



**SAVONIA**

■ OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO  
TEKNIIKAN JA LIIKENTEEN ALA

# ALUSTASTA RIIPPUMATON MOBIILISOVELLUS

TEKIJÄ: Aki Pirinen

Koulutusala Tekniikan ja liikenteen ala			
Koulutusohjelma Tietotekniikan koulutusohjelma			
Työn tekijä(t) Aki Pirinen			
Työn nimi Alustasta riippumaton mobiilisovellus			
Päiväys	15.12.2015	Sivumäärä/Liitteet	25/0
Ohjaaja(t) lehtori Jussi Koistinen, lehtori Sami Lahti			
Toimeksiantaja/Yhteistyökumppani(t) PhysioBit Oy			
Tiivistelmä <p>Opinnäytetyön aiheena oli luoda mobiilisovellus, joka toimisi kolmen suurimman mobiililaittevalmistajan laitteella. Sovelluksella oli pystyttävä viestimään viestiominaisuudella ja tallentamaan kuvia tai videoita. Tietojen piti olla käytävissä tilaajayrityksen muilla sovelluksilla. Sovelluksen on tarkoitus helpottaa liikunnan alan ammattilaisen ja hänen asiakkaidensa yhteydenpitoa ja mahdollistaa harjoitusohjelman koostaminen asiakkaalle. Sovelluksen oli tarkoitus tukea toimeksiantajan jo aiemmin toteuttamaa työpöytäsovellusta.</p> <p>Työtä toteutettiin monella eri ohjelmointiympäristöllä. Työn tekeminen aloitettiin Qt-ohjelmointiympäristöllä, joka osoittautui myöhemmin puutteellisten ominaisuuksien ja valmistajan virheen vuoksi toteutuskelvottomaksi kyseiseen työhön. Tämän jälkeen sovelluksen toteutus aloitettiin Applen iOS-laitteisiin. Sovelluksen tietokanta toimii Microsoft Azure -pilvipalvelussa, ja sitä käytetään myös laitteessa näkyvien ilmoitusten lähettämiseen. Sovelluksen ulkoasu toteutetaan yhteistyössä toimeksiantajan muotoilijoiden kanssa.</p> <p>Lopputuloksen saatiin iOS-sovellus, jolla on mahdollista viestiä viestien avulla. Kameran käyttöä ei ehditty toteuttamaan työn aikana. Sovelluksen kehitystä on mahdollista jatkaa uusilla ominaisuuksilla, esimerkiksi kalenterilla.</p>			
Avainsanat			
Etäresurssipalvelu, Qt, iOS, Swift, C++			

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Aki Pirinen			
Title of Thesis Cross Platform Mobile App			
Date	15 December 2015	Pages/Appendices	25/0
Supervisor(s) Mr. Jussi Koistinen, Lecturer and Mr. Sami Lahti, Lecturer			
Client Organization /Partners PhysioBit Ltd.			
<p>Abstract</p> <p>The subject of this thesis was to create a cross platform mobile application that would work on three major mobile manufacturers' device. The purpose of the application was to help the coach communicate with customers and to be able to create training programs for them. The application had to be able to communicate with messages, handle images and/or videos and the data had to be accessible by the client organization's other applications.</p> <p>The project was started with the Qt environment, which later proved to be unable to meet for the specifications of the applications. This was because of missing features and manufacturer's programming error. After this finding, the application was started to be developed only for Apple iOS devices. The database of the application is located in the Microsoft Azure cloud service, which is also used to deliver push notifications to the device. The user interface was created together with the client organizations designers.</p> <p>The result of this thesis was an iOS application which could communicate with messages. The application was not able to handle images and videos. The application could be used as base for further development. For example calendar component could be included.</p>			
Keywords			
cloud computing, Qt, iOS, Swift, C++			

## ESIPUHE

Kiitos PhysioBit Oy:lle mielenkiintoisesta aiheesta ja mahdollisuudesta opiskella sovelluskehitystä Applen iOS-laitteille hyvässä työilmapiirissä. Kiitokset myös lehtori Jussi Koistiselle ohjauksesta.

Aki Pirinen

# SISÄLTÖ

TERMIT JA LYHENTEET .....	6
1 JOHDANTO .....	8
2 TYÖKALUT JA TEKNIIKAT .....	9
2.1 Qt.....	9
2.2 Qt Creator .....	10
2.3 Microsoft Azure.....	10
2.3.1 Azure Mobile Services.....	10
2.3.2 Azure Notification Hub.....	10
2.4 Apple push notification service.....	11
2.5 Visual Studio 2013 Ultimate .....	12
2.6 Android Studio.....	12
2.7 Xcode 6.4.....	12
2.8 Käyttöjärjestelmät.....	13
3 SOVELLUKSEN TOTEUTUS QT:LLA.....	14
3.1 Azure Mobile Services ja Qt .....	14
3.2 Qt ja Android.....	14
3.3 Qt, Windows Phone ja iOS.....	15
3.4 Projektin eri vaiheet .....	16
4 APPLE IOS-SOVELLUKSEN TOIMINTA.....	19
4.1 Framework ja Core data .....	19
4.2 Sovellukseen kirjautuminen .....	19
4.3 Viestit .....	20
4.4 Ilmoitukset .....	21
4.5 Kameran käyttö .....	22
5 YHTEENVETO.....	23
LÄHTEET .....	24

## TERMIT JA LYHENTEET

API	Application Programming Interface (ohjelmointirajapinta) on ohjelmoinnissa käytettävä rajapinta, jolla voidaan toteuttaa ohjelmoinnin rutiineja.
IDE	Integrated development Environment (IDE) on työkalu, jolla voidaan tuottaa ohjelmistokoodia.
Java	Java on Android-sovelluskehityksessä käytettävä oliopohjainen, korkean tason ohjelmointikieli. Java on vahvasti tyypitetty kieli, joka sisältää automaattisen muistin hallinnan. (Oracle 2015)
C++	C++ on standardoitu oliopohjainen ohjelmointikieli, joka käännetään suoraan konekielelle. Pohjana toimii C-ohjelmointikieli. (cplusplus.com 2000 – 2015)
Säie	Säikeiden avulla ohjelma voi suorittaa samanaikaisesti eri toimintoja. Tämä mahdollistaa ohjelmalle nopeamman suoritusajan. (Mac Developer Library 2014a)
Ilmoitus	Ilmoitus tulee englannin kielen sanasta push notification. Mobiililaitteessa näkyvä ilmoitus, jolla viestitään käyttäjälle. Nämä ilmoitukset näkyvät laitteen ilmoituskeskuksessa.
Kit	Kit on Qt Creatorin käyttämä alustakohtainen kääntämistyökalu. Kitissä määritetään mm. kääntämisen kohde (esim. Android-puhelin tai Windows-työpöytä), kääntäjä, Qt-versio, mahdollisesti käytössä oleva virheiden paikantaja ja sekä metatietoja. (Qt 2015b)
Objective-C	Objective-C on C-kieleen pohjautuva ohjelmointikieli, jota käytetään OS -x ja iOS-sovellusten kehityksessä. C-kielen ominaisuuksien lisäksi sisältää syntaksin luokkien ja metodien määrittelemiseen. (Mac Developer Library 2014b)
Swift	Swift on iOS-, OS x - ja WatchOS-sovellusten kehittämiseen tarkoitettu ohjelmointikieli. Swift toimii yhdessä Objective-C:n kanssa. (Developer 2015a)
SDK	SDK on lyhenne sanoista Software Development Kit. SDK on paketti, joka sisältää sovelluskehitykseen tarvittavia osia kuten esim. API-kutsuja.
Framework	Framework on Applen sovelluksissa käytetty hierarkkinen hakemisto, jonka avulla jaetut resurssit saadaan yhteen pakettiin. Paketti ladataan järjestelmän muistiin, minkä jälkeen se on usean sovelluksen käytettävissä samanaikaisesti. (Mac Developer Library 2013)
JavaScript	JavaScript on tulkittu, oliopohjainen komentosarjakieli, joka on yleisesti käytössä internetsivuilla parantamaan sivujen käyttäytymistä. JavaScriptia voidaan käyttää myös muissa ympäristöissä, kuten esimerkiksi Node.js. (Mozilla developer network 2005–2015)

JSON	JSON on lyhenne sanoista JavaScript Object Notation. Se on yksinkertainen, kevyt ja helposti luettava tiedon esitysmuoto. JSON muodostuu avain–arvo-pareista. (Introducing JSON)
Node.js	Node.js on Googlen V8 JavaScript Enginen päälle rakennettu, palvelimille tarkoitettu kevyt ja tapahtumiin perustuva ympäristö (Node.js 2015).

