

Mobiilisovelluskehitys

Vesa Mäkeläinen

Tekijä tai tekijät Vesa Mäkeläinen	Ryhmätunnus tai aloitusvuosi 2011
Raportin nimi Mobiilisovelluskehitys	Sivu- ja liitesivumäärä 49 + 76
Opettajat tai ohjaajat Kai Kivimäki	
<p>Tässä opinnäytetyössä tutustutaan mobiilisovelluskehitykseen ja kolmeen yleisimpään mobiilikäyttöjärjestelmään. Produktityyppisessä työssä tuotetaan Pakettivahti-sovelluksen prototyyppi Android 4, iOS 7 ja Windows Phone 8 -järjestelmille. Sovellus automatisoi postilähetysten seuraamisen Postin verkkopalvelua käyttäen.</p> <p>Työssä tutkitaan, miten yhteisen suunnitelman perusteella on mahdollista toteuttaa kullekin järjestelmälle sen ominaispiirteet huomioiva mobiilisovellus. Tutkimus tuottaa produktin puitteissa vertailevaa tietoa valittujen järjestelmien ja niihin perustuvan sovelluskehityksen eroista. Tarkoituksena on madaltaa aloittavien mobiilisovelluskehittäjien kynnystä ja tarjota yrityksille eväitä olemassa olevien verkkopalveluiden hyödyntämiseen mobiilisovelluksista käsin.</p> <p>Teoriaosuudessa esitellään Android 4, iOS 7 ja Windows Phone 8 sekä tutustutaan kehitettävän sovelluksen kannalta olennaisiin toimintoihin. Näitä ovat sovelluksen elinkaari, taustalla tapahtuvat toiminnot ja tietojen tallennusmahdollisuudet. Case-osuudessa kuvataan iteratiivinen sovelluskehitysprosessi työvaiheittain suunnittelusta valmiiseen prototyyppiin asti.</p> <p>Saatujen tulosten perusteella havaitaan, että aloittelevakin mobiilisovelluskehittäjä pystyy toteuttamaan suunnitellun toiminnallisuuden sisältävän sovelluksen eri järjestelmille, kunhan hänellä on kokemusta olio-ohjelmoinnista, riittävästi kärsivällisyyttä ja nettiyh-teys ongelmatilanteiden ratkaisujen etsimiseksi.</p>	
Asiasanat Mobiilisovellukset, ohjelmistokehitys, Android, iOS, Windows Phone	

Degree Programme in Information Technology

Authors Vesa Mäkeläinen	Group or year of entry 2011
The title of thesis Mobile application development	Number of pages and appendices 49 + 76
Supervisor(s) Kai Kivimäki	
<p>This thesis presents mobile software development and three most commonly used mobile operating systems. The product of the thesis is a mobile application prototype, developed for Android 4, iOS 7, and Windows Phone 8 systems. The application provides automatic parcel tracking using a network service of Posti (the Finnish postal service provider).</p> <p>The study concentrates on application development for three mobile systems from a common design. It produces information by comparing software development processes between the chosen systems. The purpose of this study was to help new developers getting started with mobile application development. This information will also be useful to companies looking for solutions to provide their existing network services as mobile applications.</p> <p>The first part gives an introduction to Android 4, iOS 7, and Windows Phone 8. These systems are examined further, concentrating on the aspects used by the application to be developed. These include application life cycle events, background tasks and data storage possibilities. The second part, the case study, describes each step of the iterative mobile application development process, from design to a functional prototype.</p> <p>The results show that even a novice mobile developer is able to produce an application with designed functionality, if he or she has competence in object oriented programming, enough patience, and a network connection for finding solutions to arising problems.</p>	
Key words Mobile applications, software development, Android, iOS, Windows Phone	

Sisällys

1	Johdanto	1
1.1	Aihepiiri.....	1
1.2	Tutkimuksen syyt ja tavoitteet	2
2	Mobiilisovelluskehitys	4
2.1	Mobiilikäyttöjärjestelmät.....	4
2.2	Windows Phone 8.....	5
2.3	Android 4	11
2.4	iOS 7	16
3	Case: Pakettivahti-sovellus	21
3.1	Sovelluksen suunnittelu	21
3.2	Käyttöliittymän toteutus	23
3.3	Sovelluksen tietojen tallennus	27
3.4	Postin lähetyseurantaverkkopalvelun toiminta	33
3.5	Verkkopalvelun käyttäminen	34
3.6	Verkkopalvelun käyttäminen taustalla	36
3.7	Ilmoitus käyttäjälle saapuneesta lähetyksestä.....	39
4	Johtopäätelmät	42
4.1	Havaintoja mobiilisovelluskehityksestä	42
4.2	Tutkimustavoitteiden saavuttaminen.....	45
	Lähteet.....	47
	Liitteet.....	50
	Liite 1. Mobiilikäyttöjärjestelmien ominaisuuksien vertailu	50
	Liite 2. Pakettivahti-sovelluksen käyttötapaukset	51
	Liite 3. Lähdekoodit.	52

1 Johdanto

Tässä opinnäytetyössä toteutetaan kolmelle yleisimmälle mobiilikäyttöjärjestelmälle Postin lähetystenseurantaverkkopalvelua hyödyntävän sovelluksen prototyyppi. Toteutuksen kautta tutustutaan kuhunkin järjestelmään ja mobiilisovellusten keskeisiin toimintoihin. Työ antaa valmiuksia jatkaa mobiilisovelluskehitykseen perehtymistä.

Jo opinnäytetyöprosessin alussa tiesin haluavani mobiilisovelluskehitykseen liittyvän aiheen, jossa hyödynnetään olemassa olevaa verkkopalvelua. Lopullinen idea postilähetystä seuraavasta sovelluksesta syntyi, kun kiinalaisesta verkkokaupasta tilaamani teekannu päätyi epähuomiossa noutamattomana takaisin Kiinaan. Oli ilmaantunut selkeä tarve sovellukselle, joka ilmoittaisi saapuneiden lähetysten odottavan noutoa.

Pohdin pitkään erilaisia toteutusvaihtoehtoja. Aluksi suunnitelmissa oli vertailla iOS-sovelluskehitystä natiivityökalujen ja kahden järjestelmäriippumattoman (cross-platform) ratkaisun kesken. Tästä ajatus kääntyi pelkkien cross-platform –työkalujen käyttöön ja niillä tuotettujen sovellusten vertailuun kolmessa järjestelmässä. Lopulta päätin aloittaa mobiilisovelluskehityksen tutustumalla kunkin järjestelmän omiin työkaluihin ja menetelmiin. Järjestelmien ominaispiirteiden tultua tutuksi on myöhemmin helpompi perehtyä järjestelmäriippumattomiin ratkaisuihin ja ymmärtää niiden rajoituksia.

1.1 Aihepiiri

Opinnäytetyössäni tutustutaan sovelluskehittäjän näkökulmasta kolmen yleisimmän mobiililaitelustan käyttöjärjestelmään, kehitysympäristöön ja ohjelmointikielen. Nämä ovat Applen iOS 7 (kehitysympäristönä Xcode ja ohjelmointikielenä Objective-C), Googlen Android 4 (Eclipse ja Java) sekä Microsoftin Windows Phone 8 (Visual Studio ja C#). Opinnäytetyöni ytimen muodostaa produkti, joka koostuu kaikilla kolmella järjestelmällä toteutettavasta mobiilisovelluksen prototyypistä.

Lähtökohdaksi oletetaan mobiilisovelluskehittäjäksi ryhtyvällä olevan kokemusta olio-ohjelmoinnista esimerkiksi Java-kielellä. Tämän lisäksi opinnäytetyön sujuva seuraami-

nen edellyttäneen graafisten käyttöliittymien, XML-kielen, relaatiotietokantojen ja ORM-ratkaisujen (Object-Relational Mapping, olioiden varastoiminen relaatiotietokantaan), verkkopalveluiden käytön sekä html:n ja css:n perusteiden ymmärtämistä.

Tutkimuksen näkökulmaksi on käytännön tarpeesta valikoitunut tilanne, jossa Postin verkkosivuilla tarjottavasta lähetystenseurantapalvelusta halutaan toiminnallisuudeltaan laajennettu mobiilisovellus. Lähetystenseurantapalveluun syötetään lähetyksen yksilöivä seurantakoodi, minkä jälkeen palvelu kertoo lähetyksen etenemisvaiheet. Mobiilisovelluksen tarkoituksena on automatisoida toiminto siten, että seurantakoodi syötetään sovellukselle vain kerran. Tämän jälkeen sovellus tiedustelee itsenäisesti verkkopalvelusta lähetyksen tilaa ja antaa käyttäjälle ilmoituksen lähetyksen saavuttua noudettavaksi.

Käytössäni on testilaitteina Samsung Galaxy Note, Apple iPhone 4 ja Nokia Lumia 920. Tutkimusaihe rajautuu kehitystyössä käytettävissä olevien laitteiden myötä Androidin pääversioon 4, Windows Phonen versioon 8 ja iOS:n versioon 7. Opinnäytetyön kirjoittamisen aikaan Microsoft julkisti Windows Phone 8.1 –version, joka kuitenkin tulee jakeluun vasta toukokuusta 2014 alkaen, joten tästä käytännön syystä se rajautuu tämän työn ulkopuolelle.

1.2 Tutkimuksen syyt ja tavoitteet

Tutkimukseni selvittää miten halutunlainen mobiilisovellus voidaan toteuttaa kullakin kolmesta yleisimmästä mobiilijärjestelmästä. Opinnäytetyöni produktina syntyvillä kolmella sovellusprototyypillä on yhteinen lähtökohta ja suunnittelu. Tämän erityistapauksen kautta selvitetään kolmen valitun mobiilijärjestelmän mahdollisuuksia ja rajoituksia halutun toiminnallisuuden toteutuksessa. Tutkimuksen kohteena on siten ohjelmistokehitysprosessi suunnitelmista toiminnalliset vaatimukset täyttävään prototyyppiin asti.

Produktin muodostava sovellus perustuu siis kaikille alustoille yhteiseen suunnitelmaan. Siinä määritellään sovelluksen toiminnalliset vaatimukset ja niiden perusteella käyttöliittymän tarpeet. Haluttu toiminnallisuus koostuu neljästä osa-alueesta, joiden toteuttamista selvitän tässä työssä.

1. Sovelluksen tulee hyödyntää olemassa olevaa Postin verkkopalvelua.
2. Lähetyssurantapalveluun tehtävien kyselyiden tulisi toimia taustalla myös silloin, kun käyttäjä ei käytä sovellusta aktiivisesti.
3. Sovelluksen verkkopalvelusta saamat tiedot tulee tallentaa paikallisesti.
4. Käyttäjälle tulee ilmoittaa saapuneesta lähetyksestä myös silloin, kun sovellus ei ole auki.

Mobiiliohjelmistojen sovelluskehityksen vertailu on tarpeen, koska sovelluskehitystä yhdestä suunnitelmasta kolmeksi sovellukseksi ei juurikaan ole käsitelty aikaisemmissa tutkimuksissa. Nykyisin yhä useampi yritys haluaa IT-palvelunsa toimivan myös yleisimmillä mobiilialustoilla ja harkinnee siksi vaihtoehtoisia toteutusmahdollisuuksia. Sovelluskehittäjät pohtinevat vastaavasti, mikä olisi sopiva järjestelmä mobiilisovelluskehitystyöhön perehtymiseen.

Tutkimukseni tuottaa produktin puitteissa vertailun avulla tietoa valittujen mobiilijärjestelmien ja niihin perustuvan sovelluskehityksen eroista sekä tarjoaa näin tietoperustaa yritysten ja sovelluskehittäjien pohdintoihin. Lisäksi tutkimustulosten avulla on kenties mahdollista madaltaa yritysten kynnystä lähteä tarjoamaan olemassa olevia verkkopalveluja myös erillisinä mobiilisovelluksina.

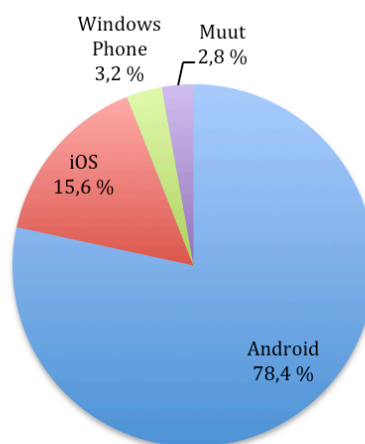
Kaiken taustalla opinnäytetyöni tavoitteena on myös oppia mobiilisovelluskehityksen menetelmiä, saada käytännön tekemisen kautta käsitys kohteena olevista järjestelmistä ja parantaa valmiuksia sovelluskehitystyön jatkamiseen opinnäytetyön jälkeenkin.

2 Mobiilisovelluskehitys

Tässä luvussa esitellään kolme yleisintä ja tässä opinnäytetyössä käytettyä mobiilikäyttöjärjestelmää. Nämä ovat Android, iOS ja Windows Phone. Tämän jälkeen perehdytään produktiosan sovelluksen toteutuksen näkökulmasta jokaisen järjestelmän keskeisiin ominaisuuksiin. Näitä ovat järjestelmän kehittyminen ja ominaisuudet, kehitysympäristö, sovelluksen elinkaari, taustalla suoritettavat toiminnot ja sovelluksen tietojen tallennusmahdollisuudet. Tarkasteltavat järjestelmäversiot ovat produktissa käytetyt Android 4, iOS 7 ja Windows Phone 8. Liitteessä 1 on järjestelmien ominaisuuksia vertaileva taulukko.

2.1 Mobiilikäyttöjärjestelmät

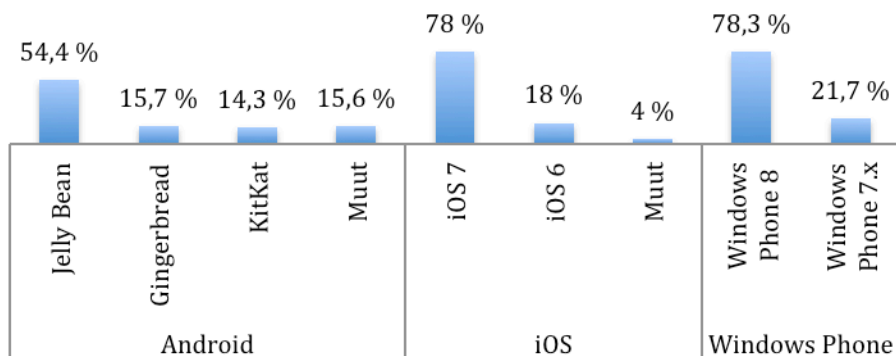
Kuviossa 1 on esitetty mobiilikäyttöjärjestelmien markkinaosuuksia. Kolme yleisimmin käytettyä mobiilikäyttöjärjestelmää vuoden 2013 tilastoissa ovat Android (78,4 %, yli 758 miljoonaa myytyä laitetta), iOS (15,6 %, yli 150 miljoonaa myytyä laitetta) ja Windows Phone (3,2 %, yli 30 miljoonaa myytyä laitetta) (Gartner 2014).



Kuvio 1. Mobiilikäyttöjärjestelmien markkinaosuuksia 2013 (Gartner 2014).

Kaikista kolmesta järjestelmästä on julkaistu useampia versioita. Eniten järjestelmäversioita on Androidilla ja vähiten uusimmalla tulokkaalla, Windows Phonella. Eri Android-versioita on myös edelleen käytössä eniten, kun taas iOS- ja Windows Phone –laitteissa uusin versio on myös yleisin. Huhtikuussa 2014 Android-järjestelmän yleisimpiin versioihin 4.1-4.3 (koodinimeltään Jelly Bean) kuului 54,4 % käytössä olevista Android-laitteista. Toiseksi suurin osa laitteista, 15,7 %, käyttää vanhempaa versiota 2.3

(Gingerbread). Viimeisintä versiota 4.4 (KitKat) käyttää 14,3 % Android-laitteista. Tällä hetkellä uusinta versiota käyttävien laitteiden osuus on huimassa kasvussa valmistajien tuodessa uusia laitteita markkinoille. Vanhemman Gingerbreadin osuus on vastaavasti loivassa laskussa, kun taas Jelly Bean -versioiden osuus on pysynyt kohtuullisen vakiona. (AppBrain 2014.) Kuviossa 2 on esitetty Androidin, iOS:n ja Windows Phonen jakautuminen eri järjestelmäversioihin.



Kuvio 2. Mobiilikäyttöjärjestelmien käytetyimmät versiot (AppBrain 2014. Appleinsider 2013. Thurrott, P. 2014).

Applen iOS:n ja Microsoftin Windows Phonen osalta tilanne on kuvion 2 perusteella huomattavasti Androidia yhtenäisempi. Vuoden 2013 lopulla 78 % Applen laitteista käytti viimeisintä versiota iOS 7 ja 18 % edellistä versiota iOS 6 (Appleinsider 2013). Tammikuussa 2014 Windows Phonen viimeisintä versiota käytti 78,3 % kaikista Windows Phone –laitteista. Loput laitteet käyttivät edellisiä versioita 7.x. Laitevalmistajista Nokia käytännössä dominoi Windows Phone –markkinoita, sillä sen valmistamia laitteita on yhteensä 78,5 % kaikista ja jopa 92,3 % käytössä olevista Windows Phone –laitteista. (Thurrott, P. 2014.)

2.2 Windows Phone 8

Microsoftin kehittämä ja lokakuussa 2012 julkaisema Windows Phone 8 on kokonaisuus, jonka muodostavat laitteistoalusta, käyttöjärjestelmä ja kokoelma web-palveluita. Se on kehitetty vuonna 2010 julkaistun Windows Phone 7:n päivitetyksi versioksi ja uuden työpöytäkäyttöön tarkoitetun Windows 8 –käyttöjärjestelmän kevennetyksi versioksi. Samalla se on edelleen yhteensopiva version 7.x sovellusten kanssa. Järjestelmä rakentuu Windows 8 –työpöytäkäyttöjärjestelmän osien ja ARM-prosessoreille suunnit-

teltujen Windows Runtime –kirjastojen sekä toisaalta Windows Phone 7 –järjestelmän ohjelmointirajapintojen (API, Application Programming Interface) varaan. Tästä johtuen järjestelmä sisältää paikoin kaksi saman toiminnallisuuden sisältävää rajapintaa, esimerkiksi Isolated Storage (WP7) ja Local Storage (Windows Runtime). (Binkley-Jones, Perga, Sync & Benoit 2014, 3 – 4, 13.)

Microsoft on määritellyt Windows Phone 8 –laitteistojen minimivaatimukset, jotta jokainen laite tarjoaisi samanlaisen käyttökokemuksen. Näyttökokoja on neljä: 800*380 (WVGA), 1280*768 (WXGA), 1280*720 (720p) ja 1920*1080 (1080p). Kapasitiivisen kosketusnäytön pitää tukea vähintään nelipisteistä kosketusta (Multitouch). Laitteissa on oltava vähintään 512 MB muistia, DirectX 9 –kiihdytystä tukeva grafiikkasuoritin sekä Snapdragon S4 tuplaydinprosessori. (Binkley-Jones, Perga, Sync & Benoit 2014, 6 – 7. WPCentral.)

Windows Phone 8 –sovellusten suoritussympäristönä on Windows 8:n tapaan muusta järjestelmästä lähes täysin eristetty hiekkalaatikko, joten ne eivät voi kommunikoida eivätkä jakaa tietojaan tai tiedostojaan suoraan toisten sovellusten kanssa. Sovellusten kehitysympäristönä on Visual Studio ja ohjelmointikielinä käytössä ovat XAML-käyttöliittymän määrittelykieltä (Extensible Application Markup Language) hyödyntävät C# ja Visual Basic sekä Direct3D-sovellusten käyttämä C++. Windows Phone 8 –sovellukset, jotka on kirjoitettu C#- tai Visual Basic –ohjelmointikielillä, suoritetaan hallittuina sovelluksina (managed applications). Tämä tarkoittaa, että niiden lähdekoodi käännetään suoritettavaksi .NET Common Language Runtime (CLR) –ympäristössä. Vain C++ -kieliset natiivisovellukset käännetään ja linkitetään suoraan konekielisiiksi. (Binkley-Jones ym. 2014, 14, 21.)

Microsoft tarjoaa kehittäjille ilmaiseksi Visual Studio Express for Windows Phone –version, joka sisältää Windows Phone SDK:n ja Blend for Visual Studio –käyttöliittymäsuunnittelutyökalun. SDK sisältää Windows Phone 8 –emulaattorin, jonka käyttäminen edellyttää Hyper V –virtualisoinnin käyttöä. Järjestelmävaatimukset ovat tällöin: 64-bittinen Windows 8 Pro –käyttöjärjestelmä, vähintään 4 GB muistia sekä laitteistopohjaista virtualisointia tukeva CPU ja BIOS. Windows Phone 8 -laitteeseen voi asentaa sovelluksia vain Windows Phone Store:sta tai yritysten sisäisiin

sovelluksiin tarkoitettusta Company Hub:sta. Sovellusten testaamiseen puhelimessa tarvitaan maksullinen rekisteröityminen Microsoftin Dev Center:iin. Jokaista kehittäjätiliä kohti voi rekisteröidä enintään kolme puhelinlaitetta ja niistä jokaisessa voi olla asennettuna samanaikaisesti enintään 10 kehitettävää sovellusta. (Binkley-Jones ym. 2014, 23–25.)

Microsoftin käyttämä ja tukema suunnittelumalli on nimeltään Model – View – View-Model (MVVM). Mallin käyttäminen ei kuitenkaan ole välttämätöntä eikä yksinkertaisissa sovelluksissa aina edes hyödyksi sen tuoman kompleksisuuden vuoksi. (Binkley-Jones ym. 2014, 49.) Mallin ideana on erottaa toisistaan tietoja käyttäjälle esittävä käyttöliittymäkerros (View), tiedot sisältävä kerros (Model), sekä näiden välillä adapterina toimiva kerros (ViewModel). ViewModelin tehtävänä on muokata Modelin raakadataa näyttökerroksen tukemaan muotoon. Riippuvuussuhteet kulkevat Model → ViewModel → View eli tietomallista näyttöön päin. Model on täysin itsenäinen tietoyksikkö. ViewModel on tietoinen vain käyttämistään Model-luokista. Vastaavasti View tuntee vain käyttämänsä ViewModelin-luokat. Suunnittelumalli mahdollistaa sovelluksen eri kerrosten samanaikaisen kehitystyön ja samalla pienemmät osakomponentit on helpompaa testata. Modulaarisuuden ansiosta myös sovelluksen käyttöliittymä on mahdollista vaihtaa muihin osiin koskematta, ja osat ovat uudelleenkäytettävissä muissa sovelluksissa. Tavoitteena on siis helpottaa sovelluksen kehitystyötä, ylläpitoa ja laajennettavuutta. (Falafel Software Inc. 2013, 84.)

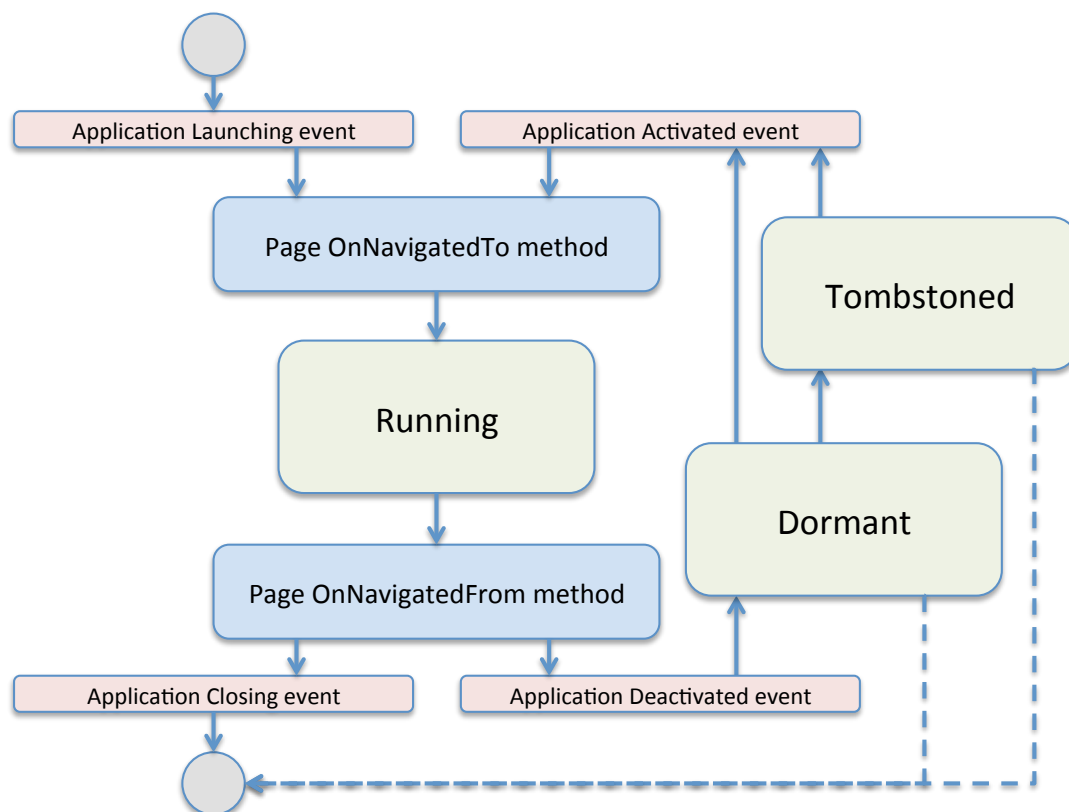
Sovelluksen elinkaari sisältää joukon tiloja, joiden läpi sovellus kulkee suorituksensa aikana. Työpöytäkäyttöjärjestelmissä käyttäjä voi avata useita sovelluksia samanaikaisesti, mutta älypuhelinien ja tablettien pienet näytöt mahdollistavat vain yhden sovelluksen käytön kerrallaan. Mobiililaitteiden on tämän vuoksi tarjottava käyttäjälle nopea vaihto eri tehtävien välillä. Niiden on oltava mahdollisesta vaatimattomasta suoritinkapasiteetista huolimatta nopeavasteisia ja pyrittävä pitämään akkukesto mahdollisimman pitkänä. Mobiilikäyttöjärjestelmät siirtävät automaattisesti sovelluksia elinkaarivaiheesta toiseen käyttäjän tai järjestelmän omien tarpeiden mukaan. Sovelluskehittäjä ei voi vaikuttaa siihen, missä elinkaarensa vaiheessa sovellus kulloinkin on. Sen sijaan järjestelmä kutsuu elinkaaren vaiheesta toiseen siirryttäessä suoritettavaksi sovelluksen metodeja, joihin kehittäjä voi lisätä tarvitsemiaan toimintoja. Näin sovellus voi esimerkiksi tallen-

taa tietojaan ennen sen suorituksen väliaikaista tai lopullista päättymistä. Tämän vuoksi sovelluksen elinkaaren ymmärtäminen on tärkeä osa mobiilisovelluskehitystä. (Falafel Software Inc. 2013, 217.)

Windows Phone 8 –sovelluksen elinkaari muodostuu kolmesta tilasta (state), niiden välisiä siirroksia toteuttavista neljästä tapahtumasta (event) sekä kahdesta suoritettavasta metodista kuvion 3 mukaisesti. Vain yksi sovellus voi olla käyttäjän käytettävissä etualalla tilassa Running. Kun käyttäjä tai järjestelmä siirtyy toiseen sovellukseen (esimerkiksi puhelun tullessa), käytetty sovellus siirtyy Dormant-tilaan. Tässä tilassa sovellus on yhä muistissa kaikkine olioineen ja tilatietoineen, mutta sen prosessien ja säikeiden suoritus on keskeytetty. Järjestelmä voi palauttaa sen Running-tilaan ilman kehittäjältä vaadittavaa lisäkooodausta. Jos käyttöjärjestelmän kuitenkin pitää vapauttaa resursseja, se siirtää sovelluspinon alimmaisen Dormant-tilassa olevan sovelluksen Tombstoned-tilaan. Tombstoned-tilassa sovelluksen käyttämä muisti vapautetaan. Jäljelle jää kuitenkin vielä tieto siitä, millä sovelluksen sivulla käyttäjä viimeksi oli. Lisäksi järjestelmä tallentaa tämän sivun kuvakaappauksen. Sovelluksen uudelleenaktivoinnin yhteydessä käyttäjä siirretään sivulle, josta hän poistui, mutta sovelluskehittäjän pitää palauttaa sovelluksen tiedot. Muistin vapauttamisesta johtuen kaikki tarvittavat tiedot on pitänyt tallentaa jo ennen Dormant-tilaan siirtymistä. Järjestelmä voi milloin tahansa poistaa Dormant- tai Tombstoned-tilassa olevan sovelluksen kokonaan muistista, eikä sovellus saa tästä tapahtumasta enää ilmoitusta. (Falafel Software Inc. 2013, 217. Binkley-Jones ym. 2014, 61–62, 70.)

Sovelluksen elinkaaren tilamuutoksiin liittyy kuvion 3 mukaisesti neljä tapahtumaa (event). Juuri ennen sovelluksen käynnistymisen päättymistä järjestelmä ilmoittaa tapahtuvaksi Application Launching. Sovelluksen siirtyessä taustalle Dormant-tilaan lähetetään Application Deactivated. Kun sovellus palaa etualalle Dormant- tai Tombstoned-tilasta lähetetään Application Activated. Näiden lisäksi käyttäjän sulkiessa sovelluksen painamalla laitteen Back-nappia tai sammuttaessa laitteen lähetetään Application Closing –tapahtuma. Näiden PhoneApplicationService-luokan tapahtumien käsittelijät (event handler) luodaan automaattisesti Visual Studio Windows Phone 8 App Template –sovelluspohjassa. Sovelluksen on reagoitava näihin tapahtumiin, jotta Microsoft

katsoisi sen toimivan oikein. (Falafel Software Inc. 2013, 217. Binkley-Jones ym. 2014, 63.)



Kuvio 3. Windows Phone 8 -sovelluksen elinkaari (Microsoft 2014a).

Tilojen ja tapahtumien lisäksi sovelluksen elinkaareen liittyy kuvion 3 mukaisesti myös kaksi PhoneApplicationPage-luokan metodia, joita järjestelmä kutsuu sovelluksen vaihtaessa tilaa. OnNavigatedTo-metodia kutsutaan välittömästi Launching- tai Activated-tapahtuman jälkeen. Sovelluksen siirryessä pois Running-tilasta kutsutaan OnNavigatedFrom-metodia juuri ennen Deactivated- tai Closing-tapahtumaa. Näiden metodien avulla saadaan koko sovelluksen tilatiedon lisäksi tallennettua myös jokaisen sivun oma tila. Sivulta voidaan tallentaa esimerkiksi käyttäjältä saadut syötteet, jotta sivun tietoken-
tät saadaan palautettua, kun sovellus myöhemmin jatkaa toimintaansa. (Falafel Software Inc. 2013, 218. Binkley-Jones ym. 2014, 70.)

Microsoft on asettanut lukuisia vaatimuksia Windows Phone 8 –sovelluksille. Sovelluk-
sen tulee käynnistyessään näyttää ensimmäinen sivu tai käynnistyskuva (splash screen
image) viiden sekunnin kuluessa. Sen tulee vastata käyttäjän toimiin viimeistään 20 se-

kunnin kuluessa käynnistyksestä. Samoin kaikkiin sovelluksen elinkaaren tapahtumiin on asetettu 10 sekunnin aikaraja, jonka ylittyessä käyttöjärjestelmä keskeyttää sovelluksen suorituksen välittömästi. Microsoft suosittelee siksi tietojen tallentamista heti sovelluksen saatua ne, eikä tallennuksen lykkäämistä elinkaaren tilan vaihtumiseen asti. (Microsoft 2014a. Microsoft 2014b.)

Microsoft on asettanut Windows Phone 8 –sovelluksille rajoituksen niiden taustalla suorittamille tehtäville. Taustalla olevat sovellukset eivät saa vaikuttaa tai hidastaa etualalla olevan sovelluksen toimintaa, eivätkä suorittaa toimenpiteitä, joiden seurauksena akun varaus voi kulua nopeasti loppuun. Käytännössä taustalla oleva sovellus voi suorittaa vain Scheduled Action Service –luokan mahdollistamia tehtäviä. Näitä ovat hälytykset (alarm), muistutukset (reminder) ja taustalla suoritettavat tehtävät (background task). Näistä viimeinen tarkoittaa pyyntöä Scheduled Action Service –luokalle tietyn sovelluksen taustatoimijan (background agent) käynnistämisestä. Tällaisen agentin avulla sovellus voi suorittaa joitain tehtäviä tausta-ajona tietyin väliajoin. Taustatoimija on tavallaan kääreluokka, joka on rekisteröity osaksi sovellusta. Sillä tulee olla metodit NotifyComplete ja Abort, jotka ilmoittavat käyttöjärjestelmälle taustalla suoritettujen tehtävien valmistumisesta tai keskeytymisestä. Taustalla voi suorittaa kahdenlaisia tehtäviä. PeriodicTask-luokan tehtävät toistuvat noin puolen tunnin välein. ResourceIntensiveTask-luokan määrittämät taustatoiminnot kuluttavat nimensä mukaisesti paljon laitteen resursseja. Käyttöjärjestelmä käynnistääkin ne vain laitteen ollessa täysin ladattu ja virtalähteeseen kytkettynä. Lisäksi laitteen on oltava yhteydessä WiFi-verkkoon tai liitettynä tietokoneeseen ja näytön lukittuna. Sovellus voi rekisteröidä itselleen vain yhden PeriodicTask-olion, ja sen toiminnalle on asetettu rajoituksia. Taustalla suoritettava tehtävä saa kestää enintään 25 sekuntia. Virransäästötila voi estää taustatehtävän suorittamisen kokonaan. Järjestelmä niputtaa tehtäviä yhteen siten, että niiden suoritusvälit voivat vaihdella 20-40 minuutin sisällä. Lisäksi ne saavat käyttää enintään 20 MB muistia laitteissa, joissa on vähintään 1 GB muistia, tai muutoin vain 11 MB. (Falafel Software Inc. 2013, 355. Binkley-Jones ym. 2014, 87, 103-104. Microsoft 2014c.)

Windows Phone 8 –sovellus voi tallentaa tietoja eri tavoin käyttötarkoituksen mukaan. Sovellus voi tallentaa esimerkiksi verkkopalvelusta uudelleen haettavissa olevan tiedon väliaikaiseen tiedostoon, jonka tiedot säilyvät Tombstoned-tilaan asti, mutta tuhoutuvat

sen jälkeen. Tämän väliaikaisen tallennuksen etuna on sen nopeus ja helppous. Pysyvään tallennukseen on tarjolla sovelluksen hiekkalaatikon tiedostojärjestelmää hyödyntävät isolated storage sekä paikallinen tietokanta. Sovelluksen tietojen väliaikaista tallentamista varten käytettävissä on PhoneApplicationService-luokan State-attribuutti. Se on avain-arvo-pareista muodostuva dictionary-tietorakenne, johon voi tallentaa merkkijonoavaimella minkä tahansa sarjallistettavan (serializable) olion. Sovellus voi tallentaa monimutkaisempia tietorakenteita suoraan tiedostoihin käyttäen asynkronisia StorageFolder- ja StorageFile-luokkia. Tällöin tallennettava tieto on kuitenkin sarjallistettava itse. Paikallinen tietokanta soveltuu parhaiten monimutkaisten tai paljon dataa sisältävien tietorakenteiden varastointiin. (Falafel Software Inc. 2013, 219, 221, 409. Binkley-Jones ym. 2014, 163–165. Microsoft 2014d.)

Windows Phone 8 sisältää Microsoft SQL Server Compact Edition – tietokantamoottorin (SQL CE). Sitä käytetään sovelluksissa LINQ to SQL –rajapinnan kautta suorien SQL-lauseiden sijaan. LINQ to SQL on object-relational mapping (ORM) –teknologia, jonka Microsoft esitteli .NET –frameworkin versiossa 3.5. Se tarjoaa sovelluksille rajapinnan olioiden tallentamiseen ja palauttamiseen paikallisesta tietokannasta. Sovellus käyttää tietokantaa erillisen tietokannan rajapintana toimivan data context –olion kautta. Tietokannan sisältämät luokat merkitään LINQ-attribuuteilla, joiden perusteella LINQ to SQL luo automaattisesti tietokannan taulut ja hallinnoi olioihin tehtäviä muutoksia. (Falafel Software Inc. 2013, 409. Binkley-Jones ym. 2014, 169–171.)

2.3 Android 4

Android on avoimeen lähdekoodiin perustuva mobiililaitteille suunnattu alusta (platform) ja käyttöjärjestelmä. Järjestelmän suunnittelussa on alusta alkaen kiinnitetty erityistä huomiota akkukäyttöisyyden edellyttämiin virransäästöominaisuuksiin ja pienikokoisten laitteiden rajoitettuun muistikapasiteettiin ja suorituskyykyyn. Käyttöjärjestelmä on silti suunniteltu toimimaan kaikenlaisissa laitteissa, eikä se aseta vaatimuksia näyttökoon tai –resoluution, käytetyn suorittimen tai piirisarjan suhteen. Laitevalmistajien ei tarvitse maksaa lisenssimaksuja avoimeen lähdekoodiin perustuvasta käyttöjärjestelmästä. Valmistajat voivat kuitenkin muokata järjestelmää ja suojata nämä järjestelmän räätä-

löinnit avoimen lähdekoodin erilaisten lisensointikäytäntöjen mukaisesti. Tämä on mahdollistanut Androidin nopean yleistymisen. (Gargenta & Nakamura 2014, 1–3.)

Google osti vuonna 2005 Android, Inc. –nimisen yrityksen, joka oli aloittanut mobiilikäyttöjärjestelmän kehitystyön. Vuonna 2007 Google julkaisi Open Handset Alliancen kautta Androidin lähdekoodin. Android SDK 1.0 julkaistiin 2008, samoin ensimmäinen Android-puhelin, HTC:n G1. Tablettimarkkinoiden kilpailun alkaessa vuonna 2011 Android oli noussut myydyimmäksi mobiilialustaksi. Nykyään Android toimii älypuhelin- lisen lisäksi monissa muissakin laitteissa ja on haastanut sulautetuissa järjestelmissä aiemmin yleisesti käytetyn linux-käyttöjärjestelmän. (Gargenta & Nakamura 2014, 3.)

Androidista on kehitetty nopeasti Googlen johdolla uusia versioita. Viimeisin on Android 4.4, eli API 19, koodinimeltään KitKat. Googlen julkaisema Android Open Source Project (AOSP) –versio on kuitenkin harvoin suoraan käytössä kuluttajalaitteissa. Tavallisesti kukin valmistaja lisää siihen omia muutoksiaan. Nämä ovat yksinkertaisimmillaan vain ulkonäöllisiä, mutta esimerkiksi HTC Sense muokkaa varsin syvästi koko järjestelmän toimintaa. Nopea kehitystyö on johtanut monenlaisten laiteominaisuuksien lisäksi myös järjestelmäversioiden sirpaloitumiseen. Laitevalmistajat ovat avoimen lähdekoodin järjestelmille tyypilliseen tapaan muokanneet omiin laitteisiinsa räätälöidyn version. Räätälöinnin laajuudesta riippuen järjestelmän päivitys on heille luonnollisesti työläämpää. Tämän vuoksi nykyään on vielä käytössä paljon Android 2.3.3 (Gingerbread, API 10) -versiota käyttäviä laitteita, joskin näiden käyttäjät ovat todennäköisesti lähiaikoina vaihtamassa laitteitaan uudempiin. Sovelluskehittäjän tulee huomioida lukuisat käytössä olevat järjestelmäversiot. Sovellukselle asetetaan API-taso, jolle se on kohdennettu, sekä myös alin mahdollinen API-taso, jolla sovellus vielä toimii. Tämä määrittää, mitä järjestelmän toimintoja sovellus voi käyttää. Valinta on tärkeä erityisesti tavoiteltaessa sovellukselle mahdollisimman laajaa laitekantaa. Eri järjestelmäversioiden lisäksi kehittäjän tulee pyrkiä varmistamaan sovelluksen toimivuus mitä erilaisimmissa laitteissa, joissa näyttökoko ja -resoluutio, muistin määrä, suoritinteho, jne. voivat vaihdella suuresti. (Gargenta & Nakamura 2014, 6–7.)

