VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Mika Salmela

DEVATUS PUSH NOTIFICATION - PALVELIN

Tekniikka ja liikenne
2014

ALKUSANAT

Opinnäytetyö on tehty Vaasan ammattikorkeakoulun tietotekniikan koulutusohjelman päättötyönä. Työ tehtiin Devatus Oy:lle ja toteutus sijoittui keväälle 2014.

Opinnäytetyön valvojana Vaasan ammattikorkeakoulun puolelta toimi yliopettaja Ghodrat Moghadampour. Ohjaajana Devatus Oy:n puolelta oli Senior Project Manager Janne Kauppila.

Haluan kiittää Devatus Oy:n henkilökuntaa sekä erityisesti Janne Kauppilaa, Software Specialist Atte Leppälää ja Ghodrat Moghadampouria tuesta opinnäytetyön tekemisessä.

Vaasassa, 25.5.2014

Mika Salmela

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Mika Salmela
Opinnäytetyön nimi	Devatus Push Notification -palvelin
Vuosi	2014
Kieli	suomi
Sivumäärä	52
Ohjaaja	Ghodrat Moghadampour

Opinnäytetyö toteutettiin Devatus Oy:lle. Tarkoituksena oli toteuttaa ympäristö, jolla mahdollistetaan Push-notifikaatioiden lähettäminen ylläpitäjän sivustolta, sovellusta käyttäville mobiililaitteille. Opinnäytetyössä toteutettiin sovellukset Android-, Windows Phone- ja iOS –alustoille.

Eri mobiilialustoille toteutettiin sovellukset, joihin käytettiin C#-, JAVA- ja Objective-C -ohjelmointikieliä. Palvelimelle toteutettiin PHP-ohjelmointikielellä ylläpitäjän sivusto sekä tarvittavat rajapinnat, joita sovellukset käyttivät. Laiteet pystyivät lähettämään tietoja palvelimelle ja tiedot tallennettiin MySQL-tietokantaan.

Opinnäytetyön tuloksena saatiin toteutettua sovellukset eri mobiilialustoille sekä ylläpitäjän sivusto, josta on mahdollisuus lähettää Push-notifikaatioita sovellusta käyttäville laittelle

Avainsanat Push -notifikaatio, mobiililaite, palvelin, ylläpitäjä

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Mika Salmela
Title	Devatus Push Notification -Server
Year	2014
Language	Finnish
Pages	52
Name of Supervisor	Ghodrat Moghadampour

The thesis was done for Devatus Oy. The main purpose of the thesis was to implement the environment, which allows the administrator to send Push notifications from the admin page to mobile devices which are using our application. In the thesis applications for Android-, Windows Phone- and iOS –platforms were implemented.

Using C#-, JAVA- and Objective-C programming languages, we are able to implement applications for different mobile platforms. On the server side, the PHP programming language was used to implement an admin page and also interfaces which the devices are using. Mobile devices will send information to the server and the information will be stored into a MySQL -database.

The result of this thesis were applications for different mobile platforms and an administrator page. The administrator page allows admin to send Push notifications to devices which are using the application.

Keywords Push notification, mobile device, server, admin

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

TERMIT JA LYHENTEET

API	Application programming interface, kutsuttavan ohjelman tunniste
APNS	Apple Push Notification Service, Applen Push -notifikaatiopalvelin
CSR	Certificate signing request, varmennepyyntö
GCM	Google Cloud Messaging, Push notifikaatiopalvelin Androidille
HTML	Hypertext Markup Language, standardoitu kuvauskieli
HTTP	Hypertext Transfer Protocol, siirtoprotokolla
JSON	JavaScript Object Notation, tiedostomuoto tiedonvälitykseen
MPNS	Microsoft Push Notification Service, Push -notifikaatiopalvelin Windows Phonelle
SSL	Secure Sockets Layer, tietoverkkosalausprotokolla
URI	Uniform Resource Identifier, merkkijono tietyn tiedon sijainnista

1 JOHDANTO

Nykyään monet suositut sovellukset käyttävät Push-notifikaatioita lähettääkseen sovelluksen käyttäjille tietoja. Push notifikaatio -teknologian avulla pystytään helposti välittämään esimerkiksi mainoksia tai tiedotteita. Esimerkkinä Spotify-sovellus, jossa kaikille sovellusta käyttäville laitteille ilmoitetaan uusista musiikkikappaleista tai albumeista, jotka ovat juuri saapuneet markkinoille.

Opinnäytetyön toimeksiantajana toimii Devatus Oy. Tarkoituksena tutkia ja toteuttaa ympäristö, jolla mahdollistetaan Push-notifikaatioiden lähettäminen ja vastaanottaminen. Toteutusvaiheessa tullaan esittämään vaihtoehtoiset ratkaisut sovelluksista eri alustoille, sivusto ylläpitäjälle, vaadittavat rajapinnat palvelimelle sekä toteutukseen vaadittavat tekniikat.

Toteutuksen jälkeen ympäristön toiminta tullaan testaamaan erilaisten testitapauksien avulla. Tutkitaan ja perehdytään, kuinka toimintoja tulisi parantaa. Opinnäytetyön lopussa käydään läpi jatkosuunnitelmat sovelluksen kehittämisen suhteen ja pohditaan opinnäytetyön onnistuvuutta.

2 DEVATUS OY

2.1 Yrityksen toiminta

Devatus Oy on nopeasti kasvava yksityisomistuksessa oleva ohjelmistokehitysyritys, joka on perustettu heinäkuussa 2010 Vaasassa. Yhtiö on kokonaisuudessaan työntekijöiden omistuksessa. Heillä on poikkeuksellisen vahva tausta useista kansainvälisistä hankkeista ohjelmistoalalta, kuten mobiilihankkeista, web-kehityksestä, liiketoiminnan ohjelmisto- ja projektinhallinnasta.

Työntekijöiden yli kymmenen vuoden IT-alan kokemuksella taataan, että asiakkaiden haasteista pystytään luomaan perusteellinen käsitys ja näin ollen niihin pystytään tarjoamaan luotettavia ratkaisuja. Yritys tarjoaa asiakkailleen mobiilisovellusten kehitystä. Tämän avulla pystytään tarjoamaan ratkaisuja esimerkiksi asiakkaiden käyttöliittymän ja sovellusten kehitykseen. Yritys myös tarjoaa yrityssovellusten kehitystä, joka sisältää asiakkaiden sen hetkisten sovellusten kehittämistä tai uusien sovelluksien toteuttamista liiketoimintaprosessien tukemiseksi. /7/

2.2 Opinnäytetyön tarkoitus

Nykyään monissa suosituissa mobiilisovelluksissa on ominaisuus, jolla pystytään välittämään sovelluksen käyttäjälle tiedotteita. Devatus Oy on toteuttanut monia sovelluksia mobiililaitteille, joilla on päivittäin suuria määriä käyttäjiä. Tarkoituksena onkin tutkia ja toteuttaa yritykselle ympäristö, jolla mahdollistetaan Push-notifikaatioiden lähettäminen ja vastaanottaminen. Yritys pystyisi käyttämään tulevaisuudessa ympäristöä uusien sovellusten kanssa sekä integroimaan valmiina oleviin ympäristöihin.

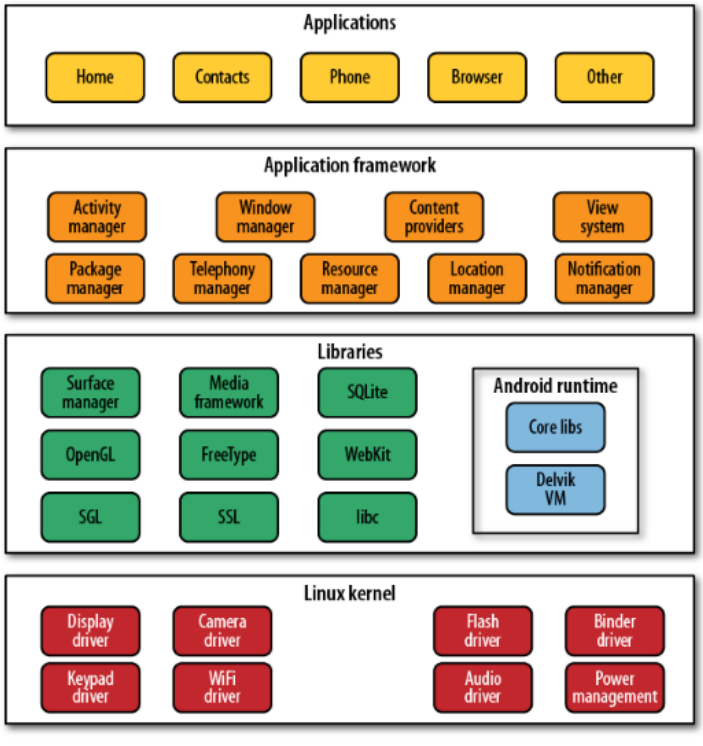
3 KÄYTETYT TEKNOLOGIAT

3.1 Mobiilialustat

3.1.1 Android

Android on yksi suosituimmista ja yleisimmistä mobiilialustoista. Alusta on Googlen omistama ja nykyään kehityksestä vastaa Open Handset Alliance.

Android on avoimen lähdekoodin alusta, joka on suunniteltu mobiililaitteille. Android käyttöjärjestelmä perustuu Linux-käyttöjärjestelmään. Alusta on avoinna aina alemmalta Linux-tasolta asti natiiveihin kirjastoihin ja sovelluksen frameworkista itse sovellukseen. Alusta perustuu sovellus-, sovellus framework-, kirjasto- ja Linux kernel –tasoista. Kehittäjillä on mahdollisuus päästä muuttamaan alustan lähdekoodia ja oikeus muokata alustaa omien vaatimusten mukaiseksi **(Kuvio 1.)**. /1/



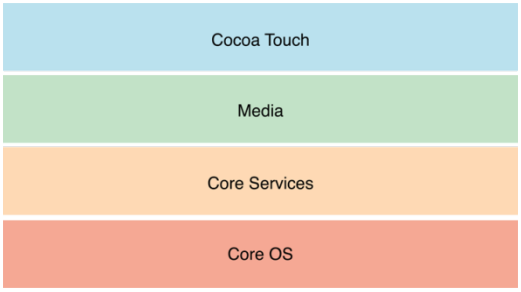
Kuvio 1. Android-arkkitehtuuri /1, 8/

Android–sovellusten kehittäjille on tarjolla ilmaiset työkalut sovellusten toteuttamiseen nopeasti ja helposti. Sovelluksien toteuttamiseen tarvitaan Android SDK, eikä edes itse Android-laite ole välttämätön. /1, 1-11/

3.1.2 iOS

iOS on Applen kehittämä käyttöjärjestelmä, joka on käytössä Applen omilla mobiililaitteilla. Käyttöjärjestelmä julkaistiin vuonna 2007, ensimmäisen iPhonen myötä. Alustan arkkitehtuuri perustuu Cocoa Touch-, Media-, Core Services- ja Core OS kerroksista **(Kuvio 2.)**.