## 1 JOHDANTO

Opinnäytetyön tilaajayritys on PhysioBit Oy. Yritys haluaa toteuttaa heidän olemassa olevaa työpöytäsovellusta tukevan, alustasta riippumattoman mobiililaitesovelluksen. Opinnäytetyön tavoitteena on luoda sovellus, jolla liikunnan alan ammattilainen (esimerkiksi fysioterapeutti tai personal trainer) voi viestiä asiakkaansa kanssa ja luoda tälle harjoitusohjelman mobiililaitteen kameran avulla.

Mobiilisovelluksen on toimittava yhdessä tilaajayrityksen muiden sovellusten kanssa. Sovelluksen ulkoasu toteutetaan yhteistyössä tilaajayrityksen muotoilijoiden kanssa. Sovelluksen käyttämä tietokanta toimii Microsoft Azure -pilvipalvelussa. Markkinoilla on tällä hetkellä sovelluksia, jotka toimivat internetsivuilla, mutta mobiililaitteelle tehtyä natiivisovellusta ei ole tiedettävästi tehty.

Tässä raportissa käydään läpi sovelluksen toteutusta kahdella eri menetelmällä, projektin ongelmia ja toteutettuja ominaisuuksia.



## 2 TYÖKALUT JA TEKNIIKAT

### 2.1 Qt

Qt on norjalaisen Trolltechin vuonna 1990 kehittämä, tällä hetkellä Digia Oyj:n omistama ja ylläpitämä sovelluskehitysympäristö, jolla on mahdollista luoda sovelluksia monille alustoille. Qt pohjautuu C++-ohjelmointikieleen. Qt:ta käyttävät mm. Panasonic Avionics, BlackBerry ja Jolla. (QT 2015a)

Qt:ssa tapahtumia käsitellään signaalien ja slottien avulla (signals and slots). Signaali rekisteröidään slottiin ja kun signaali välitetään, se käsitellään kaikissa siihen rekisteröidyissä sloteissa. Yksi signaali voidaan rekisteröidä moneen slottiin. Kuvassa 1 QNetworkAccessManager-olion finished-signaali yhdistetään netResponse-metodiin, joka toimii slottina.

```
QString DBOperations::getCustomers(const QString &id)
{
    QNetworkRequest req;
    QNetworkAccessManager *nam = new QNetworkAccessManager(this);

    connect(nam, SIGNAL(finished(QNetworkReply*)), this, SLOT (netResponse(QNetworkReply*)));
}
```

KUVA 1. Signaalin yhdistäminen slottiin

Qt sisältää myös oman QML-kielen, joka vastaa syntaksiltaan JavaScriptia. QML:n avulla sovelluksen käyttöliittymään voidaan lisätä dynaamisuutta. QML on julistava kieli, jolla olioille voidaan määrittää attribuutteja ja kuinka se reagoi toisiin olioihin. Olioiden toimintaa voidaan parantaa JavaScriptin avulla. QML-koodi ladataan sovellukseen pääsääntöisesti QML-tiedostojen avulla. QML:n ja C++:n yhdistäminen on mahdollista. C++-luokkaa voidaan kutsua QML:sta, jos se perii QObject-luokan. Kuvassa 2 WebView-olio kutsuu C++ Login -luokan loginVoid-metodia. Kuva 3 havainnollistaa, kuinka C++-metodi voidaan julkistaa QML:sta kutsuttavaksi etuliitteellä Q\_INVOKABLE.

```
Login {
    id: login
    onLoginOK: {
        console.log("hit")
        webLogin.visible = false
    }
}

WebView {
    anchors.fill: parent
    id: webLogin
    visible: true
    onLoadingChanged: login.loginVoid(url)
}
```

KUVA 2. WebView'n onLoadingChanged-tapahtumassa kutsutaan Login-komponentin loginVoid-metodia

```
public:
    Login(QObject *parent = 0);
    Q_INVOKABLE void loginVoid(QUrl url);
```

KUVA 3. Q\_INVOKABLEn käyttö

Ohjelmointikielen valinnassa pääsääntönä voidaan sanoa, että mikäli tarkoituksena on työpöytäsovellus, on suositeltavaa pidättäytyä Qt- ja C++-kielessä. Sen sijaan QML on suunniteltu mobiilisovelluksiin sen visuaalisten ominaisuuksien takia.

## 2.2 Qt Creator

Qt Creator on Qt:n kehittämä IDE. Ohjelmasta on saatavilla ilmainen versio sekä maksullisia versioita. Maksulliset versiot sisältävät enemmän ominaisuuksia kuin ilmainen versio. Teknistä tukea on mahdollista saada vain maksullisella lisenssillä.

## 2.3 Microsoft Azure

Azure on Microsoftin tarjoama pilvipalveluympäristö. Ympäristö tarjoaa mm. mahdollisuuden tiedon tallennukseen, virtuaalikoneiden ja verkkojen ylläpitämiseen, internetsivujen ylläpitämiseen sekä mobiililaitteiden hallintaan ja analytiikkaan. Azuren hinnoittelu perustuu ns. pay-as-you-go-periaatteeseen. Hinta muodostuu komponenttien käytön mukaan. Azurea käyttävät mm. Skanska, Mazda ja NBC Sports (Microsoft 2015a)

### 2.3.1 Azure Mobile Services

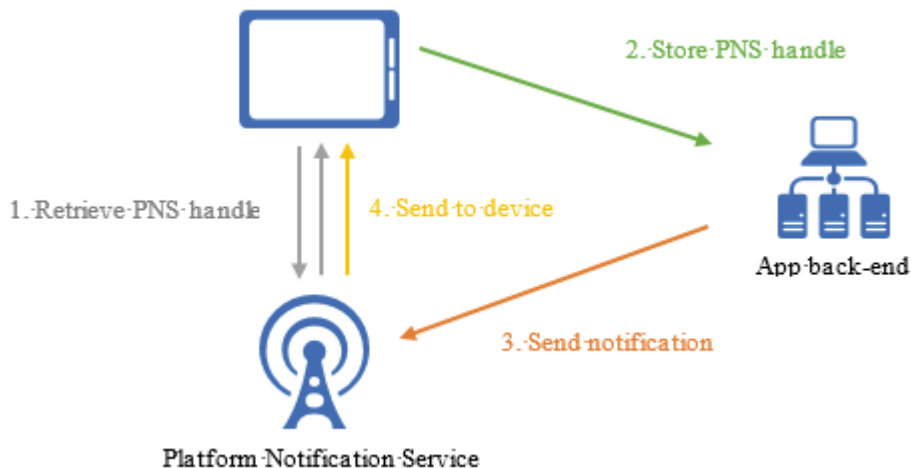
Microsoft Azure Mobile Services on Azuren palvelu, joka mahdollistaa kommunikoinnin mobiililaitteiden välillä, tiedon tallennuksen pilveen, automatisoinnin ja käyttäjien todentamisen (Microsoft 2015b).

Mobile Services toimii sovelluksen ja tietokannan välillä. Yhteen Mobile Servicesiin on mahdollista kytkeä vain yksi tietokanta, jota se käyttää tiedon hakemiseen ja tallentamiseen. Mobile Servicesin käyttö on mahdollista Microsoftin tarjoaman SDK:n tai http-kutsun avulla (esimerkki kuvassa 6). Mobile Servicesin elementtien käytettävyyttä on mahdollista rajata, esim. tietyn taulun käyttöoikeus voidaan sallia vain kirjautuneille käyttäjille.