Android on rakennettu Linux-ytimen päälle. Se tarjoaa laitteistoriippumattomuutta laitteistotason abstrahoinnilla. Se toteuttaa yksinkertaista ja tehokasta tietoturvaa suoritta-

mallalla jokaisen sovelluksen omana prosessinaan ja omana linux-käyttäjänä. Lisäksi Linux-ydin tarjoaa monia tehokkaita ominaisuuksia mm. muistinkäytön ja verkkoyhteyksien hallintaan. Järjestelmän natiivikerros on kirjoitettu C- ja C++ -kielillä. Tämän päällä toimii erityisesti mobiililaitteille suunniteltu Dalvik-virtuaalikone. Kukin Android-sovellus ajetaan omana linux-käyttäjänä ja –prosessina hiekkalaatikossaan omassa Dalvik-virtuaalikoneessa. Tyypillisesti Android-sovellukset kirjoitetaan Java-kielillä. Ohjelmakoodi käännetään ensin Java-virtuaalikoneen ymmärtämäksi tavukoodiksi (byte code) ja siitä edelleen Dalvikin tavukoodiksi. Tämän kaksivaiheisen kääntämisen ansiosta myös muut Javan tavukoodia tuottavat ohjelmointikielet ovat käytettävissä, esimerkiksi Scala, Python ja Ruby. Toisaalta Android-sovelluksia voi kirjoittaa myös virtuaalikoneet ohittaen C/C++ -kielillä suoraan natiiviympäristössä suoritettavaksi. (Gargenta & Nakamura 2014, 31–38.)

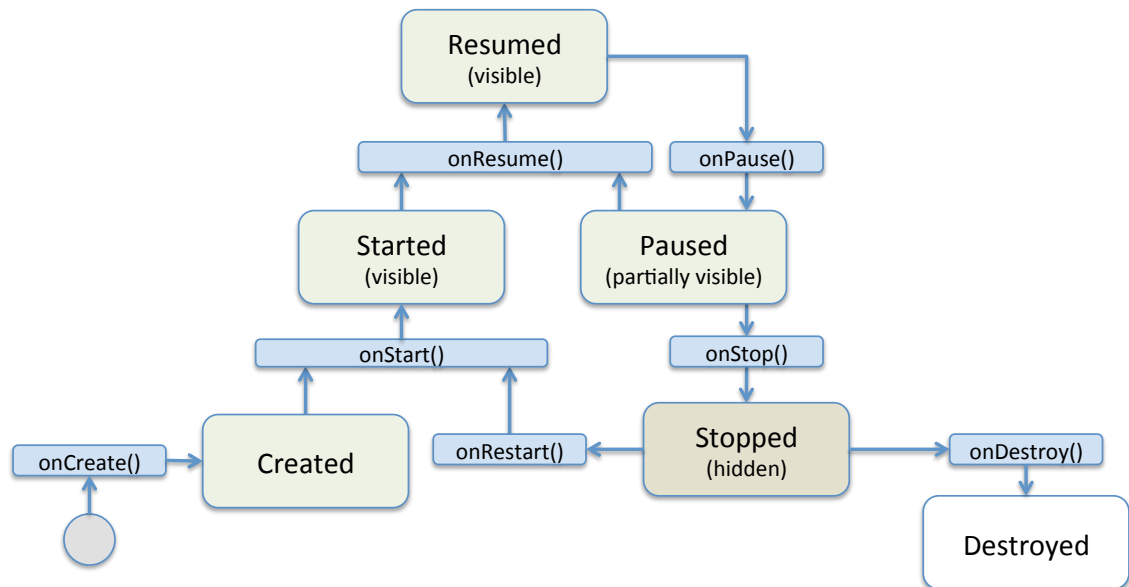
Android-kehitystyökalut ovat ilmaisia. Toistaiseksi virallinen versio on rakennettu Eclipse IDE:n (Integrated Development Environment) päälle ja saatavilla useimmille käyttöjärjestelmille (Windows, OS X, Linux). Google on kuitenkin ilmoittanut aloittaneensa siirtymän toistaiseksi vasta beta-vaiheessa olevaan avoimen lähdekoodin Android Studioon ja sen Gradle-työkaluun. Sovelluskehittimen lisäksi tarvitaan Javan Standard Edition (SE) –versio 1.6 sekä Android SDK. Android ei tue täysin Java 7:ää, eikä myöskään tarvitse kaikkia sen ominaisuuksia. Android SDK:n mukana tulee emulaattori sovellusten testaukseen. Emulaattoriin voi ladata minkä tahansa järjestelmäversion ja sen asetusten avulla sovelluksia voi testata esimerkiksi erikokoisilla näytöillä. Testaus onnistuu myös millä tahansa tietokoneeseen kytketyllä Android-laitteella, jonka kehittäjätila on asetettu käyttöön laitteen asetuksista. (Gargenta & Nakamura 2014, 42–44.)

Android-sovellusten ei tarvitse hyödyntää MVC- (Model – View – Controller) tai MVVM-tyyppistä arkkitehtuurimallia. Sovellusten käyttöliittymän voi rakentaa graafisilla työkaluilla ja XML-määrittelyillä (Extensible Markup Language). Nämä muunnetaan ohjelman suorituksen aluksi Java-luokiksi. Käyttöliittymän voi toteuttaa myös täysin ohjelmallisesti. Suositeltu tapa on määritellä käyttöliittymäkomponentit XML-tiedostona ja liittää niihin toiminnallisuus Java-koodissa. Android-sovellukset rakentuvat neljänlaisista toiminnallisista osista. Activity vastaa yhtä ikkunaa tai näkymää laitteen näytöllä. Service sisältää samanlaista toiminnallisuutta, mutta ilman käyttöliittymää. Se

onkin tarkoitettu taustalla suoritettaviin toimintoihin. Intent on viesti, joka lähetetään sovellusten osien välillä. Sillä voidaan esimerkiksi käynnistää activity tai service. Content Provider on rajapinta, jonka avulla tarjotaan tietoa jaettavaksi omiin hiekkalaatikoihinsa eristettyjen sovellusten välillä. Lopulta Broadcast Receiver on mekanismi, jonka tietty tapahtuma (esimerkiksi sille lähetty intent) aktivoi. Aktivoituessaan se voi käynnistää esimerkiksi activityn tai servicen. (Gargenta & Nakamura 2014, 63–64, 68, 70, 72, 87–88.)

Koko Android-sovelluksen elinkaari sisältyy tavallaan sovelluksen yksittäisten activityjen elinkaariin. Activityjen käynnistäminen ensimmäistä kertaa kuluttaa paljon järjestelmän resursseja. Tämän vuoksi Androidin Activity Manager huolehtii niiden elinkaaresta eli luomisesta, hallinnoinnista, sekä säilyttämisestä muistissa ja tuhoamisesta. Activity on kulloinkin yhdessä viidestä mahdollisesta tilastaan. Tilasta toiseen siirryttäessä järjestelmä kutsuu sovelluksen kyseistä tilametodia. Näiden metodien kautta kehittäjä voi ohjata sovelluksensa toimintaa tilavaihdosten yhteydessä kuvion 4 mukaisesti. (Gargenta & Nakamura 2014, 64–65.)

Starting-tilassa oleva activity ei vielä ole muistissa eikä käynnissä. Järjestelmä kutsuu sen onCreate(), onStart() ja onResume() –metodeja käynnistykseen yhteydessä kuvion 4 järjestyksessä. Saavutettuaan Running-tilan activity on näytöllä, vastaa käyttäjän syötteisiin, ja sillä on etusija järjestelmän resursseihin. Activityn onPause()-metodin kutsun kautta järjestelmä siirtää sen Paused-tilaan. Tällöin activity on yhä näytöllä, muttei etualalla, vaan tyypillisimmin dialogi-ikkunan alla. Stopped-tilassa activity ei ole enää näytöllä. Se on kuitenkin yhä muistissa ja voidaan tarvittaessa käynnistää nopeammin. Siirroksen yhteydessä järjestelmä kutsuu onStop()-metodia, jossa tulisi suorittaa tarvittavien tilatietojen tallennus. Lopulta elinkaarensa päätteeksi activity siirretään Destroyed-tilaan, jossa se poistetaan muistista. Se voi vielä vapauttaa käyttämiään resursseja onDestroy()-metodissa, mutta järjestelmä ei enää takaa tämän metodin kutsua. (Gargenta & Nakamura 2014, 65–67.)



Kuvio 4. Android Activityn elinkaari (Android a).

Android ei aseta Windows Phone 8:n tai iOS 7:n kaltaisia rajoituksia taustalla suoritettaviin tehtäviin. Sovellus voi sisältää service-olioita, mutta kehittäjän on huomioitava, että ne suoritetaan oletuksena käyttöliittymän kanssa samassa säikeessä (thread). Tämän vuoksi ne eivät saa kuluttaa liikaa järjestelmän resursseja. Androidin versiosta 3.2 (Honeycomb, API 13) alkaen mahdollisesti pitkäkestoisten tehtävien (esimerkiksi verkkotoiminnot) suorittaminen käyttöliittymäsäikeessä onkin kielletty. Ratkaisuksi Android tarjoaa tarjoaa `AsyncTask`-luokan, jolla tehtäviä voi jakaa Javan `Thread`-luokkaa yksinkertaisemmin eri säikeisiin. `Thread`-luokan säie ei voi päivittää toisessa säikeessä olevaa käyttöliittymää, koska se ei olisi thread-safe. `AsyncTask` sen sijaan voi ilmoittaa käyttöliittymäsäikeelle suorituksensa etenemisestä `doInBackground()`, `onProgressUpdate()` ja `onPostExecute()`-metodien avulla. (Gargenta & Nakamura 2014, 68, 70, 114-116.)

Android tarjoaa sovelluksen tietojen tallentamiseen erilaisia vaihtoehtoja tallennustarpeen mukaan. Yksinkertaisen tilatiedon tallennukseen voidaan käyttää `SharedPreferences`-rajapintaa, joka tarjoaa avain-arvo-pareihin perustuvan tallennuksen `Boolean`, `Float`, `Integer`, `Long`, `String` ja `Set<String>` -tyyppisille olioille. Tiedon voi tallentaa yksityisenä vain tietyn activityn käyttöön tai useampana jaettuna tietovarastona kaikkien sovelluksen activityjen käyttöön. Sovellukset voivat tallentaa monimutkaisempia tietorakenteita tiedostoihin Javan tiedostoluokkien avulla (`java.io`). Näiden lisäksi Android tukee myös xml-muotoisia tiedostoja. Tietokannan käyttäminen strukturoidun datan tallen-

tamiseen on myös mahdollista. Android ei rajoita käytettävää toteutusta, mutta tarjoaa valmiina tuen avoimen lähdekoodin SQLite-tietokannalle. (Annuzzi, Darcey & Conder 2013, 281–283, 296, 301–303. Gargenta & Nakamura 2014, 176.)

Tyypillinen SQLite-tietokannan käyttö Android-sovelluksessa tapahtuu ContentProvider-luokan laajentamisen kautta. Tällä tavoin tietokannan käytännön toteutus saadaan erotetua muusta sovelluslogiikasta ja tieto saadaan myös tarvittaessa julkaistua hiekkalaatikon rajoitusten ulkopuolelle muiden sovellusten käyttöön. Android ei tarjoa valmiista ORM-ratkaisua, vaan kehittäjän on itse toteutettava sekä tietokannan rakenne että olioiden muunnos tallennettavaan muotoon ja takaisin. Toisaalta käytettävissä on SQLiteOpenHelper-luokka, jota laajentamalla saadaan yksinkertaistettua ja tehostettua tietokantatoimintoja. Luokka tarjoaa stored procedures –perusteiset insert(), query(), update() ja delete() –metodit, joille olion tiedot välitetään ContentValues-säiliöluokan (container) avulla. ContentValues sisältää avain-arvo-pareja SharedPreferences-luokan tapaan. Muut tietokantatoiminnot, kuten taulujen luominen pitää suorittaa SQL-lauseilla execSQL()-metodia käyttäen. (Gargenta & Nakamura 2014, 176–178, 186–187. Android b.)

2.4 iOS 7

Apple mullisti älypuhelinmarkkinat esittelemällä iPhoneen tammikuussa 2007. Samalla yhtiö julkisti myös oman työpöytäkäyttöjärjestelmänsä, OS X:n, mobiiliversion iPhone OS:n, joka isoveljensä tavoin perustui Unix-tyyppiseen järjestelmäyttimeen. Tämä käyttöjärjestelmä, kuten ensimmäinen iPhonekin, aloitti uuden mobiilialustan ja aikakauden. Järjestelmä sisälsi hyvin rajoittuneen moniajon, joka mahdollisti ainoastaan musiikin kuuntelun taustasovelluksesta. Käyttäjä eristettiin laitteen tiedostojärjestelmästä. Järjestelmä esitteli käyttöliittymän kotinäkymän, johon käyttäjä pääsi palaamaan mistä tahansa sovelluksesta yhdellä napin painalluksella. Tärkein uusi ominaisuus oli kuitenkin käytettävyys. (The Verge, 2013.)

Heinäkuussa 2008 Apple esitteli iPhone OS version 2, joka sisälsi sovelluskaupan ja sovelluskehitystyökalut (SDK). Samalla Apple rajasi puhelimeen asennettavien sovellusten ainoaksi lähteeksi oman App Storensa ja alkoi valvoa siellä julkaistavia sovelluksia

tiukoin rajoituksin. Kesäkuussa 2010 julkaistun neljännen version myötä käyttöjärjestelmä nimi muuttui iOS 4:ksi. Viimeisin versio, iOS 7, julkaistiin syksyllä 2013. Se tukee edeltäjiensä tavoin Applen laajentunutta mobiililaittevalikoimaa toimien iPhoneen lisäksi iPod Touch –mediatoistimessa sekä isommilla näytöillä varustetuissa tableteissa iPad ja iPad minissä. (The Verge, 2013.)

Toisin kuin Android 4 ja Windows Phone 8, Applen iOS 7 on täysin suljettu käyttöjärjestelmä vain Applen valmistamien mobiililaitteiden käyttöön. Tämän vuoksi järjestelmän tukemien laitteiden ominaisuudet on tarkoin määriteltä. Näytöt ovat iPhoneissa kooltaan 3,5 tuumaa (480*320 point) ja 4 tuumaa (568*320 point) ja iPadeissa 7,9 tuumaa ja 9,7 tuumaa (kumpikin 1024*768 point). Tällä hetkellä myytävät laitteet on varustettu Applen kehittämällä retina-näytöllä, jossa on kaksinkertainen määrä pikseleitä. Sovelluskehittäjiä varten Apple käyttää kuitenkin määrittelyissä ”point”-yksikköä. Käyttöliittymäelementit muutetaan retina-näyttöisellä laitteella automaattisesti kertoimella 2 suurempaan resoluutioon sopiviksi. (Sharp, Sadun & Strougo, R. 2013, 15.)

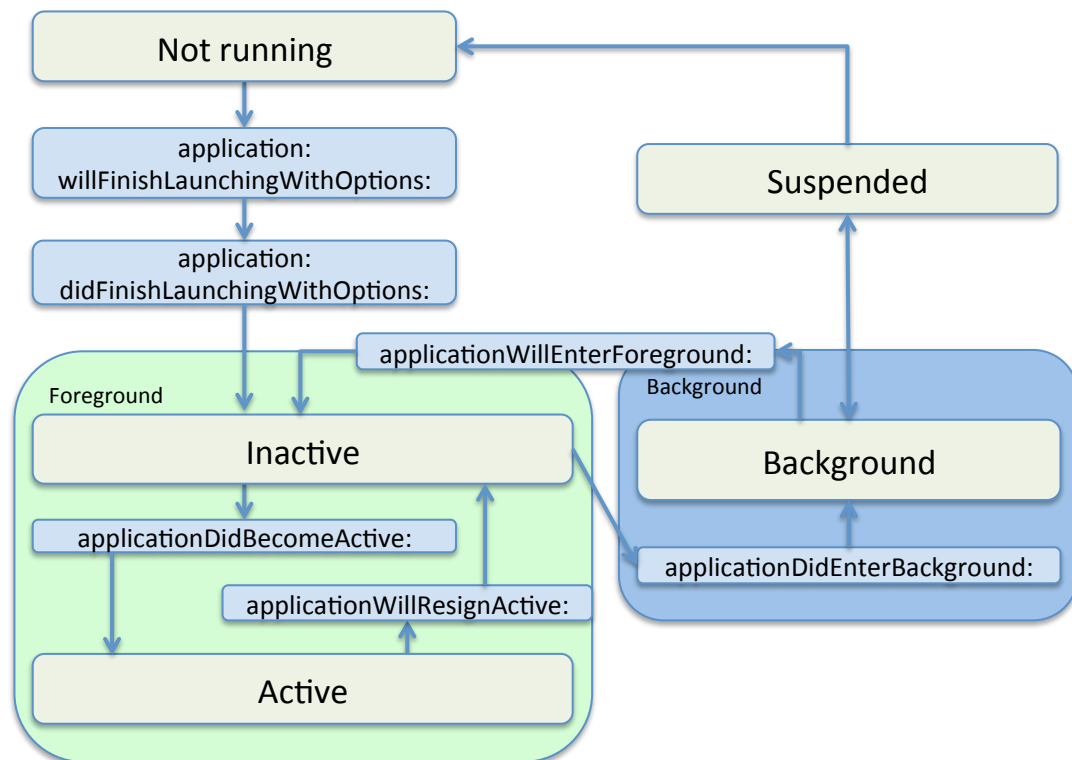
Apple tarjoaa ilmaiset kehitystyökalut Mac App Store –sovelluskaupasta ladattavana iOS SDK:n sisältävän Xcode-sovelluskehitysympäristön (IDE) muodossa. Työkalut vaativat alustaksi Applen tietokoneen, jossa on OS X -käyttöjärjestelmä. Apple on valinnut käyttöjärjestelmiensä ohjelmointikieleksi Objective-C-kielen, joka on C-kielen oliopohjainen laajennus. Standardi C-kielen päälle yhdistetään Smalltalk-kielen olio- ja viestijärjestelmä, jonka kehittämistä Apple jatkaa aktiivisesti. Sovellukset käännetään suoraan natiivisovelluksiksi, ja ne hyödyntävät Cocoa Touch ja UIKit –luokkakirjastoja (Mac OS X:n vastineet ovat Cocoa ja AppKit). Niiden avulla sovellusten käyttöliittymät on helppo toteuttaa Applen suunnitteluohjeiden mukaisesti. Lisäksi käyttöliittymäelementtien toteutus omina kirjastoinaan auttaa pitämään sovelluksen tiedostokoon varsin pienenä. SDK sisältää iOS-simulaattorin, jolla kehitettäviä sovelluksia voi testata maksutta. Simulaattori kuitenkin käyttää Mac OS X:n luokkakirjastoja, laajempaa muistikapasiteettia sekä Intel x86 –prosessoria iOS-laitteiden kirjastojen, enintään 1 GB:n muistin ja ARM-prosessorien sijaan. Liittymällä Applen maksulliseen iOS-kehittäjäohjelmaan kehittäjä voi rekisteröidä laitteita kehitys- ja testauskäyttöön. Sovellukset asennetaan omiin hiekkalaatikoihinsa, eivätkä ne voi jakaa toistensa tiedostoja tai järjestelmän mukana tulevien sovellusten dataa. Sovelluksen luoman dokumentin voi

siirtää avattavaksi toiselle sovellukselle, mutta tällöin se kopioidaan uuden sovelluksen tiedostoihin. (Sharp ym. 2013, 1–3, 5, 7–8, 11–12.)

Applen käyttämä arkkitehtuurimalli on MVC. Toteutettavat luokat jaetaan kolmeen toimintokokonaisuuteen. Model sisältää sovelluksen datan, ja kapseloi sen erillisiksi luokiksi. View tarjoaa sovelluksen visuaalisen käyttökokemuksen. Se sisältää sovelluksen visuaaliset elementit määritellen sen, miten Modelin tieto esitetään. Controller sijaitsee Modelin ja View:n välissä koordinoiden sovelluksen toimintaa ja tiedon kulkua. (Sharp ym. 2013, 318.)

Kuviossa 5 on esitetty iOS 7 –sovelluksen elinkaari. Ennen käynnistämistään tai tultuaan järjestelmän sulkemaksi sovellus on Not running –tilassa. Inactive-tilassa sovellus on yleensä vain lyhyen hetken ennen siirtymistään seuraavaan tilaan. Se on tällöin käynnissä ja etualalla, muttei vielä vastaanota tapahtumia. Täysin käynnissä oleva sovellus on Active-tilassa, kunnes siirtyy takaisin Inactive-tilaan. Tämän jälkeen sovellus voi siirtyä Background-tilaan. Siinä sovellus ei ole näkyvissä, mutta voi kuitenkin suorittaa tehtävänsä loppuun. Sovellus voi myös käynnistyä suoraan Background-tilaan Inactiven sijaan. Lopulta Suspended-tilassa järjestelmä on siirtänyt sovelluksen taustalle ja pysäyttänyt sen suorituksen ilman ennakoilmoitusta. Sovellus on yhä muistissa, mutta järjestelmä voi koska tahansa poistaa sen, mikäli aktiivinen sovellus tarvitsee lisää muistia. (Apple a.)

Kuten muissakin järjestelmissä, myös iOS-sovellus saa tiedon tilansa vaihdoksien tapahtumisesta erilaisten metodikutsujen välityksellä kuvion 5 mukaisesti. Esimerkiksi sovelluksen siirtyessä taustalta Background-tilasta Active-tilaan suoritetaan ensin `applicationWillEnterForeground:` ja varsinaisen tilanmuutoksen jälkeen `applicationDidBecomeActive:` -metodi. Sovelluksen käynnistyessä voidaan palauttaa tallennettua tilatietoa `application:willFinishLaunchingWithOptions:` -metodissa. Nämä tiedot tulee iOS 7:ssä tallentaa jo `applicationWillResignActive:` -metodissa, jotta tallennus saadaan tehtyä myös silloin, kun käyttäjä sammuttaa laitteen sulkematta sovellusta. (Neuburg 2013, 884, 887.)



Kuvio 5. iOS 7 -sovelluksen elinkaari (Apple a).

iOS 7 tukee rajoitettua moniajoa. Yksinkertaisimmillaan sovelluksen siirtyessä takalalle se voi jatkaa vielä hetken ja suorittaa käynnissä olevat tehtävänsä loppuun. Taus-talla oleva (Background) tai täysin pysähtynyt (Suspended) sovellus voi myös käyttää ajastettua ilmoitusta käyttäjän muistuttamiseen. Pitkäkestoiset taustatehtävät ovat mah-dollisia, jos sovellus joko toistaa tai tallentaa ääntä, seuraa ja ilmoittaa käyttäjän sijaintia reaaliaikaisesti, hyödyntää Voice over Internet –Protokollaa (VoIP), lataa ja prosessoi säännöllisesti pieniä määriä uutta sisältöä tai vastaanottaa säännöllisiä päivityksiä ulkoi-silta laitteilta. Esimerkiksi taustalla säännöllisesti haettava nettisivu toteutetaan applica-tion:performFetchWithCompletionHandler-metodiin sijoitetulla koodilla. Järjestelmä käynnistää sovelluksen sopivina hetkinä suoraan Background-tilaan ja kutsuu tätä me-todia. Kun sovellus metodin suorituksen lopuksi välittää järjestelmälle tiedon latauksen päätymisestä, järjestelmä siirtää sovelluksen jälleen suspended-tilaan. (Apple a. Sharp ym. 2013, 13–14.)

Sovelluksen tietojen pysyvään tallennukseen iOS tarjoaa kolme eri tarkoituksiin sovel-tuvaa menetelmää. Kaikki tallennus tapahtuu sovelluksen omaan hiekkalaatikkoon, joka on turvallisesti muiden sovellusten ulottumattomissa. Yksinkertaisten tila- ja asetustie-

tojen tallennukseen on käytettävissä `NSUserDefaults`, joka muodostuu käytännössä avain-arvo-pareja tallentavasta `NSDictionary`-luokasta. Se tallentaa sisältämänsä tiedot automaattisesti tiedostoon aina tarvittaessa. Property list –muotoisena se hyväksyy arvotyypeiksi `NSString`, `NSData`, `NSDate`, `NSNumber`, `NSArray` ja `NSDictionary`-olioita. Monimutkaisemman tiedon tallennus on mahdollista erilliseen tiedostoon. Useimmat tietoluokat (esimerkiksi `NSString`, `NSData`, `NSArray` jne) tarjoavat valmiit `writeToURL` ja `initWithContentsOfURL` –metodit jotka tallentavat ja lukevat olion suoraan URL-osoitteen määrittämään tiedostoon. `NSString` ja `NSData` suorittavat automaattisen olion sarjallistamisen tallennusta varten, mutta muut luokat käsittelevät property list –tyyppistä tietoa `NSUserDefaults`-luokan tapaan. Ratkaisuna iOS tarjoaa `NSCoding`-protokollan, jonka toteuttamalla mikä tahansa olio voi muuntua `NSData`-muotoon ja takaisin. Tällä tavoin saadaan tallennettua kaikenlaisia olioita. Jäsennellyn tiedon tallennukseen iOS sisältää `SQLite`-relaatiotietokannan, jota käytetään C-kielellä suoraan `SQL`-lauseiden avulla. iOS sisältää myös oman ORM-toteutuksen, `Core Data`n, joka tarkkailee hallinnoimiaan olioita ja tarvittaessa päivittää ne tietovarastoon, esimerkiksi `SQLite`-tietokantaan. (Neuburg 2013, 793, 796, 798, 821.)

`Core Data` frameworkia on luonnehdittu monimutkaiseksi ja vaikeaksi käyttää, mutta toisaalta se myös optimoi tehokkaasti muistinkäytön ja helpottaa erityisesti toisiinsa linkittyvien olioiden pysyvää varastointia (Neuburg 2013, 821–822). Tietojenkäsittely on jaettu `Core Data`ssa kolmeen pääkerrokseen. `Store coordinator` -kerros sisältää tiedon fyysisen tallennuksen ja tallennuspaikan, `Managed object model` –kerros määrittelee tallennettavien olioiden muodon ja lopulta `Managed object context` tarjoaa välineet hallittujen olioiden käsittelyyn muualta sovelluksesta. Olioiden attribuutit ja suhteet kuvaavia entiteettejä, jotka sijaitsevat `Managed object model` –kerroksessa, muokataan tavallisesti `Xcoden` graafisilla työkaluilla koodamisen sijaan. (Sharp ym. 2013, 319.)

3 Case: Pakettivahti-sovellus

Tässä luvussa esitellään opinnäytetyön produkti-osuus. Se koostuu Pakettivahti-sovelluksen prototyypin suunnittelusta ja toteutuksesta iOS 7, Android 4 ja Windows Phone 8 –laitteille. Sovellusprototyyppien lähdekoodit ovat liitteessä 3.

3.1 Sovelluksen suunnittelu

Sovelluksen suunnittelu alkaa sen toiminta-ajatuksen esittelyllä. Varsinaisen toiminnallisuuden määrittelemiseksi kuvataan seuraavaksi sovelluksen käsittelemän tiedon elinkaari. Tästä kuvauksesta saadaan määriteltyä tarvittavat käyttötapaukset, joiden pohjalta laaditaan kuvaus sovelluksen käyttämisestä tiedoista. Lopuksi saadaan luonnosteltua sovelluksen käyttöliittymä.

Sovelluksen tarkoituksena on saapuvien postilähetysten seuraaminen Postin verkkopalvelua hyödyntämällä ja niiden saapumisesta ilmoittaminen. Posti on julkaissut mm. tämän toiminnallisuuden sisältävän oman mobiilisovelluksensa kaikille kolmelle käytännöllä järjestelmälle. Se on kuitenkin harmillisesti rajoittunut Postin järjestelmään jo kirjattuihin lähetyksiin, eikä siten sovellu ulkomailta saapuvien lähetysten seuraamiseen ennen lähetysten ensimmäistä kirjaamista Suomessa. Suunniteltuun sovellukseen syötetään lähetysten seurantakoodi ja lähetykselle annetaan sitä kuvaava nimi. Tiedot tallennetaan paikallisesti ja lähetysten seurantatieto päivitetään säännöllisesti Postin verkkopalvelusta. Kun lähetys on valmiina noudettavissa, sovellus antaa käyttäjälle ilmoituksen. Noutaessaan lähetystä käyttäjä voi näyttää puhelimestaan lähetystunnuksen postivirkailijalle. Lähetysten tilan osoittaessa lähetysten tulleen noudetuksi sovellus poistaa lähetysten tiedot automaattisesti.

Tarkoituksena on suunnitella minimalistinen sovellus, joka toimii itsenäisesti taustalla käyttäjän syötettyä lähetystiedot. Näin käyttäjä voi huoletta unohtaa odottamansa lähetysten, kunnes sovellus ilmoittaa sen saapuneen ja odottavan noutoa.

Sovelluksen käyttötapaukset ovat liitteessä 2. Niiden perusteella voidaan havaita sovelluksen käyttävän jokaisesta lähetyksestä taulukossa 1 esitetyt tiedot. Lähetystunnuksen pituus on joko 13 tai 21 merkkiä lähetyksen tyypistä riippuen.

Taulukko 1. Lähetyksen tiedot

Nimi	Tyyppi	Selitys
Lähetystunnus	String(13 tai 21)	Postilähetyksen seurantatunnus
Nimi	String	Käyttäjän lähetykselle antama lähetystä yksilöivä nimi. Jos käyttäjä ei syötä nimeä, näytetään sen paikalla lähetyksen seurantatunnus.
Aika	DateTime	Viimeisimmän seurantatilan aika: pp.kk.vvvv hh.mm
Tila	Enum	Lähetyksen tila: ”Ei tietoja.”, ”Rekisteröity” tai ”Noudettavissa”

Sovelluksen käyttöliittymä tulee suunnitella kunkin mobiilikäyttöjärjestelmän omien käyttöliittymäsuunnitteluohjeiden mukaisesti ja käyttöliittymäkomponentteja hyödyntäen, jotta sen visuaalinen ilme ja toiminnallisuus vastaavat kyseisen järjestelmän muiden sovellusten toimintaa. Alustava hahmotelma käyttöliittymästä tietosisältöineen on esitetty kuviossa 6.



Kuvio 6. Käyttöliittymäluonnos.

Vasemmalla on näkymä, johon sovellus käynnistyy. Seurattavien lähetysten luettelo on järjestetty seurantatapahtuman ajan mukaan siten, että viimeisimmät tapahtumat ovat listan alussa. Noutoa odottavat lähetykset on korostettu. Keskellä on uuden lähetyksen

lisäysnäkö, joka käynnistyy aloitussivulta lisäysnappia painamalla. Lopuksi oikealla on lähetystunnuksen näyttönäkö, johon päästään aloitusnäköstä halutun lähetyksen riviä painamalla.

Sovellus toteutetaan inkrementaalisesti vaiheittain. Jokainen toteutusvaihe tuottaa testattavan prototyypin, jonka avulla voidaan varmistaa sovelluksen toimivuus hyväksymistestauslähtöisesti. Työvaiheet ovat

1. Käyttöliittymä
2. Lähetystietojen käsittely tietokannassa
3. Verkkopalvelun käyttäminen
4. Verkkopalvelun käyttäminen taustalla
5. Saapuneesta lähetyksestä ilmoittaminen.

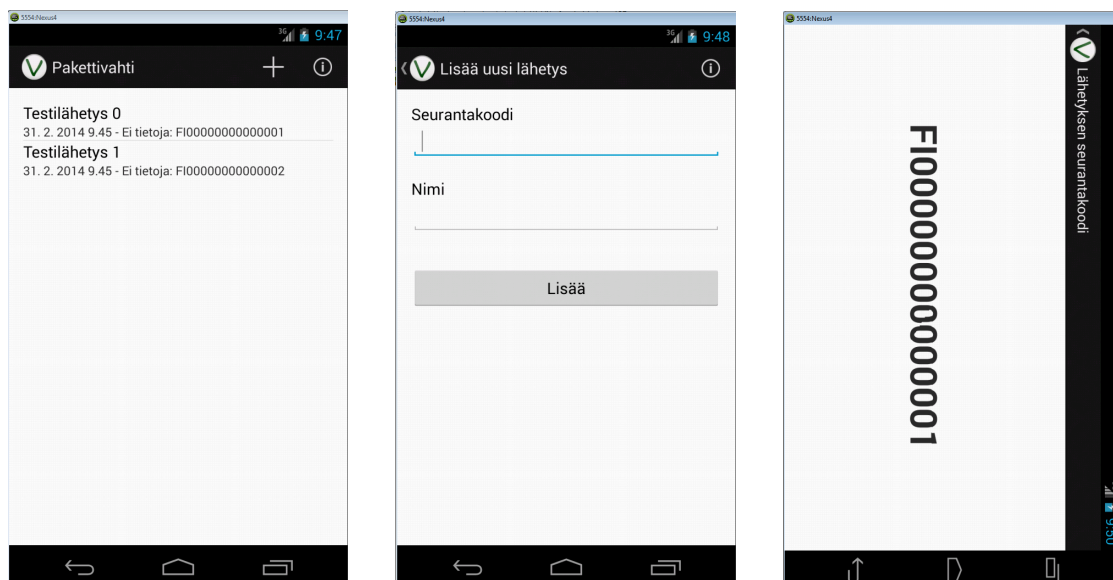
Seuraavaksi kunkin työvaiheen toteutus esitellään omana alalukunaan.

3.2 Käyttöliittymän toteutus

Kunkin sovelluksen toteutuksen aluksi rakennettiin sovellusrunko ja käyttöliittymä tarvittavin näkymin. Tässä ensimmäisessä toteutusvaiheessa lähetysten tiedot olivat yksinkertaisessa luettelossa, eikä niitä tallennettu pysyvästi.

Android-versiossa jokaista näkymää vastaa Activity, joka muodostuu käyttöliittymäelementtien sijoittelusta xml-tiedostossa ja varsinaisesta Java-luokasta. Näköstä toiseen siirtyminen tapahtuu Intent-olion välityksellä. Sille kerrotaan, minkä luokan Activity halutaan avata ja mitä tietoa sille halutaan välittää. Tietojen välitys tapahtuu avain-arvopareina intentin Extra-attribuutissa. Lisätyt tiedot saadaan prototyypin kehityksen tässä vaiheessa välitettyä päänäkömään luomalla uusi intent-olio, johon tiedot tallennetaan yhdessä kutsuneen intentin onnistuneesta päättymisestä kertovan koodin kanssa. Tämän jälkeen tietojenlisäysnäkö voidaan sulkea, jolloin päänäkö palaa näyttöpinon päällimmäiseksi. Siellä siirrytään onActivityResult-metodiin, jossa tietoja palauttava intent tunnistetaan ja sen sisältämät uuden lähetyksen tiedot lisätään luetteloon.

Jokainen sovelluksen Activity on rekisteröitävä AndroidManifest.xml-tiedostoon. Siellä määritellään myös sovelluksen käynnistykseen yhteydessä näkyviin tuleva päänäkymä. Samoin Action Barin navigointiominaisuuteen liitettävä kotinäkymä määritellään jokaiselle Activitylle tässä tiedostossa.



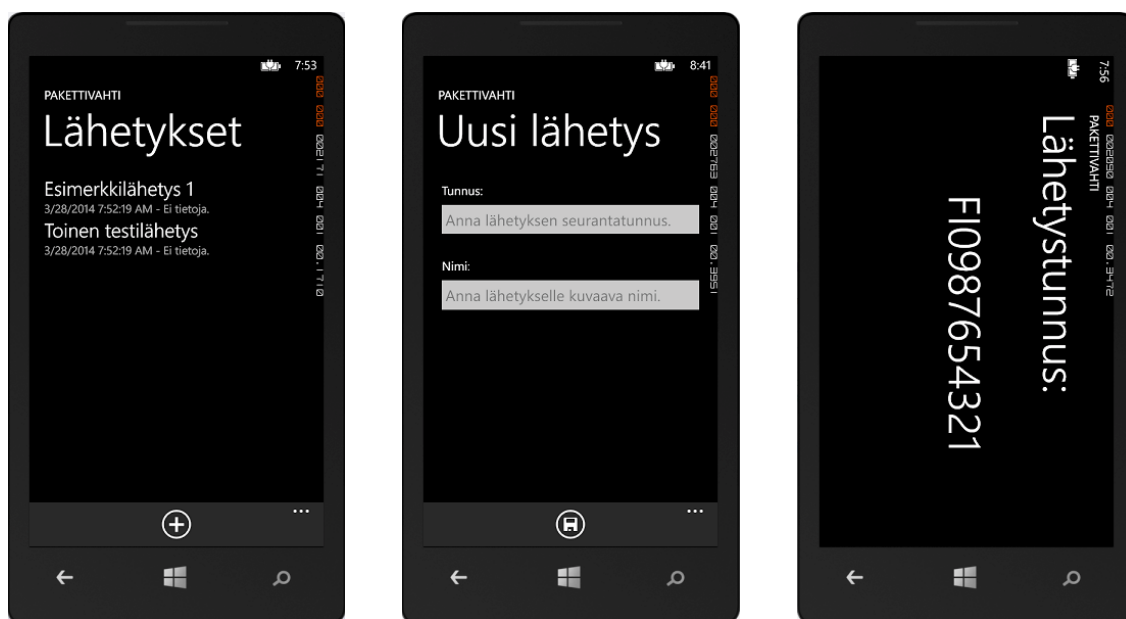
Kuvio 7. Android-version käyttöliittymä emulaattorissa.

Kuviossa 7 on esitetty Android-version käyttöliittymätoteutus. Näkymän yläosassa sijaitsee Action Bar, joka sisältää tärkeimpien toimintojen ikonit. Sovelluksen ikoniin kytkeytyy paluutoiminto, joka tuo käyttäjän takaisin sovelluksen pääsivulle. Järjestelmä hakee käyttöliittymän kaikki tekstit strings.xml-resurssitiedostosta, joten sovelluksen mahdollinen lokalisointi muille kielille onnistuu kopioimalla ja kääntämällä tämä tiedosto uudelle kohdekielelle.

Käyttöliittymän toteutus iOS-järjestelmään on esitetty kuviossa 8. Applen graafisen ilmeen uudistuksen jälkeen suunnitteluohjeiden mukainen käyttöliittymä on hyvin pelkistetty ja selkeä. Näkymän yläosassa on Navigation Bar, joka sisältää näkymän otsikon sekä päätoimintonapit, jotka iOS 7:ssä on toteutettu tekstin kaltaisina. Niiden teksti vaihtaa kieltä käytettävän laitteen lokalisaaation mukana. Simulaattorin lokalisatio on amerikkalainen, joten kuviossa toimintojen nimittekstit ovat englanninkielisiä.

män kontrolleriin sen sijaan asetetaan oma delegaatti, jonka metodit toteuttaa lähetysoi-
toliot varastoiva luokka. Lisäysnäkyä pyytää aikanaan delegaatin kautta viittauksen
uuteen lähetysoi-
lioon, johon syötetyt tiedot asetetaan.

Kuviossa 10 on esitetty Windows Phone 8 –version käyttöliittymätoteutus. Microsoftin
suunnitteluohjeiden perusteella toteutettu käyttöliittymä sisältää jokaisessa näkymässä
sovelluksen nimen ylimpänä pienikokoisilla kapitaaleilla kirjoitettuna. Sen alla on kun-
kin näkymän tai sivun nimi. Käyttöliittymä hyödyntää järjestelmän oletusasetuksia ja
mukautuu siten käyttäjän valinnan mukaiseen yleisilmeeseen, joka esimerkkikuvissa on
asetettu tummaksi. Samoin Lähetykset-sivulla luettelossa alemmalla rivillä esitetty päi-
vämäärä ja kellonaika lokalisoidaan kulloistenkin käyttäjäasetusten mukaiseen muotoon,
joka emulaattorissa on amerikkalainen.



Kuvio 10. Windows Phone 8 -sovelluksen käyttöliittymä emulaattorissa.