Cocoa Touch –kerros sisältää teknologiat, joilla tuetaan kosketus-, push notification- ja ylemmän tason palveluiden ominaisuuksia. Core Services –kerros sisältää peruspalvelut sovelluksille. Kerros myös sisältää teknologiat, joilla tuetaan sijainti-, iCloud-, sosiaalinen media- ja verkko-ominaisuuksia. Media-kerros sisältää grafiikka-, ääni- ja video-teknologiat. Kerros auttaa käyttämään sovelluksissa edellä mainittuja teknologioita, jolloin sovellukseen pystytään lisäämään erilaisia grafiikoita, ääniä ja videoita. Alemman tason ominaisuudet ovat Core OS –kerroksessa. Kerrosta käytetään mm. jos sovelluksen pitää kommunikoida lisälaitteiden kanssa, esimerkiksi bluetoothin käyttö sovelluksessa. /6/



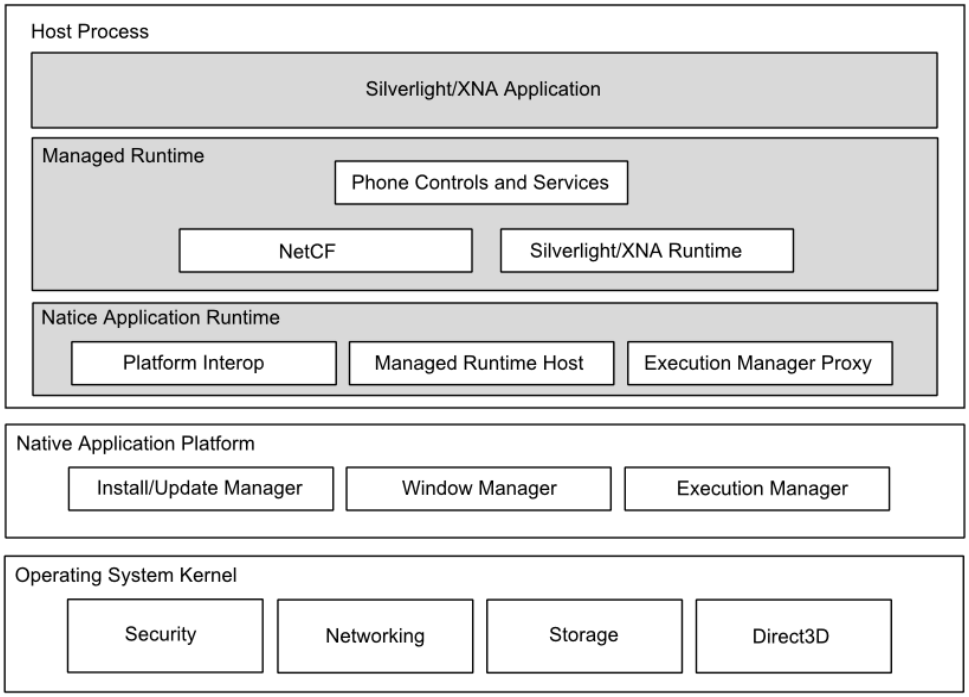
Kuvio 2. iOS arkkitehtuuri /6/

Applen tarjoama Xcode-ohjelmisto sisältää kaikki vaadittavat työkalut ja ominaisuudet, joilla pystytään kehittämään sovelluksia iOS-laitteille.

3.1.3 Windows Phone

Windows Phone on Microsoftin julkaisema käyttöjärjestelmä mobiililaitteille, joka julkaistiin vuonna 2010. Windows Phone -käyttöjärjestelmä perustuu kolmesta eri pääkerroksesta: Host Process, Native Application Platform ja Operating System Kernel **(Kuvio 3.)**.

Windows Phone -sovellukset on rakennettu Microsoftin työkalujen sekä teknologien päälle. Kaikkien kehittäjien, jotka tuntevat Microsoftin aikaisemmat teknologiat ja työkalut, on helppo omaksua sovelluksien toteuttamisen Windows Phonelle. Microsoft tarjoaa työkalut Windows Phone -sovellusten toteuttamiseksi mm. Visual Studio, Expression Blend. /2/.



Kuvio 3. Windows Phone -arkkitehtuuri /2, 8/

3.2 Palvelimen sisältämät teknologiat

3.2.1 MySQL

Monet dynaamiset web-sivustot vaativat tietokannan, johon voidaan tallentaa tietoa, jonka web-sivusto pystyy näyttämään käyttäjille tai voidaan varastoida talteen. MySQL on maailman suosituimpia avoimen lähdekoodin tietokantaohjelmistoja. MySQL:n kehittämisestä vastaa MySQL AB. Ohjelmiston nopeuden, luotettavuuden ja helppokäyttöisyyden takia, siitä on tullut monien suurtenkin yritysten valitsema ratkaisu tietokantojen toteutukseen.

MySQL-ohjelmisto sisältää MySQL-palvelimen, tiettyjä lisäohjelmia, jotka helpottavat ylläpitäjää työskentelemään tietokantojen kanssa. Ohjelmisto sisältää myös tarvittavat ohjelmistot toimiakseen. /12/

3.2.2 Apache HTTP Server

Apache HTTP Server on yksi Apache Software Foundationin hankkeista. Apachen HTTP Server, tunnetaan myös nimellä "httpd". Apache HTTP on ollut suosituin palvelinohjelmisto vuodesta 1996 lähtien.

Hankkeen tavoitteena on kehittää, ylläpitää ja tarjota avoimen lähdekoodin HTTP-palvelinohjelmaa. Apache HTTP Server tarjoaa turvallisen, tehokkaan ja laajennettavissa olevan palvelimen, joka tarjoaa HTTP-palveluja käyttäen nykyisiä HTTP-standardeja. Httpd on käytettävissä Unix- ja Windows-käyttöjärjestelmissä. /5/.

3.3 Ohjelmointikielet

3.3.1 Java

Sun Microsystemsin luoma ohjelmointikieli, joka julkaistiin 1990-luvun puolivälissä. Nykyään Sun Microsystemsin omistaa Oracle. Alustavasti Javaa on käytetty palvelinohjelmoinnissa, mutta nykyään käytössä laajemmin mm. mobiilisovellusten toteutuksessa. Java on luokkapohjainen olio-ohjelmointikieli, joka perustuu C- ja C++ -ohjelmointikieliin. Javalla toteutetut ohjelmat eivät ole käyttöjärjestelmä

riippuvaisia, joten ohjelmia ei tarvitse kääntää erikseen jokaiselle käyttöjärjestelmälle. /13/.

3.3.2 C#

C# on Microsoftin vuonna 2000 kehittämä ohjelmointikieli. C#:n tarkoituksena on olla yksinkertainen, moderni ja yleiskäyttöinen olio-ohjelmointikieli. C# perustuu C-ohjelmointikieleen ja näin ollen onkin helposti omaksuttavana niille, jotka ovat tuttuja C, C++ ja Java -ohjelmointikielten kanssa. C# tarjoaa helpotuksia C++ -ohjelmointikielen monimutkaisuuteen ja tarjoaa toimintoja, joita esim. Javasta ei löydy. /11/.

3.3.3 Objective-C

Toteutettaessa sovelluksia iOS- tai OS X -käyttöjärjestelmille, ohjelmointi tapahtuu ensisijaisesti Objective-C:llä. Objective-C perustuu C-ohjelmointikieleen, mutta tarjoaa olio-ominaisuudet. Objective-C perii syntaksin C-ohjelmointikielestä ja lisää syntaksin määrittelyyn luokkia ja metodeja. /10/

3.3.4 PHP

PHP (Hypertext Preprocessor) on skriptaus kieli, jonka alkutaipaleet alkoivat noin 1990-luvun puolivälissä. PHP:n syntaksi pohjautuu C, Java ja Pearl -ohjelmointikieliin. PHP:n avulla pystytään erityisesti luomaan dynaamisia web-sivustoja. PHP:ta pystytään käyttämään sulautettuna HTML:n sekaan **(Kuvio 4.)**. Kielellä pystytään myös käyttämään palvelin puolen ohjelmoinnissa. /14/