### 2.3.2 Azure Notification Hub

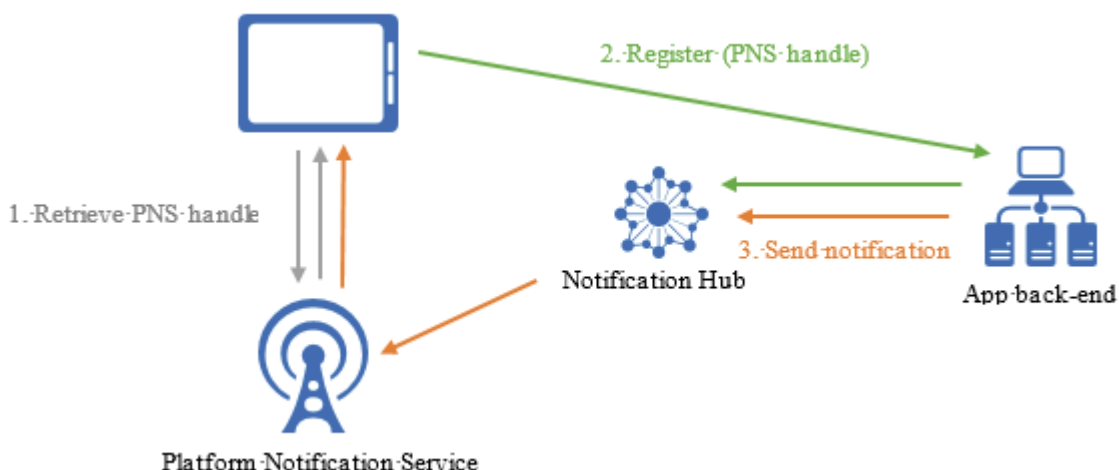
Azure Notification Hub on osa Microsoftin tarjoamaa Azure-pilvipalvelua. Notification Hub mahdollistaa ilmoitusten välityksen Windows Store -, iOS-, Android- ja Windows Phone -laitteille. Viestien lähetys onnistuu edellä mainittujen laitteiden lisäksi esimerkiksi .NET-, PHP- tai Node.js-koodilla. Ilmoitus välitetään jokaiselle alustalle sen oman Platform Notification Systemsin (PNS) avulla. Esi-merkkinä ilmoituksen lähetys Windows Store -sovellukselle. Tätä varten on ensin otettava yhteys Windows Notification Serviceen (WNS). Ilman Azure Notification Hubia ilmoituksen kulku toteutetaan

neljällä vaiheella. Käyttäjän sovellus noutaa PNS-kahvan sen omalta viestipalvelulta. Seuraavaksi kahva varastoidaan esimerkiksi tietokantaan. Viestin lähetyksessä noudetaan tallennettu kahva ja otetaan yhteys PNS-palveluun. Lopuksi PNS-palvelu välittää viestin halutulle laitteelle. Prosessi on havainnollistettu kuvassa 4.



KUVA 4. Normaali laitteeseen viestimisen kulku (Microsoft Azure 2015)

Ilmoituksen välitys Notification Hubin avulla lisää prosessiin vain yhden elementin. Jokaisen alustan oma PNS-kahva rekisteröidään Azure Notification Hubille. PNS-kahvan lisäksi on mahdollista rekisteröidä myös yksittäisiä tageja, jotka voivat olla esimerkiksi henkilön kiinnostuksen kohteita tai henkilön asuinmaa. Kun halutaan lähettää viesti yksittäiselle laitteelle tai tagilla rekisteröidyille käyttäjille, Notification Hub etsii laitteen PNS-kahvan ja välittää viestin kyseiselle PNS-palveluntarjoajalle, joka puolestaan välittää viestin oikealle laitteelle. Prosessi on havainnollistettu kuvassa 5.



KUVA 5. Laitteeseen viestiminen Notification Hubin kautta (Microsoft Azure 2015)

## 2.4 Apple push notification service

Jotta Azure Notification Hub toimisi Applen laitteissa, on jokainen sovellus rekisteröitävä erikseen APNS-palvelimelle. Tätä varten on oltava Apple Developer -tili ja profiili, joka on kytketty Xcodeen.

Tilille rekisteröidään kehitettävä tuote, jolle on mahdollista lisätä erinäisiä ominaisuuksia, mm. ilmoitukset. Ilmoituksia varten on Applelta pyydettävä sertifikaatti, joka mahdollistaa apns:n käytön. Sertifikaatin tyyppi riippuu sovelluksen toteutusympäristöstä. Sertifikaatti on pyydettävä erikseen kehitys- ja tuotantoympäristöön. Azure Notification Hubia varten sertifikaatti on muutettava .p12-tiedostomuotoon, joka ladataan Hubin asetuksiin. Tämän jälkeen Hubi osaa ohjata liikenteen oikealle APNS-palvelimelle.

Koska Azure Notification Hub sallii vain yhden sertifikaatin asettamisen, on Microsoft suositellut kahden Hubin käyttämistä, toinen testausta ja kehitystä varten ja toinen tuotantoa varten. Notification Hubin toiminta voidaan testata myös tuotantosertifikaatilla, kunhan Xcodessa valitaan sovelluksen profiilityyppi Ad-Hoc ja testauslaitteen yksilöllinen tunnus on rekisteröity Applen Developer -tilille. Tämän jälkeen sovellus on asennettava testauspuhelimeen iTunes-sovelluksen kautta.

Edellä mainituista syistä mahdolliset ilmoitusten ongelmatilanteet ovat hankalia ratkaista. On mahdollista havaita laitteen rekisteröityminen Azure Notification Hubilla, mutta kun laitteelle yritetään lähettää viestiä, saadaan virheilmoitus, että laitetta ei ole rekisteröity palvelimelle. Tässä tapauksessa virhe voi johtua väärästä profiilista. Mikäli ohjelma on käynnistetty laitteelle Xcodesta, tämä tulkitaan kehitysympäristöksi. Tässä tilanteessa laite rekisteröityy APNS kehityspalvelimelle, vaikka Azure Notification Hubille olisi asetettu tuotantosertifikaatti. Apple kertoo dokumentaatioissaan tekevänsä parhaansa ilmoituksia välittäessä, he eivät kuitenkaan takaa ilmoituksen onnistunutta välitystä (iOS Developer Library 2015).

## 2.5 Visual Studio 2013 Ultimate

Visual Studio on Microsoftin kehittämä IDE, joka mahdollistaa sovellusten kehityksen lukuisille Microsoftin alustoille mm. työpöytä, mobiili ja selain. 2013 Ultimate sisältää uuden käyttöliittymän XAML sovellusten testaukseen ja analysointiin. 2013 Ultimate on integroitavissa Azure Mobile Servicesin kanssa. (Visual Studio 2015)

## 2.6 Android Studio

Android Studio on Googlen kehittämä, IntelliJ IDEA:aan pohjautuva IDE Android-sovellusten kehittämiseen. Android Studiolla voidaan kehittää sovelluksia Android puhelimille, -TV:lle ja -Wear laitteille. Android Studion avulla sovellukseen voidaan liittää mainoksia, analytiikkaa, käyttäjän todentamista ja ilmoituksia Googlen palveluiden avulla. (Developers 2015)

## 2.7 Xcode 6.4

Applen julkaisema IDE Mac-, iPhone- ja iPad-sovellusten luontiin. Xcodesta löytyy kääntäjä Objective-C:lle ja Swiftille. Xcode sisältää hyvin toimivan emulaattorin sekä lukuisia työkaluja testausta varten, kuten esimerkiksi Zombie Detection, jonka avulla voidaan havaita mahdollisia muistin käytössä tapahtuvia virheitä. (Developer 2015b)

## 2.8 Käyttöjärjestelmät

Microsoftin valmistamista käyttöjärjestelmistä käytössä oli Windows 7 Home Edition ja 8.1 Enterprise. Mac laitteella käytetty käyttöjärjestelmä oli OS X Yosemite 10.10.4. Linux käyttöjärjestelmissä käytössä oli Ubuntu 14.04.