Windows Phone 8:n toiminnot sijoitetaan sivun alareunan App Bar –toimintopalkkiin,
jonka ikonien alle saa näkyviin toimintojen nimet kolmipisteistä ikonia painamalla. App
Bar osoittautui erityisen käteväksi uuden lähetys lisäyssivulla, jossa virtuaalinäp-
päämistö peittäisi erillisen tallennusnapin tekstikenttien alapuolelta. Järjestelmä sijoittaa
näppäimistön App Barin yläpuolelle, jolloin toimintopalkki on aina käytettävissä sa-
manaikaisesti kirjoituksen kanssa. Kuvien oikeassa reunassa olevat numerokoodit liitty-
vät emulaattorin virheenjäljitystoimintoihin, eivätkä kuulu lopulliseen käyttöliittymään.

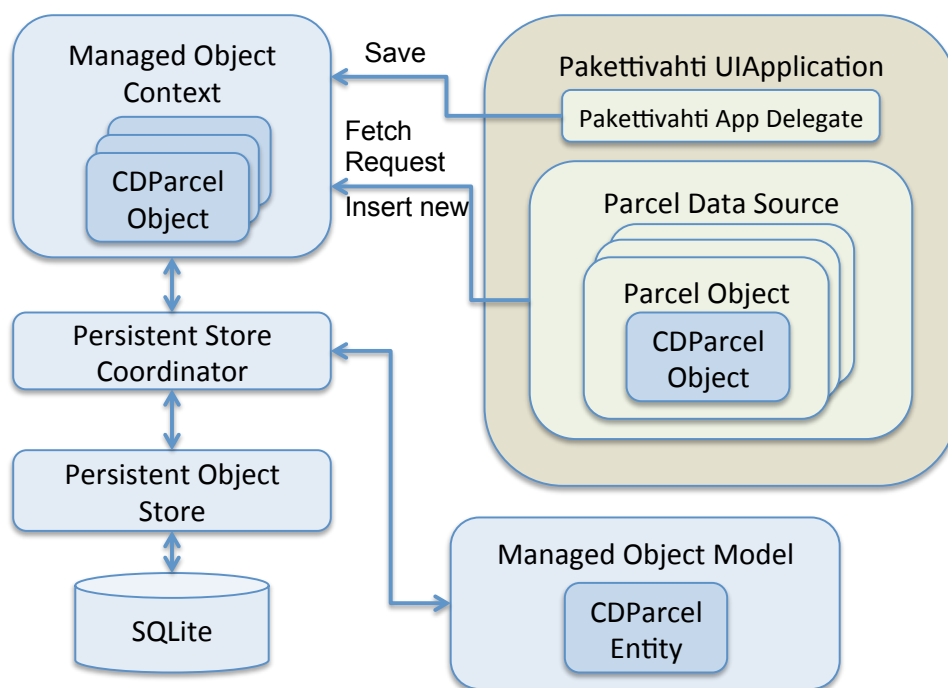
3.3 Sovelluksen tietojen tallennus

Tietojen tallennuksen yhteydessä tarkoituksena oli tutustua kunkin järjestelmän ORM-ominaisuuksiin. Näin olleen hyödynnettiin Windows Phone 8 –versiossa LINQ to SQL:ää ja iOS-versiossa Core Dataa. Koska Android ei suoraan tarjoa välineitä olioiden tallennukseen tietokantaan, toteutettiin tallennus järjestelmän olemassa olevia välineitä hyödyntäen. Ilmeisesti Androidille on saatavilla lukuisia kolmansien osapuolien kehittämia ORM-ratkaisuja. Niiden sijaan oli kuitenkin tarkoituksenmukaista tutustua tallennuksen toteutukseen järjestelmän tarjoamin menetelmin, koska näin tarjoutuu tilaisuus oppia ymmärtämään paremmin Android-sovelluksen kokonaisuutta ja järjestelmään sisältyviä ominaisuuksia.

iOS 7 tarjoaa Core Data -sovelluskehiksen (framework) kautta valmiit toiminnot olioiden tallennukseen. Xcodessa projektiin pitää linkittää Core Data framework. Tämän jälkeen voidaan hallittavaksi haluttu olio luoda Xcoden Data Model –työkalulla, jolla kuvataan tallennettava entiteetti attribuutteineen sekä muun muassa asetetaan niille mahdolliset oletusarvot. Xcode tallentaa tietomallin `ParcelDataModel.xcdatamodel/content` –tiedostoon. Kuvauksen valmistuttua työkalu generoi valmiin luokan `.h`- ja `.m`-tiedostot. Tämän luokan kantaluokkana on `NSObject`-luokan sijaan `NSManagedObject`, mikä tarkoittaa, että luokka sisältyy hallittujen olioiden kokoelmaan (Managed Object Context). Core Data tarkkailee tässä kokoelmassa oleviin olioihin tehtäviä muutoksia ja päivittää tarvittaessa taustalla käyttämänsä tietokannan vastaamaan kulloistakin tilaa. Uudet hallittaviksi tarkoitetut oliot luodaan kuvion 11 mukaisesti Managed Object Contextin sisään `InsertNew`-metodilla, jotta myös niiden muutoksia seurataan.

Kehitettävän sovelluksen `AppDelegate`en on määriteltävä käytetty tietomalli (`CDParcelEntity`) Managed Object Modelille, sekä olioiden tallennukseen käytettävä `SQLite`-tietokantatiedosto `Persistent Store Coordinator`ille kuvion 11 mukaisesti. `AppDelegate`essa määritetty Managed Object Context –olio saadaan koko sovelluksen käyttöön. Sovelluksen siirtyessä taka-alalle tai sulkeutuessa järjestelmä kutsuu `AppDelegate`en vastaavaa metodia, jossa contextia pyydetään tallentamaan mahdolliset tallentamattomat muutokset. `Parcel Data Source` hakee sovelluksen context-olion ja pyytää sen `FetchRequest`-metodin avulla hallinnoidut lähetystieto-oliot (`CDParcel`). Nämä kääritään `Parcel`-

olioiden sisään myöhemmän käsittelyn käyttämien metodien liittämiseksi. Suoraan Data Modelin automaattisesti generoimaan luokkaan (CDParcel) käsin tehdyt muutokset tai lisäykset menetetään aina Data Model -työkalua uudelleen käytettäessä.



Kuvio 11. Tiedonhallinta iOS 7:n Core Datan avulla.

Hallinnoitujen CDParcel-olioiden attribuutit eivät voi olla primitiivityyppejä vaan NSObject-kantaluokasta periytyviä olioita. Tämän vuoksi esimerkiksi kokonaisluku esitetään int-primitiivityypin sijaan NSNumber-oliona. Objective-C-kielessä taas enumeroidut tyypit eivät voi muodostua olioista. Lähetyksen tilatiedon hallinta enumeroidun tyypin avulla tapahtuu kääreluokassa Parcel. Se muuntaa CDParcel-olion tilatiedon NSNumber-oliosta kokonaislukuperusteiseen primitiivityyppiin int, jonka kanssa voi käyttää lähetyksen tilaa kuvaavia ja koodia selkeyttäviä enumeraatioita.

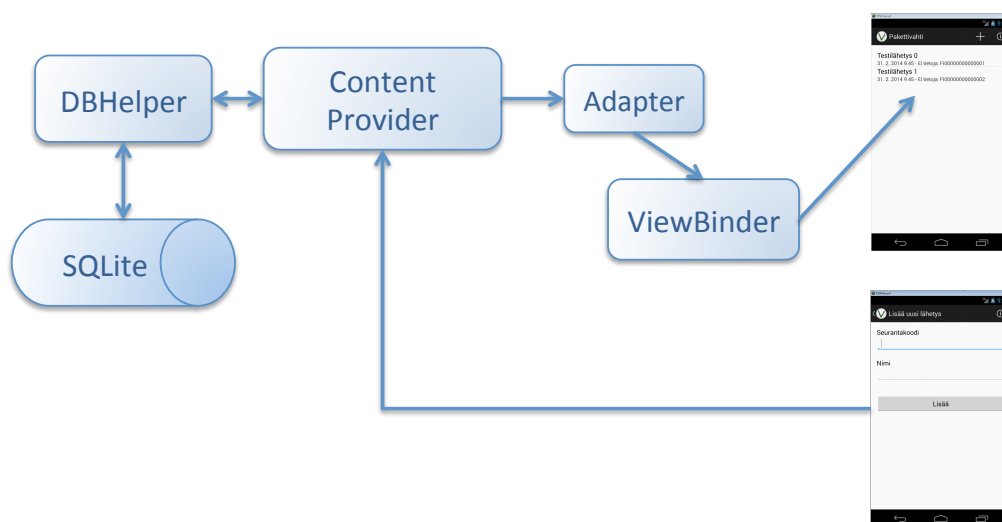
Lähetystiedot esittävän TableView:n solujen sisältö on koottu lähetysolioista merkkijonoiksi delegaattimetodissa. Tämän vuoksi mahdolliset lähetystietojen päivitykset eivät välity esitettäviin tietoihin, joten päivitysten yhteydessä on pyydettyä TableView:tä päivittämään sisältönsä. Päivitys on kuitenkin nopea toimenpide, koska TableView pyytää kaikkien lähetyksen sijaan vain kulloinkin näkyvillä olevien solujen tiedot.

Android-version toteutuksessa oli odotetusti enemmän tekemistä. Aluksi luotiin lähetysten tiedot sisältävän Parcel-luokka ja lähetysten seurantatilan määrittävä Status-enumeraatio. Näiden jälkeen oli SQLiteOpenHelper-luokasta periytyvän DBHelper-luokan sekä tietokantatoiminnoissa tarvittavat vakiomäärittelyt sisältävän ParcelContract-luokan vuoro. SQLiteOpenHelper-luokan konstruktorille annetaan tietokantatiedoston nimi ja versionumero, jonka avulla päätellään myöhemmin tarve tietokannan päivittämiseksi. DBHelper-luokkaan lisättiin uudet toteutukset tietokannan luovalle onCreate-metodille sekä tietokannan päivittäväälle onUpgrade-metodille. Tietokannan luomisen yhteydessä suoritetaan SQL-luontilause, ja metodia kutsutaan myös päivityksen yhteydessä lähetystiedot sisältävän tietokantataulun poistamisen jälkeen. SQLite sisältää automaattisen rivinumeropohjaisen kasvavan indeksoinnin, joten taulun pääavaimen sarakkeesta luodaan käytännössä rivinumeron alias. (SQLite.)

Android-sovelluksille tyypilliseen tapaan tietokantatoiminnot toteutettiin ContentProvider-luokasta periytyvän ParcelProvider-luokan kautta. Tällä tavoin tiedon tallennus on kapseloitu ja sen toteutusta voi vaihtaa muuta sovellusta muokkaamatta. Samoin toiminnot ovat tarvittaessa myöhemmin avattavissa myös muille sovelluksille. Merkittävä hyöty saadaan siitä, että provider-luokka välittää tiedon muuttuessa ilmoituksen tietojen päivittymisestä, jolloin kyseisen tai kyseisten lähetysten tiedot sisältävät olivat osaavat tarvittaessa päivittää niihin sisältyvät näkymät. Tietokannan tiedot ovat käytettävissä URI-osoitteen content://com.aueta.app.pakettivahti.dao.parcelprovider/parcels/id kautta. Tässä tietokantataulun nimi on parcels ja yksittäisen lähetystaulun rivin numero on id. Content provider -luokka palauttaa kaikki lähetykset, mikäli osoite päättyy pelkkään taulun nimeen. Toiminta on siis samantapaista kuin REST-tyyppisten verkkopalvelujen rajapinnan käyttäminen. Provider-luokka toteuttaa metodeillaan CRUD-toiminnot (Create, Read, Update, Delete) ja käyttää tietokantaa DBHelper-luokan instanssin kautta. Jotta Content Provider olisi sovelluksen käytössä, se pitää rekisteröidä sovelluksen AndroidManifest.xml-tiedostoon (Liitteessä 3, rivit 52-55). Rekisteröinnillä määritellään osoite, jossa Content Provider vastaa pyyntöihin. Authorities-määrittely sisältää käyttöoikeusnimen, joka sovelluksella pitää olla, jotta kyseinen Provider olisi sen käytettävissä. Lopuksi määritellään, että Content Provideria ei julkais- ta muiden sovellusten käyttöön.

Sovelluksen päänäkylässä lähetyksen tiedot esitetään ListView-oliossa. Tietokanta saadaan yhdistettyä suoraan luettelo-olioon SimpleCursorAdapter-olion kautta. Tälle kerrotaan, mikä luettelosolun mallin (list_item.xml -tiedostosta) kenttä yhdistyy mihinkin tietokantataulun sarakkeeseen. Koska luettelosolun määrittely muodostuu kahdesta TextView-kentästä, pitää tietokantataulun aika- ja status-sarakkeet yhdistää status-merkkijonoksi.

Tiedot yhdistetään adapterin käyttöön annettavan ViewBinder-luokasta periytyvän ParcelableViewBinder-olion avulla. Adapteri kutsuu sitä jokaisen täytettävän näyttökentän kohdalla, mikä mahdollistaa useamman tietokantasarakkeen yhdistämisen kyseiseen kenttään. Samalla on mahdollista korostaa taustaväriä muuttamalla noudettavissa olevan lähetyksen luettelosolu. Adapter saa Content Providerilta tietojen mukana myös yksittäisten tieto-olioiden URI-osoitteen. Olion tietojen muuttuessa tietokannassa Provider lähettää ilmoituksen sen URI:lle. Adapter kuuntelee sisältämiensä tietojen URI-osoitteissa ja saa siten tiedon päivityksistä, jotka se voi välittää näytölle, jossa sen tietoja on esillä. Kuviossa 12 on esitetty Android-sovelluksen tietojen tallennus- ja käyttöketju tietokannasta luetteloon sekä uuden lähetyksen lisäämisestä tietokantaan.



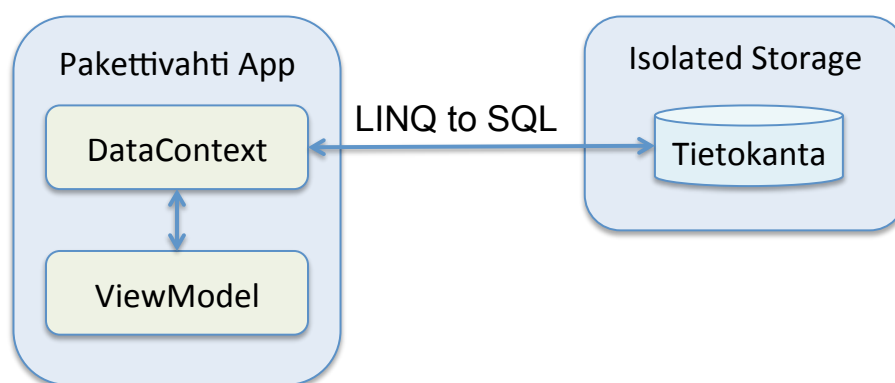
Kuvio 12. Tiedonhallinta Android-versiossa.

Koska Android ei sisällä valmista ORM-toteutusta, kävi ilmi, että sovelluksessa ei tarvita lainkaan varsinaista yksittäisen lähetyksen tiedot sisältävää luokkaa. Content Provider ei käsittele olioita vaan vastaanottaa tiedot avain-arvo-pareina. Tavallaan ListView-elementille tiedot tarjoava adapteri sisältää kokoelman ”olioita”, mutta senkin tietosisäl-

töön pääsee käsiksi vain avain-arvo-parien kautta. Koko sovelluksen toiminnallisuus on toteutettavissa ilman lähetystietoluokkaa, ja näin vältetään tässä yhteydessä tarpeettomalta olioiden muodostamiselta tietokannasta ja samoin niiden sarjallistamiselta tietokantaan.

Windows Phone 8 tarjoaa LINQ to SQL –ratkaisun olioiden pysyvään tallentamiseen tietokantaan. Tallennettava luokka merkitään [Table]-merkinnällä tietokantatauluksi. Samoin sen attribuutit merkitään tietokantataulun sarakkeiksi [Column]-merkinnällä. Näille voidaan antaa tarvittavia lisäohjeita kuten esimerkiksi Parcel.cs-tiedoston riveillä 78 ja 79 (Liite 3). Siinä Parcel-luokan id-attribuutista tehdään tietokantataulun pääavain, asetetaan tietokanta luomaan sille yksilöllisiä arvoja ja asetetaan se tarkistettavaksi aina tietojen päivityksen yhteydessä, eli päivitettävä olio valitaan sen perusteella.

Tietomalliluokan merkintöjen jälkeen kuviossa 13 esitetään DataContext-luokkaa laajentava toteutus (ParcelDataContext.cs, liitteessä 3), joka huolehtii varsinaisista tietokantatoiminnoista. Sille kerrotaan kontekstiin kuuluvat tietokantataulut, tässä tapauksessa siis lähetystiedot sisältävän taulu: Table<Parcel>, jolle LINQ to SQL toteuttaa automaattisesti tietokantatoiminnot SQL-lauseineen. Sovelluksen käyttöliittymän luetteloon lähetystiedot päätyvät ViewModel-luokan kautta.



Kuvio 13. Tietojen hallinta Windows Phone 8 -sovelluksessa.

Windows Phone 8 –sovelluksien suositeltu arkkitehtuurimalli on Model-View-ViewModel eli MVVM. ViewModelin tehtävänä on tarjota tietoa käyttöliittymäelementeille ja huolehtia vastaavasti niiden tietoihin haluamien muutosten toteutuksesta. Jotta ViewModel saisi tietoa päivittyneistä elementeistään, merkitään lähetystiedot sisältävä

luokka toteuttamaan `INotifyPropertyChanged`- ja `INotifyPropertyChanging`-rajapinnat. Jälkimmäinen optimoi muistinkäyttöä, kun taas ensimmäinen ilmoittaa varsinaisesti olion tietojen muutoksista. Myös `ViewModel`-luokka toteuttaa `INotifyPropertyChanged`-rajapinnan, jotta sen sisältämä `ObservableCollection<Parcel>` välittäisi tiedot sisällön muutoksista eteenpäin käyttöliittymäelementille. `ViewModel` käyttää kuvion 13 mukaisesti tietokantaa `DataContext`-luokan toteutuksen kautta. Aluksi tiedot luetaan kokoelmaan tietokannasta. Uusi lähetysolio tallennetaan sekä kokoelmaan että tietokantaan. Vastaavasti mahdollinen poistettava olio poistetaan luettelosta ja tietokannasta. Tämän jälkeen olioiden ylläpito on automaattista.

Itse sovelluksen muodostavaan `App.xaml.cs`-tiedostoon lisätään `ViewModel`-luokan olio, jolloin sen yksi ja sama ilmentymä saadaan koko sovelluksen käyttöön. Samalla tarkistetaan, että tietokanta on olemassa ja tarvittaessa luodaan uusi käyttäen `DataContext`-luokkaa. Sovelluksen pääsivun sisältämään `LongListSelector`iin kytketään tietoläheteeksi `ViewModel`in kokoelma `MainPage.xaml.cs`-tiedoston (liitteessä 3) rivien 94-95 mukaisesti. Luettelon lähetykset halutaan järjestää siten, että noudettavissa olevat ovat ylimpänä. Valitettavasti `ObservableCollection` ei tue järjestämistä, joten järjestäminen on toteutettava erillisen listan avulla.

Noudettavissa olevien lähetysten korostaminen taustavärin avulla onnistuu suoraan XAML-koodissa binding-ominaisuudella `MainPage.xaml`-tiedostossa (liitteessä 3) riveillä 58-59, kun toteutetaan `IValueConverter`-rajapinnan toteuttava luokka `StatusColorConverter`. `LongListSelector` kutsuu tämän luokan `Convert`-metodia, jonka avulla asetetaan palautettavaksi taustavärin nimeksi ”DarkGreen”, kun lähetykset ovat noudettavissa. Harmillisesti `Windows Phone 8` tai `Windows 8 RT` ei tue `Windows 8`:n tavoin `MultiBinding`-toimintoa, jolla olisi mahdollista koota aikaleimasta ja lähetyksen statusnumerosta oikein muotoiltu merkkijono `IMultiValueConverter`-rajapinnan avulla. Olisi toki mahdollista kytkeä statustekstikenttä esimerkiksi koko solun lähetysolioön ja rakentaa tälle oma converter-luokka. Ratkaisu ei kuitenkaan toimi, koska `PropertyChanged`-päivitysilmoituksen voi lähettää vain olion attribuuteista, ei koko oliosta. Koko olioön kytketty kenttä ei siten osaisi päivittyä yksittäisen attribuutin päivittyessä. Puutteen kiertäminen onnistuu kuitenkin luomalla lähetyksen tiedot sisältävään luokkaan oma `get`-metodi `FormattedStatus`, joka muodostaa tarvittavan merkkijonon luokan muista attri-

buuteista. Lähetyksen seurantatilan muuttuessa lähetetään PropertyChanged –ilmoitus itse statustiedon sijaan tästä yhdistelystä FormattedStatus-attribuutista, jolloin myös luettelo tietää päivittää solunsa sisällön. Toinen mahdollisuus olisi toteuttaa uusi ViewModel-luokka, joka toimii lähetyksion kääreluokkana. Tällöin olisi mahdollista ilmoittaa sen attribuuttina olevan lähetyksion muutoksesta. Varsinaisen ViewModel-luokan tulisi tällöin sisältää luettelo näistä ViewModel-käärityistä lähetyksioista.

3.4 Postin lähetyssurantaverkkopalvelun toiminta

Posti tarjoaa lähetyssurantapalvelustaan mobiililaitteille suunnatun kevennetyn version osoitteessa <https://posti.mobi/packet-tracking/AB123456789CD?lang=fi>. URL-osoitteen lopussa on lähetyksen seurantatunnuksen (esimerkissä AB123456789CD) jälkeen http-get –pyynnön parametrina palvelun kielivalinta (esimerkissä ?lang=fi, jolla pyydetään palvelusta suomenkielinen vastaussivu). Palvelu palauttaa html-sivun, jossa kerrotaan lähetyksen tilatietoja. Kuviossa 14 on sivun html-koodi sovelluksen toiminnan kannalta olennaiselta osalta. Esimerkin lähetyks on noudettu ja sen tilaksi on muutunut arkistoitu. Myös aikaisemmat tilat ovat näkyvillä.

```
<h3>Lähetyksen rekisteröinnit</h3>
<div class="events">
  <div class="ARK">
    <p>20 March 2014 14:59</p>
    <p>Lähetyks on rekisteröity.</p>
    <p>00230 Sähköinen arkistointi</p>
  </div>
  <div class="LUO">
    <p>14 maaliskuu 2014 18:35</p>
    ...
  </div>
  <div class="HYL">
    <p>13 maaliskuu 2014 07:18</p>
    ...
  </div>
  <div class="REK">
    <p>12 maaliskuu 2014 09:34</p>
    ...
  </div>
</div>
...
<footer>
```

Kuvio 14. Postin palvelun vastaussivun html-koodiesimerkki.

Koodin perusteella huomataan, että rekisteröidyn lähetyksen kirjatut tapahtumat löytyvät <div class="events"> ja <footer> -tagien välistä. Jokainen kirjaus näkyy oman div-taginsa erottamana ja jo tagin css-luokkamäärittely kertoo, minkä tilan kirjaus on kysees-

sä: REK (rekisteröity), HYL (noudettavissa), LUO (noudettu) ja ARK (arkistoitu). Kirjauksen aikaleima on luettavissa sen ensimmäisen <p>-tagin sisältä. Mikäli events-luokan div-tagia ei löydy, lähetystä ei ole vielä kirjattu järjestelmään tai lähetystunnus on virheellinen. Näiden havaintojen perusteella on mahdollista lukea palvelun palauttamasta html-sivusta yksittäisen lähetyksen tilatieto aikaleimoineen.

Sovelluskehityksen tueksi toteutettiin testauskäyttöön oma yksinkertainen palvelinohjelmisto PHP-kielellä. Se vastaanottaa http-get -pyynnön yhteydessä lähetyksen nykyisen tilakoodin ja lähettää vastauksena seuraavaa tilaa vastaavan tietosivun. Tietosivun sisältö vastaa Postin palvelun vastausta. Näin sovelluksen toimintaa on mahdollista testata toistuvasti ja hyvin nopeassa tempossa kuormittamatta Postin palvelimia ja ilman lukuisia todellisia postilähetyksiä.

3.5 Verkkopalvelun käyttäminen

Android-versiossa verkkopalvelun käyttäminen on luontevinta erillisen IntentService-luokan toteutuksen kautta (ParcelService.java, liitteessä 3). Tällöin erillisellä intentillä käynnistettävä toiminnallisuus suoritetaan onHandleIntent()-metodissa automaattisesti omassa säikeessään. Suorituksen päätyttyä IntentService pysäyttää itsensä odottamaan seuraavaa kutsua. Mikäli käynnistävään intenttiin on liitetty lisätietona lähetyksen seurantakoodi, haetaan vain tämän yhden lähetyksen tilatieto, kuten esimerkiksi uuden lähetyksen lisäyksen yhteydessä. Mikäli koodia ei ole annettu, haetaan kaikkien seurattavien lähetysten tila, kuten esimerkiksi sovelluksen käynnistyessä. Haluttu vastaussivu luetaan verkkopalvelusta HttpURLConnection-luokan avulla. Palvelimen vastaus luetaan BufferedInputStream-olioon, ja halutut tiedot erotellaan etsimällä tästä tietovirrasta. Verkosta päivitetty tieto tallennetaan Content Providerin kautta. Haettu tieto päivittyy sen ilmoittamana myös sovelluksen esittämään luetteloon. Activityjen tapaan myös service pitää rekisteröidä AndroidManifest.xml-tiedostossa sovelluksen osaksi.

Windows Phone 8 tarjoaa yksinkertaisiin verkkohakuihin soveltuvan WebClient-luokan, jossa verkkopalvelun käyttäminen tapahtuu suoraan asynkronisesti DownloadStringAsync-metodilla. Asynkroniset lataukset suoritetaan taustasäikeessä, jolloin käyttöliittymän päivitys ei katkea niiden ajaksi. WebClient-oliolle lisätään DownloadSt-

ringCompleted-tapahtumankäsittelijä, jota järjestelmä kutsuu sivun latauksen päätyttyä. Tässä metodissa käsitellään ladattu html-sivu merkkijonona tai todetaan, ettei sivua saatu ladattua. Metodille voidaan välittää DownloadStringAsync-kutsun yhteydessä tilatietona seurattavan lähetyksen id, ja tämän perusteella on tapahtumankäsittelijässä mahdollista kohdistaa päivitys oikeaan lähetysolio. Kohdeolio haetaan ViewModelista, jolloin siihen tehdyt päivitykset päivittyvät heti automaattisesti myös pääsivun LongListSelectoriin. Liitteessä 3 ParcelWebChecker.cs-tiedostossa esitettävä toteutus liittyy prototyyppiin seuraavaksi liitettävään taustatoimintoon. Tämän vuoksi se eroaa hieman tässä kerrotusta aiemmasta toteutusratkaisusta.

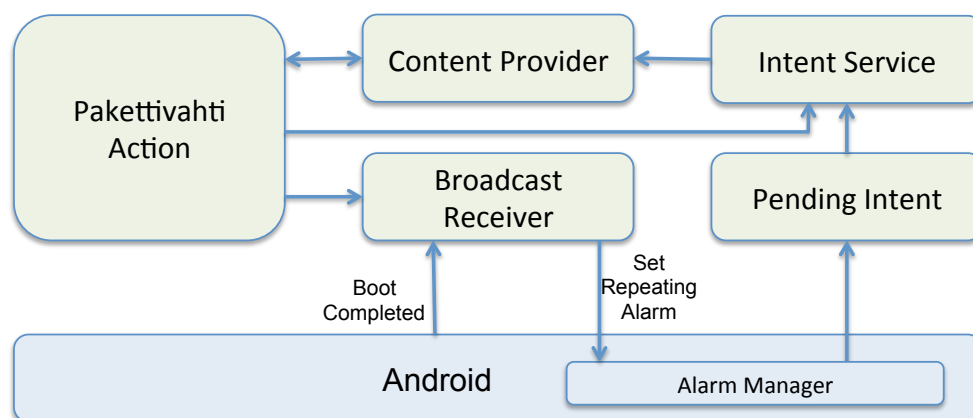
iOS 7:ssä verkkopalvelun käyttö on kätevintä toteuttaa tässä käyttöjärjestelmäversiossa uutuutena esiteltyä NSURLSession-luokkaa ja sen asynkronista NSURLSessionDownloadTask-metodia käyttäen. PakettivahtiViewController.m-tiedostossa (liitteessä 3) riveillä 190-191 metodille välitetään kutsun yhteydessä ladatun tiedon käsittelyyn tarkoitettu completion handler –metodi, jota kutsutaan uudessa säikeessä tapahtuneen latauksen päätteeksi. Näin käyttöliittymän päivitykset jatkuvat sovelluksen pääsäikeessä latauksen ja tuloksen käsittelyn aikanakin.

NSURLSessionDownloadTask-oliolle voi välittää NSURLRequest-pyyntön liitteenä mitä tahansa olioita. Kun tällä tavoin välitetään kyseisen latauksen kohteena oleva lähetysohje (Parcel), saadaan verkkopalvelusta ladattu tilatieto tallennettua taustasäikeessä suoraan oikeaan olioon. Näin voidaan muodostaa jokaisesta seurattavasta lähetysohiosta oma asynkroninen pyyntö NSURLSession sisällä, ja jokaisen pyynnön oma säike käsittelee vain kyseisen olion päivityksen. Olion päivityksen jälkeen taustasäikeestä voi lähettää komentoja pääsäikeen tehtäväjonoon suoritettavaksi rinnakkaisohjelmoinnin apuvälineitä sisältävän iOS:n Grand Central Dispatchin dispatch_async-metodin avulla (liitteen 3 PakettivahtiViewController.m-tiedostossa riviltä 358 alkaen). Näin saadaan lähetyksen tietojen päivittämisen jälkeen pyydettyä TableView-oliota päivittämään esittämänsä tiedot.

3.6 Verkkopalvelun käyttäminen taustalla

Android tarjoaa mahdollisuuden ajastettujen PendingIntent-olioiden lähettämiseen. Näin voidaan toteuttaa kuvion 15 mukainen lähetystietojen ajastettu päivitys taustalla. Sovelluksen ApplicationContext-olio sisältää viittauksen järjestelmän tarjoamiin palveluihin, joihin kuuluu mm. AlarmManager. Sen avulla saadaan verkkopalvelua käyttävä IntentService-toteutus käynnistettyä tietyin väliajoin myös silloin, kun sovellus ei ole käynnissä. Jotta ajastus olisi käytössä myös laitteen sammuttamisen ja uudelleen käynnistämisen jälkeen, on tarpeen toteuttaa BOOT_COMPLETED-eventtiä kuunteleva BroadcastReceiver. Se käynnistää ajastuksen laitteen käynnistyttyä, vaikka itse sovellus ei olisikaan käynnissä. Sovellukselle on myönnettävä AndroidManifest.xml-tiedostossa RECEIVE_BOOT_COMPLETED-lupa ja BroadcastReceiver-toteutus on myös rekisteröitävä osaksi sovellusta.

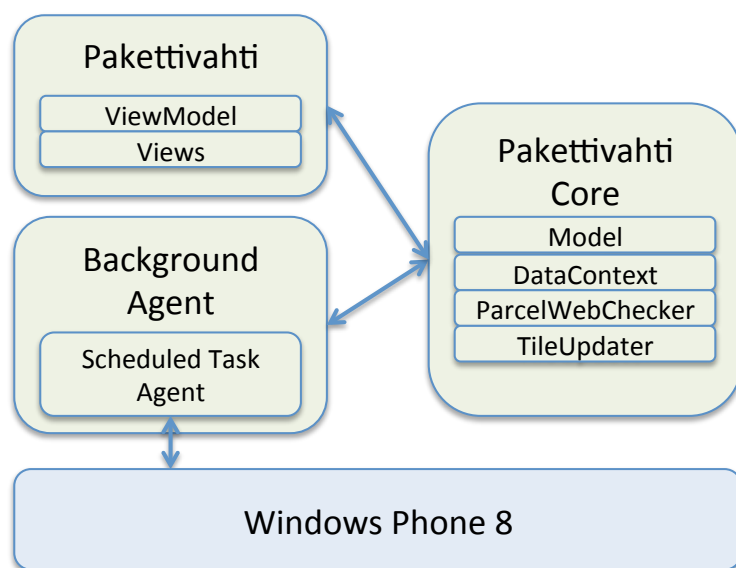
Ajastus on syytä kytkeä päälle myös sovelluksen käynnistyessä, jotta se olisi käytössä ennen laitteen ensimmäistä sovelluksen asennuksen jälkeistä uudelleenkäynnistystä. Tämä onnistuu yksinkertaisesti lähettämällä käynnistysviesti ja määrittämällä uudelleenkäynnistykseen reagoiva BroadcastReceiver kuuntelemaan myös sitä. IntentService saa päivityksen käynnistävän intentin perusteella tiedon siitä, onko kyse ajastettu vai sovelluksen pyytämä päivitys.



Kuvio 15. Android-version lähetystietojen päivitys taustalla.

Windows Phone 8 sallii kullekin sovellukselle yhden Background Agent –olion. Sovelluksen WMAppManifest.xml-tiedostoon (liitteessä 3) on lisätty tieto tausta-ajosta riveillä 21-24. Background Agent liitetään sovellukseen Visual Studiossa erillisenä projektina,

joka sisältää vain yhden luokan, ScheduledTaskAgent-luokan toteutuksen. Tämä apuprojekti tulee pitää pienikokoisena järjestelmän tausta-ajolle asettamien muistinkäyttörajoitusten vuoksi. Taustalla suoritettaville tehtäville on lisäksi annettu maksimikesto. Koska taustalla pitää päivittää lähetysten seurantatietoja verkkopalvelusta tietokantaan, erotetaan sovelluksen Visual Studio –projektista DataContext, malliluokat sekä verkkopalvelun taustatoteutus omaksi projektikseen kuvion 16 mukaisesti (TileUpdater liittyy myöhemmin esiteltävään toimintoon). Nämä lisätään viittauksina sekä varsinaiseen sovellukseen että tausta-agentille, jolloin ne ovat molempien käytössä.



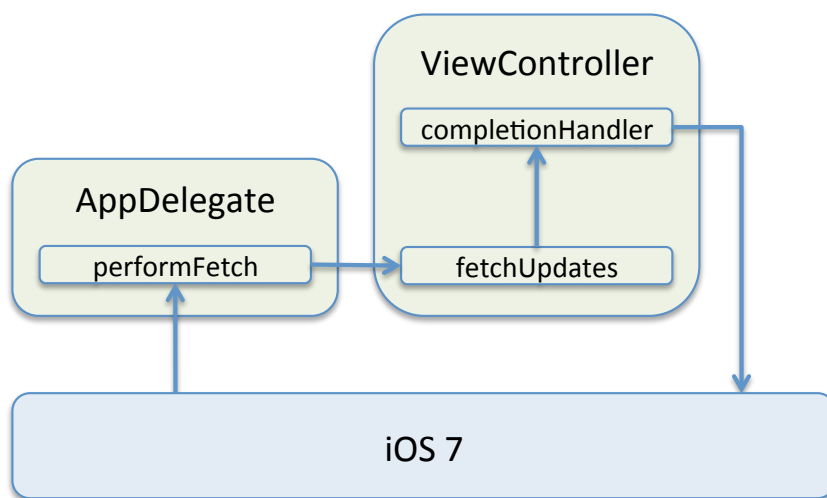
Kuvio 16. Windows Phone 8 -sovelluksen rakenne kolmena projektina.

Sovelluksen Application_Launching-metodissa lisätään sovelluksen käynnistyksen yhteydessä uusi PeriodicTask järjestelmän ScheduledActionServiceen (liittessä 3 App.xaml.cs-tiedostossa riveillä 88-126). Lisäys on yksinkertaista, koska sovelluksella voi olla vain yksi ajastettu tehtävä. ScheduledActionService tarjoaa testauksen helpottamiseksi metodin LaunchForTest, joka käynnistää tausta-ajon annetun (lyhyen) ajan kuluttua. Ajastetut tehtävät vanhentuvat tavallisesti kahdessa viikossa, joten joka kerran sovelluksen käynnistyessä poistetaan varmuuden vuoksi ajastettu tehtävä ja lisätään se uudelleen. Toisaalta järjestelmän tulisi pitää käynnissä sovelluksen LiveTileä päivittävät taustatehtävät ilman vanhentumista.

Oman haasteensa tausta-ajoon toi Windows Phonen WebClient-luokka, joka on tarkoitettu vain asynkroniseen lataukseen. Varsinainen taustatehtävän sisältävä säie ehtii päät-

tää suorituksensa ennen latauksen valmistumista. Ladattua sivua ei käsitelty lainkaan, koska käsittelijämetodia ei enää ollut. Nettietsinnöissä ratkaisuksi löytyi `TaskCompletionSource<T>`-luokka, jonka avulla WebClientin latauksen voi muuttaa synkroniseksi. Luokka kytketään `ParcelWebChecker.cs`-tiedostossa (liitteessä 3) riveillä 74-96 esitetyn koodin mukaisesti WebClientin `DownloadStringCompleted`-tapahtumankäsittelijään, ja se odottaa suorituksen etenemistä, kunnes siihen on tapahtumankäsittelijässä asetettu arvo.

iOS 7 –sovelluksessa taustalataustoiminto on otettava käyttöön valitsemalla Xcodessa sovelluksen `Background Modes` –ominaisuus käyttöön ja valitsemalla sen tiedoista `Background fetch`. Sovelluksen `AppDelegate`ssa (liitteessä 3 tiedosto `PakettivahtiAppDelegate.m`) asetetaan rivillä 27 taustapäivitysten aikaväli ja lisätään riveillä 170-179 metodi `application:performFetchWithCompletionHandler:`, jota järjestelmä kutsuu sopivin väliajoin. Tästä `AppDelegate`ten metodista käsin haetaan kuvion 17 mukaisesti sovelluksen päänäkömää hallinnoiva `rootViewController`, ja kutsutaan sen verkkopalvelusta tietoja päivittävää metodia `fetchParcelUpdatesWithCompletionHandler:` (liitteessä 3 tiedostossa `PakettivahtiViewController.m` riveillä 138-394). Kutsun yhteydessä metodille välitetään tapahtumankäsittelijä, joka lopulta palauttaa järjestelmälle tiedon taustalatauksen tuloksesta (rivit 92-135). Tällä tavoin järjestelmä tietää latauksen epäonnistumisesta, tietojen päivittymisestä tai siitä, etteivät tiedot muuttuneet. Kun metodin kutsujana on järjestelmään sijaan itse sovellus (esimerkiksi rivillä 78), ei kutsun yhteydessä välitetä lainkaan tapahtumankäsittelijää.

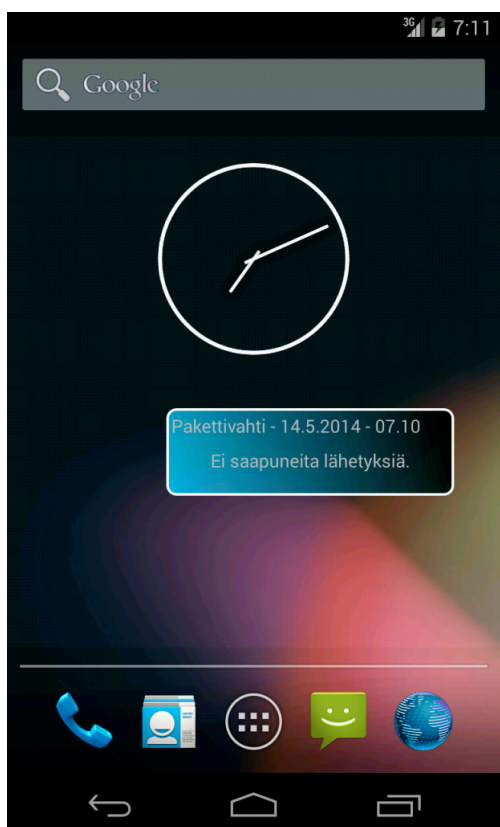


Kuvio 17. Taustahaku iOS-versiossa.

3.7 Ilmoitus käyttäjälle saapuneesta lähetyksestä

Pakettivahti-sovelluksen tulee ilmoittaa käyttäjälle noudettavaksi saapuneesta lähetyksestä myös taustapäivityksiä tehtäessä. Yksittäisen hetkellisesti huomiota herättävä ilmoituksen lisäksi on hyvä esittää käyttäjälle myös pysyvämpi tieto noudettavissa olevien lähetysten lukumäärästä.