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <?php
5          echo "Hello World!";
6      ?>
7  </body>
8  </html>
9
10
```

Kuvio 4. Esimerkki PHP:ta sulautettu HTML:n sekaan

4 SOVELLUSTEN JA PALVELIMEN TOIMINTOJEN KUVAUS

4.1 Projektin kuvaus

Push-notifikaatio on tekniikka, jonka avulla sovellusta käyttävälle laitteelle pystytään lähettämään tietoja ilman, että sovellus on käynnissä lähetys hetkellä laitteessa. Tämän takia Push-notifikaatio tekniikka on tehokkaampi keino kuin esimerkiksi polling-tekniikka. Polling-tekniikassa sovellus tulee tarkistamaan, onko esimerkiksi palvelimella oleva arvo muuttunut. Sovellus ei välttämättä tule saamaan uutta tietoa, kun taas Push notification –tekniikassa sovelluksen ei tarvitse itse käydä tekemässä toimenpiteitä vaan sovellus tullaan herättämään. Tämän ansiosta estetään turhia toimenpiteitä, joka näkyy mm. laitteen virran kulutuksessa. /2, 409-444/

Projektin tavoitteena on luoda Push-notifikaatiopalvelin, josta pystytään lähettämään ylläpitäjän syöttämiä tietoja sovellusta käyttäville laitteille. Jotta on mahdollista lähettää ja vastaanottaa Push-notifikaatioita, tulee toteuttaa alustoille sovellukset, jotka pystyvät kommunikoimaan palvelinten kanssa. Sovelluksiin tullaan toteuttamaan lisäominaisuuksina Push-notifikaation tilaaminen omaan laitteeseen sekä mahdollisuus ottaa käyttöön tai poistaa käytöstä Push-notifikaatiot.

4.2 Vaatimusten määrittely

Vaatimusmäärittelyssä esitetään, mitä palvelimelta ja sovelluksilta vaaditaan. Taulukkoon on kerätty toiminnallisuuksia ja näille annettu myös prioriteetit **(Taulukko 1.)**. Taulukosta pystytään näkemään, mitkä toiminnot tulevat olemaan tärkeitä toteutuksessa.

Taulukko 1. Vaatimusmäärittely

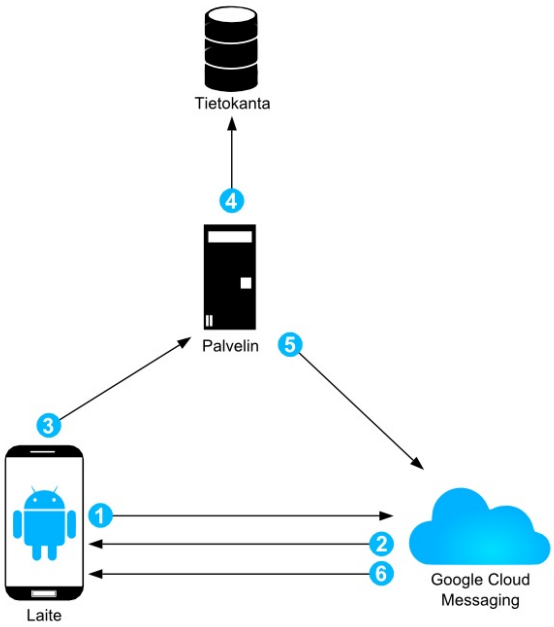
Toiminto	Prioriteetti
Sovellusten rekisteröityminen palvelimen kanssa (APNS, MPNS ja GCM).	1
Tietojen lähettäminen palvelimelle sovelluksesta.	1
Viestin lähettäminen ylläpitäjän sivustolta.	1
Viestin vastaanottaminen sovelluksessa.	1
Push -notifikaation tilaaminen sovelluksessa.	2
Tilaustunnuksen muuttaminen sovelluksessa	3
Tilaustunnuksen muuttaminen ylläpitäjän sivustolta	2
* Täytyy olla - 1, pitäisi olla - 2, kiva olla - 3.	

4.2.1 Toiminnallinen määrittely

Kaikilla alustoilla perus toiminnallisuus on samanlainen. Esimerkkinä Android -sovelluksen rekisteröityminen **(Kuvio 5.)**. Kuvasta nähdään numeroidut kohdat, joissa ensimmäinen tapahtuma on rekisteröintipyynnön lähettäminen alustalle tarkoitetulle palvelimelle. Kaikille alustoille on eri palvelimet, jotka hoitavat rekisteröintipyyntöjä. iOS-laitteille on tarjolla APNS, Windows Phone –laitteille MPNS ja Android-laitteille GCM.

Kohdassa kaksi, palautetaan laitteelle rekisteröintitunnus ja sovelluksen tulee pystyä vastaanottamaan rekisteröintitiedot. Kohdassa kolme, lähetetään saatu rekisteröintitunnus palvelimelle ja tiedot tallennetaan palvelimella MySQL-tietokantaan, joka näkyy kohdassa neljä.

Ylläpitäjä pystyy lähettämään palvelimelta ilmoituksen sovellusta käyttävälle laitteelle rekisteröintitunnusten avulla. Kohdassa viisi, ilmoitus lähetetään alustalle tarkoitetulle Push-notifkaatiopalvelimelle. Push-notifikaatiopalvelimet välittävät tiedot sovelluksille, joka näkyy kohdassa kuusi. Sovelluksen tulee pystyä ottamaan viesti vastaan ja tekemään siitä ilmoitus sovellusta käyttävälle laitteelle.

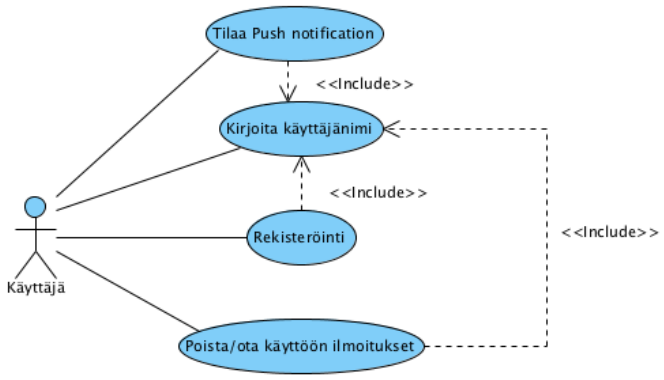


Kuvio 5. Toiminnallisuus

4.2.2 Käyttötapakaavio

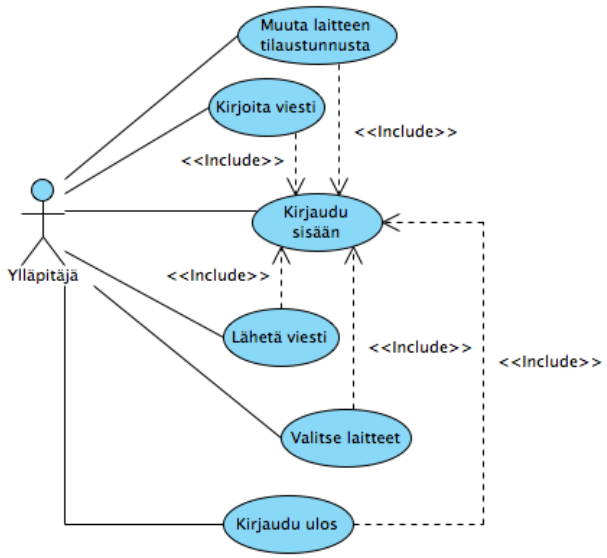
Käyttötapakaavion avulla annetaan sovelluksesta yleiskäsitys. Kaavio keskittyy siihen, mitä käyttäjä haluaa tehdä. Selvä esitystapa, jolla esitetään käyttäjän mahdollisuudet. /17/

Käyttäjällä on mahdollisuus tilata itselleen Push-notifikaation, lähettää rekisteröintitiedot palvelimelle ja muuttaa tilaustunnusta. Käyttäjältä vaaditaan käyttäjänimen syöttäminen, jonka jälkeen toiminnot ovat mahdollista toteuttaa. **(Kuvio 6.)**.



Kuvio 6. Käyttäjän käyttötapakaavio

Ylläpitäjällä on mahdollisuus kirjautua sisään, lähettää ilmoitus, valita laitteet listasta, tilaustunnuksen muuttaminen, viestin kirjoittaminen sekä ulos kirjautuminen sivustolta. Toiminnot vaativat sisään kirjautumisen **(Kuvio 7.)**.

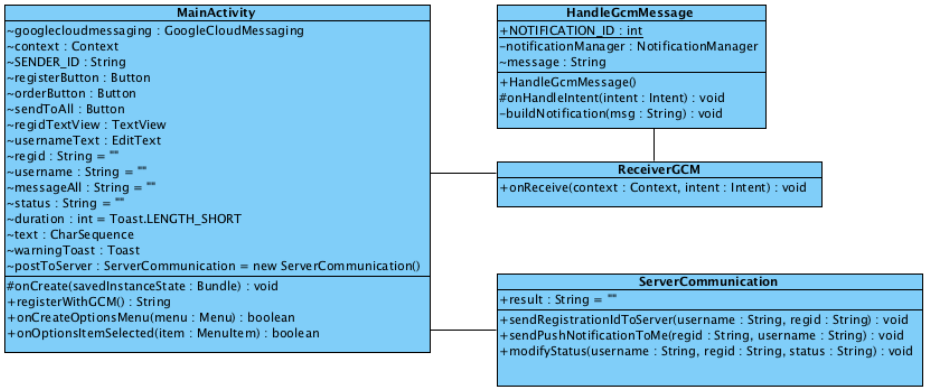


Kuvio 7. Ylläpitäjän käyttötapakaavio

4.2.3 Luokkakaavio

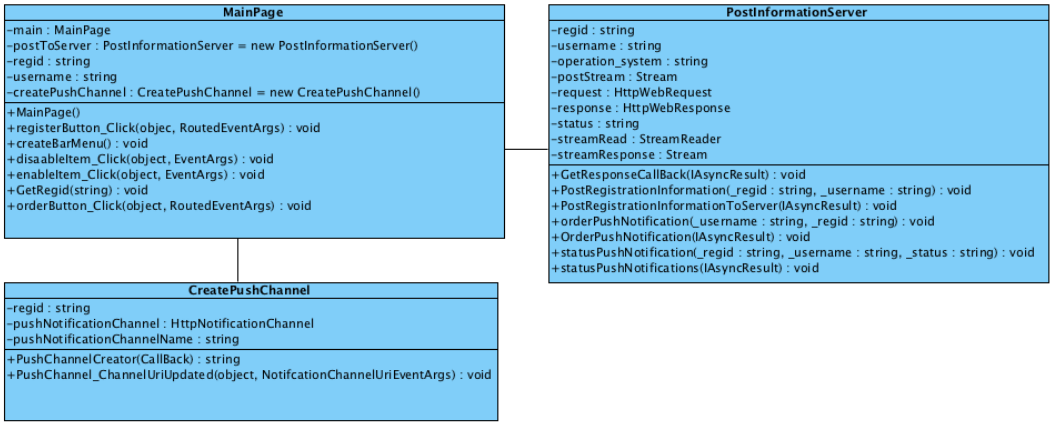
Luokkakaavioiden avulla annetaan sovelluksesta täsmällisempi kuva. Sillä esitetään millaisia objekteja järjestelmässä esiintyy ja niiden väliset staattiset suhteet. Luokkakaaviot näyttävät luokkien attribuutit ja operaatiot sekä kuinka nämä välittyvät luokkien välillä. /17/

Android-sovelluksessa tulee olemaan neljä luokkaa. MainActivity-luokka hoitaa graafisen käyttöliittymän toteutuksen sekä sisältää funktion, joka rekisteröi laitteen GCM:lle. ServerCommunication-luokka sisältää funktiot, joiden avulla pystytään kommunikoimaan palvelimen kanssa. ReceiverGCM- ja HandleGcmMessage-luokat hoitavat toimenpiteet, jotka mahdollistavat ilmoituksien saapumisen laitteeseen **(Kuvio 8.)**.



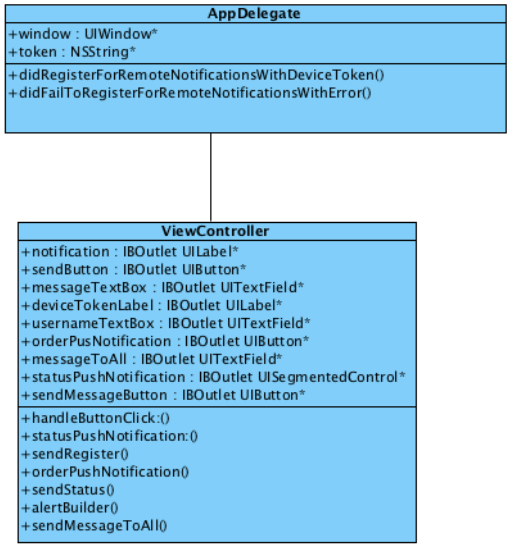
Kuvio 8. Android-sovelluksen luokkakaavio

Windows Phone -sovelluksessa tulee olemaan kolme eri luokkaa. MainPage -luokka hoitaa graafisen käyttöliittymän toteutuksen. CreatePushChannel-luokka hoitaa Push-notifkaatiokanavan luonnin sekä rekisteröintitunnuksen saamisen. PostInformationServer-luokka vastaa kommunikoinnin palvelimen kanssa **(Kuvio 9.)**.



Kuvio 9. Windows Phone –sovelluksen luokkakaavio

AppDelegate-luokassa rekisteröidään laite APNS:n kanssa ja talletetaan Device-Token-arvo. ViewController-luokka hoitaa graafisen käyttöliittymän toteutuksen sekä funktiot, jotka hoitavat kommunikoinnin palvelimen kanssa **(Kuvio 10.)**.



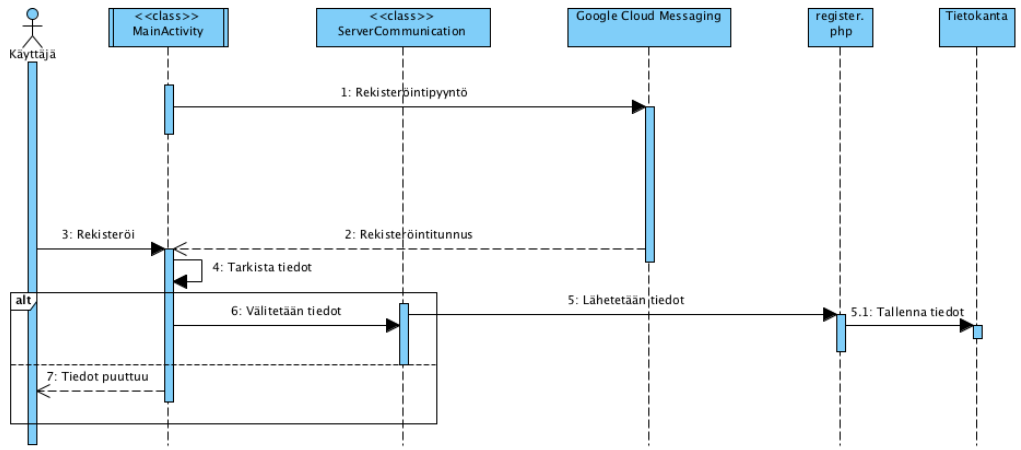
Kuvio 10. iOS-sovelluksen luokkakaavio

4.2.4 Sekvenssikaavio

Sekvenssikaavion avulla esitetään vuorovaikutus käyttäjien ja järjestelmän välillä. Kaaviosta nähdään, kuinka esimerkiksi käyttäjän syöttämät tiedot siirtyvät järjestelmässä. /17/

Sovellusten toimintaperiaate on hyvin samanlainen alustasta riippumatta. Eroavaisuuksina on luokkien ja funktioiden nimet. Windows Phone –sovelluksessa on suurin eroavaisuus, sillä sovelluksessa on yksi luokka enemmän (CreatePushChannel.cs), joka hoitaa rekisteröinnin MPNS:n kanssa. Käytetään esimerkkinä sekvenssikaavioita Android-sovelluksen toiminnoista.

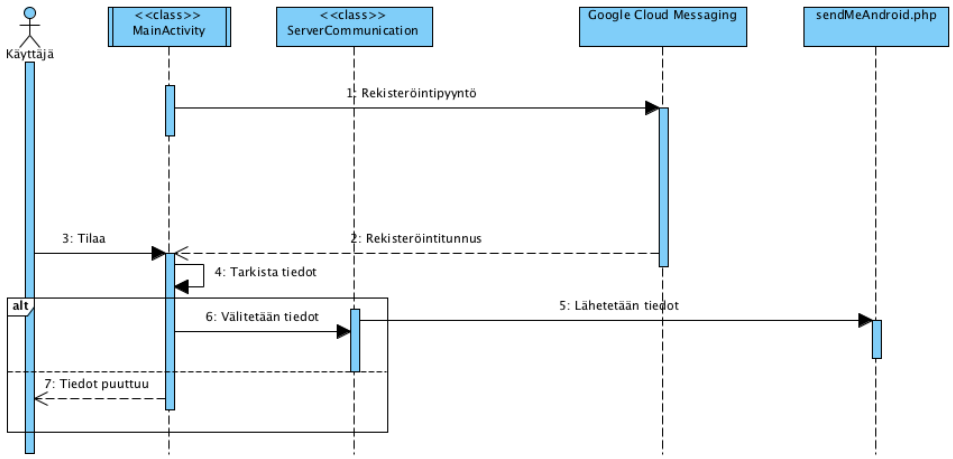
Sovellus käynnistetään ja samalla rekisteröidään laite Push -notifikaatiopalvelimelle. Käyttäjä syöttää käyttäjänimen ja painaa "Register for push notification" –nappia. Tarkistetaan onko kaikki tiedot täytetty. Jos tiedot puuttuvat, ilmoitetaan käyttäjälle asiasta. Kun tiedot on syötetty, lähetään tiedot palvelimelle **(Kuvio 11.)**.



Kuvio 11. Rekisteröintitietojen lähettäminen