### 3 SOVELLUKSEN TOTEUTUS QT:LLA

Tässä osiossa kuvataan projektin toteutusta Qt-ympäristössä, siinä ilmenneitä ongelmia sekä aikataulusta.

#### 3.1 Azure Mobile Services ja Qt

Qt:ta tai C++:aa varten ei ole tehty kirjastoa, jolla Mobile Servicesin käyttö olisi mahdollista. Käyttäjän todentamiseen voidaan kuitenkin käyttää Mobile Services APIa. Kuvassa 2 olevalle WebView-komponentille annetaan alkuosoitteeksi `https://”Mobile Servicesin nimi”.azure-mobile.net/login/google`. Näin käyttäjälle esitetään Googlen kirjautumissivu. Kun käyttäjä on syöttänyt oikean Google-tilin sähköpostin ja salasanan, WebViewin muuttuneesta url osoitteesta on mahdollista lukea käyttäjän Google-tilin id ja token. Tietokantojen tehtävät on suoritettava http-kutsun avulla. Kuvassa 6 luodaan QNetworkRequest-olio, jonka url koostuu tietokannan osoitteesta, haettava taulusta ja suodatusehdosta. Olion header-osaan lisätään aiemmin kirjautumisen yhteydessä saatu token, koska tietokannan taulujen käyttö on sallittu vain kirjautuneille käyttäjille. Kyselystä palautuva JSON-objekti puretaan kyseisen luokan netResponse-metodissa.

```
QNetworkRequest req;
QNetworkAccessManager *nam = new QNetworkAccessManager(this);

connect(nam, SIGNAL(finished(QNetworkReply*)), this, SLOT(netResponse(QNetworkReply*)));

req.setHeader(QNetworkRequest::ContentLengthHeader, QVariant(0));
req.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");
req.setRawHeader(QByteArray("X-ZUMO-AUTH"), DBOperations::token.toUtf8());

QString urlString = _msurl + "/tables/customers";
QUrl url(urlString);
QUrlQuery query;

query.addQueryItem("$filter", "(trainer eq 'XXXX012')");

url.setQuery(query);
req.setUrl(url);

nam->get(req);

return "tulee c++ kanssa";
```

KUVA 6. http-kutsun tekeminen Mobile Services customers -tauluun

#### 3.2 Qt ja Android

Ennen kuin Qt-sovelluksia voidaan suorittaa Android-laitteilla, on asennettava Java DE Development Kit, Apache Ant, Android SDK ja Android NDK. Qt Creator luo ennen sovelluksen suoritusta Androidin tarvitsemat tiedostot (mm. AndroidManifest.xml ja kaksi Java-luokkaa). Käynnistymisen yhteydessä Android ajaa muokatun activityn. Ensimmäinen Java-osuus etsii puuttuvat kirjastot (sekä Qt:sta että Javasta) ja lataa ne. Se myös välittää kaikki tapahtumat toiselle Java-osuudelle. Toinen Java-osuus kommunikoi Qt:n kanssa. Se sisältää kaiken logiikan, jotta Qt voi toimia laitteessa, kuten esim. virtuaalisen näppäimistön ja näyttöön piirtämisen.

Qt mahdollistaa Java-koodin käyttämisen sovelluksessa. Java-tiedostoihin tulee viitata Qt-projektissa. Java-koodin suoritus tapahtuu Qt:n Androidextras-kirjaston avulla. Kuvassa 7 C++-koodista suoritetaan kutsu Java-koodin QtEntering-metodiin, joka tarvitsee parametrikseen kaksi string arvoa. Tässä vaiheessa on huomioitava, että esimerkiksi Qt:n käyttämä QString-teksti ei käy suoraan Javan String-muuttujaan vaan on muutettava ennen lähetetystä AndroidJNI-objektiksi, josta tämän jälkeen otetaan Javaan sopiva jstring. Kun Java-tiedosto on saanut kutsun, se aloittaa koodin suorituksen uudessa säikeessä.

```
QtAndroid::androidActivity().callMethod<void>("QtEntering",
    "(Ljava/lang/String;Ljava/lang/String;)V",
    paramtoken.object<jstring>(),
    paramid.object<jstring>());
```

KUVA 7. Java metodin kutsuminen C++ koodilla

Ilmoitusominaisuuden lisääminen Android-sovellukseen toteutettiin käyttäjän kirjautumisen jälkeen. Jotta käyttäjän laite saadaan rekisteröityä Notification Hubille, täytyy käyttäjän Google-kirjautumisesta saatu id ja token välittää Java-koodiin Mobile Servicesille. Azure Notification Hubille rekisteröitymisen jälkeen on ilmoitusten vastaanottamista varten asetettava oma handler-luokka. Kun mobiililaitte saa viestin Notification Hubilta, viesti muutetaan Java-koodissa ilmoitukseksi, joka tämän jälkeen annetaan Androidin Notificationmanagerin esitettäväksi.

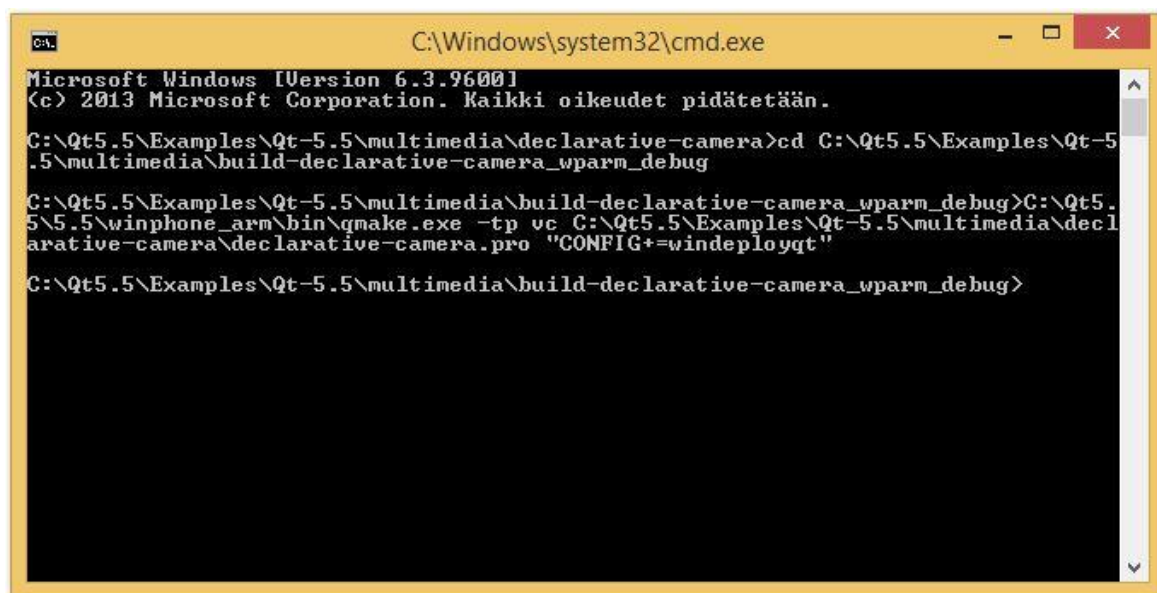
Ilmoitusten lisääminen vaatii AndroidManifestin muokkaamista. Normaalisti Qt:n kääntäjä luo manifestin automaattisesti. Qt Creatorilla on mahdollista vaihtaa esimerkiksi käytettävää SDK:ta tai muuttaa Qt kirjastojen asennustapaa laitteelle. Suurempi muokkaus on mahdollista luomalla manifestista malli (engl. template). Malli on täysin muokattavissa. Siihen voidaan lisätä mm. sovelluksen vaatimat luvat puhelimen käytöstä, tässä tapauksessa lupa vastaanottaa ilmoituksia.

Kameran toimintaa ohjaava Qt:n QCamera-luokka toimi vaihtelevasti eri Android-laitteilla. Testaus tehtiin kahdella kameraa käyttävällä ohjelmalla. Näistä ohjelmista toinen oli Qt:n esimerkeistä löytyvä declarative-camera. Testauksen laitteina oli kolmella eri puhelinta ja Samsung Galaxy tab 2 -tabletti. Näistä videotallennus onnistui ainoastaan Samsung-tabletilla, toisinaan myös tällä laitteella testausohjelmat kaatuivat vakavaan virheeseen. Virheilmoitus ei ollut yksiselitteinen, eräs tulkinta virheilmoituksesta oli, että videokuvalle ei olisi määritetty paikkaa, missä tallennettava video esitetäisiin.