Android-järjestelmä käyttää Notification-luokkaa ilmoittamaan käyttäjälle erilaisista tapahtumista NotificationManager-palvelun avulla. Saapumisilmoitus luodaan ja lähetetään tarkoitukseen toteutetussa BroadcastReceiver-tyyppisessä luokassa. Kun Intent-Service saa verkkopalvelulta tiedon siitä, että lähetykset on noudettavissa, se lähettää saapuneen lähetyksen nimen sisältävän intentin ilmoituksia lähettävälle vastaanottajalle. Tämä NotificationReceiver luo NotificationManagerin avulla ilmoituksen käyttäjälle. Ilmoitukseen liitetään pending intent, joka siirtää käyttäjän sovelluksen pääsivulle itse ilmoitusta klikkaamalla.



Kuvio 18. Android-sovelluksen widget.

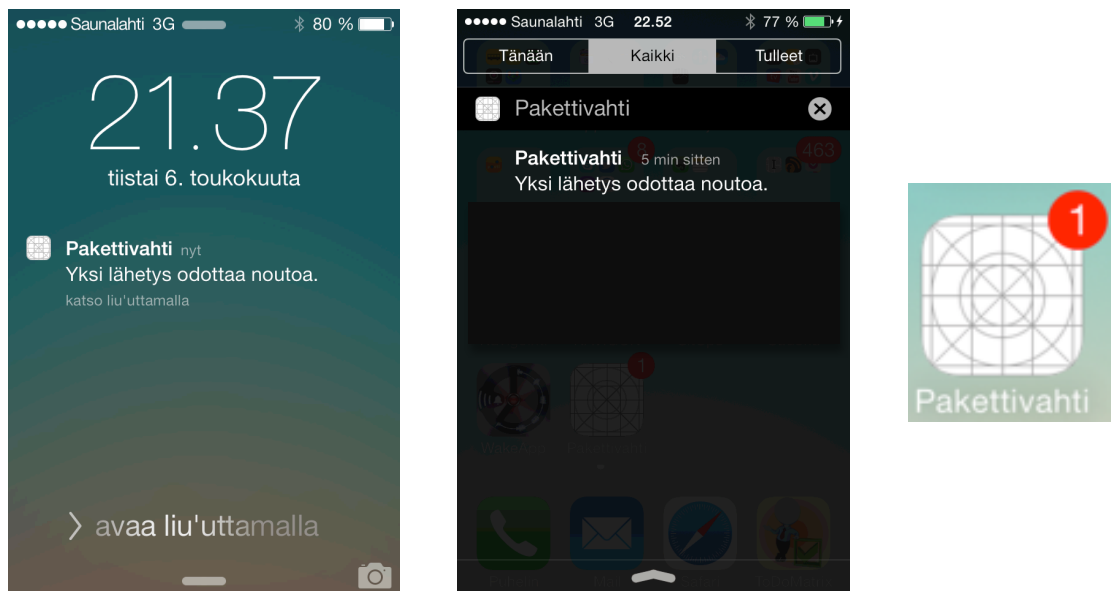
Pysyvämpään saapuneiden lukumäärän ilmoittamiseen Android-versio voi hyödyntää AppWidget-minisovellusta, jonka käyttäjä voi halutessaan lisätä laitteen kotinäkömään kuvion 18 mukaisesti. Widget peritytyy AppWidgetProvider-luokasta, joka puolestaan on BroadcastReceiver. Järjestelmä lähettää halutuin välein päivityskäskyn kutsumalla widgetin onUpdate-metodia, jossa widgetin esittämää sisältöä voi päivittää. Koska widget toimii myös BroadcastReceiverinä, voidaan sille lähettää tieto noudettavissa olevista lähetyksistä aina heti tietojen lataamisen jälkeen. Näin sovelluksessa ei tarvita lainkaan erillisiä widgetille kohdistettuja päivityskäskyjä.

Windows Phone 8 –sovellus sisältää aloitusnäytölle kiinnitettävissä olevan ikonin, Tilen. Tile voi olla LiveTile, jonka tietoja voi päivittää taustalla suoritettavista prosesseista. Pakettivahti-sovellus hyödyntää LiveTile-toimintoa FlipTilen avulla. Järjestelmä kääntää FlipTilen animoidusti tietyin väliajoin. Sen etupuolella on sovelluksen ikoni ja ohjelmallisesti asetettava numeromerkintä, johon liitetään noudettavissa olevien lähetysten lukumäärä kuvion 19 osoittamalla tavalla. Tilen kääntöpuolelle lisätään viimeimmän tarkistushetken päiväys ja kellonaika sekä tekstimuotoinen tieto siitä, onko lähetyksiä noudettavissa. Sovelluksen Tileä klikkaamalla käyttäjä pääsee siirtymään sovellukseen. Sovellus puolestaan päivittää Tilen aina pääsivulta poistuttaessa sekä taustalla tapahtuvan päivityksen päätteeksi kuviossa 16 esitettyä Core-projektin TileUpdater-luokkaa käyttäen.



Kuvio 19. Windows Phone 8 LiveTile ilmoittaa saapuneesta lähetyksestä.

iOS-järjestelmän sovellukset ja niiden taustalla toimivat prosessit voivat lähettää käyttäjälle ajastettuja muistutuksia ja ilmoituksia UILocalNotification-luokan avulla. Kun taustalla tapahtuneen päivityksen päätteeksi on löytynyt noudettavissa olevia lähetyksiä, luodaan uusi ilmoitus lähetettäväksi heti käyttäjälle. Kuviossa 20 esitetään saapumisilmoitus laitteen lukitusnäytöllä (vasen kuva), ilmoituskeskuksessa (keskellä) ja sovelluksen ikonissa. Ilmoituksen yhteydessä päivitetään myös sovelluksen ikoniin liitettävää numeromerkkiä (badge), joka ilmaisee saapuneiden lähetysten lukumäärän. Jos sovellus on ilmoituksen lähettäessään etualalla käynnissä, ei käyttäjää hälytetä eikä sovelluksen ikonin numeromerkkiä päivitetä. Ilmoitusta napauttamalla käyttäjä pääsee siirtymään sovellukseen, mikäli se ei ollut käynnissä. Sovellus päivittää aina taustalle siirtyessään sekä nettipalvelusta tietoja päivittäessään ikoninsa numeromerkkiä, jotta se myös aikaan nollaantuu. Järjestelmä piilottaa numeromerkkin, kun sen esittämä arvo on nolla.



Kuvio 20. iOS-version ilmoitus saapuneesta lähetyksestä.

4 Johtopäätelmät

Opinnäytetyön produktin valmistuttua kootaan koko projekti yhteen ja tarkastellaan sen tuloksia kokonaisuutena. Tämä luvun lopuksi arvioidaan työn tutkimustavoitteiden saavuttamista.

Karu totuus alkoi paljastua jo sovelluskehitystyön alussa: aihepiiri on hyvin laaja. Paket-tivahti-sovellus vaikutti aluksi varsin yksinkertaiselta. Mobiilikäyttöjärjestelmiin tutus-tuminen teoriaosan yhteydessä osoitti kuitenkin, että sovellukseen sisältyy monipuoli-sesti mobiilisovellusten tyypillisimpiä osa-alueita. Tämä puolestaan johti siihen, että perehdyttävää case-osuudessa oli vähintäänkin riittävästi.

Toisaalta mobiiliohjelmointiin perehtyminen on ollut erittäin antoisaa. Vaikka kolmen mobiilijärjestelmän käyttäminen onkin lisännyt opeteltavien asioiden määrää, on se sa-malla tarjonnut mahdollisuuden jäsenellä uusia asioita luontevammin kuin pelkästään yhteen järjestelmään ja sen sovelluskehitysmenetelmiin tutustumalla olisi ollut mahdol-lista. Yleisellä tasolla järjestelmät ovat varsin samankaltaisia, ja yhden yksityiskohtien erilaiset toteutusmahdollisuudet antavat uusia ideoita myös muille alustoille.

Mobiilisovelluskehitykseen tutustumisen aloitus tuntui vaikealta aiheen laajuudesta joh-tuen. Pelkästään lukemalla asioiden ymmärtäminen oli alussa kontekstin puuttumisen vuoksi hankalaa. Toteuttamalla käytännössä lähdekirjoissa esitettyjen esimerkkien kaut-ta erilaisia ratkaisuja pieniksi sovelluksiksi mobiiliohjelmointi alkoi vähitellen tuntua luontevammalta.

4.1 Havaintoja mobiilisovelluskehityksestä

Android, iOS ja Windows Phone muodostavat kukin omanlaisensa sovelluskehitysym-päristön. Jos näiden kolmen järjestelmän sovellusten kehitystyöhön on käytettävissä vain yksi tietokone, on sen oltava Applen valmistama. Windows Phone 8 -työkalut ovat käytettävissä virtualisoinnin kautta myös Apple-tietokoneissa. Androidin käyttämä Ec-lipse IDE on saatavilla suoraan yleisimmille järjestelmille. Sen sijaan Apple rajaa työka-lunsa vain valmistamissaan tietokoneissa toimivaan omaan järjestelmäänsä.

Kehitystyökalut ovat loppujen lopuksi hyvin samankaltaisia. Huomattava ero lähestymistavassa on oikeastaan siinä, että Xcoden storyboardissa ei sovelluksen käyttöliittymään kirjoiteta lainkaan ohjelmakoodia, vaan kaikki määrittelyt tehdään graafisilla työkaluilla. Android- ja Windows Phone –sovellukset edellyttävät enemmän xml- ja xaml-koodausta käyttöliittymää määriteltäessä. On kuitenkin vaikeaa ratkaista, kumpi menetelmä olisi parempi, sillä molempiin sisältyy etuja.

Androidissa ORM-ratkaisun puute teetti jonkin verran enemmän työtä tietojen hallinnan toteutuksessa. Toteutustapa kuitenkin perehdytti Android-sovellusten tyypilliseen rakenteeseen ja myöhemmissä vaiheissa se jopa nopeutti jatkokehitystä. Core Data puolestaan oli kaikista yksinkertaisin tapa tietojen hallintaan. Windows Phonen ratkaisu ei ollut työläs, mutta edellytti iOS:ää enemmän toteutukseen perehtymistä, koska LINQ ei ollut aiemmin tuttu.

Todennäköisesti suurimmat haasteet muodostuivat verkkopalvelun käytöstä sen asynkronisuuden vuoksi. Androidin ratkaisu, jossa koko IntentService suoritetaan suoraan omassa säikeessään, oli kaikkein yksinkertaisin. Windows Phone ja iOS aiheuttivat enemmän päänvaivaa tapahtumankäsittelijöineen. Onneksi ongelmat on myös huomioitu järjestelmien ohjelmointirajapinnoissa ja sovelluskehittäjille pyritään tarjoamaan mahdollisimman automaattisia työkaluluokkia. Niitä käyttämällä sovelluskehittäjän ei tarvitse ymmärtää täydellisesti rinnakkaisohjelmoinnin syvällistä olemusta.

Taustalla tapahtuva tietojen päivitys verkkopalvelusta oli työläintä toteuttaa Windows Phonelle. Androidilla tarvittiin vain päivitys-Servicen käynnistävä ajastus, muilta osin palaset tuntuivat lokahtelevan paikoilleen ja olemassa olevaa koodia oli mahdollista hyödyntää suoraan. Toisaalta tämä korostaa sovelluksen arkkitehtuurin ja kunkin järjestelmän suosimien ratkaisujen tuntemisen merkitystä. Tapahtumankäsittelijä aiheutti Windows Phone- ja iOS-versiossa eniten pohdintoja, koska se suoritetaan tavallisimmin omassa säikeessään ja siten ajastetun taustaprosessin säie on jo yleensä ehtinyt sulkeutua verkkopalvelun vastauksen saapuessa.

Ohjelmistokehitysprojehtin myötä myös järjestelmien mahdollisuudet ja rajoitukset tulivat tutummiksi. Vaikka lähtökohtana on kaikissa omaan hiekkalaatikkoonsa suljettu

sovellus, erottuu Android kuitenkin joukosta ylivoimaisesti avoimimpana. Tämän avoimuuden voi nähdä tietoturvariskinä, mutta toisaalta se avaa täysin uudenlaisia sovelluskehitysmahdollisuuksia, joiden toteuttaminen ei ole alkuunkaan mahdollista muissa järjestelmissä. Pakettivahti-sovellus hyödyntää esimerkiksi järjestelmän käynnistymisen yhteydessä lähetettävää ilmoitusta ajastetun päivitystoiminnon käyttöönottoon.

Järjestelmien ongelmat olivat jokseenkin erilaisia. Androidilla järjestelmäversioiden runsaus on haaste. Sovelluskehittäjän kannalta edullisinta olisi saada sama ohjelmakoodi toimimaan mahdollisimman suuressa osassa laitteita. Kuitenkin aina tuntuu löytyvän jokin yksittäinen tarpeellinen metodi, joka rajaa sovelluksen edellyttämän API-tason yhä uudempiin versioihin. Esimerkiksi String-luokan `isEmpty()`-metodi löytyy vasta API-tasolta 11 (Android 3.0, Honeycomb) alkaen.

Windows Phonen ongelmaksi muodostui osin keskeneräiseltä vaikuttava kirjastojen toteutus. Tiettyihin toimintoihin on toteutus useammassakin kirjastossa, toisiin taas ei lainkaan. Esimerkiksi muissa Windows 8 –versioissa on verkkopalveluiden käytössä hyödynnettävissä `HttpClient`-luokka, jolla latauksen synkronointi olisi ollut helpommin toteutettavissa. Virallisesti luokan sisältävä Microsoft HTTP Client Libraries –kirjasto ei kuitenkaan ole yhteensopiva Windows Phone 8:n kanssa. Tämän työn valmistumisen aikoihin saataville tuleva Windows Phone 8.1 korjaa huomattavan osan tämänkaltaisista ongelmista. Microsoft lupaili esittelytilaisuudessaan huhtikuussa, että uuden järjestelmäversion myötä on mahdollista kierrättää sama sovelluslogiikka vain käyttöliittymätoeutusta vaihtamalla Windows Phonen, Windows –työpöytäjärjestelmän ja jopa Xbox-pelikonsolin kesken.

Työläintä iOS-versiossa oli perehtyä Objective-C-kieleen ja sen myötä käytettäviin rakenteisiin. SmallTalk-kielestä omaksuttu oliojärjestelmä, jossa metodikutsujen sijaan lähetetään olioille viestejä muodossa `[olio viesti]`, vaati totuttelua. Koska kieli laajentaa C-kieltä, on tietyissä toiminnoissa käytössä myös suoraan C-kielisiä ratkaisuja. Esimerkiksi taustalatauksen yhteydessä kutsuttavalle metodille annettava tapahtumankäsittelijä ei ole minkään olion metodi, vaan C-kielinen funktio, jota kutsutaan muodossa `funktio(argumentit)`. Tässä kuten muissakin ongelmatilanteissa Googlen hakupalvelu oli verraton apu. Objective-C:n ja iOS:n kohdalla ongelmaksi nousi Google-hakujen myötä

tosin se, että kieli ja käyttöjärjestelmä kehittyvät nopeasti. Aiempien versioiden ratkaisut ja toteutukset eivät välttämättä toimi enää uudemmissa.

Kokonaisuutena internet on pullollaan materiaalia, jonka avulla vaikeimmatkin ongelmat tuntuvat ratkeavan. Stackoverflow.com –sivusto on loistava kysymyksiä ja vastauksia sisältävä tietolähde todennäköisesti lähes kaikenlaisiin ohjelmointiin liittyviin ongelmiin. Myös lukuisat ammattilaiset kirjoittavat blogeissaan pieniä artikkeleita, jotka valottavat monesti uudella tavalla järjestelmän omien ohjetiedostojen tietomassaa.

Projektin päättyessä on selvää, että aloittelijakin selviytyy mobiilisovelluskehityksestä, kunhan hänellä on riittävästi kärsivällisyyttä ja nettiyhteys käytettävissä. Samoin on selvää, että tämän työn puitteissa aihetta on vasta raapaistu pinnallisesti.

4.2 Tutkimustavoitteiden saavuttaminen

Tutkimuksen tarkoituksena oli selvittää miten halutunlainen mobiilisovellus voidaan toteuttaa kullakin kolmesta mobiilijärjestelmästä. Tältä osin tavoite on saavutettu, ja kolme sovellusprototyyppiä toteutettu. Case-osuudessa on kuvattu sovelluksen kehitysvaiheiden toteutuksia kussakin järjestelmässä. Samassa yhteydessä on dokumentoitu järjestelmälle tyypillisiä toteutusratkaisuja. Tuloksena syntyneet prototyyppisovellukset sisältävät halutun toiminnallisuuden ja läpäisevät hyväksymislähtöiset testit.

Koska aihepiiri osoittautui suunniteltua laajemmaksi, täytyy case-osuuden tulosten esityksessä pitäytyä melko yleisellä tasolla oppikirjamaisen yksityiskohtaisuuden sijaan. Tulokset osoittavat, että yhteisen suunnitteludokumentin pohjalta on mahdollista toteuttaa kullekin järjestelmälle sen ominaispiirteitä kunnioittava ja hyödyntävä sovellus. Toisaalta mahdollisesti rajaukset johtavat siihen, että kovin suuria eroja eri järjestelmien välille ei syntynyt. Jokaisella järjestelmällä on omat vahvuutensa ja heikkoutensa.

Olisi ollut mielenkiintoista vertailla natiivisovelluskehitystä myös kolmansien osapuolten tarjoamiin cross-platform –ratkaisuihin. Tällaisia ovat mm. avoimeen lähdekoodiin perustuvat html5- ja javascript-pohjainen PhoneGap/Cordova ja C#-kieltä käyttävät MonoTouch (iOS) ja Mono for Android. Toisaalta lähetyksen seuraamisen olisi voinut

toteuttaa sovellusten käyttämänä pilvipalveluna, joka päivittää tietojaan itsenäisesti ja ilmoittaa käyttäjän laitteella olevalle sovellukselle vasta lähetyksen saapumisesta. Nämä vaihtoehdot jäävät kuitenkin seuraavien tutkimusten aiheiksi.

Merkittävin tulos tässä opinnäytetyössä on omakohtaisen sovelluskehityskokemuksen karttuminen ja ymmärryksen kasvaminen. Projekti on antanut hyviä eväitä ja runsaasti intoa jatkaa kuhunkin järjestelmään syventymistä.

Lähteet

Android a. Starting an Activity. Luettavissa: <http://developer.android.com/training/basics/activity-lifecycle/starting.html>. Luettu 14.4.2014.

Android b. ContentValues. Luettavissa: <http://developer.android.com/reference/android/content/ContentValues.html>. Luettu 14.4.2014.

Annuzzi, J., Darcey, L., Conder, S. 2013. Introduction to Android Application Development: Android Essentials. 4th Edition. Developer's Library. Addison-Wesley Professional. Massachusetts.

AppBrain 2014. Top Android SDK versions. Luettavissa: <http://www.appbrain.com/stats/top-android-sdk-versions>. Luettu 14.4.2014.

Apple a. App States and Multitasking. Luettavissa: <https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html>. Luettu 14.4.2014.

Appleinsider 2013. iOS 7 now installed on 78% of active Apple handheld devices. Luettavissa: <http://appleinsider.com/articles/13/12/31/ios-7-now-installed-on-78-of-active-apple-handheld-devices>. Luettu 14.4.2014.

Binkley-Jones, T., Perga, M., Sync, M. & Benoit, A. 2014. Windows Phone 8 in Action. Manning Publications Co. New York.

Falafel Software Inc. 2013. Pro Windows Phone App Development. Third edition. Apress. California.

Gargenta, M. & Nakamura, M. 2014. Learnind Android: Develop Mobile Apps Using Java and Eclipse. Second Edition. O'Reilly Media. California.

Gartner 2014. Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013. Luettavissa: www.gartner.com/newsroom/id/2665715. Luettu 14.4.2014.

Microsoft 2014a. App activation and deactivation for Windows Phone 8. Luettavissa: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff817008\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff817008(v=vs.105).aspx). Luettu 14.4.2014.

Microsoft 2014b. Technical certification requirements for Windows Phone. Luettavissa: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184840\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184840(v=vs.105).aspx). Luettu 14.4.2014.

Microsoft 2014c. Background agents for Windows Phone 8. Luettavissa: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202942\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202942(v=vs.105).aspx). Luettu 14.4.2014.

Microsoft 2014d. How to preserve and restore app state for Windows Phone 8. Luettavissa: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff967547\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff967547(v=vs.105).aspx). Luettu 14.4.2014.

Neuburg, M. 2013. Programming iOS 7. Fourth Edition. O'Reilly Media. California.

Sharp, M., Sadun, E., Strougo, R. 2013. Learning iOS Development: A Hands-on Guide to the Fundamentals of iOS Programming. Addison-Wesley Professional. Massachusetts.

SQLite. SQLite Autoincrement. Luettavissa: <http://sqlite.org/autoinc.html>. Luettu 14.4.2014.

The Verge, 2013. iOS: A visual history. Luettavissa: <http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad>. Luettu 14.4.2014.

Thurrott, P. 2014. Windows Phone Device Stats: January 2014. Luettavissa:
<http://winsupersite.com/windows-phone/windows-phone-device-stats-january-2014>.
Luettu 14.4.2014.

WPCentral. Windows Phone 8 Update 3. Luettavissa:
<http://www.wpcentral.com/windows-phone-8-update-3>. Luettu 6.5.2014.

Liitteet

Liite 1. Mobiilikäyttöjärjestelmien ominaisuuksien vertailu

Ominaisuus	Android	iOS	Windows Phone
Markkinaosuudet 2013	78,4 %	15,6 %	3,2 %
Yleisimmät versiot	54,4 % Jelly Bean (4.1 - 4.3) 15,7 % Gingerbread (2.3) 14,3 % KitKat (4.4)	78 % iOS 7 18 % iOS 6	78,3 % Windows Phone 8 21,7 % Windows Phone 7.x
Julkaisuvuosi	2008 Android SDK 1.0 2013 Android 4.4 (API 19)	2007 iPhone OS 2013 iOS 7	2010 Windows Phone 7 2012 Windows Phone 8
Käyttöedellytykset	Vapaasti käytettävissä	Vain Applen omat laitteet	Microsoftin lisenssillä
Laitteistovaatimukset	Hyvin väljät, kaikenlaisia laitteita	Applen mobiililaite	Microsoftin minimiä täyttävä mobiililaite
Sovellusten eristäminen	Hiekkalaatikko		
Kehitystyökalut	Eclipse, ilmainen	Xcode, ilmainen	Visual Studio, ilmainen Express-versio
Testaus ilman laitetta	Emulaattori	Simulaattori	Emulaattori
Testaus laitteella	Ilmainen Mikä tahansa laite	Maksullinen kehittäjätili Rekisteröidyillä laitteilla	Maksullinen kehittäjätili Enintään 3 laitetta / tili Enintään 10 sovellusta / laite
Kehitysympäristö	Linux, Mac OSX, Windows	Mac OSX	Windows 8 Emulaattori: Win 8 Pro, 4 GB, Hyper V –yhteensopiva
Järjestelmäydin	Linux	Linux	Windows
Ohjelmointikieli	Java (+ Java-tavukoodia tuottavat) C/C++	Objective-C	XAML: C# ja Visual Basic Direct3D: C++
Hallittu- / natiivisovellus	Hallittu (Java) Natiivi (C/C++)	Natiivi	Hallittu (C# ja Visual Basic) Natiivi (C++)
Suunnittelumalli	(Ei vaatimuksia)	MVC (pakollinen)	MVVM (valinnainen)
Sovelluksen elinkaari	Kuusi tilaa Seitsemän metodia	Viisi tilaa Kuusi metodia	Kolme tilaa Neljä tapahtumaa Kaksi metodia
Taustatehtävät	Ei rajoituksia	Rajoitetusti vain tietyt tehtävät, esim. tietojen päivitys verkkopalvelusta	Periodic Task (n. 30 min välein enintään 25 s) Resource Intensive Task
Tietojen tallennus	Shared Preferences Tiedosto Tietokanta	NSUserDefaults Tiedosto Tietokanta	Väliaikaistiedosto Tiedosto (Isolated Storage) Tietokanta
Tietokanta & ORM	SQLite ei valmiina ORM-toimintoja	SQLite Core Data	MS SQL CE LINQ to SQL
Sovellusten jakelu	Vapaa	App Store	Windows Phone Store

Liite 2. Pakettivahti-sovelluksen käyttötapaukset

Käyttötapaus	Lähetyksen lisääminen
Esiehto	Käyttäjä on käynnistänyt sovelluksen.
Lopputulos	Sovellus on aloittanut lähetyksen seuraamisen.
Vaiheet	<ol style="list-style-type: none"> Käyttötapaus näyttää luettelon seurattavista lähetyksistä. Jokaisesta lähetyksestä näytetään sille annettu nimi sekä viimeimmän seurantatapahtuman ajan ja lähetyksen tilan. Luettelon yhteydessä on lähetyksen lisäsnappi. Käyttäjä painaa lisäsnappia. Käyttötapaus avaa näkymän, jossa täytettävänä kentät lähetystunnuksen ja lähetykselle annettavan nimen lisäämistä varten. Näiden lisäksi näkymässä on peruuta- ja lisää -napit. Käyttäjä syöttää lähetystunnuksen ja antaa lähetykselle nimen. Lopuksi hän painaa lisää-nappia. Käyttötapaus varmistaa lähetystunnuksen muodon oikeellisuuden, hakee verkkopalvelusta lähetyksen tilan ja tallentaa lähetyksen tiedot. Käyttötapaus siirtyy takaisin seurattavien lähetysten luetteloon. → 1. Käyttäjä lopettaa käyttötapauksen poistumalla sovelluksesta.
Variaatiot	<p>V4 Käyttäjä painaa peruuta-nappia. → 1.</p> <p>V4 Käyttäjä ei syötä lähetykselle nimeä. Käyttötapaus kopioi lähetystunnuksen nimikenttään. → 5.</p>
Poikkeukset	P5 Lähetystunnus ei ole oikeanmuotoinen. Käyttäjälle näytetään virheilmoitus ja pyydetään häntä tarkistamaan tunnus. → 5.

Käyttötapaus	Lähetyksen tunnuksen katsominen
Esiehto	Käyttäjä on käynnistänyt sovelluksen.
Lopputulos	Sovellus on näyttänyt lähetystunnuksen koko ruudun levyisenä.
Vaiheet	<ol style="list-style-type: none"> Käyttötapaus näyttää luettelon seurattavista lähetyksistä. Jokaisesta lähetyksestä näytetään sille annettu nimi tai lähetystunnus, jos nimeä ei ole annettu, sekä viimeimmän seurantatapahtuman ajan ja lähetyksen tilan. Luettelon yhteydessä on lähetyksen lisäsnappi. Käyttäjä valitsee lähetyksrivin listalta ja painaa sitä. Käyttötapaus avaa näkymän, jossa valitun lähetyksen lähetystunnus näytetään koko ruudun levyisenä tekstinä. Käyttäjä näyttää lähetystunnuksen postivirkailijalle ja poistuu näkymästä napauttamalla ruutua kahdesti. Käyttötapaus siirtyy takaisin seurattavien lähetysten luetteloon. → 1. Käyttäjä lopettaa käyttötapauksen poistumalla sovelluksesta.
Variaatiot	V4 Käyttäjä ei valitse riviä. → 6.

Liite 3. Lähdekoodit.

Android-versio

AboutDialog.java.....	54
activity_add_new_item.xml	55
activity_main.xml	56
activity_show_code.xml	56
add_new_item.xml	56
AddNewItemActivity.java.....	57
AndroidManifest.xml.....	60
DBHelper.java	61
list_item.xml.....	61
main.xml	61
MainActivity.java	62
NotificationReceiver.java	65
PakettivahtiWidget.java	66
ParcelBootReceiver.java	67
ParcelContract.java.....	68
ParcelProvider.java.....	69
ParcelService.java	72
ParcelViewBinder.java	76
show_code.xml	77
ShowCodeActivity.java.....	77
Status.java	78
strings.xml	79
widget_info.xml	79
widget_layout.xml	79
widget_tausta.xml	80

iOS-versio

AddParcelViewController.m	81
AddParcelViewController.h.....	82
CDParcel.h	83

CDParcel.m	83
main.m	83
Main.storyboard.....	84
PakettivahtiAppDelegate.m	88
PakettivahtiAppDelegate.h	90
PakettivahtiViewController.h	90
PakettivahtiViewController.m	91
Parcel.m	97
Parcel.h.....	98
ParcelDataModel.xcdatamodel/content.....	99
ParcelDataSourceController.h.....	99
ParcelDataSourceController.m	100
ParcelDataSourceDelegateProtocol.h	102
TrackingCodeViewController.h.....	102
TrackingCodeViewController.m.....	103

Windows Phone -versio

App.xaml.cs	104
AppResources.fi-FI.resx.....	107
MainPage.xaml.....	109
MainPage.xaml.cs	110
NewParcel.xaml.....	112
NewParcel.xaml.cs	113
Parcel.cs	114
ParcelDataContext.cs.....	116
ParcelViewModel.cs	117
ParcelWebChecker.cs	119
ShowCode.xaml.....	122
ShowCode.xaml.cs	122
StatusColorConverter.cs.....	123
StatusNames.cs	123
TileUpdater.cs	124
WMAppManifest.xml	125

```
1 package com.aueta.app.pakettivahti;
2
3 import android.app.Activity;
4 import android.app.AlertDialog;
5 import android.app.Dialog;
6 import android.content.DialogInterface;
7 import android.os.Bundle;
8 import android.support.v4.app.DialogFragment;
9 import android.support.v4.app.FragmentManager;
10
11 /**
12  * Luokka, jolla luodaan ilmoituksia käyttäjälle (dialog).
13  *
14  * @author Vesa Mäkeläinen
15  *
16  */
17 public class AboutDialog extends DialogFragment {
18
19     private Activity forActivity;    // Activity, jonne ilmoitus luodaan.
20     private int messageId;          // Dialogi-ikkunan tekstiresurssin koodi.
21
22     // Kutsujalle palautetaan uusi AboutDialog-olio.
23     public static AboutDialog newInstance()
24     {
25         return new AboutDialog();
26     }
27
28     // Asetetaan haluttu Activity vastaanottamaan dialogi-ikkuna.
29     public void setActivity(Activity activity)
30     {
31         forActivity = activity;
32     }
33
34     // Asetetaan dialogi-ikkunan tekstiresurssin koodi.
35     public void setMessage(int messageId)
36     {
37         this.messageId = messageId;
38     }
39
40     @Override
41     public Dialog onCreateDialog(Bundle savedInstanceState) {
42
43         // Luodaan dialogi-ikkuna, jossa on viestin lisäksi OK-nappi.
44         return new AlertDialog.Builder(forActivity)
45             .setMessage(messageId)
46             .setCancelable(false)
47             .setNeutralButton(R.string.app_button_ok, new DialogInterface.OnClickListener() {
48
49                 @Override
50                 public void onClick(DialogInterface dialog, int id) {
51                     // OK-napin klikkauksella suljetaan tämä dialogi.
52                     dialog.dismiss();
53                 }
54             })
55             .create();
56
57     }
58
59     public void show(FragmentManager fragmentManager, String tag)
60     {
61         super.show(fragmentManager, tag);
62     }
63 }
```

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:paddingBottom="@dimen/activity_vertical_margin"
6   android:paddingLeft="@dimen/activity_horizontal_margin"
7   android:paddingRight="@dimen/activity_horizontal_margin"
8   android:paddingTop="@dimen/activity_vertical_margin"
9   tools:context=".AddNewItemActivity" >
10
11   <TextView
12     android:id="@+id/tvCode"
13     android:layout_width="wrap_content"
14     android:layout_height="wrap_content"
15     android:layout_alignParentLeft="true"
16     android:layout_alignParentTop="true"
17     android:text="@string/add_code"
18     android:textAppearance="?android:attr/textAppearanceMedium" />
19
20   <TextView
21     android:id="@+id/tvName"
22     android:layout_width="wrap_content"
23     android:layout_height="wrap_content"
24     android:layout_alignLeft="@+id/tvCode"
25     android:layout_below="@+id/tvCode"
26     android:layout_marginTop="61dp"
27     android:text="@string/add_name"
28     android:textAppearance="?android:attr/textAppearanceMedium" />
29
30   <EditText
31     android:id="@+id/etCode"
32     android:layout_width="wrap_content"
33     android:layout_height="wrap_content"
34     android:layout_alignLeft="@+id/tvCode"
35     android:layout_alignParentRight="true"
36     android:layout_below="@+id/tvCode"
37     android:ems="10"
38     android:hint="@string/add_code_hint"
39     android:inputType="textCapCharacters" >
40
41     <requestFocus />
42   </EditText>
43
44   <EditText
45     android:id="@+id/etName"
46     android:layout_width="wrap_content"
47     android:layout_height="wrap_content"
48     android:layout_alignLeft="@+id/etCode"
49     android:layout_alignParentRight="true"
50     android:layout_below="@+id/tvName"
51     android:ems="10"
52     android:hint="@string/add_name_hint"
53     android:imeOptions="actionSend"
54     android:inputType="textCapSentences" >
55
56   </EditText>
57
58   <Button
59     android:id="@+id/btAdd"
60     android:layout_width="wrap_content"
61     android:layout_height="wrap_content"
62     android:layout_alignLeft="@+id/etName"
63     android:layout_alignRight="@+id/etName"
64     android:layout_alignParentRight="true"
65     android:layout_below="@+id/etName"
66     android:layout_marginTop="38dp"
67     android:text="@string/add_button"
68   />
69
70 </RelativeLayout>
```

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:paddingBottom="@dimen/activity_vertical_margin"
6   android:paddingLeft="@dimen/activity_horizontal_margin"
7   android:paddingRight="@dimen/activity_horizontal_margin"
8   android:paddingTop="@dimen/activity_vertical_margin"
9   tools:context=".MainActivity" >
10
11   <ListView
12     android:id="@+id/lvPaketit"
13     android:layout_width="match_parent"
14     android:layout_height="wrap_content"
15     android:layout_centerHorizontal="true"
16     android:footerDividersEnabled="true"
17     android:headerDividersEnabled="true"
18   >
19   </ListView>
20
21 </RelativeLayout>
```

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:paddingBottom="@dimen/activity_vertical_margin"
6   android:paddingLeft="@dimen/activity_horizontal_margin"
7   android:paddingRight="@dimen/activity_horizontal_margin"
8   android:paddingTop="@dimen/activity_vertical_margin"
9   tools:context=".ShowCodeActivity" >
10
11   <TextView
12     android:id="@+id/tvShowCode"
13     android:layout_width="wrap_content"
14     android:layout_height="wrap_content"
15     android:layout_centerInParent="true"
16     android:ellipsize="none"
17     android:fitsSystemWindows="true"
18     android:maxLines="@integer/Yksi"
19     android:padding="@dimen/PaddingShow"
20     android:rotation="0.0"
21     android:text="@string/show_code"
22     android:gravity="center"
23     android:textSize="42sp"
24     android:textStyle="bold"
25     android:layout_margin="2dp" />
26
27 </RelativeLayout>
```

```
1 <menu xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:omat="http://schemas.android.com/apk/res-auto" >
3
4   <item
5     android:id="@+id/action_add_about"
6     android:icon="@drawable/ic_action_about"
7     android:orderInCategory="0"
8     omat:showAsAction="ifRoom"
9     android:title="@string/action_about"/>
10
11 </menu>
12
```

```
1 package com.aueta.app.pakettivahti;
2
3 import java.util.Date;
4
5 import android.content.ContentValues;
6 import android.content.Intent;
7 import android.net.Uri;
8 import android.os.Bundle;
9 import android.support.v7.app.ActionBarActivity;
10 import android.view.KeyEvent;
11 import android.view.Menu;
12 import android.view.MenuItem;
13 import android.view.View;
14 import android.view.View.OnClickListener;
15 import android.view.inputmethod.EditorInfo;
16 import android.widget.Button;
17 import android.widget.EditText;
18 import android.widget.TextView;
19 import android.widget.TextView.OnEditorActionListener;
20
21 import com.aueta.app.pakettivahti.dao.ParcelContract;
22 import com.aueta.app.pakettivahti.domain.Status;
23 import com.aueta.app.pakettivahti.service.ParcelService;
24
25 public class AddNewItemActivity extends ActionBarActivity {
26
27     // Näillä palautettu tieto löytyy Intentistä.
28     public static final String PARCEL_CODE = "parcel_code";
29     public static final String PARCEL_NAME = "parcel_name";
30
31
32     private AboutDialog aboutDialog;          // Tietoja sovelluksesta -dialogi-ikkuna.
33     private EditText etCode;
34     private EditText etName;
35
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         super.onCreate(savedInstanceState);
39         setContentView(R.layout.activity_add_new_item);
40
41         // Haetaan tekstinsyöttökentät.
42         etCode = (EditText) findViewById(R.id.etCode);
43         etName = (EditText) findViewById(R.id.etName);
44
45         // Lisätään Lisää-napin klikkaus.
46         final Button btAdd = (Button) findViewById(R.id.btAdd);
47         btAdd.setOnClickListener(new OnClickListener() {
48
49             @Override
50             public void onClick(View v) {
51
52                 if(etCode.getText().toString().isEmpty())
53                 {
54                     // Tarkistetaan, että seurantakoodi on syötetty ja jos se puuttuu,
55                     // näytetään vinkki tekstikentässä.
56                     etCode.setHint(getString(R.string.add_code_hint));
57                 }
58                 else
59                 {
60                     // OK, palautetaan syötetyt tiedot ja suljetaan tämä Activity.
61                     returnDataAndFinish();
62                 }
63             }
64         });
65
66         // Lisätään validointi seurantakoodi-kenttään.
67         etCode.setOnEditorActionListener(new OnEditorActionListener() {
68
69             @Override
70             public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
71
72                 // Kertoo saadaanko viesti käsiteltyä omassa metodissa.
73                 // Jos saadaan, niin käsittelypyyntöä ei välitetä enää eteenpäin.
74                 boolean handled = false;
75
76
77                 if(actionId == EditorInfo.IME_ACTION_NEXT)
78                 {
79                     // Käyttäjä haluaa siirtyä seuraavaan kenttään näppäimistön NEXT-napilla.
80
81                     // Tarkistetaan, että on syötetty jotain.
82                     if(v.getText().toString().isEmpty())
83                     {
84                         // Ei saisi olla tyhjä. Näytetään vinkki.
85                         v.setHint(getString(R.string.add_code_hint));
86
87                         // Ei siirrytä eteenpäin, joten ilmoitetaan tämä tapahtuma käsitellyksi.
88                         handled = true;
89                     }
90                     else
91                     {
```

```
92 // Seurantakoodikentässä on tekstiä. Validointi tapahtuu tässä...
93 }
94 }
95 }
96
97     return handled;
98 }
99 });
100
101
102 // Lisätään lähetyksen nimikenttään näppäimistöön SEND-napin toiminta.
103 etName.setOnEditorActionListener(new OnEditorActionListener() {
104
105     @Override
106     public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
107
108         boolean handled = false;
109
110         // Jos käyttäjä painaa tässä kentässä näppäimistön SEND-nappia,
111         // tiedot lähetetään ja tämä Activity suljetaan.
112         if(actionId == EditorInfo.IME_ACTION_SEND)
113         {
114             // Ei tarvitse muita toimintoja, koska Activity suljetaan. Asetetaan muodon vuoksi.
115             handled = true;
116
117             returnDataAndFinish();
118         }
119
120         return handled;
121     }
122 });
123 }
124
125
126 @Override
127 public boolean onCreateOptionsMenu(Menu menu) {
128     // Inflate the menu; this adds items to the action bar if it is present.
129     getMenuInflater().inflate(R.menu.add_new_item, menu);
130
131     // Enable Home navigation (For old APIs with the support library.
132     getActionBar().setDisplayHomeAsUpEnabled(true);
133
134     return true;
135 }
136
137
138 @Override
139 public boolean onOptionsItemSelected(MenuItem item) {
140     switch (item.getItemId()) {
141
142         case android.R.id.home:
143
144             // Käyttäjä klikkasi Action Barin ikonin, joten poistutaan takaisin
145             // aloitus-Activityyn välittämättä mitään tietoa.
146             setResult(RESULT_CANCELED);
147             finish();
148             return true;
149
150         case R.id.action_add_about:
151
152             // Tietoja sovelluksesta.
153             showAboutDialog();
154             return true;
155
156     }
157     return super.onOptionsItemSelected(item);
158 }
159
160
161 /**
162  * Avataan dialog-ikkuna ja näytetään siinä viesti.
163  */
164 private void showAboutDialog() {
165     aboutDialog = AboutDialog.newInstance();
166     aboutDialog.setActivity(this);
167     aboutDialog.setMessage(R.string.app_about_add_text);
168     aboutDialog.show(getSupportFragmentManager(), getString(R.string.app_about_tag));
169 }
170
171
172 /**
173  * Lähetetään syötetyt tiedot ja palataan pää-Activityyn.
174  */
175 private void returnDataAndFinish()
176 {
177     // Luodaan pää-Activityyn siirtävä Intent ja luetaan syötetyt tekstit.
178     Intent returnIntent = new Intent(this, MainActivity.class);
179     String code = etCode.getText().toString();
180     String name = etName.getText().toString();
181
182     if(code.isEmpty())
```