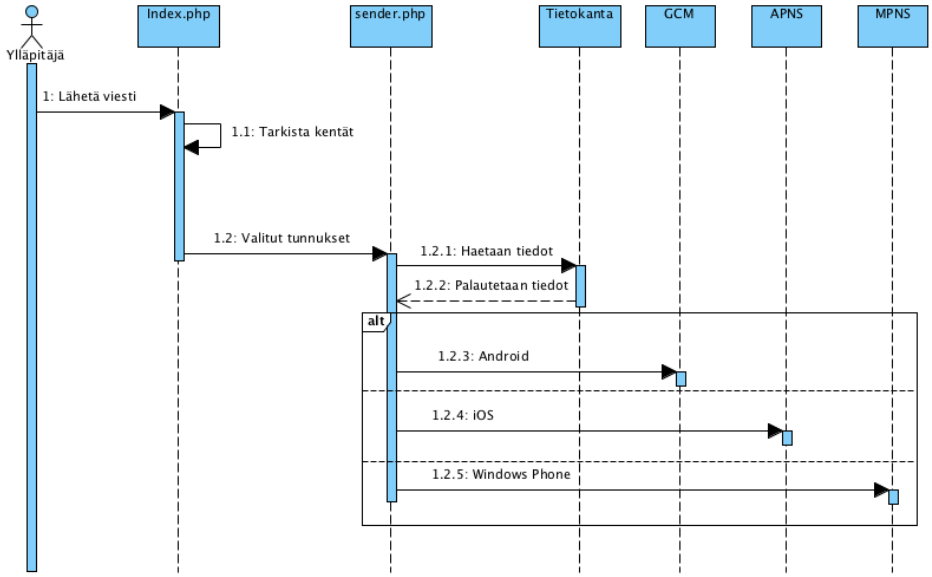
Käyttäjä pystyy tilaamaan itselleen Push-notifikaation. Laite hankkii rekisteröintitunnuksen GCM:ltä ja käyttäjän painaessa tilaus-painiketta tarkistetaan, onko käyttäjänimi syötetty. Käyttäjänimi ja rekisteröintitunnus välitetään ServerCommunication-luokalle, joka lähettää tiedot palvelimella olevalle tilausrajapinnalle.

Alustasta riippuen, kutsutaan palvelimella olevaa rajapintaa, joka hoitaa tilauksen **(Kuvio 12.)**. Tilausrajapinta välittää tiedot Push-notifikaatiopalvelimille, jotka hoitavat ilmoitusten välittämisen laitteille.



Kuvio 12. Push -notifikaation tilaaminen

Ylläpitäjän lähettäessä viestiä, tarkistetaan onko viesti-kenttä täytetty. Sender.php sisältää getInformation()-funktion, joka hakee tietokannasta valittujen laitteiden rekisteröintitunnukset ja laitetunnukset. Funktio kerää laitetunnuksien perusteella rekisteröintitunnukset oikeisiin listoihin ja välittää funktioille, jotka hoitavat lähetyksen. Lopuksi funktiot lähettävät viestin sisällön ja rekisteröintitunnukset eri alustoille tarkoitetuille Push-notifikaatiopalvelimille **(Kuvio 13.)**.

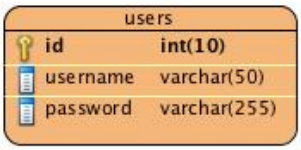


Kuvio 13. Lähettäminen ylläpitäjän sivustolta

5 TIETOKANNAN JA KÄYTTÖLIITTYMÄN SUUNNITTELU

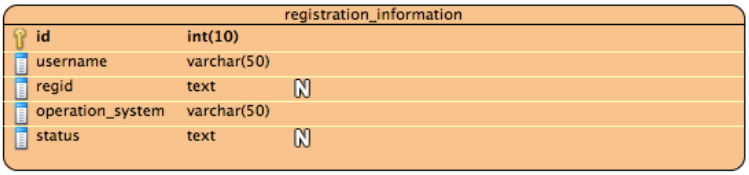
5.1 Tietokannan suunnittelu

Tietokannan tarkoituksena on tallentaa käyttäjien sekä ylläpitäjien tiedot. Käyttäjien taulukkoon tulee sarakkeet rekisteröintitunnukselle, käyttäjätunnukselle, käyttöjärjestelmän tunnukselle sekä tilaustunnukselle. Ylläpitäjien taulukoon tulee käyttäjänimi- ja salasana-sarakkeet. Ylläpitäjien tietojen avulla pystytään sallimaan käyttöoikeudet palvelimella olevalle ylläpitäjän sivustolle **(Kuvio 14.)**.



Kuvio 14. Käyttäjien tiedot

Laitteiden rekisteröintitunnukset tarvitaan, jotta pystytään jatkossa lähettämään laitteelle tietoja. Käyttäjätunnus ei ole välttämätön, mutta työssä käytetään käyttäjätunnusta selventämään listausta ylläpitäjälle. Käyttöjärjestelmän tunnuksella on suuri vastuu lähettämisessä. Tunnuksen avulla tullaan määrittämään viestin lähetystapa, sillä viestien välittämiseen käytetään eri tekniikoita riippuen alustasta. Tilaustunnus pitää tietoa siitä, onko sovelluksen käyttäjä halukas vastaanottamaan Push-notifikaatio **(Kuvio 15.)**.

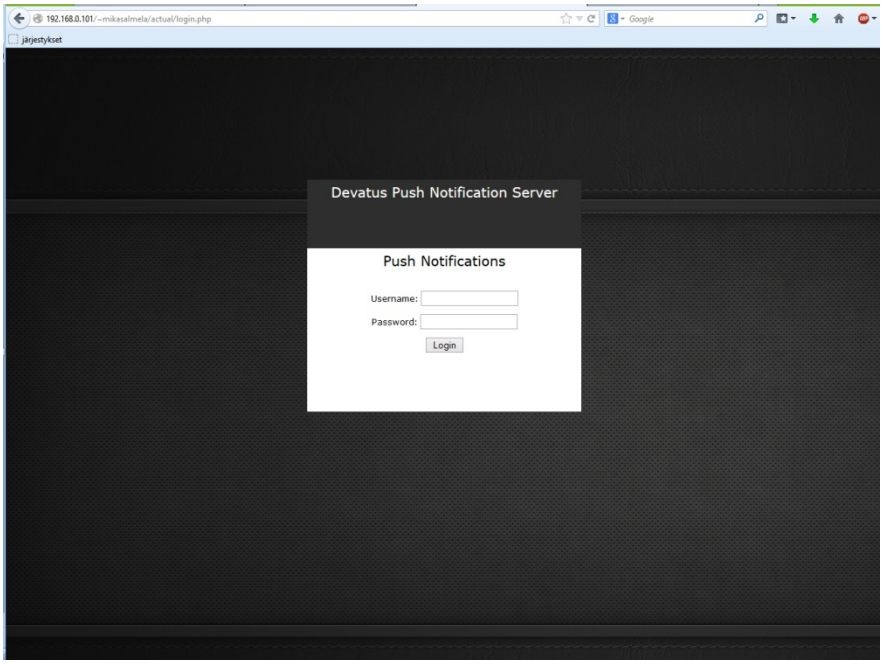


Kuvio 15. Käyttäjien rekisteröintitiedot

5.2 Käyttöliittymien eri osien suunnittelu

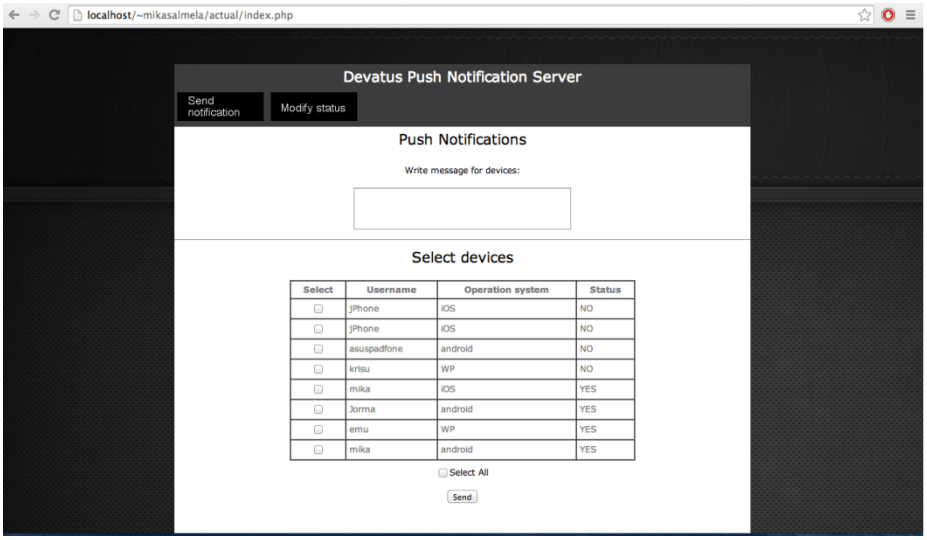
5.2.1 Ylläpitäjän sivusto

Sivustolla tulee olla kirjautuminen, jossa ylläpitäjää pyydetään syöttämään tiedot. Ylläpitäjälle näytetään kaksi tekstikenttää, käyttäjänimi ja salasana sekä kirjaudu-painike **(Kuvio 16.)**.



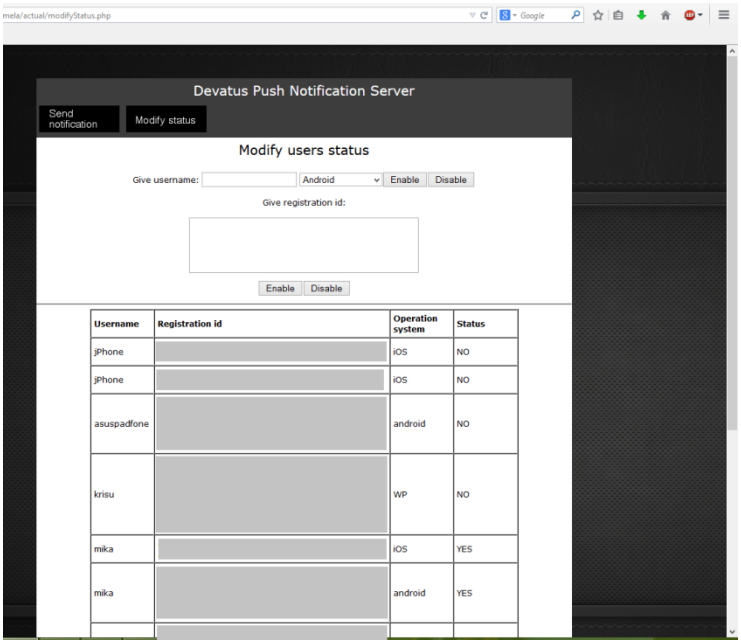
Kuvio 16. Ylläpitäjän kirjautuminen

Ylläpitäjän sivustolla on tekstikenttä, johon ylläpitäjä syöttää viestin sisällön. Taulukko, josta ylläpitäjä pystyy näkemään ja valitsemaan laitteita. Lähetä – painike, viestien lähettämiseen **(Kuvio 17.)**.



Kuvio 17. Push-notifikaation lähettäminen ylläpitäjän sivustolta

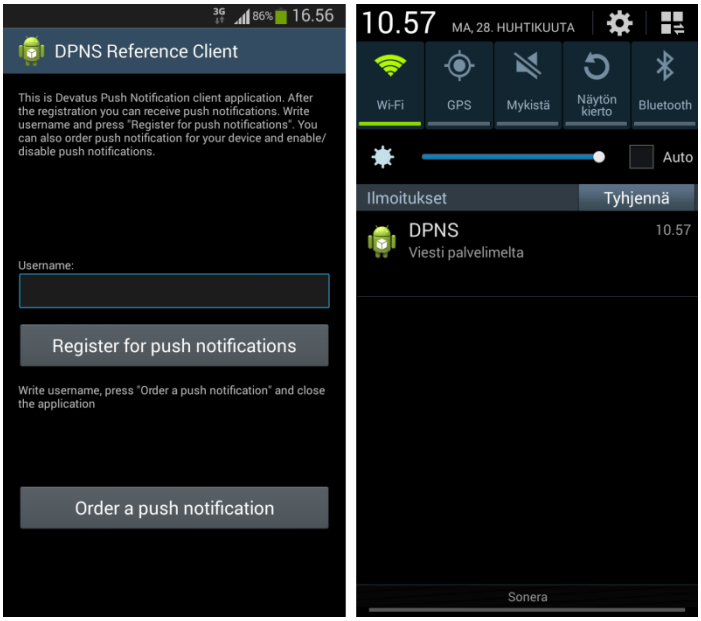
Ylläpitäjälle luodaan myös sivusto, josta pystytään muuttamaan tilaustunnusta. Muuttaminen tapahtuu, joko rekisteröintitunnuksen tai käyttäjänimen perusteella. Jos ylläpitäjä haluaa muuttaa tilaustunnusta käyttäjätunnuksen perusteella, tullaan kysymään ylläpitäjältä alustaa. Sivustolla on kaksi painiketta, joista pystytään poistamaan tai ottamaan käyttöön käyttäjälle Push-notifikaatiot **(Kuvio 18.)**.



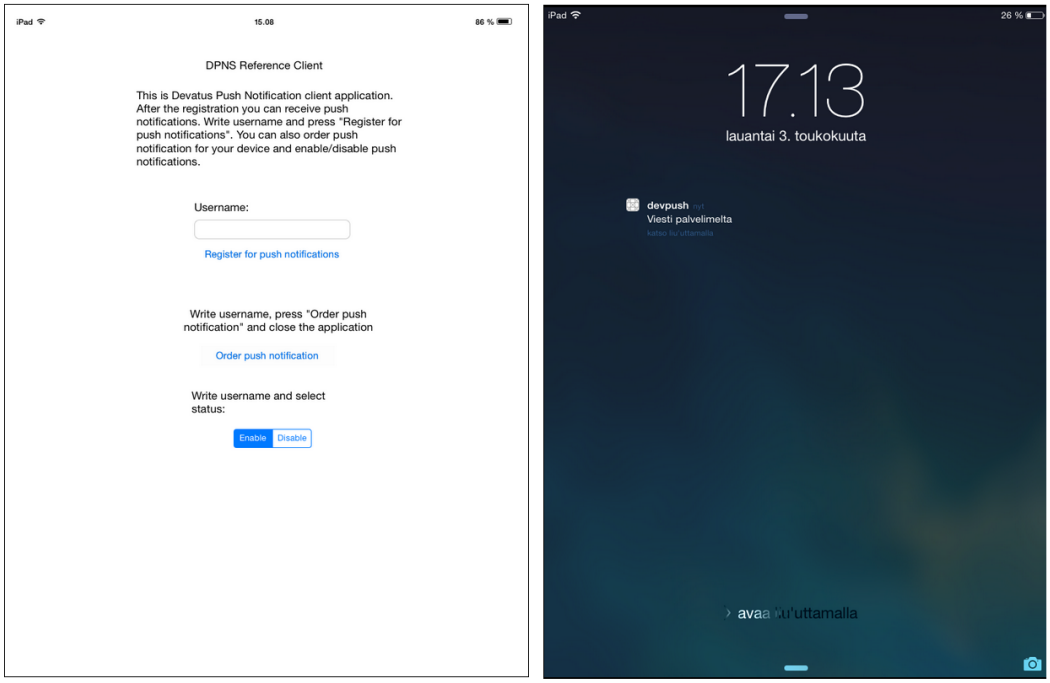
Kuvio 18. Tilaustunnuksen muuttaminen ylläpitäjän sivustolta

5.2.2 Sovellukset ja viestien näyttäminen

Jokaisen alustan sovellus tulee olemaan ulkoasultaan yksinkertainen. Tekstikenttä käyttäjänimeä varten ja toinen tekstikenttä viestin syöttämistä varten. Push-notifikaatiot tullaan näyttämään kunkin alustan ilmoituskeskuksessa. Kuvakkeina tullaan käyttämään kunkin sovelluksen vakiokuvakkeita. Kuvioista 19-21 nähdään vasemmalla reunalla sovelluksen ulkoasu ja oikealla ilmoituksen näyttäminen ilmoituskeskuksessa.



Kuvio 19. Android-sovellus



Kuvio 20. iOS-sovellus

Kuvio 21. Windows Phone -sovellus

6 SOVELLUSTEN TOTEUTUS

6.1 Yleistä toteutuksesta

Työssä toteutetaan sovellukset, joiden avulla pystytään rekisteröimään sovellusta käyttävä laite eri alustoille tarkoitettujen Push-notifikaatiopalvelinten kanssa. Sovelluksiin toteutetaan ominaisuudet, jotka mahdollistavat sovellusta käyttävän laitteen vastaanottamaan ylläpitäjän sivustolta lähetetty viesti. Lisäominaisuuksina sovelluksiin toteutetaan Push-notifikaation tilaaminen omalle laitteelle sekä mahdollisuus poistaa Push-notifikaatiot käytöstä.

6.2 Testisovellukset

6.2.1 Android

Android-alustalle on käytettävissä Google Cloud Messaging –palvelin, joka mahdollistaa viestien välittämisen. Sovelluksen tulee pystyä rekisteröitymään kyseisen palvelimen kanssa, jotta sovellusta käyttävälle laitteelle pystytään välittämään Push-notifikaatioita. Sovellukseen tulee sisällyttää Google Play Service –kirjastot, jotta on mahdollista toteuttaa Push-notifikaatiot.

Aluksi tulee luoda projekti Google API Consoleen ja aktivoida GCM-palvelimen käyttö. Projektin luonnin yhteydessä saadaan projektintunnus (SENDER_ID), jota käytetään sovelluksessa rekisteröityessä GCM:n kanssa. API Consolessa tulee myös generoida API-avain, joka määrittää mille sovellukselle Push-notificationit tullaan välittämään. Tunnukset yhdessä määrittelevät yhteyden GCM:n ja sovellusta käyttävän laitteen välille (ks. /4/).

Sovellukselle tullee antaa tiettyjä käyttöoikeuksia, jotta pystytään lähettämään tietoja omalle palvelimelle, rekisteröitymään ja vastaanottamaan viestejä GCM:ltä **(Kuvio 22.)**. Kutsutaan rekisteröintifunktiota sovelluksen käynnistyessä, joka lähettää pyynnön GCM-palvelimelle. GCM palauttaa rekisteröintitunnuksen sovellukselle. Käyttäjältä kysytään käyttäjänimi, jonka jälkeen on mahdollista painaa

rekisteröinti-nappia **(Kuvio 23.)**. Napin painalluksen jälkeen kutsutaan funktiota, joka tulee lähettämään tiedot palvelimelle **(Kuvio 24.)**. /4/.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.push.pushnotification"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />

    <application>
        <receiver
            android:name=".ReceiverGCM"
            android:permission="com.google.android.c2dm.permission.SEND" >
            <intent-filter>
                <action android:name="com.google.android.c2dm.intent.RECEIVE" />

                <category android:name="com.push.pushnotification" />
            </intent-filter>
        </receiver>

        <service android:name=".HandleGcmMessage" />
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.push.pushnotification.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Kuvio 22. Android-sovelluksen oikeudet