### 3.3 Qt, Windows Phone ja iOS

Vaikka Qt ja Windows RT (Windows Runtime) on mahdollista ohjelmoida samalla ohjelmointikielellä (C++), ei Qt-koodin suorittaminen Windows RT -laitteessa ole mahdollista suoraan Qt Creatorista. Valmiista sovelluksesta on ensin luotava Visual Studio -projekti, joka on tämän jälkeen mahdollista ajaa lopulliselle laitteelle. Projektin luonti onnistuu suorittamalla Windowsin komentorivillä komento, joka suorittaa Windows Phone arm kitissä sijaitsevan qmake-tiedoston. Qmake-tiedostolle täytyy antaa myös viittaus projektin .pro-tiedostoon ja suorittaa qmake yhdessä windeployment-työkalun

kanssa. Kuvassa 8 esitellään suoritettut komentorivin komennot, onnistunut suoritus ei tulosta viestiä komentoriville.



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Kaikki oikeudet pidätetään.

C:\Qt5.5\Examples\Qt-5.5\multimedia\declarative-camera>cd C:\Qt5.5\Examples\Qt-5.5\multimedia\build-declarative-camera_wparm_debug

C:\Qt5.5\Examples\Qt-5.5\multimedia\build-declarative-camera_wparm_debug>C:\Qt5.5\5.5\winphone_arm\bin\qmake.exe -tp vc C:\Qt5.5\Examples\Qt-5.5\multimedia\declarative-camera\declarative-camera.pro "CONFIG+=windeployqt"

