyselyn perusteella löytynyt ketään Windows-mobiililaitekäyttäjää.

Näiden osalta alkuperäinen idea monialustaisesta mobiilisovelluksesta oli oletettua isompi työ, johon ei tämän toimeksiannon puitteissa pystytty. Sovelluksesta saatiin toimiva Android-versio, jota päivitettiin testausvaiheessa ilmenneiden virheiden korjaamiseksi.

Sovelluksesta tehtiin myös infonäyttöversio, joka pystyttiin avaamaan tavallisella selaimella. Kyseisestä versiosta karsittiin suurin osa ominaisuuksista, vain postaukset laitettiin näkymään ja vierittymään automaattisesti ruudulla. Vieritysnopeutta ja postausten kokoa pystyttiin säätämään sivun asetusten avulla.

Kuusi kuukautta toimeksiannolle vaikutti aluksi sopivalta ajalta. Jo varhaisessa vaiheessa oli havaittavissa, että sovelluksen ominaisuuksia jouduttaisiin karsimaan. Seuraavan monialustaisen mobiilisovelluksen kehittämiseen tarvittaisiin enemmän tekijöitä ja selkeä suunnitelma.

LÄHTEET

Freed & Borenstein. 1996. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. Saatavilla: https://tools.ietf.org/html/rfc2045 [viitattu 21.10.2016].

GitHub. 2016a. GitHub - MariaDB/server: MariaDB server is a community developed fork of MySQL server. Started by core members of the original MySQL team, MariaDB actively works with outside developers to deliver the most featureful, stable, and sanely licensed open SQL server in the industry. Päivitetty 8.12.2016. Saatavilla: https://github.com/MariaDB/server/ [viitattu 25.5.2016].

GitHub. 2016b. GitHub - apache/cordova-plugin-device: Mirror of Apache Cordova Plugin Device. Päivitetty 8.12.2016. Saatavilla: https://github.com/apache/cordova-plugin-device [viitattu 1.11.2016].

GitHub. 2016c. GitHub - apache/cordova-plugin-camera: Mirror of Apache Cordova Plugin camera. Päivitetty 7.12.2016. Saatavilla: https://github.com/apache/cordova-plugin-camera [viitattu 25.10.2016].

GitHub. 2015d. GitHub - apache/cordova-plugin-file-transfer: Mirror of Apache Cordova Plugin file-transfer. Päivitetty 8.12.2016. Saatavilla: https://github.com/apache/cordova-plugin-file-transfer [viitattu 10.11.2016].

Google Cloud Messaging. 2016. Registering Client Apps | Cloud Messaging | Google Developers. Päivitetty 26.9.2016. Saatavilla: https://developers.google.com/cloud-messaging/registration [viitattu 14.11.2016].

Heinisuo, R. & Rauta, I. 2007. PHP ja MySQL – Tietokantapohjaiset verkkopalvelut. 4., uudistettu painos. Helsinki: Talentum Media Oy.

Kotimaisten kielten keskus & Kielikone Oy. 2016. Kielitoimiston sanakirja. Päivitetty 29.2.2016. Saatavilla: http://www.kielitoimistonsanakirja.fi/ [viitattu 8.6.2016].

Lehdonvirta, P. & Korpela, J. K. 2013. HTML5 sovellusalustana. Helsinki: RPS-yhtiöt.

Node.js. 2016. Saatavilla: https://nodejs.org/en/ [viitattu 8.6.2016].

Notepad++. 2016. Notepad++ - About. Saatavilla: https://notepad-plus-plus.org/ [viitattu 31.5.2016].

Visual Studio. 2016. Visual Studio Tools for Apache Cordova | Visual Studio. Saatavilla: https://www.visualstudio.com/en-US/explore/cordova-vs/ [viitattu 20.4.2016].

W3C & IETF Uri Planning Interest Group. 2001. URIs, URLs, and URNs: Clarifications and Recommendations 1.0. Päivitetty 21.9.2001. Saatavilla: https://www.w3.org/TR/uri-clarification/ [viitattu 21.10.2016].

W3schools.com. 2016a. HTTP Methods: GET vs. POST. Saatavilla: http://www.w3schools.com/tags/ref_httpmethods.asp [viitattu 14.11.2016].

W3schools.com. 2016b. W3schools.com: JavaScript Array reverse() Method Saatavilla: http://www.w3schools.com/jsref/jsref_reverse.asp [viitattu 16.11.2016].

WinSCP. 2014. Introducing WinSCP::WinSCP. Päivitetty 1.10.2014. Saatavilla: https://winscp.net/eng/docs/introduction/ [viitattu 30.5.2016].

Xamarin. 2013. Microsoft and Xamarin Partner Globally to Enable Microsoft Developers to Develop Native iOS and Android Apps With C# and Visual Studio. Päivitetty 13.11.2013. Saatavilla: https://www.xamarin.com/pr/microsoft-global-collaboration [viitattu 12.4.2016].

KUVALUETTELO