```
        //Rekisteröinti GCM:n kanssa asynkronisesti
        public String registerWithGCM() {
        new AsyncTask<Void, Void, String>() {
            @Override

            protected String doInBackground(Void... params) {
                try {
                      if (googlecloudmessaging == null) {
                              googlecloudmessaging =
                                      GoogleCloudMessaging.getInstance(context);
                    }
                      //rekisteröinti GCM:n kanssa, SENDER_ID:n avulla
                    regid = googlecloudmessaging.register(SENDER_ID);
                    } catch (IOException ex) {
```

```
                //Tulostetaan error-viesti
                        message = "Error :" + ex.getMessage();
                 }
               //Palautetaan rekisteröintitunnus
                 return regid;
             }
         }.execute(null, null, null);
                return regid;
}
```

Kuvio 23. Android-sovelluksen rekisteröinti GCM:n kanssa

```
//Funktio lähettää rekisteröintitiedot palvelimelle public void sendRegistra-
tionIdToServer(final Context context, final String username, final String regid) {
new AsyncTask<Void, Void, String>() {
            @Override
            protected String doInBackground(Void... params) {
        String url = "http://192.168.1.171/~mikasalmela/actual/register.php";
        try {
            //Yritetään saada luotua yhteys tietyn aikaa
            HttpParams httpParameters = new BasicHttpParams();
            HttpConnectionParams.setConnectionTimeout(httpParameters, 10000);
            HttpConnectionParams.setSoTimeout(httpParameters, 10000);
            //Tehdään HTTP yhteys
            HttpClient httpclient = new DefaultHttpClient(httpParameters);
            HttpPost httpPost = new HttpPost(url);
            //JSON tiedoista
            String json = "";
            JSONObject jsonObject = new JSONObject();
            jsonObject.accumulate("username", username);
            jsonObject.accumulate("reg_id", regid);
            jsonObject.accumulate("operation_system", "android");
            jsonObject.accumulate("status", "YES");
            json = jsonObject.toString();
            StringEntity se = new StringEntity(json);
            httpPost.setEntity(se);
            httpPost.setHeader("Accept", "application/json");
            httpPost.setHeader("Content-type", "application/json");
            //Lähetetään tiedot palvelimelle
            HttpResponse httpResponse = httpclient.execute(httpPost);
            //Vastauksen saanti
            InputStream in = httpResponse.getEntity().getContent();
            BufferedReader reader = new BufferedReader(new InputStreamReader(in));
               StringBuilder str = new StringBuilder();
               String line = null;
            while((line = reader.readLine()) != null){
                str.append(line + "\n");
            }
               in.close();
               result = str.toString();
               //Tulostetaan saatu vastaus logiin
               Log.d("Response: ",result);

        }catch (ConnectTimeoutException e)
            {
               //Jos ei pystytä luomaan yhteyttä tulostetaan virheilmoitus
               e.printStackTrace();
               result = "Connection failed";
               } catch (JSONException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
               } catch (ClientProtocolException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
               } catch (IOException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();

               }
               //Palautetaan saatu tieto
               return result;}
            protected void onPostExecute(String result) {
               if(result.equals(""))
               {
               //Tehdään toast-ilmoitus joka saadaan palvelimelta vastaukseksi
               result = "Connection timeout";
               }
                Toast.makeText(context, result, Toast.LENGTH_LONG).show();
            }

        }.execute(null, null, null);
}
```

Kuvio 24. Android-sovelluksen rekisteröintitietojen lähetys

ReceiverGCM-luokka hoitaa kuuntelun, jonka GCM pystyy herättämään. GCM herättää sovelluksen, jolloin sovellus käynnistyy taustalle. Tämän avulla pystytään estämään sovellusta menemästä lepotilaan viestin käsittelyn aikana **(Kuvio 25.)**. HandleGcmMessage-luokka toteuttaa toimenpiteet, mihin saapunutta viestiä tullaan käyttämään. /4/.