C:\Qt5.5\Examples\Qt-5.5\multimedia\build-declarative-camera_wparm_debug>

```

KUVA 8. Qmake-tiedoston suoritus declarative-camera esimerkkiprojektista.

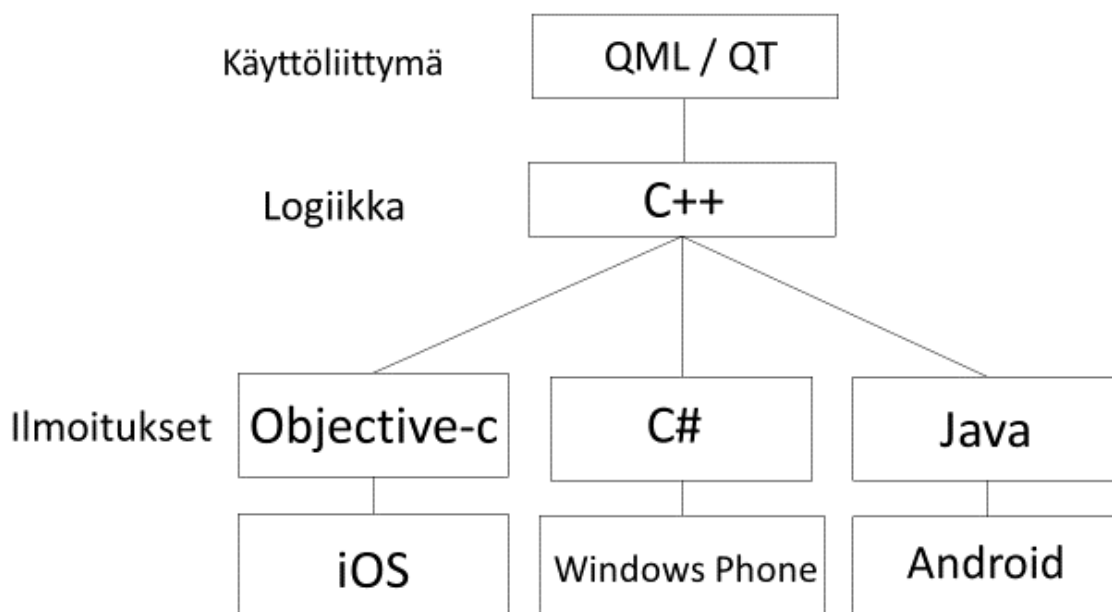
Qt:n internetsivujen esittämistä varten tehty QWebView-luokka ei toimi Windows Phone -laitteilla. Qt:n Windows Phone kit ei sisällä QWebView-luokkaa, tästä syystä käännetty Visual Studio -projekti ei tunnista käytössä olevaa luokkaa. Ongelmaa yritettiin ratkaista lisäämällä QWebView-kirjasto toisesta kitistä. Kirjasto kopioitui projektin mukana asennushakemistoon, mutta tämä ei ratkaissut ongelmaa. Ongelmaan kysyttiin ratkaisua Qt:n foorumilla, mutta tähän viestiin ei koskaan vastattu.

Apple iOS laitteiden testaus täytyy tehdä Mac tietokoneelta, johon on asennettu Xcode. Sovelluksen kääntäminen Qt Creatorissa suorittaa qmake-tiedoston, joka kääntää tiedostot Xcode-projektiksi. Tämän jälkeen Xcode-projekti on mahdollista suorittaa laitteessa.

### 3.4 Projektin eri vaiheet

Projekti aloitettiin tutkimalla, millä tekniikalla sovelluksen ominaisuudet olisi mahdollista toteuttaa. Qt:n lisäksi vaihtoehtoina olivat Xamarin ja Phonegap (Apache Cordova). Xamarin jäi pois lähtökohdaisesta hintansa takia, sekä parhaimmillaankin ainoa uudelleen käytettävä osuus olisi ollut käyttöliittymä. Muut toiminnot olisi täytynyt toteuttaa laitteen omalla ohjelmointikielellä. Phonegap ei soveltunut html-ohjelmointikielestä johtuen. Tilaajayrityksessä ei ollut html-, css- ja JavaScript-ohjelmoinnista tarvittavaa kokemusta, jotta sovelluksen ylläpito olisi ollut mahdollista projektin jälkeen. Qt valittiin toteutusympäristöksi kolmesta syystä. Tilaajayrityksessä oli osaamista Qt:sta, kaupallisen lisenssin hinta oli houkutteleva ja C++-kehityskielenä oli itselleni mieleinen valinta. Projektin alussa suurimpia haasteita arveltiin olevan ilmoitusten esittäminen ja laitteen kameran käyttö. Ilmoituksia varten Qt:ssa ei ole omaa kirjastoja, joten jokaiselle alustalle oli tehtävä tätä varten oma osuus sen omalla ohjelmointikielellä kuvan 9 mukaisesti.





KUVA 9. Sovelluksen suunniteltu rakenne Qt:lla toteutettuna

Qt:hen tutustumisen jälkeen aloitettiin ilmoitusten toteutus Android-alustalle. Android valikoitui ensimmäiseksi alustaksi, koska tutkimuksissa havaittiin Qt:n ja ilmoitusten yhdistämisen olevan mahdollista. Samalla oppisi paremmin käyttämään Qt:ta. Java-osuus luotiin Android Studiota hyväksi käyttäen, näin käytössä oli koodin tarkasteluun ohjelman tarjoama intellisense.

Seuraavaksi testausalustaksi valittiin Windows-mobiililaitteet (Windows Phone). Tutkimuksissa ei löytänyt yhtään tapausta, jossa Qt ja ilmoitukset olisi yhdistetty tälle alustalle. Toteutuksen arveltiin olevan mahdollista kahdella eri tapaa. Ensimmäinen vaihtoehto oli tehdä notification hubille rekisteröityminen ns. valmiiseen sovellukseen. Koska Qt-projekti oli käännettävä Visual Studion omaksi projekti ennen suoritusta, lisättäisiin rekisteröityminen ilman varsinaista kosketusta Qt:n koodiin. Ongelmaksi tässä tapauksessa saattoi osoittautua sovelluksen käyttäjän todentaminen, mikä oli tehtävä ennen notification hubille rekisteröitymistä. Toinen vaihtoehto oli kääntää projekti Visual Studiolle, tehdä notification hubille rekisteröityminen omaksi luokakseen ja tämän jälkeen ladata tämä luokka Qt:n QLibrary-luokalla sovellukseen.

Yllättäen ongelmaksi muodostunut internetsivun esittäminen Windows Phone -laitteilla olisi ollut mahdollista kiertää Googlella käytössä olevan API:n avulla. Käyttäjän todentaminen on mahdollista lähettämällä Google-tilin tunnus ja salasana internetsivun osoiterivin parametrina, mutta tätä toimintatapaa ei haluttu käyttää sen tietoturvariskin takia. Tilaajayrityksen kanssa oltiin samaa mieltä asiasta, ettei käyttäjän tietojen kysyminen muualla kuin Googlen kirjautumissivulla anna hyvää kuvaa sovelluksesta. Windows-mobiililaitteiden pienen markkinaosuuden vuoksi päätettiin, että sovellusta ei toteuta tälle alustalle.

Applen iOS-laitteelle ilmoitusten toiminnan testaamisen sijaan, seuraavaksi testattiin kameran toimintaa. Tämä päätös tehtiin yrityksen senhetkisen laitetilanteen vuoksi. Kameran toiminnassa testattiin sekä valokuvaus että videokuvaus. iOS-laitteella kameran käyttö onnistui, mutta Android-

laitteilla huomattujen ongelmien takia projektin jatkoa jouduttiin miettimään uudestaan. Sovellusta kehitettiin ilmaisella lisenssillä, joten Qt:lta oli mahdotonta kysyä teknistä tukea kohdattuihin ongelmiin. Projektin alussa Qt:lla oli mobiililaitteiden sovelluskehitystä varten maksullinen lisenssi, jonka hinta oli 29 \$/kk. Projektin aikana Qt muutti hinnoitteluaan ja poisti mobiililaitelisanssin käytöstä. Uusi hinta tilaajayritykselle olisi ollut 350 \$/kk.

Kameran käyttö ja varsinkin videokuvaus oli kuitenkin yksi sovelluksen päätoiminnoista, eikä siitä ollut valmiita luopumaan. Ongelmien ratkaisuun ei haluttu käyttää projektin loppuaikaa, koska tavoitteena oli saada mahdollisimman pitkälle kehitetty sovellus. Riskinä olisi ollut myös huono ylläpidettävyys, jos ongelmat olisi onnistuttu ratkaisemaan vain väliaikaisesti. Seuraavaksi aloitettiin tutkinta projektin toteuttamisesta toisella tekniikalla. Tutkimuksissa kävi ilmi, että yleisesti markkinoilla käytössä olevilla tekniikoilla jokin ominaisuus olisi jouduttu jättämään pois jollakin mobiilialustalla tai sovelluksen tekoon käytetty aika ei olisi eronnut juurikaan siitä, mitä se olisi, jos sovellus tehtäisiin erikseen jokaiselle alustalle. Lopullisena tuloksena päätettiin, että sovelluksen toteutuksesta alustasta riippumattomana luovutaan kokonaan.

Koska sovellus oli tehtävä jokaiselle mobiilialustalle alustan omalla ohjelmointikielellä, oli seuraavaksi valittava alusta, jolle sovellus toteutetaan ensimmäisenä. Vaihtoehtoina oli joko Android tai iOS, joista valittiin iOS. Perusteena valinnalle oli oma mielenkiintoni iOS-sovelluskehitykseen.

## 4 APPLE IOS-SOVELLUKSEN TOIMINTA

Tässä luvussa kerrotaan sovelluksen toiminnasta iOS-laitteissa. Sovelluksen kaikki toiminnot on toteutettu Swift-kielillä.

### 4.1 Framework ja Core data

Microsoft on luonut frameworkin Mobile Servicesin käyttämistä varten. Framework on kirjoitettu Objective-C-kielillä. Jotta frameworkin käyttö olisi mahdollista Swift-kielillä tehdyssä sovelluksessa, on luotava Objective-C bridging header -tiedosto. Tähän tiedostoon sijoitetaan import käsky tarvittavien framework luokkien header tiedostoihin. Projektin asetuksiin on asetettava bridging header -tiedoston nimi, jotta se huomioidaan käännöksen aikana. Tämän jälkeen frameworkia voidaan käyttää Swiftin syntaksilla.

Core Data mahdollistaa tiedon tallennuksen laitteeseen tietokannoista tutulla rakenteella. Core Data tarvitsee toimiakseen luonnoksen sen tiedoista. Managed object modelin avulla määritetään tietojen rakenne, joka koostuu entiteyistä. Jokainen entity sisältää attribuutteja, joille on määrätty minkä tyyppistä tietoa ne ovat (esimerkki kuvassa 10). Core Dataa käsitellään sen oman frameworkin avulla. Core Dataa tieto haetaan Managed object contextiin kopioina ja tiedot muokataan managed objectien avulla. Objektia voidaan muokata halutulla tavalla. Muokattu objekti täytyy tallentaa manuaalisesti managed object contextista Core Dataan. (Mac Developer Library 2014c)

ENTITIES

E

 Authority

E

 Customers

E

 Groups

E

 Images

E

 Messages

E

 Updates

FETCH REQUESTS

CONFIGURATIONS

C

 Default

▼ Attributes

Attribute ^	Type	
<div>S</div> client	String	↕
<div>S</div> id	String	↕
<div>S</div> message	String	↕
<div>D</div> ms_createdAt	Date	↕
<div>S</div> nickname	String	↕
<div>S</div> sender	String	↕

+ -

KUVA 10. Core datan rakenne Xcode:ssa

### 4.2 Sovellukseen kirjautuminen

Ennen sovelluksen käyttöä käyttäjän on rekisteröidyttävä tilaajayrityksen toisen sovelluksen avulla järjestelmään. Tätä prosessia varten käyttäjä tarvitsee Google-tilin, jota käytetään apuna käyttäjän todentamisessa. Ensimmäistä kertaa sovelluksen avaava käyttäjä ohjataan Microsoft Azure Mobile Services API:n avulla Googlen kirjautumissivulle. Kun käyttäjä on todennettu Googlen palvelun avulla, varmistetaan palvelimella olevan API:n avulla, että kyseinen Google-tili on olemassa tietokannassa. Onnistuneen kirjautumisen jälkeen käyttäjän tilin id ja Googlen luoma token tallennetaan laitteen muistiin. Näin käyttäjä ei tarvitse kirjautua sisään aina kun sovellus avataan. Käyttäjän token voi

vanhentua tai se voidaan peruuttaa. Näissä tapauksissa käyttäjää pyydetään kirjautumaan uudelleen.

Kirjautumisen jälkeen laitteen PNS-kahva ja käyttäjän id rekisteröidään tätä sovellusta varten luodulle Notification Hubille. Normaalisti nämä tiedot rekisteröidään Notification Hubille, jota Mobile Service käyttää. Syy tälle toimenpiteelle on selvennetty luvussa 4.4. Kirjautumisen jälkeen suoritetaan haku tietokannasta. Koska Mobile Services framework suorittaa tietokantaoperaatiot asynkronisesti, tietokannan hakuja ketjutetaan completion handlerien avulla. Completion handleria käyttävän metodin esittely on havainnollistettu kuvassa 11. Ketjun seuraavaa metodia kutsutaan completion metodissa. Käyttäjälle esitetään latausikkuna, josta poistutaan sovelluksen etusivulle vasta kun kaikki tiedot on haettu.