```
183     {
184         // Ei palauteta tyhjää tietoa vaan Cancel-viesti.
185         setResult(RESULT_CANCELED);
186     }
187     else
188     {
189         // Jos nimikenttä on tyhjä, kopioidaan siihen lähetystunnus.
190         if(name.isEmpty())
191         {
192             name = code;
193         }
194
195         // Tallennetaan lisätty lähetys tietokantaan.
196
197         ContentValues values = new ContentValues();
198
199         values.put(ParcelContract.Column.CODE, code);
200         values.put(ParcelContract.Column.NAME, name);
201         values.put(ParcelContract.Column.STATUS, Status.DEFAULT.ordinal());
202         values.put(ParcelContract.Column.TIME, new Date().getTime());
203
204         Uri uri = getContentResolver().insert(ParcelContract.CONTENT_URI, values);
205
206         // Haetaan nettipalvelusta uuden lähetyksen tilatieto (taustalla servicen avulla).
207         Intent parcelService = new Intent(this, ParcelService.class);
208         parcelService.putExtra(AddNewItemActivity.PARCEL_CODE, code);
209         startService(parcelService);
210
211         // Ilmoitetaan toiminto onnistuneeksi ja liitetään palautettava data.
212         setResult(RESULT_OK, returnIntent);
213     }
214
215     // Poistutaan tästä Activitystä.
216     finish();
217 }
218
219 }
```

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.aueta.app.pakettivahti"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk
8          android:minSdkVersion="11"
9          android:targetSdkVersion="18" />
10
11     <uses-permission android:name="android.permission.INTERNET" />
12     <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
13
14     <application
15         android:allowBackup="true"
16         android:icon="@drawable/ic_launcher"
17         android:label="@string/app_name"
18         android:theme="@style/AppTheme" >
19
20         <receiver
21             android:icon="@drawable/ic_launcher"
22             android:label="Pakettivahti-Widget"
23             android:name="com.aueta.app.pakettivahti.widget.PakettivahtiWidget" >
24
25             <intent-filter >
26                 <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
27                 <action android:name="com.aueta.app.pakettivahti.action.UPDATE_WIDGET" />
28             </intent-filter>
29
30             <meta-data
31                 android:name="android.appwidget.provider"
32                 android:resource="@xml/widget_info" />
33             </receiver>
34
35             <receiver android:name="com.aueta.app.pakettivahti.receiver.ParcelBootReceiver" >
36                 <intent-filter>
37                     <action android:name="android.intent.action.BOOT_COMPLETED" />
38                     <action android:name="android.intent.action.QUICKBOOT_POWERON" /> <!-- HTC -->
39                     <action android:name="com.aueta.app.pakettivahti.action.START_PARCELSERVICE" />
40                 </intent-filter>
41             </receiver>
42
43             <receiver android:name="com.aueta.app.pakettivahti.receiver.NotificationReceiver"
44                 android:exported="false" >
45                 <intent-filter>
46                     <action android:name="com.aueta.app.pakettivahti.action.PARCEL_ARRIVED" />
47                 </intent-filter>
48             </receiver>
49
50             <service android:name="com.aueta.app.pakettivahti.service.ParcelService" />
51
52             <provider
53                 android:name="com.aueta.app.pakettivahti.dao.ParcelProvider"
54                 android:authorities="com.aueta.app.pakettivahti.dao.ParcelProvider"
55                 android:exported="false" />
56
57             <activity
58                 android:name="com.aueta.app.pakettivahti.MainActivity"
59                 android:label="@string/app_name"
60                 android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >
61                 <intent-filter>
62                     <action android:name="android.intent.action.MAIN" />
63
64                     <category android:name="android.intent.category.LAUNCHER" />
65                 </intent-filter>
66             </activity>
67
68             <activity
69                 android:name="com.aueta.app.pakettivahti.AddNewItemActivity"
70                 android:label="@string/title_activity_add_new_item"
71                 android:parentActivityName="com.aueta.app.pakettivahti.MainActivity"
72                 android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
73                 android:windowSoftInputMode="stateVisible" >
74                 <meta-data
75                     android:name="android.support.PARENT_ACTIVITY"
76                     android:value="com.aueta.app.pakettivahti.MainActivity" />
77             </activity>
78
79             <activity
80                 android:name="com.aueta.app.pakettivahti.ShowCodeActivity"
81                 android:label="@string/title_activity_show_code"
82                 android:parentActivityName="com.aueta.app.pakettivahti.MainActivity" >
83                 <meta-data
84                     android:name="android.support.PARENT_ACTIVITY"
85                     android:value="com.aueta.app.pakettivahti.MainActivity" />
86
87             </activity>
88
89     </application>
90
91 </manifest>
    
```



```

1 package com.aueta.app.pakettivahti.dao;
2
3 import android.content.Context;
4 import android.database.sqlite.SQLiteDatabase;
5 import android.database.sqlite.SQLiteOpenHelper;
6
7 public class DBHelper extends SQLiteOpenHelper {
8
9     // Lokiinkirjoitusta varten luokan nimitägi.
10    private static final String TAG = DBHelper.class.getSimpleName();
11
12    public DBHelper(Context context) {
13        super(context, ParcelContract.DB_NAME, null, ParcelContract.DB_VERSION);
14    }
15
16    @Override
17    public void onCreate(SQLiteDatabase database) {
18        // Kun tietokanta otetaan käyttöön, luodaan varsinainen tietokantataulu.
19
20        String sql = String.format(
21            "CREATE TABLE %s (%s INTEGER PRIMARY KEY, %s TEXT, %s TEXT, %s INTEGER, %s INTEGER)",
22            ParcelContract.TABLE, ParcelContract.Column.ID, ParcelContract.Column.NAME,
23            ParcelContract.Column.CODE, ParcelContract.Column.STATUS, ParcelContract.Column.TIME);
24        database.execSQL(sql);
25    }
26
27    @Override
28    public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {
29        // Tietokantaa päivitetään uuteen versioon.
30
31        // Poistetaan tietokantataulu ja luodaan tietokanta uudelleen.
32        String sql = String.format("DROP TABLE IF EXISTS %s", ParcelContract.TABLE);
33        database.execSQL(sql);
34        onCreate(database);
35    }
36 }

```

```

1 <?xml version="1.0" encoding="utf-8"?>
2
3 <LinearLayout
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     android:layout_width="fill_parent"
6     android:layout_height="fill_parent"
7     android:orientation="vertical">
8
9     <TextView
10        android:id="@+id/tvItemName"
11        android:layout_width="fill_parent"
12        android:layout_height="wrap_content"
13        android:textAppearance="?android:attr/textAppearanceMedium"
14        android:layout_weight="2" />
15
16     <TextView
17        android:id="@+id/tvItemStatus"
18        android:layout_width="fill_parent"
19        android:layout_height="wrap_content"
20        android:textAppearance="?android:attr/textAppearanceSmall"
21        android:layout_weight="1" />
22
23 </LinearLayout>

```

```

1 <menu xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:omat="http://schemas.android.com/apk/res-auto" >
3     <item
4         android:id="@+id/action_about"
5         android:icon="@drawable/ic_action_about"
6         android:orderInCategory="1"
7         omat:showAsAction="ifRoom"
8         android:title="@string/action_about"
9         android:titleCondensed="@string/action_about_condensed"
10        />
11     <item
12         android:id="@+id/action_new"
13         android:icon="@drawable/ic_action_new"
14         android:orderInCategory="0"
15         omat:showAsAction="ifRoom"
16         android:title="@string/action_new" />
17 </menu>

```

```
1 package com.aueta.app.pakettivahti;
2
3 import java.util.Date;
4
5 import android.app.LoaderManager.LoaderCallbacks;
6 import android.content.CursorLoader;
7 import android.content.Intent;
8 import android.content.SharedPreferences;
9 import android.content.pm.ActivityInfo;
10 import android.database.Cursor;
11 import android.os.Bundle;
12 import android.preference.PreferenceManager;
13 import android.support.v4.widget.SimpleCursorAdapter;
14 import android.support.v7.app.ActionBarActivity;
15 import android.util.Log;
16 import android.view.Menu;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.widget.AdapterView;
20 import android.widget.AdapterView.OnItemClickListener;
21 import android.widget.ListView;
22
23 import com.aueta.app.pakettivahti.dao.ParcelContract;
24 import com.aueta.app.pakettivahti.dao.ParcelViewBinder;
25 import com.aueta.app.pakettivahti.service.ParcelService;
26
27 public class MainActivity extends ActionBarActivity implements LoaderCallbacks<Cursor>
28 {
29
30     private final int NEW_PARCEL_REQUEST = 0; // Pyydetään lisättävän lähetyksen tiedot lisäysactivityltä.
31     private AboutDialog aboutDialog; // Tietoja sovelluksesta -dialogi-ikkuna.
32
33
34     // ParcelProviderin käyttö. Kerrotaan, mistä kentistä tietoja luetaan.
35     private static final String[] FROM = { ParcelContract.Column.ID, ParcelContract.Column.NAME,
36                                           ParcelContract.Column.CODE, ParcelContract.Column.STATUS,
37                                           ParcelContract.Column.TIME };
38
39     // Ja mihin käyttöliittymän elementteihin niitä sijoitetaan.
40     private static final int[] TO = { R.id.tvItemName, R.id.tvItemStatus };
41
42     // Yksilöivä tunniste.
43     private static final int LOADER_ID = 118;
44
45     private SimpleCursorAdapter adapter; // ListView:n adapteri.
46
47     private ListView listView; // ListView, jossa seurattavat lähetykset näytetään.
48
49     protected ListView getListView()
50     {
51         if(listView == null)
52         {
53             listView = (ListView) findViewById(R.id.lvPaketit);
54         }
55
56         return listView;
57     }
58
59
60     @Override
61     protected void onCreate(Bundle savedInstanceState) {
62         super.onCreate(savedInstanceState);
63         setContentView(R.layout.activity_main);
64
65         // Asetetaan adapteri ja viewBinder.
66         adapter = new SimpleCursorAdapter(this, R.layout.list_item, null, FROM, TO, 0);
67         adapter.setViewBinder(new ParcelViewBinder());
68
69         // Kytetään ne listView:hun.
70         listView = getListView();
71         listView.setAdapter(adapter);
72         getLoaderManager().initLoader(LOADER_ID, null, this);
73
74         // Haetaan asetukset käyttöön.
75         SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
76
77         // Katsotaan edellinen tietojenhaku aika, oletuksena 0 eli päivitetään joka tapauksessa.
78         long lastUpdateTime = preferences.getLong("lastUpdated", 0);
79
80         // Päivitetään näytön tiedot vain, jos edellisestä päivityksestä on kulunut yli puoli tuntia.
81         // = 30 minuuttia * 60 sekuntia * 1000 ms
82         if((new Date().getTime() - lastUpdateTime) > 30*60*1000)
83         {
84             Log.d("OnActivityResult: ", "Haetaan lähetystiedot palvelimelta.");
85
86             // Päivitetään kaikki seurattavat lähetykset verkkopalvelusta.
87             Intent statusIntent = new Intent(this, ParcelService.class);
88             startService(statusIntent);
89         }
90
91         // Varmistetaan, että ajastettu päivitys nettipalvelusta on käynnissä lähettämällä käynnistysviesti.
```

```

92     Intent startServiceIntent = new Intent("com.aueta.app.pakettivahti.action.START_PARCELSERVICE");
93     sendBroadcast(startServiceIntent);
94
95     Log.d("OnActivityResult: ", "Lähetetty päivityspalvelun käynnistyspyyntö.");
96
97     // Asetetaan listView:n toiminto, jolla napauttamalla siirrytään lähetystunnuksen näyttöön.
98     listView.setOnItemClickListener(new OnItemClickListener() {
99
100         @Override
101         public void onItemClick(AdapterView<?> listView, View v, int position, long id) {
102
103             // Haetaan cursor-olio, joka sisältää valitun ListView:n solun sisältämän olion.
104             Cursor cursor = (Cursor) listView.getAdapter().getItem(position);
105
106             // Haetaan valitun lähetyksen seurantakoodi.
107             String selectedCode = cursor.getString(cursor.getColumnIndex(ParcelContract.Column.CODE));
108
109             // Näytetään valitun lähetyksen koodi uudessa Activityssä.
110             Intent showCodeIntent = new Intent(MainActivity.this, ShowCodeActivity.class);
111             showCodeIntent.putExtra(AddNewItemActivity.PARCEL_CODE, selectedCode);
112             startActivity(showCodeIntent);
113         }
114     });
115 }
116
117 @Override
118 public boolean onCreateOptionsMenu(Menu menu) {
119     // Inflate the menu; this adds items to the action bar if it is present.
120     getMenuInflater().inflate(R.menu.main, menu);
121
122     return super.onCreateOptionsMenu(menu);
123 }
124
125 @Override
126 protected void onStart() {
127
128     // Käännetään näyttö pystyyn.
129     setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
130
131     super.onStart();
132 }
133
134 @Override
135 public boolean onOptionsItemSelected(MenuItem item) {
136
137     // Käyttäjä on painanut Action Barin menunappia.
138
139     switch(item.getItemId())
140     {
141
142     case R.id.action_about:
143         // Tietoja sovelluksesta.
144         showAboutDialog();
145         return true;
146
147     case R.id.action_new:
148         // Uuden lähetyksen lisääminen. Avataan lisäys-Activity.
149         Intent addNewItem = new Intent(this, AddNewItemActivity.class);
150         startActivityForResult(addNewItem, NEW_PARCEL_REQUEST);
151         return true;
152     }
153
154     return super.onOptionsItemSelected(item);
155 }
156
157 @Override
158 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
159
160     // Tutkitaan syöttikö käyttäjä uuden lähetyksen tiedot ja käsitellään ne tarvittaessa.
161     if(requestCode == NEW_PARCEL_REQUEST)
162     {
163         // On palattu lähetyksen lisäys-Activitystä.
164         if(resultCode == RESULT_OK)
165         {
166             // Uuden lähetyksen lisäys onnistui.
167         }
168     }
169 }
170
171 /**
172  * Avataan dialog-ikkuna, ja näytetään siinä viesti.
173  */
174 private void showAboutDialog() {
175
176     aboutDialog = AboutDialog.newInstance();
177     aboutDialog.setActivity(this);
178     aboutDialog.setMessage(R.string.app_about_text);
179     aboutDialog.show(getSupportFragmentManager(), getString(R.string.app_about_tag));
180 }
181
182 @Override

```

```
183     public android.content.Loader<Cursor> onCreateLoader(int id, Bundle args) {
184         if(id != LOADER_ID)
185         {
186             return null;
187         }
188
189         // Haetaan lähetystiedot.
190         return new CursorLoader(this, ParcelContract.CONTENT_URI, FROM, null, null, ParcelContract.DEFAULT_SORT);
191     }
192
193     @Override
194     public void onLoadFinished(android.content.Loader<Cursor> loader, Cursor cursor) {
195
196         // Vaihdetään tiedot sisältävä cursor uuteen.
197         adapter.swapCursor(cursor);
198     }
199
200     @Override
201     public void onLoaderReset(android.content.Loader<Cursor> arg0) {
202
203         adapter.swapCursor(null);
204     }
205 }
206 }
```

```
1 package com.aueta.app.pakettivahti.receiver;
2
3 import android.R;
4 import android.app.Notification;
5 import android.app.NotificationManager;
6 import android.app.PendingIntent;
7 import android.content.BroadcastReceiver;
8 import android.content.Context;
9 import android.content.Intent;
10
11 import com.aueta.app.pakettivahti.MainActivity;
12 import com.aueta.app.pakettivahti.dao.ParcelContract;
13
14 public class NotificationReceiver extends BroadcastReceiver {
15
16     public static final int NOTIFICATION_ID = 118;
17
18     @Override
19     public void onReceive(Context context, Intent intent) {
20
21         // Haetaan contextin notificationManager, jolla luodaan ilmoitus.
22         NotificationManager notificationManager =
23             (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
24
25         // Haetaan saapuneen lähetyksen nimi intentistä.
26         String name = intent.getStringExtra(ParcelContract.COLUMN_NAME);
27
28         // Muodostetaan Pending Intent, joka vie sovelluksen pääsivulle, kun käyttäjä koskettaa ilmoitusta.
29         PendingIntent mainActivityIntent = PendingIntent.getActivity(context, 0,
30             new Intent(context, MainActivity.class), PendingIntent.FLAG_ONE_SHOT);
31
32         // Kootaan ja lähetetään ilmoitus.
33         Notification notification = new Notification.Builder(context).setContentTitle("Saapunut!")
34             .setContentText("'" + name + "' saapunut.")
35             .setSmallIcon(R.drawable.ic_dialog_alert)
36             .setContentIntent(mainActivityIntent)
37             .setAutoCancel(true)
38             .getNotification();
39
40         notificationManager.notify(NOTIFICATION_ID, notification);
41     }
42 }
```

```
1 package com.aueta.app.pakettivahti.widget;
2
3 import java.text.SimpleDateFormat;
4 import java.util.Date;
5 import java.util.Locale;
6
7 import android.app.PendingIntent;
8 import android.appwidget.AppWidgetManager;
9 import android.appwidget.AppWidgetProvider;
10 import android.content.ComponentName;
11 import android.content.Context;
12 import android.content.Intent;
13 import android.util.Log;
14 import android.widget.RemoteViews;
15
16 import com.aueta.app.pakettivahti.MainActivity;
17 import com.aueta.app.pakettivahti.R;
18
19 public class PakettivahtiWidget extends AppWidgetProvider {
20
21     private static final String TAG = PakettivahtiWidget.class.getSimpleName();
22
23     @Override
24     public void onReceive(Context context, Intent intent) {
25         super.onReceive(context, intent);
26
27         // Widget on vastaanottanut Broadcastin.
28
29         // Haetaan intentistä noudettavissa olevien lähetysten määrä, oletuksena 0.
30         int updateCount = intent.getIntExtra("updateCount", 0);
31
32         AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
33
34         // Päivitetään tieto.
35         updateWidget(context, appWidgetManager, appWidgetManager.getAppWidgetIds(
36             new ComponentName(context, PakettivahtiWidget.class)), updateCount);
37     }
38
39     private void updateWidget(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds,
40                               int updateCount)
41     {
42         // Asetetaan Widgetin teksti noudettavissa olevien lähetysten määrän mukaan.
43         String widgetText;
44
45         if(updateCount > 1)
46         {
47             widgetText = "" + updateCount + " saapunutta lähetystä.";
48         }
49         else if(updateCount > 0)
50         {
51             widgetText = "Yksi saapunut lähetys.";
52         }
53         else
54         {
55             widgetText = "Ei saapuneita lähetyksiä.";
56         }
57
58         // Luodaan PendingIntent, jonka avulla Widgettiä koskettamalla päästään itse sovellukseen.
59         Intent mainActivityIntent = new Intent(context, MainActivity.class);
60         PendingIntent goToMainActivity = PendingIntent.getActivity(context, 0, mainActivityIntent,
61             PendingIntent.FLAG_UPDATE_CURRENT);
62
63         // Muotoillaan ajan esitys merkkijonoksi.
64         SimpleDateFormat simpleDateFormat = new SimpleDateFormat("d.M.yyyy - HH.mm", Locale.getDefault());
65
66         // Käsitellään kaikki Pakettivahtiwidgetit.
67         for(int widgetId : appWidgetIds)
68         {
69             String localizedTime = simpleDateFormat.format(new Date().getTime());
70
71             // Päivitetään widgetin teksti.
72             RemoteViews view = new RemoteViews(context.getPackageName(), R.layout.widget_layout);
73             view.setTextViewText(R.id.updateTime, "Pakettivahti - " + localizedTime);
74
75             view.setTextViewText(R.id.updateText, widgetText);
76
77             // Päivitetään kosketustoiminto.
78             view.setOnClickPendingIntent(R.id.layout, goToMainActivity);
79
80             // Lähetetään Widgetin päivitykset.
81             appWidgetManager.updateAppWidget(widgetId, view);
82         }
83     }
84
85     @Override
86     public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
87         // Järjestelmä on kutsunut widgetin päivitysmetodia.
88     }
89 }
90 }
```

```
1 package com.aueta.app.pakettivahti.receiver;
2
3 import android.app.AlarmManager;
4 import android.app.PendingIntent;
5 import android.content.BroadcastReceiver;
6 import android.content.Context;
7 import android.content.Intent;
8 import android.util.Log;
9
10 import com.aueta.app.pakettivahti.service.ParcelService;
11
12 public class ParcelBootReceiver extends BroadcastReceiver {
13
14     // Lokiinkirjoituksen tagiksi luokan nimi.
15     private static final String TAG = ParcelBootReceiver.class.getSimpleName();
16
17     public static final int REQUEST_CODE = 1234567;
18
19     // Asetetaan oletuspäivitysväliksi tunti.
20     private static final long DEFAULT_INTERVAL = AlarmManager.INTERVAL_HOUR;
21
22     @Override
23     public void onReceive(Context context, Intent intent) {
24
25         // Luodaan ajastettu intent, jolle asetetaan tunnus, jolla se löydetään tarvittaessa myöhemminkin.
26
27         Intent timedIntent = new Intent(context, ParcelService.class);
28
29         // Lisätään ajastettuun intenttiin tieto siitä, että kyseessä on ajastettu toiminto.
30         timedIntent.putExtra("ajastettu", true);
31
32         // FLAG_UPDATE_CURRENT päivittää mahdollisesti jo käynnissä olevan intentin, tai luo uuden.
33         PendingIntent timedOperation = PendingIntent.getService(context, REQUEST_CODE,
34             timedIntent, PendingIntent.FLAG_UPDATE_CURRENT);
35
36         // Haetaan järjestelmän Alarm Service.
37         AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
38
39         // Asetetaan toistuva tehtävä.
40         alarmManager.setInexactRepeating(AlarmManager.RTC, System.currentTimeMillis(), DEFAULT_INTERVAL,
41             timedOperation);
42     }
43 }
44 }
```

```
1 package com.aueta.app.pakettivahti.dao;
2
3 import android.net.Uri;
4 import android.provider.BaseColumns;
5
6
7 /**
8  * Lähetysten tietokantatalennukseen liittyvät vakioarvot sisältävä luokka.
9  *
10 * @author Vesa Mäkeläinen
11 *
12 */
13 public class ParcelContract {
14
15     // Tietokantaan liittyvät vakioarvot.
16
17     public static final String DB_NAME = "parcels.db"; // Tietokantatiedoston nimi.
18     public static final int DB_VERSION = 2; // Tietokannan versionumero.
19     public static final String TABLE = "parcel"; // Lähetysten tiedot sisältävän tietokantataulun nimi.
20
21     // Hakutulosten oletuslajittelu statuksen mukaan laskevassa järjestyksessä.
22     public static final String SORTING = Column.STATUS + " DESC";
23
24     // Tietokantataulun sarakkeiden nimet.
25     public class Column
26     {
27         public static final String ID = BaseColumns._ID;
28         public static final String NAME = "name";
29         public static final String CODE = "code";
30         public static final String STATUS = "status";
31         public static final String TIME = "time";
32     }
33
34
35     // Content Providerin vakioarvot.
36
37     // Authority määrittää, kuka saa käyttää content provideria.
38     public static final String AUTHORITY = "com.aueta.app.pakettivahti.dao.ParcelProvider";
39
40     // Content providerin yksilöivä osoite.
41     public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/" + TABLE);
42
43     // Content provider vastaa yhden lähetyksen tiedot palauttaen:
44     // (vastauksen MIME-tyyppi on ennen /-merkkiä)
45     public static final int PARCEL_ITEM = 1;
46     public static final String PARCEL_TYPE_ITEM =
47         "vnd.android.cursor.item/vnd.com.aueta.app.pakettivahti.provider.parcel";
48
49     // Tai kaikki lähetytiedot palauttaen:
50     public static final int PARCEL_DIR = 2;
51     public static final String PARCEL_TYPE_DIR =
52         "vnd.android.cursor.dir/vnd.com.aueta.app.pakettivahti.provider.parcel";
53
54     public static final String DEFAULT_SORT = Column.STATUS + " DESC";
55 }
```



```
1 package com.aueta.app.pakettivahti.dao;
2
3 import android.content.ContentProvider;
4 import android.content.ContentUris;
5 import android.content.ContentValues;
6 import android.content.UriMatcher;
7 import android.database.Cursor;
8 import android.database.sqlite.SQLiteDatabase;
9 import android.database.sqlite.SQLiteQueryBuilder;
10 import android.net.Uri;
11 import android.text.TextUtils;
12 import android.util.Log;
13
14
15 /**
16  * ParcelProvider on Content Provider -olio, jonka kautta lähetystietojen
17  * tietokantatoiminnot ovat käytettävissä.
18  *
19  * @author Vesa Mäkeläinen
20  *
21  */
22 public class ParcelProvider extends ContentProvider {
23
24     private static final String TAG = ParcelProvider.class.getSimpleName();
25
26     private DBHelper dbHelper;
27
28     // Tämän avulla selvitetään, onko URI-osoite yhteen oloon, vai kaikkiin.
29     private static final UriMatcher sURIMatcher = new UriMatcher(UriMatcher.NO_MATCH);
30
31     static
32     {
33         sURIMatcher.addURI(ParcelContract.AUTHORITY, ParcelContract.TABLE, ParcelContract.PARCEL_DIR);
34         sURIMatcher.addURI(ParcelContract.AUTHORITY, ParcelContract.TABLE + "/" + "#",
35                             ParcelContract.PARCEL_ITEM);
36     }
37
38     @Override
39     public int delete(Uri uri, String selection, String[] selectionArgs) {
40
41         // Lähetysten tietojen poistaminen.
42
43         String where;
44
45         // Poistetaanko yksi vai monta?
46         switch (sURIMatcher.match(uri))
47         {
48             case ParcelContract.PARCEL_DIR:
49
50                 // Jos on annettu rajausehto, otetaan se käyttöön.
51                 where = (selection == null) ? "1" : selection;
52                 break;
53
54             case ParcelContract.PARCEL_ITEM:
55                 long id = ContentUris.parseId(uri);
56                 where = ParcelContract.Column.ID + "=" + id + (TextUtils.isEmpty(selection) ?
57                                                                     "" : "and ( " + selection + " )");
58                 break;
59
60             default:
61                 throw new IllegalArgumentException("Illegal uri: " + uri);
62         }
63
64         // Suoritetaan tietokantaoperaatio. Saadaan muokattujen rivien määrä.
65         SQLiteDatabase database = dbHelper.getWritableDatabase();
66         int changedRows = database.delete(ParcelContract.TABLE, where, selectionArgs);
67
68         // Lähetetään muutoksesta ilmoitus.
69         if(changedRows > 0)
70         {
71             getContext().getContentResolver().notifyChange(uri, null);
72         }
73
74         return changedRows;
75     }
76
77     @Override
78     public String getType(Uri uri) {
79
80         // Palauttaa URI:n tyyppin: yksi lähetys tai monta.
81
82         switch(sURIMatcher.match(uri))
83         {
84             case ParcelContract.PARCEL_DIR:
85                 return ParcelContract.PARCEL_TYPE_DIR;
86
87             case ParcelContract.PARCEL_ITEM:
88                 return ParcelContract.PARCEL_TYPE_ITEM;
89
90             default:
91                 throw new IllegalArgumentException("Illegal uri: " + uri);
92         }
93     }
94 }
```

```
92     }
93 }
94
95 @Override
96 public Uri insert(Uri uri, ContentValues values) {
97     Log.d("Provider", "Insert values: " + values);
98
99     // Palautettava URI, joka sisältää lisätyn lähetyksen olion URI:n.
100     Uri returnUri = null;
101
102     // Tarkistetaan, että annettu URI on oikea.
103     if (sURIMatcher.match(uri) != ParcelContract.PARCEL_DIR)
104     {
105         throw new IllegalArgumentException("Illegal uri: " + uri);
106     }
107
108     // Avataan tietokantayhteys kirjoittamista varten.
109     SQLiteDatabase database = dbHelper.getWritableDatabase();
110
111     // Lisätään annetut arvot tietokantaan, joka palauttaa lisätyn rivin id:n.
112     long rowId = database.insertWithOnConflict(ParcelContract.TABLE, null, values,
113                                             SQLiteDatabase.CONFLICT_IGNORE);
114
115     // Virhetapauksessa tietokanta palauttaa -1.
116     if(rowId != -1)
117     {
118         // Luetaan lisätylle oliolle luotu id.
119         long id = rowId; //values.getAsLong(ParcelContract.Column.ID);
120
121         // Luodaan palautettava URI, joka sisältää lisätyn olion id:n
122         returnUri = ContentUris.withAppendedId(uri, id);
123
124         // Ilmoitetaan, että kyseisen URIn osoittama tieto on muuttunut.
125         getContext().getContentResolver().notifyChange(uri, null);
126     }
127
128     // Palautetaan tallennettuun olioon osoittava URI.
129     return returnUri;
130 }
131
132 @Override
133 public boolean onCreate()
134 {
135     dbHelper = new DBHelper(getContext());
136     return true;
137 }
138
139 @Override
140 public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
141                     String sortOrder) {
142
143     // Rakennetaan kysely siihen tarkoitetulla oliolla.
144     SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
145     queryBuilder.setTables(ParcelContract.TABLE);
146
147     // Tutkitaan saatu uri.
148     switch (sURIMatcher.match(uri))
149     {
150     case ParcelContract.PARCEL_DIR:
151         break;
152
153     case ParcelContract.PARCEL_ITEM:
154
155         // On pyydetty yksittäistä lähetystä, joten lisätään rajausehto sen id:n perusteella.
156         queryBuilder.appendWhere(ParcelContract.Column.ID + "=" + uri.getLastPathSegment());
157         break;
158
159     default:
160         // Vääränlainen uri.
161         throw new IllegalArgumentException("Illegal uri: " + uri);
162     }
163
164     // Valitaan palautettavan tiedon lajittelu, joko oletus tai annettu.
165     String orderBy = (TextUtils.isEmpty(sortOrder)) ? ParcelContract.DEFAULT_SORT : sortOrder;
166
167     // Avataan tietokantayhteys lukua varten ja tehdään haku.
168     SQLiteDatabase database = dbHelper.getReadableDatabase();
169     Cursor cursor = queryBuilder.query(database, projection, selection, selectionArgs, null, null, orderBy);
170
171     // Ilmoitetaan cursor-oliolle sen sisältämän datan uri,
172     // jotta se tietää myöhemmin päivittää sisältöään saatuaan muutosviestin.
173     cursor.setNotificationUri(getContext().getContentResolver(), uri);
174
175     // Palautetaan pyydetyn datan sisältävä cursor.
176     return cursor;
177 }
178
179 @Override
180 public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
```

```
183
184     // Päivitetään tietokannassa olevaa tietoa.
185
186     // SQL-lauseen rajaavan osan kokoamista varten.
187     String where;
188
189     // URIn pitää sisältää lähetyksen id, jotta muutokset voidaan kohdentaa oikein,
190     // joten tietolistauksen URI ei kelpaa.
191     switch (sURIMatcher.match(uri))
192     {
193
194         // Tietolistauksen URI.
195         case ParcelContract.PARCEL_DIR:
196             where = selection;
197             break;
198
199         // Yksittäisen lähetyksen URI.
200         case ParcelContract.PARCEL_ITEM:
201
202             // Luetaan annetun URIn lopusta id ja kootaan sen avulla SQL:n WHERE-rajaus.
203             long id = ContentUris.parseId(uri);
204             where = ParcelContract.Column.ID + "=" + id + (TextUtils.isEmpty(selection) ?
205                                                         "" : " AND ( " + selection + " )");
206             break;
207
208         default:
209             // Vääräntyyppinen URI.
210             throw new IllegalArgumentException("Illegal uri: " + uri);
211     }
212
213     Log.d(TAG, "Päivitetään tietokantaan arvot: " + values);
214
215     // Avataan tietokantayhteys kirjoitusta varten ja tehdään päivitys.
216     SQLiteDatabase database = dbHelper.getWritableDatabase();
217     int changedRows = database.update(ParcelContract.TABLE, values, where, selectionArgs);
218
219     // Tietokanta palauttaa muutettujen rivien lukumäärän.
220     if(changedRows > 0)
221     {
222         // Ilmoitetaan onnistuneen päivityksen jälkeen muutoksesta.
223         getContext().getContentResolver().notifyChange(uri, null);
224     }
225
226     return changedRows;
227 }
228
229 }
```

```
1 package com.aueta.app.pakettivahti.service;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.net.HttpURLConnection;
7 import java.net.MalformedURLException;
8 import java.net.URL;
9 import java.text.ParseException;
10 import java.text.SimpleDateFormat;
11 import java.util.Date;
12
13 import android.app.IntentService;
14 import android.content.ContentValues;
15 import android.content.Intent;
16 import android.content.SharedPreferences;
17 import android.database.Cursor;
18 import android.net.Uri;
19 import android.os.IBinder;
20 import android.preference.PreferenceManager;
21 import android.util.Log;
22
23 import com.aueta.app.pakettivahti.AddNewItemActivity;
24 import com.aueta.app.pakettivahti.dao.ParcelContract;
25 import com.aueta.app.pakettivahti.domain.Status;
26
27 public class ParcelService extends IntentService {
28
29     // Lokiikirjoituksen tagi.
30     static final String TAG = "ParcelService";
31
32     public ParcelService() {
33
34         // IntentService tarvitsee konstruktoriinsa nimen.
35         super(TAG);
36     }
37
38     @Override
39     public IBinder onBind(Intent arg0) {
40
41         // Tämä on unbound service, eli tämän elinkaari ei ole kytkäksissä käynnistäneeseen activityyn.
42         // Palautetaan siis null, koska binderia ei ole.
43
44         return null;
45     }
46
47     @Override
48     public void onCreate() {
49         // Service on luotu.
50         super.onCreate();
51     }
52
53     @Override
54     protected void onHandleIntent(Intent intent) {
55         // Servicen tässä toteutettu työskentely tapahtuu omassa säikeessä.
56         // Työn valmistuttua kutsutaan saman tien onDestroy().
57
58         // TESTIKÄYTTÖÖN!!
59         boolean onAjastettu = intent.getBooleanExtra("ajastettu", false);
60
61         // Luetaan intentistä selvitettävän lähetyksen seurantakoodi.
62         String codeFromIntent = intent.getStringExtra(AddNewItemActivity.PARCEL_CODE);
63
64         // Lasketaan saapuneita, nollataan laskuri.
65         int arrivals = 0;
66
67         Cursor cursor = null;
68
69         if(codeFromIntent == null || codeFromIntent.isEmpty())
70         {
71             // Ei saatu koodia, joten haetaan kaikkien seurattavien lähetysten tiedot.
72
73             // Haetaan kaikki lähetykset.
74             cursor = getContentResolver().query(ParcelContract.CONTENT_URI, null, null, null, null);
75         }
76         else
77         {
78             // Haetaan annettu lähetykset.
79             cursor = getContentResolver().query(ParcelContract.CONTENT_URI, null, "code = ?",
80                 new String[] {codeFromIntent}, null);
81         }
82
83         // Lopetetaan, jos ei ole seurattavia lähetyksiä.
84         if(!cursor.moveToFirst())
85         {
86             // Cursor on tyhjä. Ei seurattavia lähetyksiä.
87             return;
88         }
89
90         // Jos haetaan kaikki lähetykset, tallennetaan hakuaika asetustietoihin.
```

```
92     if(codeFromIntent == null || codeFromIntent.isEmpty())
93     {
94         // Haetaan asetukset käyttöön. Haetaan myös asetuksia kirjoittava editori.
95         SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
96         SharedPreferences.Editor editor = preferences.edit();
97
98         // Asetetaan päivitysaika nykyhetkeen.
99         long lastUpdateTime = new Date().getTime();
100
101         // Tallennetaan päivitysaika.
102         editor.putLong("lastUpdated", lastUpdateTime).commit();
103     }
104
105     // Käydään lähetykset läpi yksi kerrallaan. Nyt kursorissa on ainakin yksi lähetyk.
106     do
107     {
108         // Haetaan käsiteltävän lähetyksen seurantakoodi.
109         String code = cursor.getString(cursor.getColumnIndex(ParcelContract.Column.CODE));
110
111         // Haetaan status testipalvelinta varten.
112         int oldStatus = cursor.getInt(cursor.getColumnIndex(ParcelContract.Column.STATUS));
113
114         Status newStatus = Status.UNKNOWN;
115
116         // TESTAUSTA varten pelkkä salaamaton yhteys.
117         HttpURLConnection urlConnection = null;
118
119         // Oikean palvelimen kanssa käytetään salattua yhteyttä.
120         //HttpsURLConnection urlConnection = null;
121
122         try
123         {
124             // Muodostetaan Itellan seurantapalvelun url-osoite.
125             //URL url = new URL("https://posti.mobi/packet-tracking/" + code + "?lang=fi");
126
127             // Testipalvelin vastaanottaa statustiedon ja palauttaa seuraavaa tilaa vastaavan sivun.
128             URL url = new URL("http://pakettivahti.aueta.com/testipalvelu.php?status=" + oldStatus );
129
130             // Oikean palvelimen yhteys.
131             urlConnection = (HttpURLConnection) url.openConnection();
132
133             // Testipalvelimen yhteys.
134             urlConnection = (HttpURLConnection) url.openConnection();
135
136
137             // Kääritään input stream puskuroiduksi.
138             InputStream in = new BufferedInputStream(urlConnection.getInputStream());
139
140             int read = 0;
141             boolean readFailed = false;
142             boolean shouldReadTime = false;
143             String timestamp = "";
144             newStatus = Status.UNKNOWN;
145
146             // Luetaan merkkejä, kunnes tullaan loppuun (-1) tai löydetään etsitty.
147             while((read = in.read()) != -1)
148             {
149                 // Onko luettu '<', jolloin pitää tutkia tarkemmin seuraavat merkit.
150                 if(read == '<')
151                 {
152                     String tag = "";
153                     timestamp = "";
154
155                     // Etsitään <div class="ARK"> -tyyppistä tagia.
156                     // Luetaan seuraavat 14 merkkiä eli jälkimmäiseen lainausmerkkiin asti.
157                     for(int i = 0; i < 14; i++)
158                     {
159                         read = in.read();
160
161                         if(read != -1)
162                         {
163                             tag += "" + (char)read;
164                         }
165                         else
166                         {
167                             // Loppui kesken, ei tulosta.
168                             readFailed = true;
169                             break;
170                         }
171                     }
172
173                     // Ei tutkita enempää, jos ei luettu riittävästi merkkejä.
174                     if(readFailed)
175                     {
176                         continue;
177                     }
178
179                     //Log.d(TAG, "Luettu tagi: " + tag);
180
181                     if(tag.startsWith("div class="))
182                     {
```

```
183 // On oikeanlainen tagi, tutkitaan lisää.
184
185 // luetaan 3 viimeistä merkkiä ja etsitään status.
186 String statusName = tag.substring(tag.length() - 3 - 1);
187
188 if(statusName.contains("ARK"))
189 {
190     // Arkistoitu.
191     newStatus = Status.ARCHIVED;
192     shouldReadTime = true;
193 }
194 else if (statusName.contains("LUO"))
195 {
196     // Noudettu.
197     newStatus = Status.FINISHED;
198     shouldReadTime = true;
199 }
200 else if (statusName.contains("HYL"))
201 {
202     // Odottaa noutoa.
203     newStatus = Status.READY;
204     shouldReadTime = true;
205 }
206 else if (statusName.contains("REK"))
207 {
208     // Rekisteröity.
209     newStatus = Status.REGISTERED;
210     shouldReadTime = true;
211 }
212
213 // Luetaan aikaleima, jos tarpeen.
214 if(shouldReadTime)
215 {
216     // Seuraava rivi on muotoa: <p>13 March 2014 18:23</p>
217
218     // Luetaan pois <p>-tagi. Ja sitä ennen loppuun edellinen tagi.
219     while(in.read() != '>');
220     while(in.read() != '>');
221
222     // Kootaan p-tagin sisältö merkkijonoksi.
223     while(((read = in.read()) != -1) && (read != '<'))
224     {
225         timestamp += "" + (char)read;
226     }
227     break;
228 }
229 }
230 }
231 }
232 }
233
234 ContentValues values = new ContentValues();
235
236 // Muodostetaan tämän lähetyksen yksilöivä uri.
237 Uri itemUri = Uri.parse(ParcelContract.CONTENT_URI + "/" +
238     cursor.getLong(cursor.getColumnIndex(ParcelContract.Column.ID)));
239
240 if(newStatus != Status.UNKNOWN)
241 {
242     // Lähetystieto on päivittynyt, joten päivitetään muutos myös tietokantaan.
243     values.put(ParcelContract.Column.STATUS, newStatus.ordinal());
244
245     // Muunnetaan aikaleima millisekunneiksi tallennusta varten.
246     long time = parseTimestamp(timestamp);
247
248     if(time != 0)
249     {
250         // Tallennetaan vain, jos saatiin luotua aika.
251         values.put(ParcelContract.Column.TIME, time);
252     }
253 }
254 else
255 {
256     // Ei rekisteröity vielä. Tutkitaan, muuttuiko ensimmäisen tarkastuksen myötä.
257     if(oldStatus == Status.DEFAULT.ordinal())
258     {
259         // Ensimmäinen tarkistus nettipalvelusta, joten status muuttuu tuntemattomaksi
260         // (ei vielä rekisteröity).
261         values.put(ParcelContract.Column.STATUS, Status.UNKNOWN.ordinal());
262     }
263
264     // Muodostetaan uusi seuranta-aika.
265     values.put(ParcelContract.Column.TIME, new Date().getTime());
266 }
267
268 // Päivitetään tietokantaan.
269 int counter = getContentResolver().update(itemUri, values, null, null);
270
271 // Noutoa odottavat.
272 if(newStatus == Status.READY)
```

```
274         {
275             // Lasketaan saapuneet.
276             arrivals++;
277
278             // Ilmoitetaan käyttäjälle saapuneesta lähetyksestä lähettämällä ilmoitus,
279             // joka sisältää lähetyksen nimen.
280             Intent arrivedIntent = new Intent("com.aueta.app.pakettivahti.action.PARCEL_ARRIVED");
281             arrivedIntent.putExtra(ParcelContract.Column.NAME,
282                                 cursor.getString(cursor.getColumnIndex(ParcelContract.Column.NAME)));
283
284             sendBroadcast(arrivedIntent);
285         }
286         else if(newStatus.ordinal() >= Status.FINISHED.ordinal())
287         {
288             // Poistetaan noudettu lähetyk.
289             getContentResolver().delete(itemUri, null, null);
290         }
291
292         } catch (MalformedURLException e) {
293             e.printStackTrace();
294         } catch (IOException e) {
295             e.printStackTrace();
296         }
297         finally
298         {
299             // Jos yhteys on avattu, suljetaan se.
300             if(urlConnection != null)
301             {
302                 urlConnection.disconnect();
303             }
304         }
305     } while(cursor.moveToNext());
306
307     // Päivitetään mahdolliseen widgettiin noudettavissa olevien lukumäärä.
308     Intent intentToWidget = new Intent("com.aueta.app.pakettivahti.action.UPDATE_WIDGET");
309     intentToWidget.putExtra("updateCount", arrivals);
310
311     sendBroadcast(intentToWidget);
312 }
313
314 /**
315  * Muuttaa merkkijonona saamansa aikaleiman millisekunneiksi.
316  *
317  * @param timestamp merkkijono, josta aika muunnetaan.
318  * @return luettu merkkijono millisekunneina.
319  */
320 private long parseTimestamp(String timestamp)
321 {
322     // Jos päivämäärämerkkijono on tyhjä, palautetaan nolla.
323     if(timestamp == null || timestamp.isEmpty())
324     {
325         return 0;
326     }
327
328     // Asetetaan muotoilu, josta luetaan. "20 March 2014 14:59"
329     SimpleDateFormat dateFormat = new SimpleDateFormat("dd MMMM yyyy HH:mm");
330     Date dateFromString = null;
331
332     // Muutetaan aikaleima merkkijonosta Date-olioksi.
333     try {
334         dateFromString = dateFormat.parse(timestamp);
335     } catch (ParseException e) {
336         e.printStackTrace();
337     }
338
339     if(dateFromString == null)
340     {
341         Log.d(TAG, "Aikaleiman luku epäonnistui: " + timestamp);
342         return 0;
343     }
344     else
345     {
346         // Kaikki onnistui, palautetaan aika millisekunneina.
347         return dateFromString.getTime();
348     }
349 }
350
351 @Override
352 public void onDestroy() {
353     // Service tuhoetaan.
354     super.onDestroy();
355 }
356
357 }
```

```
1 package com.aueta.app.pakettivahti.dao;
2
3 import java.text.SimpleDateFormat;
4 import java.util.Date;
5 import java.util.Locale;
6
7 import android.database.Cursor;
8 import android.graphics.Color;
9 import android.support.v4.widget.SimpleCursorAdapter.ViewBinder;
10 import android.view.View;
11 import android.widget.LinearLayout;
12 import android.widget.TextView;
13
14 import com.aueta.app.pakettivahti.R;
15 import com.aueta.app.pakettivahti.domain.Status;
16
17
18 public class ParcelViewBinder implements ViewBinder {
19
20     @Override
21     public boolean setViewValue(View view, Cursor cursor, int columnIndex)
22     {
23         // Valitaan käsiteltävä rivi.
24         switch(view.getId())
25         {
26
27             case R.id.tvItemName: // Lähetyksen nimi.
28
29                 // Muunnetaan saatu view TextView:ksi, koska list-itemin nimikenttä on sellainen.
30                 TextView tName = (TextView)view;
31
32                 // Haetaan ja asetetaan nimi tekstikenttään.
33                 tName.setText(cursor.getString(cursor.getColumnIndex(ParcelContract.Column.NAME)));
34
35                 // Ilmoitetaan, että tämä sijoitus on tehty, eikä muita bindereita tarvita enää tälle kentälle.
36                 return true;
37
38             case R.id.tvItemStatus: // Statusrivi.
39
40                 // Statusriville pitää koota aikaleima ja statusteksti.
41                 // Luetaan aluksi aika ja muutetaan se Date-olioksi.
42                 int timeColumn = cursor.getColumnIndex(ParcelContract.Column.TIME);
43                 long timeLong = cursor.getLong(timeColumn);
44                 Date time = new Date(timeLong);
45
46                 // Luodaan lokalisoitu aikaformaatti, jossa esitetään päivä ja kuukausi numeroin
47                 // ja kellonaika minuutin tarkkuudella.
48                 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("d.M.yyyy - HH.mm",
49                                                                                               Locale.getDefault());
50
51                 String localizedTime = simpleDateFormat.format(time);
52
53                 // Haetaan samoin status-koodi.
54                 int statusColumn = cursor.getColumnIndex(ParcelContract.Column.STATUS);
55                 int statusInt = cursor.getInt(statusColumn);
56                 Status status = Status.values()[statusInt];
57
58                 // Luodaan näytettävä status-rivi.
59                 String statusText = localizedTime + " - " + status;
60
61                 // Haetaan TextView, ja sijoitetaan status-rivi siihen.
62                 TextView textView = (TextView) view;
63                 textView.setText(statusText);
64
65                 // Jos lähetyksen on noudettavissa, korostetaan tietosolu taustaväriä.
66                 if(statusInt == 3)
67                 {
68                     LinearLayout cell = (LinearLayout)(view.getParent());
69                     cell.setBackgroundColor(Color.GREEN);
70                 }
71                 else
72                 {
73                     LinearLayout cell = (LinearLayout)(view.getParent());
74                     cell.setBackgroundColor(Color.WHITE);
75                 }
76
77                 // Palautetaan true merkiksi siitä, että data saatiin bindattua.
78                 return true;
79
80             default:
81
82                 // Joku muu saa käsitellä mahdolliset muut kentät.
83                 return false;
84
85         }
86     }
87
88 }
89
90
91 }
```



```
1 <menu xmlns:android="http://schemas.android.com/apk/res/android" >
2
3     <item
4         android:id="@+id/action_settings"
5         android:orderInCategory="100"
6         android:showAsAction="never"
7         android:title="@string/action_settings"/>
8
9 </menu>
```

```
1 package com.aueta.app.pakettivahti;
2
3 import android.app.Activity;
4 import android.content.pm.ActivityInfo;
5 import android.os.Bundle;
6 import android.view.Menu;
7 import android.widget.TextView;
8
9 public class ShowCodeActivity extends Activity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_show_code);
15
16         // Käännetään näyttö vaakasuuntaan.
17         this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
18
19         // Haetaan seurantakoodi intentin tiedoista.
20         String code = getIntent().getStringExtra(AddNewItemActivity.PARCEL_CODE);
21
22         // Näytetään koodi.
23         TextView tvShowCode = (TextView) findViewById(R.id.tvShowCode);
24         tvShowCode.setText(code);
25     }
26
27     @Override
28     public boolean onCreateOptionsMenu(Menu menu) {
29         // Inflate the menu; this adds items to the action bar if it is present.
30         //getMenuInflater().inflate(R.menu.show_code, menu);
31         return true;
32     }
33 }
```

```
1 package com.aueta.app.pakettivahti.domain;
2
3
4 /**
5  * Enum-luettelo, joka sisältää lähetyksen elinkaaren vaiheet.
6  *
7  * Tulostaa myös toString-metodeilla jokaisen vaiheen statustekstin.
8  *
9  * @author Vesa
10  *
11  */
12 public enum Status {
13
14     // Oletusarvo, kun uusi lähetyk on luotu järjestelmään, eikä tietoja ole vielä haettu nettipalvelusta.
15     DEFAULT
16     {
17         @Override
18         public String toString()
19         {
20             return "----- Odotetaan tietoja. -----";
21         }
22     },
23
24     // Lähetyksestä ei ole rekisteröity nettipalveluun.
25     UNKNOWN
26     {
27         @Override
28         public String toString()
29         {
30             return "Ei tietoja.";
31         }
32     },
33
34     // Lähetyk on rekisteröity nettipalveluun.
35     REGISTERED
36     {
37         @Override
38         public String toString()
39         {
40             return "Rekisteröity.";
41         }
42     },
43
44     // Lähetyk on toimitettu perille ja odottaa noutoa.
45     READY
46     {
47         @Override
48         public String toString()
49         {
50             return "Noudettavissa.";
51         }
52     },
53
54     // Lähetyk on noudettu.
55     FINISHED
56     {
57         @Override
58         public String toString()
59         {
60             return "Noudettu.";
61         }
62     },
63
64     // Lähetyksen tiedot on arkistoitu Postin palvelun arkistoon.
65     ARCHIVED
66     {
67         @Override
68         public String toString()
69         {
70             return "Arkistoitu.";
71         }
72     }
73 }
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="app_name">Pakettivahti</string>
4   <string name="action_about">Tietoja sovelluksesta</string>
5   <string name="action_about_condensed">Tietoja</string>
6   <string name="action_new">Uusi</string>
7   <string name="app_about_tag">Tietoja sovelluksesta</string>
8   <string name="app_about_text">Pakettivahti ilmoittaa saapuneista postilähetyksistä.\n\nLisää uusi lähetys painamalla</string>
9   <string name="app_about_add_text">Anna lähetysten seurantakoodi ja keksi paketille nimi.</string>
10  <string name="app_button_ok">OK</string>
11  <string name="title_activity_add_new_item">Lisää uusi lähetys</string>
12  <string name="action_settings">Settings</string>
13
14  <string name="show_code">FI1234546567890123456FI</string>
15
16  <string name="add_code">Seurantakoodi</string>
17  <string name="add_name">Nimi</string>
18  <string name="add_button"> Lisää </string>
19  <string name="add_code_hint">Syötä lähetysten seurantakoodi.</string>
20  <string name="add_name_hint">Anna lähetykselle kuvaava nimi.</string>
21  <string name="title_activity_show_code">Lähetysten seurantakoodi</string>
22 </resources>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
3   android:initialLayout="@layout/widget"
4   android:minHeight="52dp"
5   android:minWidth="146dp"
6   android:updatePeriodMillis="0" >
7
8 </appwidget-provider>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:id="@+id/layout"
4   android:orientation="vertical"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:layout_margin="8dp"
8   android:background="@drawable/widget_tausta" >
9
10
11   <TextView
12     android:id="@+id/updateTime"
13     style="@android:style/TextAppearance.Small"
14     android:layout_width="match_parent"
15     android:layout_height="wrap_content"
16     android:layout_gravity="top"
17     android:gravity="left"
18     android:layout_margin="4dp"
19     android:text="Pakettivahti" >
20   </TextView>
21
22
23   <TextView
24     android:id="@+id/updateText"
25     style="@android:style/TextAppearance.Small"
26     android:layout_width="match_parent"
27     android:layout_height="wrap_content"
28     android:layout_gravity="bottom"
29     android:gravity="center_horizontal"
30     android:layout_margin="4dp"
31     android:text="Haetaan tietoja..." >
32   </TextView>
33
34 </LinearLayout>
35
36
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <shape xmlns:android="http://schemas.android.com/apk/res/android"
3     android:shape="rectangle" >
4
5     <stroke
6         android:width="2dp"
7         android:color="#FFFFFF" />
8
9     <gradient
10        android:angle="225"
11        android:endColor="#DD2ECCFA"
12        android:startColor="#DD000000" />
13
14    <corners
15        android:bottomLeftRadius="7dp"
16        android:bottomRightRadius="7dp"
17        android:topLeftRadius="7dp"
18        android:topRightRadius="7dp" />
19
20 </shape>
```

```
1 //
2 // AddParcelViewController.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import "AddParcelViewController.h"
10 #import "Parcel.h"
11
12 @interface AddParcelViewController ()
13
14 @property (weak, nonatomic) IBOutlet UITextField *tfCode;
15 @property (weak, nonatomic) IBOutlet UITextField *tfName;
16
17 @end
18
19 @implementation AddParcelViewController
20
21 #pragma mark - UIViewController
22
23 - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
24 {
25     self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
26     if (self) {
27         // Custom initialization here
28     }
29     return self;
30 }
31
32 - (void)viewDidLoad
33 {
34     [super viewDidLoad];
35     // Do any additional setup after loading the view.
36 }
37
38
39 - (void)didReceiveMemoryWarning
40 {
41     [super didReceiveMemoryWarning];
42     // Dispose of any resources that can be recreated.
43     NSLog(@"DidReceiveMemoryWarning.");
44 }
45
46 - (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
47 {
48     // Poistetaan virtuaalinäppäimistö näkyvistä, jos kosketetaan taustaa tekstikentän ulkopuolella.
49     [self.view endEditing:YES];
50     [super touchesBegan:touches withEvent:event];
51 }
52 }
53
54 #pragma mark - TextField Delegate Methods
55
56 - (BOOL)textFieldShouldReturn:(UITextField *)textField
57 {
58     // Käyttäjä on napauttanut virtuaalinäppäimistön Enter-nappia.
59
60     // Kumpi tekstikenttä?
61     if(textField.tag == 1)
62     {
63         // On lähetystunnuskenttä. Siirretään fokus nimikenttään.
64         [textField resignFirstResponder];
65         [self.tfName becomeFirstResponder];
66
67         // Ei tehdä mitään muuta (tekstikentän omaa return-käsittelijää ei suoriteta).
68         return NO;
69     }
70     else
71     {
72         // On nimikenttä. Tallennetaan uusi lähetys.
73         [self saveAndExit];
74         return NO;
75     }
76 }
77
78 - (void)saveAndExit
79 {
80     // Jos lähetykselle ei ole annettu koodia, ei tallenneta.
81     if([self.tfCode.text isEqualToString:@""])
82     {
83         // Tyhjä on. Ei saa tallentaa eikä lopettaa.
84         NSLog(@"Koodikenttä on tyhjä!");
85         return;
86     }
87
88     // Jos delegaatti löytyy, pyydetään uusi alustettu hallinnoitu lähetysolio, johon annetut tiedot lisätään.
89     if (self.delegate) {
90         Parcel *parcelToAdd = [self.delegate getNewParcel];
91     }
```

```
92     // Asetetaan lähetyksen tiedot.
93     [parcelToAdd setCode:self.tfCode.text];
94
95     if([self.tfName.text isEqualToString:@""])
96     {
97         // Nimikenttä on tyhjä. Kopioidaan koodi nimeksi.
98         [parcelToAdd setName:self.tfCode.text];
99     }
100    else
101    {
102        [parcelToAdd setName:self.tfName.text];
103    }
104
105    // Valmis! Lähetyks on tallessa. Suljetaan näkymä.
106    [self dismissViewControllerAnimated:YES completion:nil];
107 }
108 else
109 {
110     // Virhe: delegaattia ei asetettu.
111     NSLog(@"Delegaattia ei asetettu.");
112     abort();
113 }
114 }
115
116
117 #pragma mark - Actions
118
119 - (IBAction)cancelButtonTap:(UIBarButtonItem *)sender {
120
121     // Poistutaan näkymästä tallentamatta tietoja.
122     [self dismissViewControllerAnimated:YES completion:nil];
123 }
124
125 - (IBAction)saveButtonTap:(UIBarButtonItem *)sender {
126
127     // Tallennetaan ja poistutaan.
128     [self saveAndExit];
129 }
130
131 @end
```

```
1 //
2 // AddParcelViewController.h
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10 #import "ParcelDataSourceDelegateProtocol.h"
11
12 @interface AddParcelViewController : UIViewController <UITextFieldDelegate>
13
14 @property (nonatomic, weak) id <ParcelDataSourceDelegateProtocol> delegate;
15
16 @end
```

```
1 //
2 // CDParcel.h
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 3.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import <CoreData/CoreData.h>
11
12
13 @interface CDParcel : NSObject
14
15 @property (nonatomic, retain) NSString * code;
16 @property (nonatomic, retain) NSString * name;
17 @property (nonatomic, retain) NSDate * updateTime;
18 @property (nonatomic, retain) NSNumber * status;
19
20 @end
```

```
1 //
2 // CDParcel.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 3.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import "CDParcel.h"
10
11
12 @implementation CDParcel
13
14 @dynamic code;
15 @dynamic name;
16 @dynamic updateTime;
17 @dynamic status;
18
19 @end
```

```
1 //
2 // main.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10
11 #import "PakettivahtiAppDelegate.h"
12
13 int main(int argc, char * argv[])
14 {
15     @autoreleasepool {
16         return UIApplicationMain(argc, argv, nil, NSStringFromClass([PakettivahtiAppDelegate class]));
17     }
18 }
```