```
//Herättää sovelluksen kun viesti saapuu
public class ReceiverGCM extends WakefulBroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        //Pitää huolen, että HandleGcmMessage luokka hoitaa toimenpiteet
         ComponentName component = new        Component-
Name(context.getPackageName(),HandleGcmMessage.class.getName());
         //Käynnistetään palvelu ja pidetään laite päällä
         startWakefulService(context, (intent.setComponent(component)));
         setResultCode(MainActivity.RESULT_OK);
    }
}
```

Kuvio 25. Android-sovelluksen herätys

Vastaanotettua tietoa pystytään käyttämään mihin tarkoitukseen vain. Vastaanotetun tiedon sisältö sijoitetaan muuttujaan, jonka jälkeen se välitetään parametrina funktiolle, joka toteuttaa ilmoituksen **(Kuvio 26.)**. Sovelluksessa tullaan luomaan ilmoitus ilmoituskeskukseen. Saadun viestin sisältö sijoitetaan ilmoituksen sisältöön, jonka jälkeen ilmoitus tulee näkyviin ilmoituskeskukseen **(Kuvio 27.)**.

```
  //Hoitaa merkkijonon käsittelyn
  @Override
  protected void onHandleIntent(Intent intent) {
          //Vastaantotetaan merkkijono, joka saadaan GCM:ltä
          String extras = intent.getExtras().getString("message");
          //Välitetään saatu viesti ilmoituksen luonti funktiolle
          buildNotification(extras.toString());
  }
```

Kuvio 26. Android-sovelluksen herättäminen

```
//Luodaan ilmoitus
 private void buildNotification(String msg) {
     notificationManager = (NotificationManager)
             this.getSystemService(Context.NOTIFICATION_SERVICE);
     //Kun ilmoitus saapuu käytetään default-soittoääntä, jonka käyttäjä on
     asettanut laitteeseen
     Uri uri= RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
     //Toimenpide mitä tapahtuu kun ilmoitusta painetaan
     //Tässä sovelluksessa siirrytään MainActivityyn
      PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
                                    new Intent(this, MainActivity.class), 0);
      //Luodaan ilmoituksesta halutunlainen
      NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
      .setSound(uri)
      .setSmallIcon(R.drawable.ic_launcher)
      .setContentTitle("DPNS")
      .setStyle(new NotificationCompat.BigTextStyle()
      //Asetetaan saatu viesti ilmoituksen sisältöön
      .bigText(msg))
      .setContentText(msg);

      mBuilder.setContentIntent(contentIntent);
      notificationManager.notify(NOTIFICATION_ID++, mBuilder.build());
 }
```

Kuvio 27. Android-sovelluksen ilmoituksen toteutus

6.2.2 Windows Phone

Sovellus lähettää pyynnön Push Client Servicelle, joka hoitaa pyynnön eteenpäin MPNS:lle. MPNS palauttaa URIn sovellukselle, jota käytetään viestien lähettämisessä palvelimelta.

Kun sovellus käynnistetään, kutsutaan CreatePushChannel-luokassa olevaa CreatePushChannel-funktiota, joka luo Push Channel –yhteyden. Yhteyden nimi voidaan itse määrittää. Tämän avulla luodaan yhteys Push Client Servicen kanssa, joka ottaa yhteyden MPNS:ään. MPNS palauttaa URIn, jonka avulla laitteeseen pystytään lähettämään Push-notifikaatioita **(Kuvio 28.)**. /18/

```
        public string PushChannelCreator(CallBack cb)
         {
             callBack = cb;
             //´Tarkistetaan löytyykö jo kyseinen kanava
             pushNotificationChannel = HttpNotificationChannel.Find(channelName);

             // Jos kanavaa ei löydy, luodaan uusi yhteys push servicen kanssa
             if (pushNotificationChannel == null)
             {
                 pushNotificationChannel= new HttpNotificationChannel(channelName);
                 //Rekisteröidään kaikki eventit ennen kuin yritetään avata kanava
                 pushNotificationChannel.ChannelUriUpdated += new
                                         EventHandler<NotificationChannelUriEventArgs>
                                         (PushChannel_ChannelUriUpdated);
                //Avataan kanava
                 pushNotificationChannel.Open();
                 //Sidotaan uusi kanava toast-eventin kanssa
                 pushNotificationChannel.BindToShellToast();
             }
             else
             {
                 //Kanava jo avattu, rekisteröidään eventille
                 pushNotificationChannel.ChannelUriUpdated += new
                                         EventHandler<NotificationChannelUriEventArgs>
                                         (PushChannel_ChannelUriUpdated);
                //Rekisteröintitunnus
                regid = pushNotificationChannel.ChannelUri.ToString();
             }
                //Palautetaan rekisteröintitunnus
                return regid;
         }

         void PushChannel_ChannelUriUpdated(object sender,
                NotificationChannelUriEventArgs e)
         {
             Dispatcher.BeginInvoke(() =>
             {
                 //Jos saadaan uusi rekisteröintitunnus
                 regid = e.ChannelUri.ToString();
                                 //Palautetaan rekisteröintitunnus
                 callBack(regid);
             });
         }
     }
}
```

Kuvio 28. Windows Phonen rekisteröinti

Sovelluksessa tiedot lähetetään palvelimella olevalle rekisteröintirajapinnalle, joka vastaanottaa saadut tiedot **(Kuvio 29.)**. Sovellukseen ilmoituksen saapuessa toteutetaan Toast-ilmoitus. Windows Phonessa on palvelu, joka hoitaa vastaanottamisen ja toteuttaa ilmoituksen saaduista tiedoista. Tämän ansiosta sovelluksessa erikseen ei tarvitse toteuttaa vastaanottamista.

```
      public void PostRegistrationInformation(string _regid, string _username)
        {
               //Otetaan talteen vaadittavat tiedot
               username = _username;
               regid = _regid;
               //Pyyntö rajappinnalle joka hoitaa rekisteröinnin
               request = (HttpWebRequest)WebRequest.Create(
                      "http://192.168.1.171/~mikasalmela/actual/register.php");
               request.ContentType = "text/json";
               request.Method = "POST";
               //Yritetään saada vastaus
               request.BeginGetRequestStream(new
                             AsyncCallback(PostRegistrationInformationToServer),
                                                                    request);
        }

        private void PostRegistrationInformationToServer(IAsyncResult      asyn-
chronousResult)
        {
            request = (HttpWebRequest)asynchronousResult.AsyncState;
            postStream = request.EndGetRequestStream(asynchronousResult);
            //Asetetaan välitettävät tiedot
            string postData = "{\"reg_id\":\"" + regid + "\"" + "," +
                              "\"username\":\"" + username + "\"" + "," +
                              "\"operation_system\":\""+ operation_system +"\"" +
                              "," + "\"status\":\"YES\"}";
            byte[] postValueArray = Encoding.UTF8.GetBytes(postData);
            //Kirjoitetaan request streamiin
            postStream.Write(postValueArray, 0, postData.Length);
            postStream.Close();
            request.BeginGetResponse(new AsyncCallback
                                          (GetResponseCallBack),request);
        }
        //Käisttelee lähetyksen ja näyttää palvelimelta saadun vastauksen
        private void GetResponseCallBack(IAsyncResult asynchronousResult)
        {
            try{
            request = (HttpWebRequest)asynchronousResult.AsyncState;
            response = (HttpWebResponse)
                        request.EndGetResponse(asynchronousResult);
            streamResponse = response.GetResponseStream();
            streamRead = new StreamReader(streamResponse);
            string read = streamRead.ReadToEnd();
                       //Tulostetaan saatu vastaus
            Debug.WriteLine("Response : " + read);
                       //Ilmoitetaan saatu vastaus käyttäjälle
            Dispatcher.BeginInvoke(() => { MessageBox.Show(read); });
            streamResponse.Close();
            streamRead.Close();
            response.Close();
            }
            catch (Exception ex)
            {
               //Ilmoitetaan käyttäjälle jos ei saada yhteyttä palvelimeen
                Dispatcher.BeginInvoke(() => { MessageBox.Show("error"); });
            }
        }
```

Kuvio 29. Windows Phone, rekisteröintitietojen lähettäminen

6.2.3 iOS

Kun toteutetaan sovellusta iOS:lle, tulee olla tietyt sertifikaatit ja vaadittavat asetukset Applen Dev Centerissä, jotta sovellukseen pystytään lisäämään Push-notifikaatiot (ks. /8/.). Sovelluksen käynnistyessä tulee ottaa yhteys APNS-palvelimen kanssa, jolta saadaan deviceToken-arvo **(Kuvio 30.)**. Arvoa tullaan käyttämään viestien lähettämiseen palvelimelta. APNS vastaa rekisteröintipyyntöön ja generoi deviceToken-arvon uniikista laitesertifikaatista. Arvo sisältää laitteen tunnisteen. Tämän jälkeen APNS purkaa arvon laiteavaimen avulla ja palauttaa sen laitteelle. DeviceToken-arvo joudutaan jäsentämään uudelleen, sillä vastaanotetussa arvossa on välilyönnit tietyn väliajoin. Kun käyttäjä painaa rekisteröinti-nappia, tullaan lähettämään tiedot palvelimelle **(Kuvio 31.)**. /8/ /9/

```
//Sovelluksen käynnistyessä rekisteröidään APNS-palvelimen kanssa
- (void)application:(UIApplication*)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData*)deviceToken
{
      //Tulstetaan deviceToken-arvo logiin
      NSLog(@"My token is: %@", deviceToken);
      //Sijoitetaan saatu arvo token muuttujaan jota pystytään käyttämään jatkossa
      token = deviceToken;
}

// Ongelmia deviceToken –arvon saamisessa
- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
{
      //Tulostetaan virhe
      NSLog(@"Failed to get token, error: %@", error);
}
```

Kuvio 30. iOS-laitteen rekisteröinti

```
//Funktio hoitaa rekisteröintitietojen lähettämisen palvelimelle
- (void) sendRegister
{
    NSMutableURLRequest *request = [[NSMutableURLRequest alloc] initWithURL:[NSURL
URLWithString:@"http://192.168.1.171/~mikasalmela/actual/register.php"]];
    // Luodaan POST pyyntö palvelimelle
    [request setHTTPMethod:@"POST"];
    [request addValue:@"postValues" forHTTPHeaderField:@"METHOD"];
    NSString *username = self.usernameTextBox.text;
    // Otetaan token arvo app delegatesta
    AppDelegate *appDelegate = (AppDelegate *)[UIApplication sharedApplica-
tion].delegate;
    //Parsitaan token arvosta ylimääräiset välilyönnit pois
    NSString *tokenString = [[[appDelegate.token description]
                              stringByTrimmingCharactersInSet:[NSCharacterSet
                              characterSetWithCharactersInString:@"<>"]]
                              stringByReplacingOccurrencesOfString:@" "
```

```
                                withString:@""];
    //Kerätään tiedot, jotka lähetetään palvelimelle
    NSMutableDictionary *dictionary = [[NSMutableDictionary alloc] init];
    [dictionary setValue:tokenString forKey:@"reg_id"];
    [dictionary setValue:@"iOS" forKey:@"operation_system"];
    [dictionary setValue:username forKey:@"username"];
    [dictionary setValue:@"YES" forKey:@"status"];

    //Käytetään jsonia
    NSData *postdata = [NSJSONSerialization dataWithJSONObject:dictionary
                        options:0 error:nil];

    NSLog([[NSString alloc] initWithData:postdata encoding:NSUTF8StringEncoding]);
    //Asetetaan data bodyyn
    [request setHTTPBody:postdata];
    [request addValue:[NSString stringWithFormat:@"%d",postdata.length]
forHTTPHeaderField:@"Content-Length"];
    [request setTimeoutInterval:20.0];

    [NSURLConnection sendAsynchronousRequest:request queue:[[NSOperationQueue al-
loc] init] completionHandler:^(NSURLResponse *response, NSData *data, NSError
*error){
        if (data){
               //Tulostetaan vastaanotetut tiedot logiin
               dataNSLog([[NSString alloc] initWithData:data
                       encoding:NSUTF8StringEncoding]);
        }
        else if (error){
            NSLog(@"%@",error);
            dispatch_async(dispatch_get_main_queue(), ^{
                [self alertBuilder:@"Connection failed"];
            });
        }
    }];
}
```

Kuvio 31. iOS-sovelluksen tietojen lähettäminen

Jokaiselle alustalle tullaan toteuttamaan toiminnot, jotka mahdollistavat Push-notifikaation tilaamisen itselleen sekä käyttäjä pystyy ottamaan käyttöön tai poistamaan käytöstä ilmoitukset. Ominaisuudet toteutetaan samoin kuin funktiot, jotka hoitavat rekisteröintitietojen lähettämisen **(Kuviot 24. ja 29. ja 31.)**. Kutsutaan palvelimella olevaa rajapintaa, jolle lähetetään tarvittavat tiedot. Esimerkiksi Android-sovellukseen toteutettu Push-notifikaation tilaaminen **(Kuvio 32.)** sekä tilaustunnuksen muuttaminen **(Kuvio 33.)**. Muuttaminen suoritetaan "Order push notification" -painiketta painaessa.