```
public func loadMessagesFromAPI(id:String, completion: () -> Void) {
```

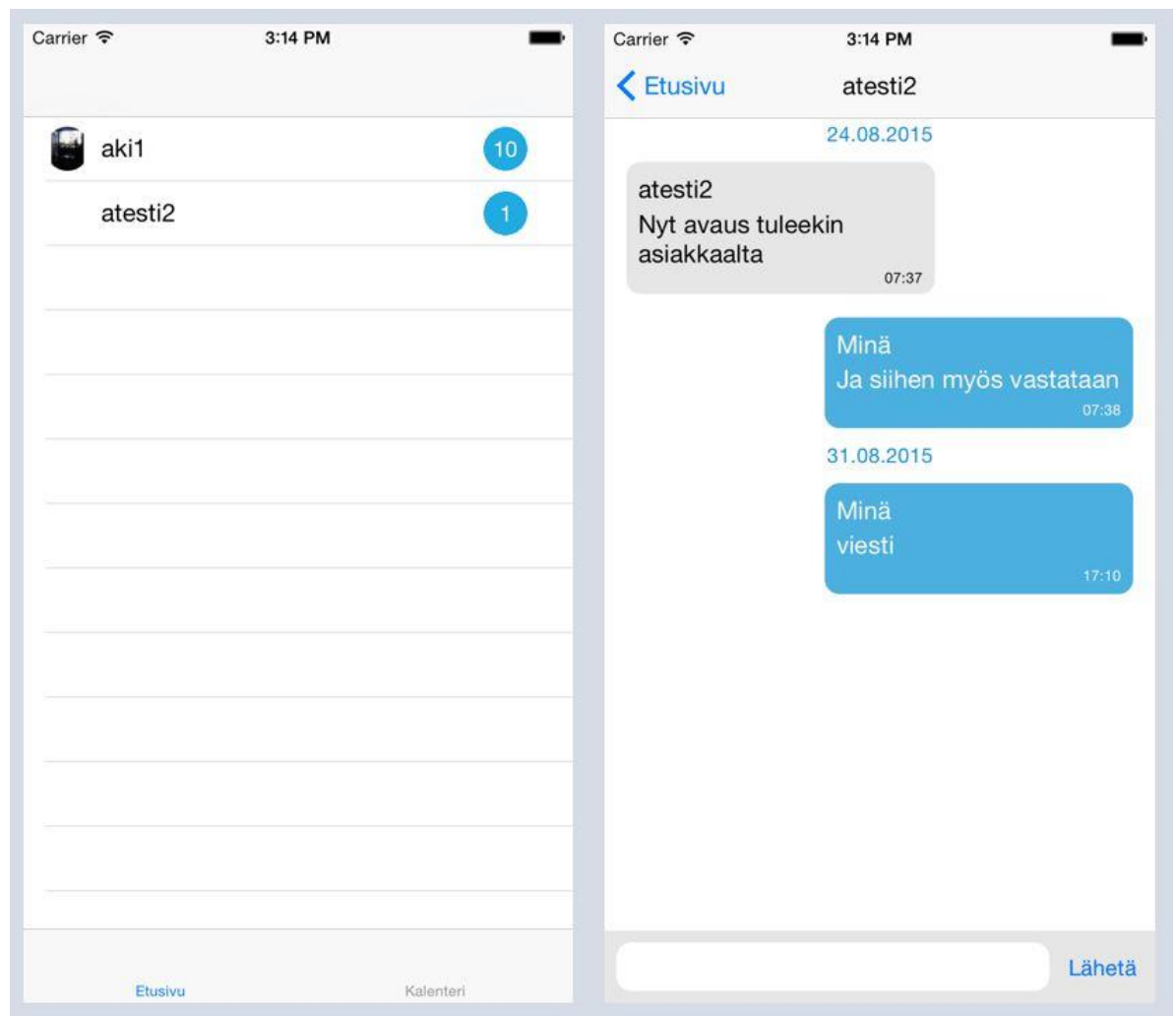
Kuva 11. Viestien hausta vastaavan metodin esittely

### 4.3 Viestit

Sovelluksen käyttäjät voivat viestiä viestiominaisuuden avulla. Viestit haetaan tietokannasta palvelimen API:n kautta. Viesteistä vastaava API tarvitsee parametrina käyttäjän yksilöllisen id:n. API palauttaa JSON-aulun kaikista viesteistä, joihin käyttäjä on osallisena. Taulun yksittäinen tietue pitää sisällään mm. viestin lähettäjän, vastaanottajan, tallennusajan ja viestin. Viestit tallennetaan tämän jälkeen puhelimen muistiin ja samalla päivitetään muistissa olevaa uusien viestien laskuria.

Viesti-ikkunan päänäkyssä esitetään keskustelukumppaneiden nimi ja uusien viestien määrä. Päänäkymä on Table View -komponentti, jonka jokainen solu sisältää kuva-, tekstikentän ja painikkeen. Kuvakentässä esitetään käyttäjän profiilikuva, tekstikentässä käyttäjän nimimerkki ja painikkeen tekstinä on uusien viestien määrä. Mikäli käyttäjä ei ole lähettänyt uusia viestejä, piilotetaan painike näkyvistä. Kun käyttäjä klikkaa viesti-ikkunasta keskustelukumppanin nimeä, ohjataan käyttäjä uudelle sivulle, jossa esitetään tämän käyttäjän kanssa vaihdetut viestit. Viesti-ikkunoiden ulkoasu on esitetty kuvassa 12.

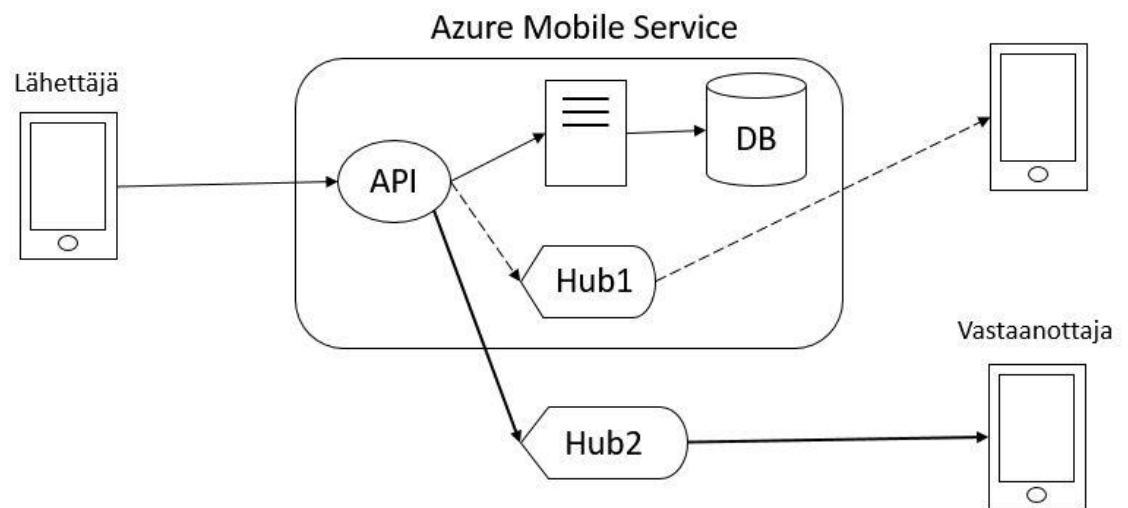
Viestit haetaan puhelimen muistista ennen viestisivun esittämistä ja samaan aikaan muistiin tallennettu uusien viestien määrä nollataan. Sivulla tietojen esityksessä Table View:ssa käytetään kolmea erilaista solukomponenttia. Yksi solu sisältää tekstikentän, johon kirjoitetaan päivämäärä. Toinen ja kolmas solu sisältävät molemmat tekstikentät viestin lähettäjälle, viestille ja lähetysajalle. Solujen ero on taustavärisä ja näiden käyttö riippuu viestin lähettäjästä. Harmaalla taustavärillä oleva solu on tarkoitettu keskustelukumppanin viesteille, kun taas sinisellä taustalla olevaa solua käytetään omien viestien esittämiseen. Mikäli käyttäjä saa viestin keskustelukumppaniltaan, lähettää viesteistä vastaava palvelimen API ilmoituksen käyttäjän laitteeseen. Kun käyttäjä avaa sovelluksen seuraavan kerran, suoritetaan viestien haku tietokannasta. Ilman ilmoitusta uusien viestien haku tapahtuu ainoastaan kerran päivässä, tällä vähennetään tarpeetonta viestien hakua.



KUVA 12. Vasemmalla viestien pääsivu ja oikealla henkilön viestinäkömä

#### 4.4 Ilmoitukset

Koska sovellus toimii samassa Mobile Servicesissa, mihin on jo asetettu sertifikaatti toista sovellusta varten, täytyi ilmoituksen lähetys toteuttaa vaihtoehtoisella tavalla. Azureen luodaan uusi Notification Hub, johon asetetaan kehityssertifikaatti tätä kyseistä sovellusta varten. Tämän jälkeen jokainen sovellukseen tarkoitettu ilmoitus ohjataan API:n avulla pois Mobile Servicen omasta Notification Hubista tätä sovellusta varten luotuun Hubiin. Tämä prosessi on selvennetty kuvassa 13. Kuvassa myös esitetään tiedon tallennus tietokantaan (DB). Viesteistä vastaavan API:n Node.js-koodiin täytyi tehdä muutos. Ensin varmistetaan, että ilmoitus on tarkoitus toimittaa tähän sovellukseen. Seuraavaksi luodaan yhteys uuteen NotificationHubServiceen, johon sovelluksen laitteet on rekisteröity. Tämän jälkeen NotificationHubService:lla lähetetään ilmoitus Apple push notification servicelle ja parametrina annetaan käyttäjän Google-tilin id.



KUVA 13. Viestin ohjaus oikealle laitteelle.

#### 4.5 Kameran käyttö

Alkuperäisessä suunnitelmassa ollut kameran käyttö sovelluksessa jätetään toteuttamatta. Tämän ominaisuuden lisääminen sovellukseen vaatii suurta tietokannan muokkaamista. Tietokantaa ei voida muokata projektin aikana, koska tämä vaatisi muutoksia myös yrityksen muissa sovelluksissa.

## 5 YHTEENVETO

Projektin tavoitteena oli luoda alustasta riippumaton mobiilisovellus, jolla olisi mahdollista viestiä toiseen laitteeseen ja käyttää puhelimen kameraa apuna harjoitusohjelman laadinnassa. Sovelluksen oli tarkoitus toimia Applen iOS-, Android- ja Windows Phone -laitteilla. Sovelluksen käyttämä tietokanta toimi Microsoftin Azure-pilvipalvelussa. Sovelluksen oli toimittava yhdessä tilaajayrityksen muiden sovellusten kanssa. Sovelluksen ulkoasu oli tarkoitus toteuttaa yhdessä yrityksen muotoilijoiden kanssa. Projekti aloitettiin tutkimalla, millä ohjelmointiympäristöllä toteutus olisi mahdollinen. Tutkinnan jälkeen ympäristöksi valittiin Qt. Projektin alussa suurimmaksi ongelmaksi arvioitiin ilmoitusten vastaanottamista ja laitteen kameran käyttöä. Ilmoituksia varten jokaiselle valitulle mobiilialustalle täytyisi tehdä sen omalla ohjelmointikielellä osuus. Projektin aikataulutusta suunniteltiin näiden haasteiden perusteella.