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="5056"
3     systemVersion="12E55" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none"
4     useAutolayout="YES" initialViewController="vXZ-lx-hvc">
5     <dependencies>
6         <plugin identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="3733"/>
7     </dependencies>
8     <scenes>
9         <!--Pakettivahti View Controller-->
10        <scene sceneID="ufC-wZ-h7g">
11            <objects>
12                <viewController modalTransitionStyle="flipHorizontal" id="vXZ-lx-hvc"
13                    customClass="PakettivahtiViewController" sceneMemberID="viewController">
14                    <layoutGuides>
15                        <viewControllerLayoutGuide type="top" id="jyV-Pf-zRb"/>
16                        <viewControllerLayoutGuide type="bottom" id="2fi-mo-0CV"/>
17                    </layoutGuides>
18                    <view key="view" contentMode="scaleToFill" id="kh9-bI-dsS">
19                        <rect key="frame" x="0.0" y="0.0" width="320" height="568"/>
20                        <autoresizingMask key="autoresizingMask" flexibleMaxX="YES" flexibleMaxY="YES"/>
21                        <subviews>
22                            <tableView clipsSubviews="YES" contentMode="scaleToFill" fixedFrame="YES"
23                                alwaysBounceVertical="YES" showsHorizontalScrollIndicator="NO"
24                                dataMode="prototypes" style="plain" separatorStyle="default"
25                                rowHeight="40" sectionHeaderHeight="22" sectionFooterHeight="22"
26                                translatesAutoresizingMaskIntoConstraints="NO" id="dU4-sT-8o5">
27                                <rect key="frame" x="0.0" y="72" width="320" height="496"/>
28                                <autoresizingMask key="autoresizingMask" widthSizable="YES"
29                                    heightSizable="YES"/>
30                                <color key="backgroundColor" white="1" alpha="1"
31                                    colorSpace="calibratedWhite"/>
32                                <prototypes>
33                                    <tableViewCell contentMode="scaleToFill" selectionStyle="blue"
34                                        hidesAccessoryWhenEditing="NO" indentationLevel="1"
35                                        indentationWidth="0.0" reuseIdentifier="Cell"
36                                        labelText="7D9-eh-YNU" detailTextLabel="z6A-IF-taC"
37                                        style="IBUITableViewCellStyleSubtitle" id="bLD-Hi-54w">
38                                        <rect key="frame" x="0.0" y="22" width="320" height="40"/>
39                                        <autoresizingMask key="autoresizingMask"/>
40                                        <tableViewCellContentView key="contentView" opaque="NO"
41                                            clipsSubviews="YES" multipleTouchEnabled="YES"
42                                            contentMode="center" tableViewCell="bLD-Hi-54w"
43                                            id="F6y-xY-4hn">
44                                            <rect key="frame" x="0.0" y="0.0" width="320" height="39"/>
45                                            <autoresizingMask key="autoresizingMask"/>
46                                            <subviews>
47                                                <label opaque="NO" clipsSubviews="YES"
48                                                    multipleTouchEnabled="YES" contentMode="left"
49                                                    text="Teekannu" lineBreakMode="tailTruncation"
50                                                    baselineAdjustment="alignBaselines"
51                                                    adjustsFontSizeToFit="NO" id="7D9-eh-YNU">
52                                                    <rect key="frame" x="15" y="1" width="77" height="22"/>
53                                                    <autoresizingMask key="autoresizingMask"/>
54                                                    <fontDescription key="fontDescription" type="system"
55                                                        pointSize="18"/>
56                                                    <color key="textColor"
57                                                        cocoaTouchSystemColor="darkTextColor"/>
58                                                    <nil key="highlightedColor"/>
59                                                </label>
60                                                <label opaque="NO" clipsSubviews="YES"
61                                                    multipleTouchEnabled="YES" contentMode="left"
62                                                    text="10.3.2014 17.36 - Ei tietoja."
63                                                    lineBreakMode="tailTruncation"
64                                                    baselineAdjustment="alignBaselines"
65                                                    adjustsFontSizeToFit="NO" id="z6A-IF-taC">
66                                                    <rect key="frame" x="15" y="23" width="154"
67                                                        height="15"/>
68                                                    <autoresizingMask key="autoresizingMask"/>
69                                                    <fontDescription key="fontDescription" type="system"
70                                                        pointSize="12"/>
71                                                    <color key="textColor"
72                                                        cocoaTouchSystemColor="darkTextColor"/>
73                                                    <nil key="highlightedColor"/>
74                                                </label>
75                                            </subviews>
76                                        </tableViewCellContentView>
77                                </tableViewCell>
78                                <connections>
79                                    <segue destination="S4i-bk-n2L" kind="modal"
80                                        identifier="ShowCode" id="WNz-iq-KLJ"/>
81                                </connections>
82                            </tableViewCell>
83                        </prototypes>
84                        <connections>
85                            <outlet property="dataSource" destination="vXZ-lx-hvc" id="kZ8-bo-M3C"/>
86                            <outlet property="delegate" destination="vXZ-lx-hvc" id="LeU-gM-Veu"/>
87                        </connections>
88                    </tableView>
89                    <navigationBar contentMode="scaleToFill" fixedFrame="YES"
90                        translatesAutoresizingMaskIntoConstraints="NO" id="wb1-0q-Klx">
91                        <rect key="frame" x="0.0" y="20" width="320" height="44"/>
92                        <autoresizingMask key="autoresizingMask" widthSizable="YES"/>
```



```
92         flexibleMaxY="YES"/>
93     <items>
94         <navigationItem title="Pakettivahti" id="6q4-Zf-a5S">
95             <barButtonItem key="leftBarButtonItem" systemItem="refresh"
96                 id="D8o-oK-OTf">
97                 <connections>
98                     <action selector="refreshButtonTap:"
99                         destination="vXZ-lx-hvc" id="Bzd-nm-aC2"/>
100                 </connections>
101             </barButtonItem>
102             <barButtonItem key="rightBarButtonItem" systemItem="add"
103                 id="IsH-8q-wcg">
104                 <connections>
105                     <action selector="addButtonTap:" destination="vXZ-lx-hvc"
106                         id="OrR-SK-Qwk"/>
107                 </connections>
108             </barButtonItem>
109         </navigationItem>
110     </items>
111 </navigationBar>
112 </subviews>
113 <color key="backgroundColor" white="1" alpha="1" colorSpace="custom"
114     customColorSpace="calibratedWhite"/>
115 </view>
116 <nil key="simulatedTopBarMetrics"/>
117 <simulatedOrientationMetrics key="simulatedOrientationMetrics"/>
118 <connections>
119     <outlet property="tableView" destination="dU4-sT-8o5" id="G12-tn-cUd"/>
120     <segue destination="GVW-cy-vPs" kind="modal" identifier="AddParcel"
121         id="YLK-2N-wh0"/>
122 </connections>
123 </viewController>
124 <placeholder placeholderIdentifier="IBFirstResponder" id="x5A-6p-PRh"
125     sceneMemberID="firstResponder"/>
126 </objects>
127 </scene>
128 <!--Tracking Code View Controller-->
129 <scene sceneID="SvT-n9-ohU">
130     <objects>
131         <viewController id="S4i-bk-n2L" customClass="TrackingCodeViewController"
132             sceneMemberID="viewController">
133             <layoutGuides>
134                 <viewControllerLayoutGuide type="top" id="zkb-AJ-pLx"/>
135                 <viewControllerLayoutGuide type="bottom" id="W5R-q6-L2J"/>
136             </layoutGuides>
137             <view key="view" contentMode="scaleToFill" id="etv-w6-nKl">
138                 <rect key="frame" x="0.0" y="0.0" width="568" height="320"/>
139                 <autoresizingMask key="autoresizingMask" flexibleMaxX="YES" flexibleMaxY="YES"/>
140                 <subviews>
141                     <label opaque="NO" clipsSubviews="YES" userInteractionEnabled="NO"
142                         contentMode="left" horizontalHuggingPriority="251"
143                         verticalHuggingPriority="251" fixedFrame="YES" text="FI123456789FI"
144                         textAlignment="center" lineBreakMode="characterWrap" numberOfLines="2"
145                         baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
146                         preferredMaxLayoutWidth="528"
147                         translatesAutoresizingMaskIntoConstraints="NO" id="sh5-NG-y3R">
148                         <rect key="frame" x="20" y="139" width="528" height="41"/>
149                         <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
150                             flexibleMaxY="YES"/>
151                         <fontDescription key="fontDescription" type="boldSystem" pointSize="30"/>
152                         <nil key="highlightedColor"/>
153                     </label>
154                     <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
155                         contentHorizontalAlignment="center" contentVerticalAlignment="center"
156                         buttonType="roundedRect" lineBreakMode="middleTruncation"
157                         translatesAutoresizingMaskIntoConstraints="NO" id="cgd-5N-gYX">
158                         <rect key="frame" x="0.0" y="0.0" width="568" height="320"/>
159                         <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
160                             flexibleMaxY="YES"/>
161                         <inset key="contentEdgeInsets" minX="0.0" minY="-48" maxX="0.0" maxY="0.0"/>
162                         <state key="normal" title="Button">
163                             <color key="titleShadowColor" white="0.5" alpha="1"
164                                 colorSpace="calibratedWhite"/>
165                         </state>
166                         <connections>
167                             <action selector="returnButtonDoubleTap:" destination="S4i-bk-n2L"
168                                 eventType="touchDownRepeat" id="KT7-oP-fJh"/>
169                         </connections>
170                     </button>
171                 </subviews>
172                 <color key="backgroundColor" white="1" alpha="1" colorSpace="custom"
173                     customColorSpace="calibratedWhite"/>
174                 <simulatedOrientationMetrics key="simulatedOrientationMetrics"
175                     orientation="landscapeRight"/>
176             </view>
177             <navigationItem key="navigationItem" id="6Gg-3r-eDQ"/>
178             <simulatedOrientationMetrics key="simulatedOrientationMetrics"
179                 orientation="landscapeRight"/>
180             <connections>
181                 <outlet property="btReturn" destination="cgd-5N-gYX" id="vMG-dg-6tn"/>
182                 <outlet property="lblCode" destination="sh5-NG-y3R" id="s2t-1l-b3A"/>
```

```
183         </connections>
184     </viewController>
185     <placeholder placeholderIdentifier="IBFirstResponder" id="d1R-9S-aec"
186         userLabel="First Responder" sceneMemberID="firstResponder"/>
187 </objects>
188 <point key="canvasLocation" x="844" y="573"/>
189 </scene>
190 <!--Add Parcel View Controller-->
191 <scene sceneID="nR9-oe-9i2">
192     <objects>
193         <viewController modalTransitionStyle="flipHorizontal" id="GVW-cy-vPs"
194             customClass="AddParcelViewController" sceneMemberID="viewController">
195             <layoutGuides>
196                 <viewControllerLayoutGuide type="top" id="cg1-hQ-7Qe"/>
197                 <viewControllerLayoutGuide type="bottom" id="Ala-Xz-uJc"/>
198             </layoutGuides>
199             <view key="view" contentMode="scaleToFill" id="dfU-nl-4Du">
200                 <rect key="frame" x="0.0" y="0.0" width="320" height="568"/>
201                 <autoresizingMask key="autoresizingMask" flexibleMaxX="YES" flexibleMaxY="YES"/>
202                 <subviews>
203                     <navigationBar contentMode="scaleToFill" fixedFrame="YES"
204                         translatesAutoresizingMaskIntoConstraintsIntoConstraints="NO" id="7Ih-M0-SOT">
205                         <rect key="frame" x="0.0" y="20" width="320" height="44"/>
206                         <autoresizingMask key="autoresizingMask" widthSizable="YES"
207                             flexibleMaxY="YES"/>
208                         <items>
209                             <navigationItem title="Lisää uusi lähetyks" id="iaA-SK-Xto">
210                                 <barButtonItem key="leftBarButtonItem" systemItem="cancel"
211                                     id="Ido-4s-v6w">
212                                     <connections>
213                                         <action selector="cancelButtonTap:" destination="GVW-cy-vPs"
214                                             id="UII-Zs-wdV"/>
215                                     </connections>
216                                 </barButtonItem>
217                                 <barButtonItem key="rightBarButtonItem" systemItem="save"
218                                     id="WgW-h5-j0h">
219                                     <connections>
220                                         <action selector="saveButtonTap:" destination="GVW-cy-vPs"
221                                             id="47F-LF-esE"/>
222                                     </connections>
223                                 </barButtonItem>
224                             </navigationItem>
225                         </items>
226                     </navigationBar>
227                     <label opaque="NO" clipsSubviews="YES" userInteractionEnabled="NO"
228                         contentMode="left" horizontalHuggingPriority="251"
229                         verticalHuggingPriority="251" fixedFrame="YES" text="Lähetystunnus:"
230                         lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
231                         adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraintsIntoConstraints="NO"
232                         id="r0B-Ft-I9i">
233                         <rect key="frame" x="20" y="80" width="117" height="21"/>
234                         <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
235                             flexibleMaxY="YES"/>
236                         <fontDescription key="fontDescription" type="system" pointSize="17"/>
237                         <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
238                         <nil key="highlightedColor"/>
239                     </label>
240                     <label opaque="NO" clipsSubviews="YES" userInteractionEnabled="NO"
241                         contentMode="left" horizontalHuggingPriority="251"
242                         verticalHuggingPriority="251" fixedFrame="YES" text="Nimi:"
243                         lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
244                         adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraintsIntoConstraints="NO"
245                         id="RKP-iT-hjL">
246                         <rect key="frame" x="20" y="147" width="40" height="21"/>
247                         <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
248                             flexibleMaxY="YES"/>
249                         <fontDescription key="fontDescription" type="system" pointSize="17"/>
250                         <nil key="highlightedColor"/>
251                     </label>
252                     <textField opaque="NO" clipsSubviews="YES" tag="1" contentMode="scaleToFill"
253                         fixedFrame="YES" contentHorizontalAlignment="left"
254                         contentVerticalAlignment="center" borderStyle="roundedRect"
255                         placeholder="Syötä seurantakoodi tähän." minimumFontSize="17"
256                         translatesAutoresizingMaskIntoConstraintsIntoConstraints="NO" id="DcS-Dh-uSN">
257                         <rect key="frame" x="20" y="109" width="280" height="30"/>
258                         <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
259                             flexibleMaxY="YES"/>
260                         <fontDescription key="fontDescription" type="system" pointSize="14"/>
261                         <textInputTraits key="textInputTraits"
262                             autocapitalizationType="allCharacters" autocorrectionType="no"
263                             returnKeyType="next" enablesReturnKeyAutomatically="YES"/>
264                         <connections>
265                             <outlet property="delegate" destination="GVW-cy-vPs" id="Vx0-jl-JCg"/>
266                         </connections>
267                     </textField>
268                     <textField opaque="NO" clipsSubviews="YES" tag="2" contentMode="scaleToFill"
269                         fixedFrame="YES" contentHorizontalAlignment="left"
270                         contentVerticalAlignment="center" borderStyle="roundedRect"
271                         placeholder="Anna lähetykselle nimi." minimumFontSize="17"
272                         translatesAutoresizingMaskIntoConstraintsIntoConstraints="NO" id="l2j-GW-Hfe">
273                         <rect key="frame" x="20" y="176" width="280" height="30"/>
```

```
274         <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
275             flexibleMaxY="YES"/>
276         <fontDescription key="fontDescription" type="system" pointSize="14"/>
277         <textInputTraits key="textInputTraits" autocapitalizationType="sentences"
278             autocorrectionType="no" returnKeyType="done"
279             enablesReturnKeyAutomatically="YES"/>
280         <connections>
281             <outlet property="delegate" destination="GVW-cy-vPs" id="2Cw-zg-lRz"/>
282         </connections>
283     </textField>
284 </subviews>
285 <color key="backgroundColor" white="1" alpha="1" colorSpace="custom"
286     customColorSpace="calibratedWhite"/>
287 </view>
288 <nil key="simulatedTopBarMetrics"/>
289 <nil key="simulatedBottomBarMetrics"/>
290 <simulatedOrientationMetrics key="simulatedOrientationMetrics"/>
291 <connections>
292     <outlet property="tfCode" destination="DcS-Dh-uSN" id="nxc-3f-0Bg"/>
293     <outlet property="tfName" destination="l2j-GW-Hfe" id="cMZ-Qg-j6J"/>
294 </connections>
295 </viewController>
296 <placeholder placeholderIdentifier="IBFirstResponder" id="emF-Pd-0a0"
297     userLabel="First Responder" sceneMemberID="firstResponder"/>
298 </objects>
299 <point key="canvasLocation" x="841" y="-179"/>
300 </scene>
301 </scenes>
302 <simulatedMetricsContainer key="defaultSimulatedMetrics">
303     <simulatedStatusBarMetrics key="statusBar"/>
304     <simulatedOrientationMetrics key="orientation"/>
305     <simulatedScreenMetrics key="destination" type="retina4"/>
306 </simulatedMetricsContainer>
307 </document>
```

```
1 //
2 // PakettivahtiAppDelegate.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import "PakettivahtiAppDelegate.h"
10 #import "PakettivahtiViewController.h"
11
12 @implementation PakettivahtiAppDelegate
13
14 @synthesize managedObjectContext = _managedObjectContext;
15 @synthesize managedObjectModel = _managedObjectModel;
16 @synthesize persistentStoreCoordinator = _persistentStoreCoordinator;
17
18
19
20 #pragma mark - AppDelegate
21
22 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
23 {
24     // Override point for customization after application launch.
25
26     // Asetetaan taustalla tapahtuvien nettihakujen aikaväli.
27     [application setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum];
28
29     return YES;
30 }
31
32 - (void)applicationDidEnterBackground:(UIApplication *)application
33 {
34     // Use this method to release shared resources, save user data, invalidate timers,
35     // and store enough application state information to restore your application to its current state
36     // in case it is terminated later.
37     // If your application supports background execution, this method is called instead of
38     // applicationWillTerminate: when the user quits.
39
40     // Päivitetään sovelluksen ikonin luku, joka kertoo noutoa odottavien lähetysten määrän.
41     [self updateBadge];
42 }
43
44 - (void)applicationWillEnterForeground:(UIApplication *)application
45 {
46     // Called as part of the transition from the background to the inactive state; here you can undo many
47     // of the changes made on entering the background.
48 }
49
50 - (void)applicationDidBecomeActive:(UIApplication *)application
51 {
52     // Restart any tasks that were paused (or not yet started) while the application was inactive.
53     // If the application was previously in the background, optionally refresh the user interface.
54 }
55
56 - (void)applicationWillResignActive:(UIApplication *)application
57 {
58     // Sent when the application is about to move from active to inactive state. This can occur for certain
59     // types of temporary interruptions (such as an incoming phone call or SMS message) or when the user
60     // quits the application and it begins the transition to the background state.
61     // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates.
62     // Games should use this method to pause the game.
63
64     // Tallennetaan lähetystietojen muutokset.
65     [self saveContext];
66 }
67
68 - (void)applicationWillTerminate:(UIApplication *)application
69 {
70     // Called when the application is about to terminate. Save data if appropriate.
71     // See also applicationWillResignActive:.
72
73     // Tallennetaan lähetystietojen muutokset.
74     [self saveContext];
75 }
76
77 - (void)updateBadge
78 {
79     PakettivahtiViewController *pakettivahtiViewController =
80         (PakettivahtiViewController *)self.window.rootViewController;
81     [pakettivahtiViewController updateAppIconBadge];
82 }
83
84
85 #pragma mark - CoreData
86
87 // Returns the managed object context for the application.
88 // If the context doesn't already exist, it is created and bound to the persistent store coordinator
89 // for the application.
90 - (NSManagedObjectContext *)managedObjectContext
91 {
```

```

92     if (_managedObjectContext != nil) {
93         return _managedObjectContext;
94     }
95
96     NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
97     if (coordinator != nil) {
98         _managedObjectContext = [[NSManagedObjectContext alloc] init];
99         [_managedObjectContext setPersistentStoreCoordinator:coordinator];
100     }
101     return _managedObjectContext;
102 }
103
104 // Returns the managed object model for the application.
105 // If the model doesn't already exist, it is created from the application's model.
106 - (NSManagedObjectContext *)managedObjectContext
107 {
108     if (_managedObjectContext != nil) {
109         return _managedObjectContext;
110     }
111     NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"momd" withExtension:@"momd"];
112     _managedObjectContext = [[NSManagedObjectContext alloc] initWithContentsOfURL:modelURL];
113     return _managedObjectContext;
114 }
115
116 // Returns the persistent store coordinator for the application.
117 // If the coordinator doesn't already exist, it is created and the application's store added to it.
118 - (NSPersistentStoreCoordinator *)persistentStoreCoordinator
119 {
120     if (_persistentStoreCoordinator != nil) {
121         return _persistentStoreCoordinator;
122     }
123
124     NSURL *storeURL = [[self applicationDocumentsDirectory] URLByAppendingPathComponent:@"MySQLDataFileName"];
125
126     NSError *error = nil;
127     _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
128                                     initWithManagedObjectContext:[self managedObjectContext]];
129     if (![self _persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
130                                     configuration:nil URL:storeURL options:nil error:&error]) {
131
132         //Replace this implementation with code to handle the error appropriately.
133         NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
134         abort();
135     }
136
137     return _persistentStoreCoordinator;
138 }
139
140 - (void)saveContext
141 {
142     NSError *error = nil;
143     NSManagedObjectContext *managedObjectContext = self.managedObjectContext;
144     if (managedObjectContext != nil) {
145         if ([managedObjectContext hasChanges] && ![managedObjectContext save:&error]) {
146             // Replace this implementation with code to handle the error appropriately.
147             // abort() causes the application to generate a crash log and terminate.
148             // You should not use this function in a shipping application,
149             // although it may be useful during development.
150             NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
151             abort();
152         }
153     }
154 }
155
156
157 #pragma mark - Application's Documents directory
158
159 // Returns the URL to the application's Documents directory.
160 - (NSURL *)applicationDocumentsDirectory
161 {
162     return [[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
163                                     inDomains:NSUserDomainMask] lastObject];
164 }
165
166
167 #pragma mark - Background Fetch Delegate
168
169 // Järjestelmä kutsuu tätä, kun on aika tehdä taustahaku.
170 - (void)application:(UIApplication *)application
171     performFetchWithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler
172 {
173     // Haetaan pääkontrolleri ja kutsutaan sen taustahakumetodia.
174     PakettivahtiViewController *pakettivahtiViewController =
175         (PakettivahtiViewController *)self.window.rootViewController;
176     [pakettivahtiViewController fetchParcelUpdatesWithCompletionHandler:^(UIBackgroundFetchResult result) {
177         completionHandler(result);
178     }];
179 }
180
181 @end

```