```
//Tilausrajapinnan kutsu
String url = "http://192.168.1.171/~mikasalmela/actual/sendmeAndroid.php";
.
//Lähetettävät tiedot
String json = "";
JSONObject jsonObject = new JSONObject();
jsonObject.accumulate("reg_id", regid);
jsonObject.accumulate("username", username);
json = jsonObject.toString();
.
```

Kuvio 32. Push-notifikaation tilaaminen

```
//Tilausrajapinnan kutsu
String url = "http://192.168.1.171/~mikasalmela/actual/status.php";
.
//Lähetettävät tiedot
String json = "";
JSONObject jsonObject = new JSONObject();
jsonObject.accumulate("username", username);
jsonObject.accumulate("reg_id", regid);
jsonObject.accumulate("status", status);
json = jsonObject.toString();
.
```

Kuvio 33. Tilaustunnuksen muuttaminen.

6.3 Ylläpitäjän sivusto

Ylläpitäjän sivuston tärkeimpänä tehtävänä on tulostaa taulukko kaikista laitteista, antaa ylläpitäjälle mahdollisuus valita listasta laitteet, joille halutaan viestin lähetettävän, tekstikenttä viestiä varten sekä lähetä-painike. Ylläpitäjän sivusto toteutetaan PHP-ohjelmointikielellä, joka on upotettu HTML:n sekaan.

Listataan kaikki MySQL-tietokannassa olevat laitteet taulukkoon ja sijoitetaan taulukon ensimmäiseen sarakkeeseen valintaruutu, jokaisen rekisteröityneen laitteen kohdalle. Valintaruudulle sijoitetaan tietokannasta saatu tunnuksen id-arvo. Ylläpitäjän painaessa lähetä-painiketta, välitetään parametrina id:eiden arvot funktiolle, joka hakee tietokannasta valittujen laitteiden tiedot **(Kuvio 34.)**.

```
<?php

//Sisällytetään sender.php tiedosto, joka sisältää tietojen keruun ja viestin
lähettämisen
    require_once('sender.php');

    //Jos painetaan lähetä-painiketta
    if (isset($_POST['Button'])) {
        //Tarkistetaan puuttuuko tietoja
        if(empty($_POST['message']))
            {
                //Toteutetaan javascriptillä alert-viesti
                echo "<script type='text/javascript'>checkMessage();</script>";
            }
             else{
                        //Lista id:tä varten
                        $send_id = array();
                        if(empty($_POST['id']))
                        {
                          echo "<script
                                type='text/javascript'>checkDevices();</script>";
                        }

                    //Käydään läpi kaikki valitut id:t
                    foreach($_POST['id'] as $db_id)
                    {
                        $send_id[] = $db_id;
```

```
                }
                $message = $_POST['message'];
                //Välitetään valitut id:t ja viestin sisältö parametrina
                funktiolle
                getInformation($send_id, $message);
            }
    }
?>
```

Kuvio 34. Ylläpitäjän sivusto, laitteiden valitseminen

Sender.php -tiedostossa on funktio getInformation(), joka hakee rekisteröintitunnukset ja niille kuuluvat laitetunnukset tietokannasta. Funktio hoitaa tarkistuksen, mikä alusta on kyseessä ja sen perusteella välittää listat rekisteröintitunnuksista funktioille, jotka hoitavat lähetyksen **(Kuvio 35.)**.

```
//Haetaan rekisteröintitunnuksia vastaavat tiedot tietokannasta
function getInformation($send_id, $message)
    {
        $android_regids = array();
        $ios_regids = array();
        $wp_regids = array();
        //Yhteys tietokantaan
        $con=mysqli_connect("localhost","root","root","pushnotification");
        //Käydään läpi kaikki valitut rekisteröintitunnukset
        for($i=0; $i<count($send_id); ++$i)
        {

            //Haetaan tietokannasta tiedot
            $result = mysqli_query($con,"SELECT * FROM registration_information
                    WHERE id='$send_id[$i]'");
            //Käydään läpi kaikki tulokset
            while($row = mysqli_fetch_array($result))
            {
                //Tarkistetaan mikä alusta kyseessä ja tarkistetaan tilaustunnus
                if($row['operation_system'] == 'android')
                {
                    if($row['status'] != 'NO')
                    {
                        $android_regids[] = $row['regid'];
                    }
                }
                else if($row['operation_system']== 'iOS')
                {
                    if($row['status'] != 'NO')
                    {
                        $ios_regids[] = $row['regid'];
                    }
                }
                else if($row['operation_system'] == 'WP')
                {
                    if($row['status'] != 'NO')
                    {
                        $wp_regids[] = $row['regid'];
                    }
                }
            }
        }
        //Suljetaan yhteys tietokantaan
        mysqli_close($con);
        //Välitetään parametreina listat rekisteröintitunnuksista
        send_android_pushnotification($android_regids, $message);
        send_wp_pushnotification($wp_regids, $message);
        send_ios_pushnotification($ios_regids, $message);
    }
```

Kuvio 35. Valittujen laitteiden haku tietokannasta

Jokaiselle alustalle joudutaan toteuttamaan erilaiset lähetystavat, sillä jokaisen alustan Push-notifikaatiopalvelin käyttää hieman erilaista tekniikkaa, jolla pystytään ottamaan yhteyttä ja välittämään tietoja kyseisille palvelimille.

PHP tukee libcurl-kirjastoa, jonka avulla pystytään kommunikoimaan palvelimien kanssa. Libcurl tukee mm. http- sekä https-protokollia. Tätä kirjastoa käytetään hyväksi kun toteutetaan funktiot, joiden avulla lähetetään Windows Phone ja Android laitteille Push-notifikaatioita. Androidille lähetettäessä tullaan käyttämään luotua API-avainta, jotta pystytään olemaan varmoja mille sovellukselle Push-notifikaatiot lähetetään. Kaikki rekisteröintitunnukset lähetetään yhtenä listana GCM:lle, joka osaa itse käydä kaikki rekisteröintitunnukset läpi. API-avainta ei tule näyttää turvallisuussyistä. **(Kuvio 36.)**. /3/ /15/

```
function send_android_pushnotification($android_regids, $message) {
            if(!empty($android_regids)){
                //Määritellään API-avain, jonka avulla GCM tietää mille
                //projektille lähettää tiedot
                define("GOOGLE_API_KEY", "pidettävä salassa");
                //GCM osoite lähettämistä varten
                define("GOOGLE_GCM_URL",
                "https://android.googleapis.com/gcm/send");
                $gcm_regids =array_unique($android_regids);
                //Lisätään kaikki rekisteröintitunnukset listaan ja välitetään
               //lista GCM:lle. GCM välittää viestit eteenpäin.
                $sendingfields = array(
                                        'registration_ids' => $gcm_regids,
                                        'data' => array("message" => $message),
                                        );
                $headers = array(
                                'Authorization: key=' . GOOGLE_API_KEY,
                                'Content-Type: application/json'
                                );

                //Asetetaan curl yhteys
                $ch = curl_init();
                curl_setopt($ch, CURLOPT_URL, GOOGLE_GCM_URL);
                curl_setopt($ch, CURLOPT_POST, true);
                curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
                curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
                curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
                curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($sendingfields));

                //Suoritetaan pyyntö
                $result = curl_exec($ch);
                //Jos ongelmia niin lopetetaan prosessi.
                if ($result === FALSE)
                {
                    die('Problem occurred: ' . curl_error($ch));
                }
                //Suljetaan yhteys
                curl_close($ch);
            }
        }
```

Kuvio 36. Android-laitteille lähettäminen

Windows Phone –laitteille lähettäminen tapahtuu melkein samoin kuin Android-laitteille. Windows Phonelle lähetettäessä ei tarvitse välttämättä käyttää tunnistetta, jonka avulla web-palvelimen ja MPNS:n yhteys todennettaisiin. MPNS-palvelimelle tullaan lähettämään yksitellen rekisteröintitunnukset, jotka ylläpitäjä on valinnut **(Kuvio 37.)**. /18/

```
function send_wp_pushnotification($wp_regids, $message){
    if(!empty($wp_regids))
    {
         // Tarkistetaan onko samoja rekisteröintitunnuksia
        $mpns_regids =array_unique($wp_regids);
        //Toast-ilmoituksen elementti, johon sijoitetaan ylläpitäjän
       //kirjoittama viesti. Windows Phone luo tämän perusteella ilmoituksen
        $toastMessageElements = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" .
                "<wp:Notification xmlns:wp=\"WPNotification\">" .
                "<wp:Toast>" .
                "<wp:Text1>" . "DPNS" . "</wp:Text1>" .
                "<wp:Text2>" . $message . "</wp:Text2>" .
                "</wp:Toast> " .
                "</wp:Notification>";
         //Käydään läpi kaikki rekisteröintitunnukset ja lähetetään ilmoitus
        for($i=0; $i<count($mpns_regids); ++$i)
        {
            //Asetetaan curl yhteys
            $ch = curl_init();
            //Rekisteröintitunnus toimii curl:ssa osoitteena
            curl_setopt($ch, CURLOPT_URL, $mpns_regids[$i]);
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
            curl_setopt($ch, CURLOPT_POST, true);
            curl_setopt($ch, CURLOPT_HEADER, true);

            // Lisätään header
            $httpHeaders=array('Content-type: text/xml; charset=utf-8',
                                        'X-WindowsPhone-Target: toast',
                                        'Accept: application/*',
                                        'X-NotificationClass: 2',
                                        'Content-Length:'.strlen
                                                ($toastMessageElements));
            curl_setopt($ch, CURLOPT_HTTPHEADER, $httpHeaders);
            //lisätään elementti
            curl_setopt($ch, CURLOPT_POSTFIELDS, $toastMessageElements);
            //Suoritetaan toimenpide
            $output = curl_exec($ch);
            //Suljetaan yhteys
            curl_close($ch);
            //echo $output;
        }
    }
}
```

Kuvio 37. Windows Phone -laitteille lähettäminen

APNS:lle lähetettäessä tietoja, joudutaan käyttämään sertifikaatteja yhteyden luomiseen. Toteutustapa tulee olemaan erilainen kuin Android- ja Windows Phone –laitteille. Yhteyden luominen APNS:n ja palvelimen kanssa, tapahtuu PHP:ssa olevan stream_socket_client -funktion avulla **(Kuvio 38.)**. Lähetyksessä käytetään SSL-sertifikaatista, private keystä sekä CSR:stä luotua ”ck.pem” –

tiedostoa. Tiedosto sisältää vaadittavat sertifikaatit, joiden avulla pystytään luomaan yhteys APNS:n kanssa (ks. /8/). ”Passphrase” –muuttuja sisältää salasanan, jotta ”ck.pem” –tiedostoa pystytään käyttämään. Nämä tiedot tulee pitää turvallisuussyistä piilotettuna. /8/ /16/