Projektin edetessä huomattiin, että Qt ei sovellu tämän sovelluksen toteutusympäristöksi, joten toteutuksesta alustasta riippumattomana päätettiin luopua. Koska sovellus tulisi toteuttamaan jokaiselle laitteelle sen omalla ohjelmointikielellä, aloitettiin toteutus iOS-laitteelle.

Projektin alussa ilmenneiden ongelmien takia opinnäytetyön aikana ei ehditty ottaa vielä ollenkaan kantaa, kuinka sovellus käsittelee kuvaa ja/tai videota. Tämä ominaisuus asettaa haasteita myös tietokannalle, sillä otettu kuva tai video on kyettävä suojaamaan niin, että kenelläkään ei ole mahdollista avata sitä luvatta. Harjoitusohjelman luonnissa tarvitaan dataa toisesta tietokannasta. Koska Mobile Service voi käyttää vain yhtä tietokantaa, olisi tähän toiseen kantaan päästävä käsiksi jollain muulla tavalla. Tietokantoja tullaan muokkaamaan siten, että vältetään haulta kahdesta paikasta. Sovelluksen kehitystä jatketaan tämän projektin jälkeen.

Projekti oli kokonaisuudessaan erittäin opettava. Pidän tulevaisuuden kannalta tärkeänä iOS-laitteista osaamista ja Swift-kielen hallintaa. Jälkiviisaana voi sanoa, että tilaajayrityksen kannalta toteutuksesta alustasta riippumattomana olisi pitänyt luopua heti alussa ja toteuttaa sovellus jokaiselle alustalle sen omalla ohjelmointikielellä.

## LÄHTEET

CPLUSPLUS.COM 2000 – 2015. A Brief Description. [Viitattu 2015-07-05]. Saatavissa:

<http://www.cplusplus.com/info/description/>

DEVELOPER 2015a. Swift. [Viitattu 2015-08-09]. Saatavissa: <https://developer.apple.com/swift/>

DEVELOPER 2015b. Xcode IDE. [Viitattu 2015-12-15]. Saatavissa:

<https://developer.apple.com/xcode/features/>

DEVELOPERS 2015. Features. [Viitattu 2015-12-15]. Saatavissa:

<http://developer.android.com/tools/studio/studio-features.html>

IOS DEVELOPER LIBRARY 2015. Local and Remote Notification Programming Guide. [Viitattu 2015-09-09]. Saatavissa:

<https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html>

INTRODUCING JSON. [Viitattu 2015-09-14]. Saatavissa: <http://json.org/>

MAC DEVELOPER LIBRARY 2013. Framework Programming Guide. [Viitattu 2015-08-23]. Saatavissa:

[https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/BPFrameworks/Concepts/WhatAreFrameworks.html#//apple\\_ref/doc/uid/20002303-BBCEIJFI](https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/BPFrameworks/Concepts/WhatAreFrameworks.html#//apple_ref/doc/uid/20002303-BBCEIJFI)

MAC DEVELOPER LIBRARY 2014a. About Threaded Programming. [Viitattu 2015-07-26]. Saatavissa:

<https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Multithreading/AboutThreads/AboutThreads.html>

MAC DEVELOPER LIBRARY 2014b. About Objective-C. [Viitattu 2015-08-09]. Saatavissa:

<https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>

MAC DEVELOPER LIBRARY 2014c. Core Data Programming. [Viitattu 2015-08-23]. Saatavissa:

[https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CoreData/Articles/cdBasics.html#//apple\\_ref/doc/uid/TP40001650-TP1](https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CoreData/Articles/cdBasics.html#//apple_ref/doc/uid/TP40001650-TP1)

MICROSOFT AZURE 2015. Azure Notification Hubs. [Viitattu 2015-07-28]. Saatavissa:

<https://azure.microsoft.com/en-us/documentation/articles/notification-hubs-overview/>

MICROSOFT 2015a. Azure Documentation. [Viitattu 2015-07-05]. Saatavissa:

<http://azure.microsoft.com/en-us/documentation/>



MICROSOFT 2015b. Mobile Services Documentation. [Viitattu 2015-07-05]. Saatavissa: <http://azure.microsoft.com/en-us/documentation/services/mobile-services/>

MOZILLA DEVELOPER NETWORK 2005–2015. About JavaScript. [Viitattu 2015-10-14]. Saatavissa: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/About\\_JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript)

NODE.JS 2015. [Viitattu 2015-10-11]. Saatavissa: <https://nodejs.org/en/>

ORACLE 2015. The Java® Language Specification. Chapter 1. Introduction. [Viitattu 2015-09-23]. Saatavissa: <http://docs.oracle.com/javase/specs/jls/se8/html/index.html>

QT 2015a. About Qt. [Viitattu 2015-07-05]. Saatavissa: [http://wiki.qt.io/About\\_Qt](http://wiki.qt.io/About_Qt)

QT 2015b. Qt Documentation. Adding Kits. [Viitattu 2015-08-02]. Saatavissa: <http://doc.qt.io/qtcreator/creator-targets.html>

VISUAL STUDIO 2015. What's New About Visual Studio 2013. [Viitattu 2015-12-15]. Saatavissa: <https://www.visualstudio.com/en-us/news/2013-oct-17-vs.aspx>