```
1 //
2 //  PakettivahtiAppDelegate.h
3 //  Pakettivahti
4 //
5 //  Created by Vesa Mäkeläinen on 2.5.2014.
6 //  Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10 #import <CoreData/CoreData.h>
11
12 #define MyModelURLFile      @"ParcelDataModel"
13 #define MySQLDataFileName  @"ParcelStore.sqlite"
14 #define MyStatusCount      5
15
16 @interface PakettivahtiAppDelegate : UIResponder <UIApplicationDelegate>
17
18 @property (strong, nonatomic) UIWindow *window;
19
20 @property (readonly, strong, nonatomic) NSManagedObjectContext *managedObjectContext;
21 @property (readonly, strong, nonatomic) NSManagedObjectModel *managedObjectModel;
22 @property (readonly, strong, nonatomic) NSPersistentStoreCoordinator *persistentStoreCoordinator;
23
24 - (void)saveContext;
25 - (NSURL *)applicationDocumentsDirectory;
26
27 @end
```

```
1 //
2 //  PakettivahtiViewController.h
3 //  Pakettivahti
4 //
5 //  Created by Vesa Mäkeläinen on 2.5.2014.
6 //  Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10
11 @interface PakettivahtiViewController : UIViewController
12
13 // Tausta-hakuja varten.
14 -(void)fetchParcelUpdatesWithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler;
15
16 // Ikonin laskurin päivitys.
17 - (void)updateAppIconBadge;
18
19 @end
```

```

1 //
2 // PakettivahtiViewController.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import "PakettivahtiViewController.h"
10 #import "ParcelDataSourceDelegateProtocol.h"
11 #import "CDParcel.h"
12 #import "ParcelDataSourceController.h"
13
14
15 @interface PakettivahtiViewController ()
16
17 @property (weak, nonatomic) IBOutlet UITableView *tableView;
18 @property (nonatomic, strong) ParcelDataSourceController *datasource;
19 @property BOOL newUpdatesFound;
20 @property (nonatomic, strong) NSDate *lastUpdateTime;
21
22 @end
23
24 @implementation PakettivahtiViewController
25
26 #pragma mark - UIViewController
27
28 - (void)viewDidLoad
29 {
30     [super viewDidLoad];
31     // Do any additional setup after loading the view, typically from a nib.
32
33     // Alustetaan lähetystietolähde.
34     self.datasource = [[ParcelDataSourceController alloc] init];
35
36     // Nollataan päivitystiedot.
37     self.newUpdatesFound = false;
38     self.lastUpdateTime = [NSDate date];
39
40     // Käynnistetään päivitysten haku netistä.
41     [self fetchParcelUpdatesWithCompletionHandler:nil];
42 }
43
44 - (void)viewWillAppear:(BOOL)animated
45 {
46     // Päivitetään tableView.
47     [self.tableView reloadData];
48
49     // Käynnistetään päivitysten haku netistä.
50     [self fetchParcelUpdatesWithCompletionHandler:nil];
51 }
52
53 - (void)viewWillDisappear:(BOOL)animated
54 {
55 }
56
57
58 - (void)didReceiveMemoryWarning
59 {
60     [super didReceiveMemoryWarning];
61     // Dispose of any resources that can be recreated.
62     NSLog(@"DidReceiveMemoryWarning.");
63 }
64
65
66 #pragma mark - Actions
67
68 - (IBAction)addButtonTap:(UIBarButtonItem *)sender {
69     // Siirrytään lähetyksen lisäsnäkymään.
70     [self performSegueWithIdentifier:@"AddParcel" sender:self];
71 }
72
73
74 - (IBAction)refreshButtonTap:(UIBarButtonItem *)sender {
75     // Kutsutaan päivitysmetodia ilman completionHandleria.
76     [self fetchParcelUpdatesWithCompletionHandler:nil];
77 }
78
79
80 #pragma mark - Update Badge
81
82 - (void)updateAppIconBadge
83 {
84     int count = [self.datasource countParcelsWaitingForPickUp];
85     [UIApplication sharedApplication].applicationIconBadgeNumber = count;
86 }
87
88
89 #pragma mark - Fetch Updates from Web
90
91

```

```

92 - (void)reportFetchCompletionResult:(UIBackgroundFetchResult)result
93     WithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler
94 {
95     if(self.newUpdatesFound)
96     {
97         // Lähetys on päivittynyt. Etsitään noudettavissa olevat lähetykset.
98         int readyParcelCount = [self.datasource countParcelsWaitingForPickUp];
99
100         if(readyParcelCount > 0)
101         {
102             // On valmiina odottavia lähetyksiä. Lähetetään käyttäjälle ilmoitus.
103
104             // Peruutetaan mahdolliset aiemmat.
105             [[UIApplication sharedApplication] cancelAllLocalNotifications];
106             UILocalNotification *notification = [[UILocalNotification alloc] init];
107
108             // Lähetetään ilmoitus heti, eli asetetaan nykyhetki ja muut ilmoituksen tiedot.
109             notification.fireDate = [NSDate date];
110             notification.alertBody = readyParcelCount > 1 ?
111                 @"Lähetykset odottavat noutoa." : @"Yksi lähetyks odottaa noutoa.";
112             notification.soundName = UILocalNotificationDefaultSoundName;
113
114             // Asetetaan sovelluksen ikoniin numeromerkki osoittamaan valmiiden lähetysten lukumäärää.
115             notification.applicationIconBadgeNumber = readyParcelCount;
116             [[UIApplication sharedApplication] scheduleLocalNotification:notification];
117         }
118
119         // Päivitetään sovelluksen ikonin lukumäärää ilmaiseva numero.
120         [UIApplication sharedApplication].applicationIconBadgeNumber = readyParcelCount;
121
122         // Nollataan päivitystieto.
123         self.newUpdatesFound = false;
124
125         // Asetetaan tulos.
126         result = UIBackgroundFetchResultNewData;
127     }
128     else
129     {
130         result = UIBackgroundFetchResultNoData;
131     }
132
133     // Välitetään tulos eteenpäin.
134     completionHandler(result);
135 }
136
137 - (void)fetchParcelUpdatesWithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler
138 {
139     // Päivitetään kaikkien lähetysten tiedot nettipalvelusta.
140
141     // Jos edellisestä päivityksestä on kulunut alle 30 sekuntia, ei päivitetä,
142     // jos pyytö tulee tausta-agentilta.
143     if((fabs([self.lastUpdateTime timeIntervalSinceNow]) < 30) && completionHandler)
144     {
145         // Ei tarvetta päivittää. Lopetaan tällä erää tähän.
146         completionHandler(UIBackgroundFetchResultNoData);
147         return;
148     }
149
150     // Onko päivitettävää?
151     int parcelCount = [self.datasource countParcels];
152     if(parcelCount == 0)
153     {
154         // Ei lähetyksiä, ei tarvetta päivittää.
155         return;
156     }
157
158     // Nollataan aluksi päivitystieto.
159     self.newUpdatesFound = false;
160
161     // Haetaan yhteinen session-olio.
162     NSURLSession *session = [NSURLSession sharedSession];
163
164     // Käsitellään jokainen lähetyks yksi kerrallaan.
165     for(int i = 0; i < parcelCount; i++)
166     {
167         Parcel *parcel = [self.datasource getParcelAtIndex:i];
168
169         // Muodostetaan lähetyksen kysely-URL merkkijonoksi ja URL-olioksi. HUOM! Testipalvelin.
170         NSString *textUrl = [NSString stringWithFormat:@"%d",
171             @"http://pakettivahti.aueta.com/testipalvelu.php?status=", [parcel getStatus]];
172         NSURL *url = [NSURL URLWithString:textUrl];
173
174         // Luodaan http-pyyntö.
175         NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url];
176
177         // Lisätään käsiteltävä lähetysolio pyyntöön,
178         // jotta päivitykset saadaan kohdistettua oikeaan lähetykseen.
179         [NSURLProtocol setProperty:parcel forKey:@"parcel" inRequest:request];
180
181         if(i == (parcelCount - 1))
182 
```



```

183     {
184         // Lisätään viimeisen lähetyksen yhteyteen vastauksen antaminen completionHandlerille.
185         [NSURLProtocol setProperty:completionHandler forKey:@"completionHandler" inRequest:request];
186     }
187
188
189     // Ladataan tulossivu palvelimelta ja käsitellään vastaus uudessa säikeessä completionHandlerilla.
190     NSURLSessionDownloadTask *task = [session downloadTaskWithRequest:request
191                                     completionHandler:^(NSURL *location, NSURLResponse *response, NSError *error)
192     {
193         if(error)
194         {
195             NSLog(@"Lataus epäonnistui: %@", error);
196             return;
197         }
198
199         // Tarkistetaan http-vastauksen statuskoodi (200 = ok).
200         NSInteger status = [(NSHTTPURLResponse *)response statusCode];
201         if(status != 200)
202         {
203             NSLog(@"Statuskoodi ei ole 200. Virhe.");
204             return;
205         }
206
207         // Muuttuiko tieto.
208         BOOL parcelWasUpdated = false;
209
210         // Pyydetty sivu on ladattu laitteelle tilapäiseen tiedostoon.
211         // Luetaan se sieltä merkkijonoksi.
212         NSStringEncoding *encoding = NULL;
213         NSError *parsingError;
214         NSString *htmlString = [NSString stringWithContentsOfURL:location
215                             usedEncoding:encoding error:&parsingError];
216
217         // Haetaan liitetty lähetysoolio, jonka tiedot on ladattu.
218         Parcel *parcel = [NSURLProtocol propertyForKey:@"parcel" inRequest:request];
219
220         // Oletuksena uusi status on tuntematon.
221         // (muttei enää odota tietoja default-statusella.)
222         int newStatus = UNKNOWNstatus;
223
224         // Etsitään merkkijono, joka aloittaa lähetyksen tilatiedot.
225         NSRange eventsRange = [htmlString rangeOfString:@"<div class=\"events\">"];
226
227         if(eventsRange.location != NSNotFound)
228         {
229             // Löytyi. Leikataan turha alkuosa sivusta pois.
230             htmlString = [htmlString substringFromIndex:(eventsRange.location +
231                                                         eventsRange.length)];
232
233             // Etsitään merkkijono "ARK", eli arkistoitu.
234             NSRange statusRange = [htmlString rangeOfString:@"ARK"];
235
236             if(statusRange.location != NSNotFound)
237             {
238                 // Löytyi, joten uusi status on Arkistoitu.
239                 newStatus = ARCHIVEDstatus;
240             }
241             else
242             {
243                 // Jatketaan etsintää etsimällä statusta noudettu eli "LUO".
244                 statusRange = [htmlString rangeOfString:@"LUO"];
245
246                 if(statusRange.location != NSNotFound)
247                 {
248                     newStatus = FINISHEDstatus;
249                 }
250                 else
251                 {
252                     // Yritetään löytää noudettavissa oleva "HYL"-status.
253                     statusRange = [htmlString rangeOfString:@"HYL"];
254
255                     if(statusRange.location != NSNotFound)
256                     {
257                         newStatus = READYstatus;
258                     }
259                     else
260                     {
261                         // Viimeinen toivo: onko edes rekisteröity?
262                         statusRange = [htmlString rangeOfString:@"REK"];
263                         if(statusRange.location != NSNotFound)
264                         {
265                             newStatus = REGISTEREDstatus;
266                         }
267                     }
268                 }
269             }
270
271             // Jos sivulta löytynyt status ei ole mikään näistä, ei lueta aikaleimaa.
272             // Ei lueta aikaa myöskään, jos status ei ole muuttunut.
273             if((statusRange.location != NSNotFound) && ([parcel getStatus] != newStatus))

```

```

274     {
275         // Päivittynyt status on löytynyt, luetaan sen aikaleima.
276
277         // Siirretään statustiedon löytöpaikka <div class="***"> -tagin
278         // ja sitä seuraavan <p>-tagin loppuun.
279         statusRange.location = statusRange.location + [@"***"><p>" length];
280
281         // Leikataan merkkijono siitä loppuun.
282         NSString *cutHtmlString = [htmlString substringFromIndex:statusRange.location];
283
284         // Aikaleima alkaa leikatun merkkijonon alusta.
285         NSRange timestampRange;
286         timestampRange.location = 0;
287
288         // Etsitään aikaleiman lopputagi.
289         NSRange timestampEndRange = [cutHtmlString rangeOfString:@"</p>"];
290
291         if(timestampEndRange.location == NSNotFound)
292         {
293             NSLog(@"Aikaleimaa ei löytynyt.");
294         }
295         else
296         {
297             // Asetetaan aikaleiman pituus.
298             timestampRange.length = timestampEndRange.location - timestampRange.location;
299
300             // Leikataan aikaleima.
301             NSString *timestampString = [cutHtmlString substringWithRange:timestampRange];
302
303             //NSLog(@"Löytynyt aikaleima: %@", timestampString);
304
305             // Annetaan aikaleiman muoto
306             // ja luetaan aika sen perusteella merkkijonosta NSDate-olioksi.
307             NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
308
309             // Pitää käyttää amerikkalaista lokalisaatiota,
310             // jotta aikaleima tulkitaan oikein.
311             [dateFormatter setLocale:[NSLocale alloc]
312                                     initWithLocaleIdentifier:@"en-US"]];
313
314             // Aikaleima on muodossa "12 March 2014 09:34", asetetaan muotoilu.
315             [dateFormatter setDateFormat:@"dd MMMM yyyy HH:m"];
316
317             NSDate *dateFromTimestamp = [dateFormatter dateFromString:timestampString];
318
319             // Päivitetään seurattavan lähetyksen aikaleima.
320             [parcel setUpdateTime:dateFromTimestamp];
321
322             // Asetetaan tieto tilapäivityksestä.
323             parcelWasUpdated = true;
324         }
325     }
326 }
327
328 else
329 {
330     // Sivulta ei löytynyt statustietoja.
331     // Päivitetään lähetykselle aikaleimaksi tarkistusaika.
332     [parcel setUpdateTime:[NSDate date]];
333 }
334
335 // Päivitetään lähetyksen status (joko palvelusta löytynyt tai "ei tietoja").
336 [parcel setStatus:newStatus];
337
338 // Oliko viimeinen lähetysolio?
339 BOOL isLast = false;
340
341 if([NSURLProtocol propertyForKey:@"completionHandler" inRequest:request] != nil)
342 {
343     // On viimeinen. Vain viimeiselle on annettu completion handler.
344     isLast = true;
345 }
346
347 // Onko noudettu?
348 BOOL canBeRemoved = false;
349
350 if([parcel getStatus] >= FINISHEDstatus)
351 {
352     // Lähetyks on noudettu ja voidaan poistaa.
353     canBeRemoved = true;
354 }
355
356 // Päivitetään toiminta lähettämällä pääsäikeelle komentoja suoritettavaksi.
357 dispatch_async(dispatch_get_main_queue(), ^{
358     // Suoritetaan pääsäikeen komentojonossa.
359
360     if(canBeRemoved)
361     {
362         // Poistetaan lähetysolio.
363         [self.datasource deleteParcel:parcel];
364     }
365 }

```

```

365         else
366         {
367             // Asetetaan tieto uusista päivityksistä.
368             if(parcelWasUpdated)
369             {
370                 self.newUpdatesFound = true;
371             }
372
373             // Asetetaan päivitysaika.
374             self.lastUpdateTime = [NSDate date];
375
376             // Jos oli viimeinen haettava lähetys,
377             // lähetetään valmistumistieto ja päivitetään päivitysaika.
378             if(isLast)
379             {
380                 [self reportFetchCompletionResult:UIBackgroundFetchResultNewData
381                    WithCompletionHandler:completionHandler];
382             }
383         }
384
385         // Lajitellaan ja päivitetään luettelo.
386         [self.datasource sort];
387         [self.tableView reloadData];
388     });
389
390     // Käynnistetään taustasäikeen tehtävä.
391     [task resume];
392 }
393
394 }
395
396 #pragma mark - Navigation
397
398 // In a storyboard-based application, you will often want to do a little preparation before navigation
399 - (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
400 {
401     if ([[segue identifier] isEqualToString:@"AddParcel"])
402     {
403         // Haetaan kohteena oleva kontrolleri ja asetetaan sille (lisäysnäkyhälle) delegaatti.
404         [[segue destinationViewController] setDelegate:self.datasource];
405     }
406     else if ([[segue identifier] isEqualToString:@"ShowCode"])
407     {
408         // Siirrytään seurantakoodin näyttöön.
409
410         // Poimitaan valittu rivi ja sen lähetys.
411         NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
412
413         Parcel *selectedParcel = [self.datasource getParcelAtIndex:indexPath row];
414
415         // Välitetään koodi kohteena olevalle kontrollerille.
416         [[segue destinationViewController] setCode:[selectedParcel getCode]];
417     }
418 }
419
420
421 #pragma mark - UITableViewDataSourceDelegate Methods
422
423 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
424 {
425     // Pakettivahti näyttää luettelossa vain yhden osion.
426     return 1;
427 }
428
429 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
430 {
431     // Osion rivien määrä on taulukossa olevien lähetysten määrä.
432     return [self.datasource countParcels];
433 }
434
435 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
436 {
437     // Luodaan uusi TableViewCell annetulle mallille.
438     static NSString *CellIdentifier = @"Cell";
439     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier
440                             forIndexPath:indexPath];
441
442     // Haetaan solussa näytettävä lähetys.
443     NSIndexPath *indexPath = [indexPath row];
444     Parcel *parcel = [self.datasource getParcelAtIndex:indexPath];
445
446     if(parcel)
447     {
448         // Löytyi haluttu lähetys. Asetetaan solun tiedot siitä.
449         cell.textLabel.text = [parcel getName];
450
451         NSDate *date = [parcel getUpdateTime];
452
453         // Muotoillaan päiväys ja aika.
454         NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
455     }

```

```
456 [dateFormatter setLocale:[NSLocale currentLocale]];
457 [dateFormatter setDateFormat:@"%dd.MM.yyyy HH.mm"];
458
459 NSString *formattedDate = [dateFormatter stringFromDate:date];
460
461 // Asetetaan solun tietoriville aikaleima ja status.
462 cell.detailTextLabel.text = [NSString stringWithFormat:@"%@" - %@",
463                             formattedDate, [Parcel getNameForStatus:[parcel getStatus]]];
464
465 // Jos lähetys on noudettavissa, korostetaan solun tausta vihreällä.
466 if ([parcel getStatus] == READYstatus)
467 {
468     // On noudettavissa, vaihdetaan taustaväri vihertäväksi.
469     [cell setBackgroundColor:[UIColor colorWithRed:0.5 green:0.9 blue:0.5 alpha:1.0]];
470 }
471 else
472 {
473     // Ei ole noudettavissa, taustaväri on valkoinen.
474     [cell setBackgroundColor:[UIColor colorWithRed:1.0 green:1.0 blue:1.0 alpha:1.0]];
475 }
476 }
477
478 // Palautetaan täytetty solu.
479 return cell;
480 }
481
482 @end
```

```
1 //
2 // Parcel.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 3.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import "Parcel.h"
10
11 @interface Parcel()
12
13 @property (nonatomic, strong) CDParcel *cdParcel;
14
15 @end
16
17
18 @implementation Parcel
19
20 #pragma mark - Status Names Declaration
21
22 static NSString * const statusNames[] = { @"----- Odotetaan tietoja -----",
23                                           @"Ei tietoja.",
24                                           @"Rekisteröity.",
25                                           @"Noudettavissa.",
26                                           @"Noudettu.",
27                                           @"Arkistoitu." };
28
29
30 #pragma mark - Class methods
31
32 + (NSString *)getNameForStatus:(ParcelStatus)status
33 {
34     return statusNames[status];
35 }
36
37 + (NSNumber *)getValueForStatus:(ParcelStatus)status
38 {
39     return [NSNumber numberWithInt:status];
40 }
41
42 #pragma mark - Instance methods
43
44 - (Parcel *)initWithCDParcel:(CDParcel *)cdParcel
45 {
46     self = [super init];
47     if (self) {
48         _cdParcel = cdParcel;
49     }
50     return self;
51 }
52
53 - (CDParcel *)getCDParcel
54 {
55     return self.cdParcel;
56 }
57
58 - (ParcelStatus)getStatus
59 {
60     return [_cdParcel.status intValue];
61 }
62
63 - (void)setStatus:(ParcelStatus)status
64 {
65     if(status >= DEFAULTstatus && status <= ARCHIVEDstatus)
66     {
67         // Voidaan asettaa turvallisesti.
68         [_cdParcel setStatus:[NSNumber numberWithInt:status]];
69     }
70 }
71
72 - (NSString *)getCode
73 {
74     return _cdParcel.code;
75 }
76
77 - (void)setCode:(NSString *)code
78 {
79     if (![code isEqualToString:@""])
80     {
81         [_cdParcel setCode:code];
82     }
83 }
84
85 - (NSString *)getName
86 {
87     return _cdParcel.name;
88 }
89
90 - (void)setName:(NSString *)name
91 {
```

```
92     if(![name isEqualToString:@""])
93     {
94         [_cdParcel setName:name];
95     }
96 }
97
98 - (NSDate *)getUpdateTime
99 {
100     return _cdParcel.updateTime;
101 }
102
103 - (void)setUpdateTime:(NSDate *)date
104 {
105     [_cdParcel setUpdateTime:date];
106 }
107
108
109 @end
```

```
1 //
2 // Parcel.h
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 3.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import "CDParcel.h"
11
12 @interface Parcel : NSObject
13
14 typedef NS_ENUM(int, ParcelStatus)
15 {
16     DEFAULTstatus = 0, // = 0
17     UNKNOWNstatus, // = 1, etc.
18     REGISTEREDstatus,
19     READYstatus,
20     FINISHEDstatus,
21     ARCHIVEDstatus // = 5
22 };
23
24 + (NSString *)getNameForStatus:(ParcelStatus)status;
25 + (NSNumber *)getValueForStatus:(ParcelStatus)status;
26
27 - (Parcel *)initWithCDParcel:(CDParcel *)cdParcel;
28 - (CDParcel *)getCDParcel;
29 - (ParcelStatus)getStatus;
30 - (void)setStatus:(ParcelStatus)status;
31 - (NSString *)getCode;
32 - (void)setCode:(NSString *)code;
33 - (NSString *)getName;
34 - (void)setName:(NSString *)name;
35 - (NSDate *)getUpdateTime;
36 - (void)setUpdateTime:(NSDate *)date;
37
38 @end
```

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <model userDefinedModelVersionIdentifier="" type="com.apple.IDECoreDataModeler.DataModel" documentVersion="1.0" last
3   <entity name="CDParcel" representedClassName="CDParcel" syncable="YES">
4     <attribute name="code" attributeType="String" syncable="YES"/>
5     <attribute name="name" attributeType="String" syncable="YES"/>
6     <attribute name="status" optional="YES" attributeType="Integer 16" defaultValueString="0" syncable="YES"/>
7     <attribute name="updateTime" attributeType="Date" syncable="YES"/>
8   </entity>
9   <elements>
10     <element name="CDParcel" positionX="-416" positionY="-45" width="128" height="103"/>
11   </elements>
12 </model>
```

```
1 //
2 // ParcelDataSourceController.h
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 4.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import "ParcelDataSourceDelegateProtocol.h"
11
12 @interface ParcelDataSourceController : NSObject <ParcelDataSourceDelegateProtocol>
13
14 - (NSUInteger)countParcels;
15 - (Parcel *)getParcelAtIndex:(NSUInteger)index;
16 - (NSUInteger)countParcelsWaitingForPickUp;
17 - (void) sort;
18 - (void) deleteParcel:(Parcel *)parcel;
19
20 @end
```

```
1 //
2 // ParcelDataSourceController.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 4.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import "ParcelDataSourceController.h"
10 #import "PakettivahtiAppDelegate.h"
11 #import "ParcelDataSourceDelegateProtocol.h"
12 #import "Parcel.h"
13 #import "CDParcel.h"
14
15 @interface ParcelDataSourceController() <ParcelDataSourceDelegateProtocol>
16
17
18 @property (nonatomic, strong)NSMutableArray *parcels;
19 @property (nonatomic, strong)NSManagedObjectContext *managedObjectContext;
20
21 @end
22
23
24 @implementation ParcelDataSourceController
25
26 #pragma mark - ParcelDataSource methods
27
28 - (NSMutableArray *)parcels
29 {
30     if(!_parcels)
31     {
32         _parcels = [NSMutableArray alloc] init];
33     }
34
35     return _parcels;
36 }
37
38 - (instancetype)init
39 {
40     self = [super init];
41     if (self) {
42         // Haetaan AppDelegate käyttöön.
43         PakettivahtiAppDelegate *appDelegate = [[UIApplication sharedApplication] delegate];
44
45         // Ja sieltä Core Datan hallinnoimat oliot.
46         _managedObjectContext = appDelegate.managedObjectContext;
47
48         // Haetaan luettelo Core Datan tallentamista lähetyksistä.
49         NSError *error = nil;
50         NSFetchRequest *fetchRequest = [NSFetchRequest fetchRequestWithEntityName:@"CDParcel"];
51
52         // Alustetaan ja täytetään taulukko lähetyksillä.
53         NSArray *tempParcels = [NSArray alloc] initWithArray:[_managedObjectContext
54                                                                 executeFetchRequest:fetchRequest error:&error]];
55
56         if(error != nil)
57         {
58             NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
59             abort();
60         }
61
62         // Kopioidaan CDParcel-oliot Parcel-olioiksi lopulliseen käyttötaulukkoon.
63         for (CDParcel *cdParcel in tempParcels) {
64             Parcel *parcel = [[Parcel alloc] initWithCDParcel:cdParcel];
65             [self.parcels addObject:parcel];
66         }
67
68         // Järjestetään taulukko.
69         [self sort];
70     }
71
72     return self;
73 }
74
75 - (void)sort
76 {
77     // Järjestetään taulukko statuksen perusteella laskevaan järjestykseen
78     // ja toissijaisesti aikaleiman perusteella laskevaan.
79     [self.parcels sortUsingComparator:^(NSComparisonResult(id obj1, id obj2) {
80         int status1 = [(Parcel *)obj1 getStatus];
81         int status2 = [(Parcel *)obj2 getStatus];
82
83         if(status1 == status2)
84         {
85             // Sama status kummallakin, pitää verrata vielä aikaleimaa.
86             NSDate *date1 = [(Parcel *)obj1 getUpdateTime];
87             NSDate *date2 = [(Parcel *)obj2 getUpdateTime];
88
89             // Palautetaan aikajärjestyksessä, ei käänteisessä.
90             return ([date1 compare:date2]) * 1;
91         }
92     })];
93 }
```



```
92         else
93         {
94             // Pelkkä statusvertailu riittää. Palautetaan -1 jos status1 on suurempi, muutoin 1.
95             return status1 > status2 ? -1 : 1;
96         }
97     }];
98 }
99
100 -(NSInteger)countParcels
101 {
102     return self.parcels.count;
103 }
104
105 -(Parcel *)getParcelAtIndex:(NSInteger)index
106 {
107     if(index >= [self.parcels count])
108     {
109         // Indeksä on liian suuri, taulukossa ei ole niin monta lähetystä.
110         // Jotain meni pieleen. Palautetaan tyhjää.
111         NSLog(@"Indeksi liian suuri.");
112         return nil;
113     }
114
115     return [self.parcels objectAtIndex:index];
116 }
117
118 -(NSInteger)countParcelsWaitingForPickUp
119 {
120     // Kootaan valmiit lähetykset muokattavaan taulukkoon.
121     NSMutableArray *tempParcels = [NSMutableArray alloc] init];
122
123     for (Parcel *parcel in self.parcels) {
124         if([parcel getStatus] == READYstatus)
125         {
126             [tempParcels addObject:parcel];
127         }
128     }
129
130     // Palautetaan taulukko, jota ei voi muokata.
131     return [tempParcels count];
132 }
133
134 -(void) deleteParcel:(Parcel *)parcel
135 {
136     // Poistetaan annettu lähetys Core Datan hallitusta contextista.
137     [self.managedObjectContext deleteObject:[parcel getCParcel]];
138
139     NSError *error = nil;
140
141     // Tallennetaan muuttunut tilanne ja tarkistetaan onnistuminen.
142     if(![self.managedObjectContext save:&error])
143     {
144         NSLog(@"Poisto epäonnistui: %@", [error userInfo]);
145     }
146
147     // Poistetaan myös tyhjä kääreolio.
148     [self.parcels removeObject:parcel];
149 }
150
151
152 #pragma mark - ParcelDataSourceDelegate Methods
153
154 -(Parcel *) getNewParcel
155 {
156     // Muodostetaan uusi hallinnoitu CDparcel-olio ja kääritään se Parcel-olioksi.
157     CDParcel *addedCDParcel = [NSEntityDescription insertNewObjectForEntityForName:@"CDParcel"
158                                     inManagedObjectContext:self.managedObjectContext];
159
160     Parcel *addedParcel = [[Parcel alloc] initWithCDParcel:addedCDParcel];
161
162     // Asetetaan oletusstatus ja päivitysaika nykyhetkeen.
163     [addedParcel setStatus:DEFAULTstatus];
164     [addedParcel setUpdateTime:[NSDate date]];
165
166     // Lisätään uusi lähetys luetteloon ja järjestetään luettelo uudelleen.
167     [self.parcels addObject:addedParcel];
168     [self sort];
169
170     // Palautetaan luotu, kääritty ja alustettu sekä luetteloon lisätty hallinnoitu lähetysolio.
171     return addedParcel;
172 }
173
174 @end
```

```
1 //
2 // ParcelDataSourceDelegateProtocol.h
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import "Parcel.h"
11
12 @protocol ParcelDataSourceDelegateProtocol <NSObject>
13
14 // Palauttaa ViewControllerille lähetysolion.
15 - (Parcel *) getNewParcel;
16
17 @end
```

```
1 //
2 // TrackingCodeViewController.h
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10
11 @interface TrackingCodeViewController : UIViewController
12
13 @property (nonatomic, strong) NSString *code;
14
15 @end
```

```
1 //
2 // TrackingCodeViewController.m
3 // Pakettivahti
4 //
5 // Created by Vesa Mäkeläinen on 2.5.2014.
6 // Copyright (c) 2014 Vesa Mäkeläinen. All rights reserved.
7 //
8
9 #import "TrackingCodeViewController.h"
10
11 @interface TrackingCodeViewController ()
12
13 @property (weak, nonatomic) IBOutlet UILabel *lblCode;
14 @property (weak, nonatomic) IBOutlet UIButton *btReturn;
15
16 @end
17
18 @implementation TrackingCodeViewController
19
20 #pragma mark - UIViewController
21
22 - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
23 {
24     self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
25     if (self) {
26         // Custom initialization
27     }
28     return self;
29 }
30
31 - (void)viewDidLoad
32 {
33     [super viewDidLoad];
34     // Do any additional setup after loading the view.
35
36     // Poistetaan buttonin teksti näkyviltä. Se auttaa vain interface builderissä...
37     [self.btReturn setTitle:@"" forState:UIControlStateNormal];
38
39     // Lisätään lähetyksen seurantakoodi labeliin.
40     self.lblCode.text = self.code;
41 }
42
43 - (void)didReceiveMemoryWarning
44 {
45     [super didReceiveMemoryWarning];
46     // Dispose of any resources that can be recreated.
47     NSLog(@"DidReceiveMemoryWarning.");
48 }
49
50 #pragma mark - Actions
51
52 - (IBAction)returnButtonDoubleTap:(UIButton *)sender {
53     // Suljetaan näyttö.
54     [self dismissViewControllerAnimated:YES completion:nil];
55 }
56
57 @end
```

```
1 using System;
2 using System.Diagnostics;
3 using System.Resources;
4 using System.Windows;
5 using System.Windows.Markup;
6 using System.Windows.Navigation;
7 using Microsoft.Phone.Controls;
8 using Microsoft.Phone.Shell;
9 using Pakettivahti.Resources;
10 using Pakettivahti.Core.DAO;
11 using Pakettivahti.ViewModel;
12 using Microsoft.Phone.Scheduler; // ParcelDataContext
13
14 namespace Pakettivahti
15 {
16     public partial class App : Application
17     {
18         /// <summary>
19         /// Provides easy access to the root frame of the Phone Application.
20         /// </summary>
21         /// <returns>The root frame of the Phone Application.</returns>
22         public static PhoneApplicationFrame RootFrame { get; private set; }
23
24         // Koko sovelluksen käyttöön tarkoitettu ViewModel.
25         private static ParcelViewModel viewModel;
26         public static ParcelViewModel ViewModel
27         {
28             get { return viewModel; }
29         }
30
31         /// <summary>
32         /// Constructor for the Application object.
33         /// </summary>
34         public App()
35         {
36             // Global handler for uncaught exceptions.
37             UnhandledException += Application_UnhandledException;
38
39             // Standard XAML initialization
40             InitializeComponent();
41
42             // Phone-specific initialization
43             InitializePhoneApplication();
44
45             // Language display initialization
46             InitializeLanguage();
47
48             // Show graphics profiling information while debugging.
49             if (Debugger.IsAttached)
50             {
51                 // Display the current frame rate counters.
52                 Application.Current.Host.Settings.EnableFrameRateCounter = true;
53
54                 // Prevent the screen from turning off while under the debugger by disabling
55                 // the application's idle detection.
56                 // Caution:- Use this under debug mode only.
57                 // Application that disables user idle detection will continue to run
58                 // and consume battery power when the user is not using the phone.
59                 PhoneApplicationService.Current.UserIdleDetectionMode = IdleDetectionMode.Disabled;
60             }
61
62             // Paikallisen tietokantayhteyden connection string.
63             string connectionString = @"Data Source=isostore:/Parcels.sdf";
64
65             // Luodaan tietokanta, jos sellaista ei vielä ole.
66             using (var db = new ParcelDataContext(connectionString))
67             {
68                 if (!db.DatabaseExists())
69                 {
70                     // Ei löytynyt, joten luodaan uusi tietokanta.
71                     db.CreateDatabase();
72                 }
73             }
74
75             // Lisätään ViewModel-olio.
76             viewModel = new ParcelViewModel(connectionString);
77
78             // Ladataan tietokannasta lähetysten tiedot.
79             viewModel.LoadFromDatabase();
80
81         }
82
83         // Code to execute when the application is launching (eg, from Start)
84         // This code will not execute when the application is reactivated
85         private void Application_Launching(object sender, LaunchingEventArgs e)
86         {
87             PeriodicTask updateTask;
88
89             // Etsitään aiemmin ajastettu taustatehtävä.
90             updateTask = ScheduledActionService.Find("NotificationUpdateTask") as PeriodicTask;
91         }
```

```

92
93         // Jos aiempi ajastus löytyi, poistetaan se.
94         if (updateTask != null)
95         {
96             if (!updateTask.IsEnabled)
97             {
98                 System.Diagnostics.Debug.WriteLine("Käyttäjä estänyt taustatoiminnon!");
99             }
100             ScheduledActionService.Remove(updateTask.Name);
101         }
102
103         // Yritetään ajastaa tehtävä (mahdollisesti uudelleen, jos aiempi löytyi).
104         try
105         {
106             updateTask = new PeriodicTask("NotificationUpdateTask");
107             // Näkyy käyttäjälle asetusten taustatehtävät-näkymässä Pakettivahti-sovelluksen kohdalla.
108             updateTask.Description = "Päivittää sovelluksen Tileen tiedon saapuneista lähetyksistä.";
109             ScheduledActionService.Add(updateTask);
110             //System.Diagnostics.Debug.WriteLine("Lisätty taustatehtävä.");
111         }
112         catch (InvalidOperationException)
113         {
114             updateTask = null;
115             System.Diagnostics.Debug.WriteLine("Tehtävän lisäys epäonnistui.");
116         }
117
118         // Asetetaan testiajastus 61 sekuntiin. Tämä tosin toimii vain laitteella,
119         // ei näemmä emulaattorissa. MS sanoo, että vähintään 60 sekuntia, muuten ei ehkä käynnisty.
120
121         if (updateTask != null)
122         {
123             // Ajastetaan testausta varten suoritus jo 61 sekunnin päähän.
124             // HUOM! Tämä ajastus koskee vain seuraavaa suorituskertaa, joten pitää uusia joka kerran.
125             ScheduledActionService.LaunchForTest(updateTask.Name, TimeSpan.FromSeconds(61));
126         }
127     }
128
129     // Code to execute when the application is activated (brought to foreground)
130     // This code will not execute when the application is first launched
131     private void Application_Activated(object sender, ActivatedEventArgs e)
132     {
133     }
134
135     // Code to execute when the application is deactivated (sent to background)
136     // This code will not execute when the application is closing
137     private void Application_Deactivated(object sender, DeactivatedEventArgs e)
138     {
139     }
140
141     // Code to execute when the application is closing (eg, user hit Back)
142     // This code will not execute when the application is deactivated
143     private void Application_Closing(object sender, ClosingEventArgs e)
144     {
145     }
146
147     // Code to execute if a navigation fails
148     private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs e)
149     {
150         if (Debugger.IsAttached)
151         {
152             // A navigation has failed; break into the debugger
153             Debugger.Break();
154         }
155     }
156
157     // Code to execute on Unhandled Exceptions
158     private void Application_UnhandledException(object sender, ApplicationUnhandledExceptionEventArgs e)
159     {
160         if (Debugger.IsAttached)
161         {
162             // An unhandled exception has occurred; break into the debugger
163             Debugger.Break();
164         }
165     }
166
167     #region Phone application initialization
168
169     // Avoid double-initialization
170     private bool phoneApplicationInitialized = false;
171
172     // Do not add any additional code to this method
173     private void InitializePhoneApplication()
174     {
175         if (phoneApplicationInitialized)
176             return;
177
178         // Create the frame but don't set it as RootVisual yet; this allows the splash
179         // screen to remain active until the application is ready to render.
180         RootFrame = new PhoneApplicationFrame();
181         RootFrame.Navigated += CompleteInitializePhoneApplication;
182     }