```
function send_ios_pushnotification($ios_regids, $message){
    if(!empty($ios_regids)){
        // Avaimen salasana
        $passphrase ='salainen';

        $context = stream_context_create();
        stream_context_set_option($context, 'ssl', 'local_cert', 'ck.pem');
        stream_context_set_option($context, 'ssl', 'passphrase', $passphrase);
        //Tarkistetaan onko samoja rekisteröintitunnuksia
        $apns_regids =array_unique($ios_regids);
        // Avataan yhteys APNS:lle.
        // SSL-security yhtey, joten tarvitaan sertifikaatit
        // ck.pem sisältää sertifikaatit ja passhphrase sisältää salasanan
        // jolla pystytään purkamaan. Luodaan yhteys
        //gateway.sandbox.push.apple.com is url for Developer to APNS
        $fp = stream_socket_client(
                  'ssl://gateway.sandbox.push.apple.com:2195', $err,
                  $errstr, 60, STREAM_CLIENT_CONNECT|STREAM_CLIENT_PERSISTENT,
                  $context);

         //Käydään läpi kaikki rekisteröintitunnukset
        for ($i = 0; $i < count($apns_regids); ++$i) {

            // Luodaan payload body
            $body['aps'] = array(
                                'alert' => $message,
                                'sound' => 'default'
                                );
            // Enkoodataan payload JSON:ksi
            $payload = json_encode($body);
            // Luodaan binääri ilmoituksesta
           $binarynotification = chr(0) . pack('n', 32) . pack('H*',
           $apns_regids[$i]) . pack('n', strlen($payload)) . $payload;

            // Lähetetään APNS:lle
            $apns_result = fwrite($fp, $binarynotification,
            strlen($binarynotification));

        }
        // Suljetaan yhteys
        fclose($fp);
    }
}
```

Kuvio 38. iOS-laitteille lähettäminen

6.4 Palvelimen toiminnot

6.4.1 Laitteen rekisteröintitietojen tallentaminen

Palvelimelle luodaan rajapinta, jonka avulla pystytään tallentamaan tiedot tietokantaan. Rajapinta ottaa vastaan tiedot, jotka sovelluksesta tullaan lähettämään.

Saadut tiedot sijoitetaan muuttujiin jatkotoimenpiteitä varten. Avataan yhteys tietokantaan ja valitaan kaikki tietokannassa olevat rekisteröintitunnukset. Tarkistetaan, onko tietokannassa jo kyseiset tiedot. Jos tietokannasta ei löydy kyseistä tietoa, tallennetaan vastaanotetut tiedot tietokantaan ja suljetaan yhteys. **(Kuvio 39.)**.

```
<?php
    //Vastaanotetaan JSON-tiedot
    $jsonString = file_get_contents('php://input');
    $jsonArray = json_decode($jsonString, true);

    //Otetaan vastaan tiedot muuttujiin
    $regid = $jsonArray['reg_id'];
    $username = $jsonArray['username'];
    $system = $jsonArray['operation_system'];
    $status = $jsonArray['status'];

    //Tietokantayhteyden luonti
    $con=mysqli_connect("localhost","root","root","pushnotification");
    //Tarkastetaan onko saatu rekisteröintitunnus tyhjä
    if(empty($regid))
    {
        echo "Registration failed";
    }
    else{
      //Valitaan löytyykö tietokannasta jo tiedot
      $result = mysqli_query($con,"SELECT username, regid FROM
      registration_information WHERE regid='$regid' AND username='$username'");

    //Jos ei niin lisätään tietokantaa
    if(mysqli_num_rows($result) == 0)
        {
            $sql="INSERT INTO registration_information (username, regid,
               operation_system, status) VALUES ('$username', '$regid', '$system',
               '$status')";
            mysqli_query($con, $sql);
            echo "Registration success";
        }
    else
    {
        //Viesti, joka mahdollista vastaanottaa sovelluksessa
        echo "Registration information already exists";
    }
    //Suljetaan yhteys
    mysqli_close($con);
    }
?>
```

Kuvio 39. Rekisteröintitietojen tallentaminen

6.4.2 Tilaustunnusten muuttaminen

Sovelluksissa on mahdollisuudet tilata Push -notifikaatioita ja muuttaa tilaustunnusta. Palvelimelle tulee toteuttaa rajapinta, joka mahdollistaa nämä toimenpiteet. Kun käyttäjä haluaa muuttaa tilaustunnustaan, välitetään tiedot palvelimella olevalle rajapinnalle, joka hoitaa tilaustunnuksen muuttamisen **(Kuvio 40.)**.

```
<?php
      //Vastaanotetaan JSON-tiedot
      $jsonString = file_get_contents('php://input');
      $jsonArray = json_decode($jsonString, true);

      //Otetaan vastaan tiedot muuttujiin
      $username = $jsonArray['username'];
      $regid = $jsonArray['regid'];
      $status = $jsonArray['status'];


      $con=mysqli_connect("localhost","root","root","pushnotification");
      //Päivitetään käyttäjänimeä vastaavan rekisteröintitunnuksen tilaa
      mysqli_query($con,"UPDATE registration_information SET status='$status' WHERE
      regid='$regid' AND username='$username'");
      echo "Status updated";
      //Suljetaan yhteys
      mysqli_close($con);
?>
```

Kuvio 40. Tilaustunnusten muuttaminen

Kun käyttäjä tilaa itselleen Push-notifikaation, toteutus tapahtuu samoin kuin aikaisemmin mainitut lähetys-funktiot **(Kuviot 36. ja 37. ja 38.)**. Funktioista on toteutettu jokaista alustaa kohden omat PHP-tiedostot, joissa otetaan vastaan käyttäjänimi ja rekisteröintitunnus. Tämän jälkeen tullaan välittämään tiedot parametreina lähetys-funktioille, jotka hoitavat lähetyksen.

7 TOIMINTOJEN TESTAUS

Sovelluksia testattiin toteutuksen aikana aina, kun jokin toiminto saatiin valmiiksi. Kun sovelluksen tärkeimmät toiminnot olivat valmiina, pystyttiin testaamaan myös Push-notifikaation lähettämistä ylläpitäjän sivustolta. Toteutuksen jälkeen toteutettiin kolme testitapausta, joiden avulla pystytään varmistamaan tärkeimpien toimintojen onnistuvuus sekä mahdolliset virheet **(Taulukko 2.)**.

Taulukko 2. Testitapaukset

ID	Testitapaukset	Odotettu tulos	Tulos
A1	Rekisteröintitietojen lähettäminen palvelimelle sovelluksesta	Laite saa rekisteröintitunnuksen ja tiedot tallentuvat tietokantaan	OK
A2	Ylläpitäjän sivustolta pystytään lähettämään alustasta riippumatta, monelle laitteelle Push-notifikaatio	Kaikki laitteet vastaanottavat ilmoituksen	OK
A3	Sovelluksista pystytään tilaamaan Push-notifikaation omaan laitteeseen.	Kaikki vastaanottavat ilmoituksen.	OK

8 YHTEENVETO

Opinnäytetyön tuloksena saatiin toimiva ympäristö, joka mahdollistaa Push-notifikaatioiden lähettämisen sekä vastaanottamisen sovellusta käyttävällä laitteella. Tärkeimmät toiminnot saatiin toteutettua, joita ovat mm. laitteen rekisteröinti, tietojen välittäminen sovelluksista palvelimelle, Push-notifikaatioiden lähettäminen ylläpitäjän sivustolta sekä Push-notifikaation vastaanottaminen sovellusta käyttävällä laitteella.

Työ oli sinänsä haastava, koska toteutuksen myötä tuli opetella monia uusia työkaluja, teknologioita sekä ohjelmointikieliä mm. PHP ja Objective-C. Työn alussa tuntui menevän liian paljon aikaa pelkästään työkalujen ja teknologioiden tutkimiseen. Loppujen lopuksi voidaan todeta, että työ oli todella mielenkiintoinen ja opetti paljon uutta, josta tulee varmasti olemaan hyötyä tulevaisuudessa.

Tulevaisuutta ajatellen, sovellusta tulisi vielä kehittää paremmaksi. Toteutuksen aikana testattiin ympäristöä aina, kun saatiin jokin toiminto valmiiksi. Tämän myötä tuli huomattua, että ympäristöä tulisi kehittää vakaammaksi sekä keskittyä viestien välittämisen nopeuteen. Sillä työssä havaittiin ongelmia lähettämisessä ylläpitäjän sivustolta, kun laitteita oli suurempia määriä. Ympäristön vakaus tuntui riippuvan päivästä, joinakin päivinä kaikki Push-notifikaatiot eivät välittyneet laitteille ja hetken odottelun jälkeen toistettiin toimenpiteet onnistuneesti.

LÄHTEET

Kirjallisuuslähteet

/1/ Argenta, M. 2011. Learning Android. Sebastopol. O'Reilly Media, Inc.

/2/ Whitechapel, A. 2012. Windows Phone 7 Development Internals. Sebastopol. O'Reilly Media, Inc.

Verkkolähteet

/3/ Android Developer. GCM HTTP Connection Server.
Viitattu 6.3.2014.
http://developer.android.com/google/gcm/http.html

/4/ Android Developer. Implementing GCM Client.
Viitattu 6.3.2014.
http://developer.android.com/google/gcm/client.html

/5/ Apache. HTTP Server Project.
Viitattu 20.4.2014.
http://httpd.apache.org/

/6/ Apple Developer Libary. iOS Technology Overview.
Viitattu 4.5.2014.
https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/Introduction/Introduction.html

/7/ Devatus Oy.
Viitattu 6.5.2014.
http://devatus.fi/index.html

/8/ iOS Developer Library. Provisioning and Development.
Viitattu 10.4.2014.
https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ProvisioningDevelopment.html#//apple_ref/doc/uid/TP40008194-CH104-SW1

/9/ iOS Developer Library. Scheduling, Registering, and Handling Notifications
Viitattu 25.4.2014.
https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/IPhoneOSClientImp.html#//apple_ref/doc/uid/TP40008194-CH103-SW1

/10/ Mac Developer Library. About Objective-C.
Viitattu 20.3.2014.
https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

/11/ Microsoft. Visual C# resources.
Viitattu 20.4.2014.
http://msdn.microsoft.com/en-US/vstudio/hh341490.aspx

/12/ MySQL. Reference Manual
Viitattu 25.4.2014.
http://dev.mysql.com/doc/refman/5.7/en/introduction.html

/13/ Oracle. Java Language Specification.
Viitattu 25.4.2014.
http://docs.oracle.com/javase/specs/jls/se8/html/jls-1.html#jls-1.6

/14/ PHP Documentation. Introduction.
Viitattu 25.4.2014.
http://www.php.net/manual/en/intro-whatis.php

/15/ PHP Documentation. cURL.
Viitattu 12.4.2014.
http://www.php.net/manual/en/intro.curl.php

/16/ PHP Documentation. Stream Functions.
Viitattu 25.4.2014.
http://www.php.net/manual/en/function.stream-socket-client.php

/17/ Visual Paradigm. UML and SysML Modeling.
Viitattu 20.5.2014.
http://www.visual-paradigm.com/features/uml-and-sysml-modeling/

/18/ Windows Phone Dev Center. How to send and receive toast notifications for Windows Phone 8.
Viitattu 10.4.2014.
http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202967%28v=vs.105%29.aspx