```

```

183
184         // Handle navigation failures
185         RootFrame.NavigationFailed += RootFrame_NavigationFailed;
186
187         // Handle reset requests for clearing the backstack
188         RootFrame.Navigated += CheckForResetNavigation;
189
190         // Ensure we don't initialize again
191         phoneApplicationInitialized = true;
192     }
193
194     // Do not add any additional code to this method
195     private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs e)
196     {
197         // Set the root visual to allow the application to render
198         if (RootVisual != RootFrame)
199             RootVisual = RootFrame;
200
201         // Remove this handler since it is no longer needed
202         RootFrame.Navigated -= CompleteInitializePhoneApplication;
203     }
204
205     private void CheckForResetNavigation(object sender, NavigationEventArgs e)
206     {
207         // If the app has received a 'reset' navigation, then we need to check
208         // on the next navigation to see if the page stack should be reset
209         if (e.NavigationMode == NavigationMode.Reset)
210             RootFrame.Navigated += ClearBackStackAfterReset;
211     }
212
213     private void ClearBackStackAfterReset(object sender, NavigationEventArgs e)
214     {
215         // Unregister the event so it doesn't get called again
216         RootFrame.Navigated -= ClearBackStackAfterReset;
217
218         // Only clear the stack for 'new' (forward) and 'refresh' navigations
219         if (e.NavigationMode != NavigationMode.New && e.NavigationMode != NavigationMode.Refresh)
220             return;
221
222         // For UI consistency, clear the entire page stack
223         while (RootFrame.RemoveBackEntry() != null)
224         {
225             ; // do nothing
226         }
227     }
228
229     #endregion
230
231     // Initialize the app's font and flow direction as defined in its localized resource strings.
232     private void InitializeLanguage()
233     {
234         try
235         {
236             // Set the font to match the display language defined by the
237             // ResourceLanguage resource string for each supported language.
238             //
239             // Fall back to the font of the neutral language if the Display
240             // language of the phone is not supported.
241             //
242             // If a compiler error is hit then ResourceLanguage is missing from
243             // the resource file.
244             RootFrame.Language = XmlLanguage.GetLanguage(AppResources.ResourceLanguage);
245
246             // Set the FlowDirection of all elements under the root frame based
247             // on the ResourceFlowDirection resource string for each
248             // supported language.
249             //
250             // If a compiler error is hit then ResourceFlowDirection is missing from
251             // the resource file.
252             FlowDirection flow = (FlowDirection)Enum.Parse(typeof(FlowDirection),
253                 AppResources.ResourceFlowDirection);
254             RootFrame.FlowDirection = flow;
255         }
256         catch
257         {
258             // If an exception is caught here it is most likely due to either
259             // ResourceLangauge not being correctly set to a supported language
260             // code or ResourceFlowDirection is set to a value other than LeftToRight
261             // or RightToLeft.
262
263             if (Debugger.IsAttached)
264             {
265                 Debugger.Break();
266             }
267
268             throw;
269         }
270     }
271 }
272

```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <root>
3   <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4     xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
5     <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
6     <xsd:element name="root" msdata:IsDataSet="true">
7       <xsd:complexType>
8         <xsd:choice maxOccurs="unbounded">
9           <xsd:element name="metadata">
10             <xsd:complexType>
11               <xsd:sequence>
12                 <xsd:element name="value" type="xsd:string" minOccurs="0" />
13               </xsd:sequence>
14               <xsd:attribute name="name" use="required" type="xsd:string" />
15               <xsd:attribute name="type" type="xsd:string" />
16               <xsd:attribute name="mimetype" type="xsd:string" />
17               <xsd:attribute ref="xml:space" />
18             </xsd:complexType>
19           </xsd:element>
20           <xsd:element name="assembly">
21             <xsd:complexType>
22               <xsd:attribute name="alias" type="xsd:string" />
23               <xsd:attribute name="name" type="xsd:string" />
24             </xsd:complexType>
25           </xsd:element>
26           <xsd:element name="data">
27             <xsd:complexType>
28               <xsd:sequence>
29                 <xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1" />
30                 <xsd:element name="comment" type="xsd:string" minOccurs="0" msdata:Ordinal="2" />
31               </xsd:sequence>
32               <xsd:attribute name="name" type="xsd:string" use="required" msdata:Ordinal="1" />
33               <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
34               <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
35               <xsd:attribute ref="xml:space" />
36             </xsd:complexType>
37           </xsd:element>
38           <xsd:element name="resheader">
39             <xsd:complexType>
40               <xsd:sequence>
41                 <xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1" />
42               </xsd:sequence>
43               <xsd:attribute name="name" type="xsd:string" use="required" />
44             </xsd:complexType>
45           </xsd:element>
46         </xsd:choice>
47       </xsd:complexType>
48     </xsd:element>
49   </xsd:schema>
50   <resheader name="resmimetype">
51     <value>text/microsoft-resx</value>
52   </resheader>
53   <resheader name="version">
54     <value>2.0</value>
55   </resheader>
56   <resheader name="reader">
57     <value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=4.0.0.0, Culture=neutral,
58       PublicKeyToken=b77a5c561934e089</value>
59   </resheader>
60   <resheader name="writer">
61     <value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=4.0.0.0, Culture=neutral,
62       PublicKeyToken=b77a5c561934e089</value>
63   </resheader>
64   <data name="ResourceFlowDirection" xml:space="preserve">
65     <value>LeftToRight</value>
66     <comment>Controls the FlowDirection for all elements in the RootFrame.
67       Set to the traditional direction of this resource file's language</comment>
68   </data>
69   <data name="ResourceLanguage" xml:space="preserve">
70     <value>fi-FI</value>
71     <comment>Controls the Language and ensures that the font for all elements in the RootFrame
72       aligns with the app's language. Set to the language code of this resource file's language.</comment>
73   </data>
74   <data name="ApplicationTitle" xml:space="preserve">
75     <value>PAKETTIVAHTI</value>
76   </data>
77   <data name="AppBarButtonText" xml:space="preserve">
78     <value>add</value>
79   </data>
80   <data name="AppBarMenuItemText" xml:space="preserve">
81     <value>Menu Item</value>
82   </data>
83   <data name="Add" xml:space="preserve">
84     <value>Lisää</value>
85   </data>
86   <data name="FirstPage" xml:space="preserve">
87     <value>Lähetys</value>
88   </data>
89   <data name="NewAddButton" xml:space="preserve">
90     <value>Tallenna</value>
91   </data>
```

```
92 <data name="NewCode" xml:space="preserve">
93   <value>Tunnus:</value>
94 </data>
95 <data name="NewCodeHint" xml:space="preserve">
96   <value>Anna lähetyksen seurantatunnus.</value>
97 </data>
98 <data name="NewName" xml:space="preserve">
99   <value>Nimi:</value>
100 </data>
101 <data name="NewNameHint" xml:space="preserve">
102   <value>Anna lähetykselle kuvaava nimi.</value>
103 </data>
104 <data name="NewPage" xml:space="preserve">
105   <value>Uusi lähety</value>
106 </data>
107 <data name="ShowCodePage" xml:space="preserve">
108   <value>Lähetystunnus:</value>
109 </data>
110 </root>
```



```
1 <phone:PhoneApplicationPage
2   x:Class="Pakettivahti.MainPage"
3   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5   xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
6   xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
7   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
8   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
9   xmlns:local="clr-namespace:Pakettivahti.ViewModel"
10  mc:Ignorable="d"
11  FontFamily="{StaticResource PhoneFontFamilyNormal}"
12  FontSize="{StaticResource PhoneFontSizeNormal}"
13  Foreground="{StaticResource PhoneForegroundBrush}"
14  SupportedOrientations="Portrait" Orientation="Landscape"
15  shell:SystemTray.IsVisible="True">
16
17  <!-- ApplicationBar -menu sivun alareunassa. Lisätään valinnat. -->
18  <phone:PhoneApplicationPage.ApplicationBar>
19    <shell:ApplicationBar>
20      <shell:ApplicationBarIconButton Text="Lisää uusi" IconUri="/Assets/AppBar/new.png"
21        Click="newButton_Click" />
22      <shell:ApplicationBar.MenuItems>
23        <shell:ApplicationBarMenuItem x:Name="UpdateMenuItem" Text="Päivitä tiedot"
24          Click="UpdateMenuItem_Click" />
25      </shell:ApplicationBar.MenuItems>
26    </shell:ApplicationBar>
27  </phone:PhoneApplicationPage.ApplicationBar>
28
29  <!--LayoutRoot is the root grid where all page content is placed-->
30  <Grid x:Name="LayoutRoot" Background="Transparent">
31    <Grid.RowDefinitions>
32      <RowDefinition Height="Auto"/>
33      <RowDefinition Height="*" />
34    </Grid.RowDefinitions>
35
36    <Grid.Resources>
37      <local:StatusColorConverter x:Key="StatusColorConverter" /></local:StatusColorConverter>
38    </Grid.Resources>
39
40    <!--TitlePanel contains the name of the application and page title-->
41    <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
42      <TextBlock Text="{Binding Path=LocalizedResources.ApplicationTitle,
43        Source={StaticResource LocalizedStrings}}" Style="{StaticResource PhoneTextNormalStyle}"
44        Margin="12,0"/>
45      <TextBlock Text="{Binding Path=LocalizedResources.FirstPage,
46        Source={StaticResource LocalizedStrings}}" Margin="9,-7,0,0"
47        Style="{StaticResource PhoneTextTitle1Style}" />
48    </StackPanel>
49
50    <!--ContentPanel - place additional content here-->
51    <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
52      <phone:LongListSelector x:Name="longlistselectorParcels" Tap="longlistselectorParcels_Tap" >
53        <phone:LongListSelector.ItemTemplate>
54          <DataTemplate>
55            <!-- Yksittäisen lähetyksen tiedot sisältävä luettelosolu. -->
56
57            <StackPanel Background="{Binding Status,
58              Converter={StaticResource StatusColorConverter}}" >
59              <TextBlock x:Name="tbName" Text="{Binding Name}"
60                Style="{StaticResource PhoneTextTitle2Style}" />
61              <TextBlock x:Name="tbStatus" Text="{Binding FormattedStatus}"
62                Style="{StaticResource PhoneTextSmallStyle}" />
63            </StackPanel>
64          </DataTemplate>
65        </phone:LongListSelector.ItemTemplate>
66      </phone:LongListSelector>
67    </Grid>
68  </Grid>
69
70 </phone:PhoneApplicationPage>
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Net;
5 using System.Windows;
6 using System.Windows.Controls;
7 using System.Windows.Navigation;
8 using Microsoft.Phone.Controls;
9 using Microsoft.Phone.Shell;
10 using Pakettivahti.Resources;
11
12
13 using Pakettivahti.Core.Model;
14 using System.Collections.ObjectModel;
15 using System.IO.IsolatedStorage;
16 using Pakettivahti.Core.DAO;
17 using System.Globalization;
18 using Pakettivahti.Core.Service; // ParcelDataContext
19
20 namespace Pakettivahti
21 {
22     public partial class MainPage : PhoneApplicationPage
23     {
24         // Constructor
25         public MainPage()
26         {
27             InitializeComponent();
28
29             // Asetetaan sivun datacontextiksi ViewModel.
30             this.DataContext = App.ViewModel;
31
32             // Lisätään tapahtumankäsittelijä sivun latautumisen valmistumiseen.
33             this.Loaded += MainPage_Loaded;
34
35         }
36
37         private void MainPage_Loaded(object sender, RoutedEventArgs e)
38         {
39             // Suoritetaan, kun käyttöliittymäelementit on ladattu.
40
41             // Kytetään lähetysluettelo longlistselectoriin ja järjestetään se statuksen mukaan.
42             // False = ViewModel ei hae kaikkia uudestaan tietokannasta.
43             updateList(false);
44
45             // Päivitetään kaikki lähetykset verkkopalvelusta.
46             // Null = päivitetään kaikki, eli ei anneta yhtä lähetystä.
47             checkStatus(null);
48
49         }
50
51         protected override void OnNavigatedTo(NavigationEventArgs e)
52         {
53             base.OnNavigatedTo(e);
54         }
55
56         protected override void OnNavigatedFrom(NavigationEventArgs e)
57         {
58             // Tallennetaan mahdolliset muutokset tietokantaan.
59             App.ViewModel.SaveChanges();
60
61             // Lasketaan noudettavissa olevien lähetysten määrä.
62             List<Parcel> parcelReadyCheck = App.ViewModel.Parcels.OrderByDescending(p => p.Status).ToList();
63
64             int count = 0;
65
66             foreach (Parcel parcel in parcelReadyCheck)
67             {
68                 if (parcel.Status == StatusNames.READY)
69                 {
70                     count++;
71                 }
72             }
73
74             // Päivitetään sovelluksen tile.
75             TileUpdater tileUpdater = new TileUpdater();
76             tileUpdater.UpdateDefaultTile(count);
77
78             base.OnNavigatedFrom(e);
79         }
80
81         public void updateList(Boolean shouldLoadFromDB)
82         {
83             // Päivittää käyttöliittymän seurattavien lähetysten luettelon.
84
85             if (shouldLoadFromDB)
86             {
87                 // Päivitetään tietokannasta lähetystiedot ViewModeliin.
88                 App.ViewModel.LoadFromDatabase();
89             }
90         }
91     }
```

```
92
93     // Päivitetään luettelo statuksen mukaan järjestetyksi.
94     longlistselectorParcels.ItemsSource =
95         App.ViewModel.Parcels.OrderByDescending(p => p.Status).ToList();
96
97 }
98
99 private void checkStatus(Parcel newParcel)
100 {
101     // Tarkistaa verkkopalvelusta saadun lähetysohion tai kaikkien lähetysten tiedot.
102
103     // Kootaan luettelo päivitettävistä lähetysohioista.
104     ObservableCollection<Parcel> parcelsToCheck;
105
106     // Jos on saatu lähetysohio, tarkistetaan vain sen tiedot.
107     if (newParcel != null)
108     {
109         // Lisätään haettava lähetysohio listaan.
110         parcelsToCheck = new ObservableCollection<Parcel>();
111         parcelsToCheck.Add(newParcel);
112     }
113     else
114     {
115         // Ei saatu yhtä lähetysohioa, joten haetaan kaikkien tiedot. Järjestyksellä ei nyt ole väliä.
116         parcelsToCheck = App.ViewModel.Parcels;
117     }
118
119     // Otetaan verkkopalvelu käyttöön ja välitetään sille päivitettävien luettelo
120     // sekä käyttöliittymän päivittävä metodi.
121     ParcelWebChecker webService = new ParcelWebChecker(parcelsToCheck, updateList);
122
123     // Käynnistetään päivitys. True = asynkronisesti, eli lataus tapahtuu taustasäikeessä.
124     webService.checkStatus(true);
125
126 }
127
128
129 private void newButton_Click(object sender, EventArgs e)
130 {
131     // Käyttäjä haluaa lisätä uuden lähetyksen. Siirrytään lisäyssivulle.
132     this.NavigationService.Navigate(new Uri("/Views/NewParcel.xaml", UriKind.Relative));
133 }
134
135
136 private void longlistselectorParcels_Tap(object sender, System.Windows.Input.GestureEventArgs e)
137 {
138     // Käyttäjä on valinnut listalta lähetyksen.
139
140     // Etsitään listalta valittu lähetysohio.
141     Parcel selectedParcel = longlistselectorParcels.SelectedItem as Parcel;
142
143     if (selectedParcel != null)
144     {
145         // Nollataan valinta, jottei jatkossa näytettäisi samaa, jos klikataan tyhjää listaa.
146         longlistselectorParcels.SelectedItem = null;
147
148         // Siirrytään koodinnäyttöön ja välitetään koodi sinne.
149         this.NavigationService.Navigate(new Uri("/Views/ShowCode.xaml?code=" + selectedParcel.Code,
150             UriKind.Relative));
151     }
152 }
153
154 private void UpdateMenuItem_Click(object sender, EventArgs e)
155 {
156     // Käyttäjä on valinnut Application Barin menusta "Päivitä tiedot".
157
158     // Haetaan tiedot nettipalvelusta. Null = kaikki lähetykset.
159     checkStatus(null);
160 }
161
162 }
163 }
```

```

1 <phone:PhoneApplicationPage
2   x:Class="Pakettivahti.NewParcel"
3   xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
4   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
5   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
6   xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
7   xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
8   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
9   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
10  FontFamily="{StaticResource PhoneFontFamilyNormal}"
11  FontSize="{StaticResource PhoneFontSizeNormal}"
12  Foreground="{StaticResource PhoneForegroundBrush}"
13  SupportedOrientations="Portrait" Orientation="Portrait"
14  mc:Ignorable="d"
15  shell:SystemTray.IsVisible="True">
16
17  <phone:PhoneApplicationPage.ApplicationBar>
18    <shell:ApplicationBar>
19      <shell:ApplicationBarIconButton Text="Tallenna" IconUri="/Assets/AppBar/save.png"
20        Click="ApplicationBarIconButton_Click" />
21    </shell:ApplicationBar>
22  </phone:PhoneApplicationPage.ApplicationBar>
23
24  <!--LayoutRoot is the root grid where all page content is placed-->
25  <Grid x:Name="LayoutRoot" Background="Transparent">
26    <Grid.RowDefinitions>
27      <RowDefinition Height="Auto"/>
28      <RowDefinition Height="*" />
29    </Grid.RowDefinitions>
30
31    <!--TitlePanel contains the name of the application and page title-->
32    <StackPanel Grid.Row="0" Margin="12,17,0,28">
33      <TextBlock Text="{Binding Path=LocalizedResources.ApplicationTitle,
34        Source={StaticResource LocalizedStrings}}" Style="{StaticResource PhoneTextNormalStyle}" />
35      <TextBlock Text="{Binding Path=LocalizedResources.NewPage,
36        Source={StaticResource LocalizedStrings}}" Margin="9,-7,0,0"
37        Style="{StaticResource PhoneTextTitle1Style}" />
38    </StackPanel>
39
40    <!--ContentPanel - place additional content here-->
41    <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
42      <Grid.RowDefinitions>
43        <RowDefinition Height="135" />
44        <RowDefinition Height="135" />
45      </Grid.RowDefinitions>
46
47      <StackPanel Grid.Row="0" Margin="6,0,0,0">
48        <TextBlock HorizontalAlignment="Left" Margin="12,12,0,-10" TextWrapping="Wrap"
49          Text="{Binding Path=LocalizedResources.NewCode,
50            Source={StaticResource LocalizedStrings}}"
51          VerticalAlignment="Top" Width="434" Height="36"/>
52        <toolkit:PhoneTextBox x:Name="txtCode" HorizontalAlignment="Left" Height="71"
53          Margin="0,0,0,0" TextWrapping="Wrap" Hint="{Binding LocalizedResources.NewCodeHint,
54            Source={StaticResource LocalizedStrings}}" VerticalAlignment="Top" Width="444"
55          KeyUp="txtCode_KeyUp" />
56      </StackPanel>
57
58      <StackPanel Grid.Row="1" Margin="6,0,0,0">
59        <TextBlock HorizontalAlignment="Left" Margin="12,0,0,-10" TextWrapping="Wrap"
60          Text="{Binding Path=LocalizedResources.NewName,
61            Source={StaticResource LocalizedStrings}}"
62          VerticalAlignment="Top" Width="434" Height="36"/>
63        <toolkit:PhoneTextBox x:Name="txtName" HorizontalAlignment="Left" Height="72"
64          Margin="0,0,0,0" TextWrapping="Wrap" Hint="{Binding Path=LocalizedResources.NewNameHint,
65            Source={StaticResource LocalizedStrings}}" VerticalAlignment="Top" Width="444"
66          KeyUp="txtName_KeyUp" />
67      </StackPanel>
68
69
70    </Grid>
71  </Grid>
72
73  </phone:PhoneApplicationPage>

```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Net;
4 using System.Windows;
5 using System.Windows.Controls;
6 using System.Windows.Navigation;
7 using Microsoft.Phone.Controls;
8 using Microsoft.Phone.Shell;
9 using Pakettivahti.Core.DAO; // Data Context
10 using Pakettivahti.Core.Model; // Parcel
11 using System.Windows.Input;
12
13
14 namespace Pakettivahti
15 {
16     public partial class NewParcel : PhoneApplicationPage
17     {
18         public NewParcel()
19         {
20             InitializeComponent();
21
22             // Asetetaan sivun data contextiksi ViewModel.
23             this.DataContext = App.ViewModel;
24         }
25
26         private void ApplicationBarIconButton_Click(object sender, EventArgs e)
27         {
28             // Käyttäjä on koskettanut tallennusikonkia.
29             tryToSaveAndExit();
30         }
31
32         private void tryToSaveAndExit()
33         {
34             // Yritetään tallentaa syötetty lähetys seurattavaksi ja poistua lisäyssivulta.
35
36             // Luetaan syötetyt nimi ja seurantakoodi.
37             string name = txtName.Text;
38             string code = txtCode.Text;
39
40             if (code.Length == 0)
41             {
42                 // Koodi ei saa olla tyhjä. Näytetään virheilmoitus ja siirretään käyttäjä syöttämään koodi.
43                 MessageBox.Show("Syötä lähetystunnus!");
44                 txtCode.Focus();
45             }
46             else
47             {
48                 // Tallennetaan ja poistutaan vain jos tunnus on annettu.
49                 if (name.Length == 0)
50                 {
51                     // Nimen ollessa tyhjä, kopioidaan lähetystunnus siihen.
52                     name = code;
53                 }
54
55                 // Luodaan uusi lähetysolio.
56                 Parcel newParcel = new Parcel();
57                 newParcel.Name = name;
58                 newParcel.Code = code;
59                 newParcel.Status = StatusNames.DEFAULT;
60                 newParcel.LastEventTime = DateTime.Now;
61
62                 // Lisätään uusi lähetys ViewModeliin, josta se tallentuu tietokantaan.
63                 App.ViewModel.AddParcel(newParcel);
64
65                 // Palataan takaisin päänäkömään.
66                 if (NavigationService.CanGoBack)
67                 {
68                     NavigationService.GoBack();
69                 }
70             }
71         }
72
73         private void txtCode_KeyUp(object sender, System.Windows.Input.KeyEventArgs e)
74         {
75             if (e.Key == Key.Enter)
76             {
77                 // Jos Käyttäjän painama näppäin oli Enter, siirrytään syöttämään lähetyksen nimeä.
78                 txtName.Focus();
79             }
80         }
81
82         private void txtName_KeyUp(object sender, System.Windows.Input.KeyEventArgs e)
83         {
84             if (e.Key == Key.Enter)
85             {
86                 // Jos käyttäjän painama näppäin oli Enter, tallennetaan lähetys.
87                 tryToSaveAndExit();
88             }
89         }
90     }
91 }
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data.Linq; // Binary
5 using System.Data.Linq.Mapping;
6 using System.Linq;
7 using System.Runtime.CompilerServices;
8 using System.Text;
9 using System.Threading.Tasks;
10
11 namespace Pakettivahti.Core.Model
12 {
13
14     // Luokka edustaa tietokantataulua.
15     [Table]
16     public class Parcel : INotifyPropertyChanged, INotifyPropertyChanging
17     {
18
19         // Tämä määrittely parantaa tietävästi merkittävästi päivitysnopeutta.
20         // [Column(IsVersion = true)]
21         // private Binary _version;
22
23         // Automaattinen päivitys otetaan käyttöön, jotta olioiden muutokset välittyisivät mm. luetteloihin.
24         public event PropertyChangedEventHandler PropertyChanged;
25
26         // Tarvittavat set-metodit kutsuvat tätä.
27         // Vain päivitysaika ja status voivat muuttua olion luomisen jälkeen, joten vain niistä kutsutaan.
28         private void NotifyPropertyChanged([CallerMemberName] String propertyName = "")
29         {
30             // Jos joku kuuntelee ja on asettanut tapahtumankäsittelijän, lähetetään muutoksesta ilmoitus.
31             if (PropertyChanged != null)
32             {
33                 {
34                     PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
35                 }
36             }
37
38             // Sama vaihtumisilmoituksille, eli tieto siitä, että property on juuri muuttumassa.
39             // Tämän lisääminen auttaa rajoittamaan muistinkäyttöä muutosten jäljittämisen yhteydessä.
40             public event PropertyChangingEventHandler PropertyChanging;
41
42             private void NotifyPropertyChanging(string propertyName)
43             {
44                 if (PropertyChanging != null)
45                 {
46                     {
47                         PropertyChanging(this, new PropertyChangingEventArgs(propertyName));
48                     }
49                 }
50
51                 private int id;
52
53                 private string name;
54
55                 private string code;
56
57                 private DateTime lastEventTime;
58
59                 private int status;
60
61                 public Parcel()
62                 {
63                     {
64                         }
65                     }
66
67                 public Parcel(int id, string name, string code, DateTime eventTime, int status)
68                 {
69                     {
70                         this.id = id;
71                         this.name = name;
72                         this.code = code;
73                         this.lastEventTime = eventTime;
74                         this.status = status;
75                     }
76                 }
77
78                 [Column(IsPrimaryKey = true, IsDbGenerated = true, DbType = "INT NOT NULL Identity",
79                     CanBeNull = false, UpdateCheck = UpdateCheck.Always)]
80                 public int Id
81                 {
82                     get { return id; }
83                     set
84                     {
85                         if (id != value)
86                         {
87                             id = value;
88                         }
89                     }
90                 }
91             }
92         }
```

```
92     // UpdateCheck on Never, koska nimeä ei luomisen jälkeen päivitetä.
93     [Column(CanBeNull = false, UpdateCheck = UpdateCheck.Never)]
94     public string Name
95     {
96         get { return name; }
97         set
98         {
99             name = value;
100         }
101     }
102
103     // UpdateCheck on Never, koska koodia ei luomisen jälkeen päivitetä.
104     [Column(CanBeNull = false, UpdateCheck = UpdateCheck.Never)]
105     public string Code
106     {
107         get { return code; }
108         set
109         {
110             code = value;
111         }
112     }
113
114     // Tarkistetaan vain kun aikaa päivitetään.
115     [Column(CanBeNull = false, UpdateCheck = UpdateCheck.WhenChanged)]
116     public DateTime LastEventTime
117     {
118         get { return lastEventTime; }
119         set
120         {
121             NotifyPropertyChanging("LastEventTime");
122             lastEventTime = value;
123             NotifyPropertyChanged("LastEventTime");
124         }
125     }
126
127     // Tarkistetaan päivitettäessä.
128     [Column(CanBeNull = false, UpdateCheck = UpdateCheck.WhenChanged)]
129     public int Status
130     {
131         get { return status; }
132         set
133         {
134             if ((value >= StatusNames.STATUS_MIN) && (value <= StatusNames.STATUS_MAX)
135                 && (status != value))
136             {
137                 // Viimeinen ehto varmistaa, että arvo tosiaan muuttuu.
138
139                 // Ilmoitetaan Status-kentän muuttumisesta.
140                 NotifyPropertyChanging("Status");
141                 NotifyPropertyChanging("FormattedStatus");
142                 status = value;
143                 NotifyPropertyChanged("Status");
144                 NotifyPropertyChanged("FormattedStatus");
145             }
146             else
147             {
148                 status = StatusNames.DEFAULT;
149             }
150         }
151     }
152
153     // Tällä yhdistetään aikaleima ja statustiedot yhdeksi riviksi käyttöliittymää varten.
154     public string FormattedStatus
155     {
156         get
157         {
158             // Päivämäärä ja kellonaika, G edustaa yleistä päivämäärä + kellonaika sekunneilla
159             // -merkintää lokalisaatioasetuksen mukaisesti esitettynä.
160             string result = lastEventTime.ToString("G") + " - " + StatusNames.Names[status];
161
162             return result;
163         }
164     }
165 }
166 }
167 }
```

```
1 using Pakettivahti.Core.Model; // Parcel
2 using System;
3 using System.Collections.Generic;
4 using System.Data.Linq; // DataContext
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace Pakettivahti.Core.DAO
10 {
11     public class ParcelDataContext : DataContext
12     {
13         // Tietokannan yhteysmerkkijono, jossa kerrotaan tietokantatiedoston sijainti ja nimi.
14         public static string ConnectionString = @"Data Source=isostore:/Parcels.sdf";
15
16         public ParcelDataContext(string connectionString)
17             : base(connectionString) { }
18
19         // Tietokantataulussa Parcels on Parcel-olioita.
20         public Table<Parcel> Parcels;
21     }
22 }
```



```
1 using Pakettivahti.Core.DAO; // ParcelDataContext
2 using Pakettivahti.Core.Model; // Parcel
3 using System;
4 using System.Collections.Generic;
5 using System.Collections.ObjectModel; // ObservableCollection
6 using System.ComponentModel; // INotifyPropertyChanged
7 using System.Data.Linq;
8 using System.Diagnostics; // Debug
9 using System.Linq;
10 using System.Text;
11 using System.Threading.Tasks;
12
13 namespace Pakettivahti.ViewModel
14 {
15     public class ParcelViewModel : INotifyPropertyChanged
16     {
17         // LINQ to SQL data context.
18         private ParcelDataContext parcelDB;
19
20         public ParcelViewModel(string parcelDBConnectionString)
21         {
22             parcelDB = new ParcelDataContext(parcelDBConnectionString);
23         }
24
25         // Lähetysolioiden luettelo.
26         private ObservableCollection<Parcel> _parcels;
27
28         public ObservableCollection<Parcel> Parcels
29         {
30             // Palautetaan statuksen mukaan järjestetty kopio luettelosta.
31             // Itse luettelo ei ole järjestettävissä.
32             get {
33
34                 return _parcels; }
35             set
36             {
37                 _parcels = value;
38                 NotifyPropertyChanged("Parcels");
39             }
40         }
41
42         // Haetaan tietokannasta lähetystiedot.
43         public void LoadFromDatabase()
44         {
45             // Luodaan kysely kaikkien lähetysten haulle.
46             var allParcelsInDB = from Parcel parcel in parcelDB.Parcels
47                                 orderby parcel.Status descending select parcel;
48
49             // Toteutetaan kysely ja päivitetään luettelo.
50             // (Deferred loading = kysely vasta, kun observable collection luodaan)
51             Parcels = new ObservableCollection<Parcel>(allParcelsInDB);
52
53             // Tehdään kopioluettelo poistoja varten.
54             List<Parcel> copyOfParcels = new List<Parcel>(allParcelsInDB);
55
56             // Poistetaan noudetut lähetykset.
57             foreach (Parcel parcel in copyOfParcels)
58             {
59                 if (parcel.Status >= StatusNames.FINISHED)
60                 {
61                     DeleteParcel(parcel);
62                 }
63             }
64
65         }
66
67         // Lähetysten lisääminen luetteloon ja tietokantaan.
68         public void AddParcel(Parcel newParcel)
69         {
70             try
71             {
72                 // Lisätään lähetys data contexttiin.
73                 parcelDB.Parcels.InsertOnSubmit(newParcel);
74
75                 // Tallennetaan tietokannan muutokset.
76                 SaveChanges();
77
78                 // Lisätään lähetys myös luetteloon.
79                 Parcels.Add(newParcel);
80             }
81             catch (Exception e)
82             {
83                 Debug.WriteLine(e);
84             }
85         }
86
87         // Poistetaan lähetys luettelosta ja tietokannasta.
88         public void DeleteParcel(Parcel parcelToBeDeleted)
89
90
91
```

```
92     {
93         // Poistetaan lähetys luettelosta.
94         Parcels.Remove(parcelToBeDeleted);
95
96         // Poistetaan se data contextista.
97         parcelDB.Parcels.DeleteOnSubmit(parcelToBeDeleted);
98
99         // Tallennetaan tietokannan muutokset.
100        SaveChanges();
101    }
102
103    // Tallennetaan data contextin muutokset tietokantaan.
104    public void SaveChanges()
105    {
106        try
107        {
108            parcelDB.SubmitChanges();
109        }
110        catch (Exception e)
111        {
112            // Jos jostain syystä tulisi rinnakkaisuuden kanssa ongelmia, ratkaistaan ylikirjoittamalla.
113            foreach (ObjectChangeConflict occ in parcelDB.ChangeConflicts)
114            {
115                // Ylikirjoitetaan tietokannan arvot.
116                occ.Resolve(RefreshMode.OverwriteCurrentValues);
117            }
118        }
119    }
120
121    // Muutosilmoitukset.
122
123    public event PropertyChangedEventHandler PropertyChanged;
124
125    private void NotifyPropertyChanged(string propertyName)
126    {
127        if (PropertyChanged != null)
128        {
129            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
130        }
131    }
132
133    }
134
135    }
136 }
```

```
1 using Pakettivahti.Core.DAO;
2 using Pakettivahti.Core.Model;
3 using System;
4 using System.Collections.Generic;
5 using System.Collections.ObjectModel;
6 using System.Globalization;
7 using System.Linq;
8 using System.Net;
9 using System.Text;
10 using System.Threading.Tasks;
11
12
13 namespace Pakettivahti.Core.Service
14 {
15     public class ParcelWebChecker
16     {
17         // Mahdollinen käyttöliittymän päivitysmetodi.
18         private Action<Boolean> listViewUpdater;
19
20         // Tarkistettavien lähetysten luettelo.
21         private ObservableCollection<Parcel> parcelsToCheck;
22
23         public ParcelWebChecker(ObservableCollection<Parcel> parcelsToCheck,
24                                 Action<Boolean> listViewUpdaterMethod)
25         {
26             this.parcelsToCheck = parcelsToCheck;
27             listViewUpdater = listViewUpdaterMethod;
28         }
29
30         public void checkStatus(Boolean isAsync)
31         {
32             // Päivitetään lähetykset verkkopalvelusta.
33
34             // Käsitellään luokalle konstruktorissa annettu lähetysten luettelo.
35             foreach(Parcel parcel in parcelsToCheck)
36             {
37
38                 // Haetaan nettipalvelusta annetun lähetyksen tiedot, joko asynkronisesti tai synkronoidusti.
39
40                 // Kun tarkoitus on lukea pelkästään yksi sivu yksinkertaisella http-get-pyyntöllä,
41                 // käytetään WebClient-oliota.
42                 WebClient webClient = new WebClient();
43
44                 // Muodostetaan Postin palveluun osoite lähetystunnuksen perusteella.
45                 // Pyydetään englanninkielinen sivu.
46                 Uri serviceUri = new Uri("https://posti.mobi/package-tracking/"
47                                         + new Parcel.Code + "?lang=en");
48
49                 // Testipalvelun osoite. Testipalvelin vastaanottaa lähetyksen nykyisen tilan
50                 // ja palauttaa seuraavaa tilaa vastaavan sivun.
51                 Uri serviceUri = new Uri("http://pakettivahti.aueta.com/testipalvelu.php?status="
52                                         + parcel.Status);
53
54                 if (isAsync)
55                 {
56                     // Asynkroninen haku. Lähetetään pyyntö ja käsitellään vastaus aikanaan, kun se on saatu.
57
58                     // Asetetaan toiminto, joka tehdään latauksen päätteeksi.
59                     webClient.DownloadStringCompleted +=
60                         new DownloadStringCompletedEventHandler(statusDownloadCompleted);
61
62                     // Ladataan sivu. Käyttämällä DownloadStringAsync-metodia ei tarvitse huolehtia
63                     // verkkoyhteyden sulkemisesta, se tapahtuu automaattisesti.
64                     // Välitetään asynkronisen latauksen mukaan myös käsiteltävän lähetyksen tunnistetieto,
65                     // jotta luettu tieto voidaan tallentaa siihen.
66                     webClient.DownloadStringAsync(serviceUri, parcel.Id);
67                 }
68                 else
69                 {
70                     // Synkronoitu haku. Odotetaan vastausta ja käsitellään se odottelun päätteeksi.
71                     // Kutsu on tullut tausta-agentilta.
72
73                     // Luodaan olio, johon ladatun html-sivun sisältö tallennetaan merkkijonona.
74                     var tcs = new TaskCompletionSource<String>();
75
76                     // Ja asetetaan siihen ladattu sivu latauksen päätteeksi uudessa tehtäväkäsittelijässä.
77                     webClient.DownloadStringCompleted += (sender, EventArgs) =>
78                     {
79                         // Palautetaan joko virheilmoitus tai ladattu sivu.
80                         if (EventArgs.Error == null)
81                         {
82                             tcs.SetResult(EventArgs.Result);
83                         }
84                         else
85                         {
86                             tcs.SetException(EventArgs.Error);
87                         }
88                     };
89
90                     // Ladataan sivu. Käyttämällä DownloadStringAsync-metodia ei tarvitse huolehtia
91                     // verkkoyhteyden sulkemisesta, se tapahtuu automaattisesti.
```

```
92         webClient.DownloadStringAsync(serviceUri);
93
94         // Odotetaan siis latauksen valmistumista TaskCompletionSource:n avulla.
95         // Käsitellään palvelimelta saatu vastaus tälle lähetykselle.
96         processPageToParcel(tcs.Task.Result, parcel.Id);
97     }
98
99     }
100 }
101
102
103 private void statusDownloadCompleted(object sender, DownloadStringCompletedEventArgs e)
104 {
105     // Asynkronisen latauksen tapahtumankäsittelijä, joka suoritetaan latauksen päätyttyä.
106
107     // Lataus on valmistunut. Tutkitaan mahdollinen virhe.
108     if (e.Error != null)
109     {
110         System.Diagnostics.Debug.WriteLine("Verkkopalveluun yhdistäminen epäonnistui: "
111             + e.Error.Message);
112     }
113     else
114     {
115         // Haetaan lähetyksen id ja käsitellään vastaanotettu html-sivu.
116         int id = (Int32)e.UserState;
117         processPageToParcel(e.Result, id);
118     }
119 }
120
121
122
123
124 private void processPageToParcel(string page, int parcelId)
125 {
126     // Luetaan HTML-sivusta annetulle lähetykselle tilatieto.
127
128     Parcel parcel = null;
129
130     // Haetaan luettelosta lähetykset, jolle luettu sivu käsitellään.
131     parcel = parcelsToCheck.Single(x => x.Id == parcelId);
132
133     // Jatketaan vain, jos luettelosta löytyi lähetysoolio.
134     if (parcel == null)
135     {
136         System.Diagnostics.Debug.WriteLine("Lähetystä ei löytynyt.");
137         return;
138     }
139
140     // Etsitään näiden avulla html-koodista oikea kohta.
141     string startTag = "events";
142     string testTag = "hasTouchEvent";
143     string endTag = "footer";
144     int start = page.IndexOf(startTag) + startTag.Length + 2; // 2 = lisätään " ja >-merkit.
145     int test = page.IndexOf(testTag) + testTag.Length + 2;
146     int end = page.IndexOf(endTag);
147
148     if (start < end)
149     {
150         // Sivulla on lähetykselle seurantatietoja ennen footeria.
151
152         // Leikataan sivusta kiinnostava osa jatkotutkimuksia varten.
153         string partOfPage = page.Substring(start, end - start);
154
155         // Erotetaan ensimmäisen <p>-tagin sisältö päivämääräaroksi.
156         startTag = "<p>";
157         endTag = "</p>";
158
159         start = partOfPage.IndexOf(startTag) + startTag.Length;
160         end = partOfPage.IndexOf(endTag);
161
162         string dateString = partOfPage.Substring(start, end - start);
163
164         // Varmuuden vuoksi päivämäärään asetetaan jokin arvo.
165         DateTime date = DateTime.MinValue;
166
167         try
168         {
169             // Muunnetaan luettu päivämäärä merkkijonosta DateTime-olioksi.
170             date = DateTime.ParseExact(dateString, "dd MMMM yyyy HH:mm", CultureInfo.InvariantCulture);
171         }
172         catch (Exception e)
173         {
174             // Epäonnistunut muunnos. Jotain meni pieleen sivun latauksessa.
175             System.Diagnostics.Debug.WriteLine("Tietosivun muunnos päivämääräksi epäonnistui: "
176                 + e.StackTrace);
177         }
178
179         // Määritellään lähetyksen tilatieto.
180         if (partOfPage.Contains("ARK"))
181         {
182
```

```
183         parcel.Status = StatusNames.ARCHIVED;
184         if (date != DateTime.MinValue)
185         {
186             parcel.LastEventTime = date;
187         }
188     }
189     else if (partOfPage.Contains("LUO"))
190     {
191         parcel.Status = StatusNames.FINISHED;
192         if (date != DateTime.MinValue)
193         {
194             parcel.LastEventTime = date;
195         }
196     }
197     else if (partOfPage.Contains("HYL"))
198     {
199         parcel.Status = StatusNames.READY;
200         if (date != DateTime.MinValue)
201         {
202             parcel.LastEventTime = date;
203         }
204     }
205     else if (partOfPage.Contains("REK"))
206     {
207         parcel.Status = StatusNames.REGISTERED;
208         if (date != DateTime.MinValue)
209         {
210             parcel.LastEventTime = date;
211         }
212     }
213 }
214 }
215 else
216 {
217     // Ei ole rekisteröity vielä, koska ei tietoja löydy. Lähetystunnushan on oikea...
218     parcel.Status = StatusNames.UNKNOWN;
219
220     // Päivitetään aikaleimaksi tarkastusaika, eli nykyhetki.
221     parcel.LastEventTime = DateTime.Now;
222 }
223
224 // Yritetään päivittää lähetysluettelo.
225 updateListView();
226 }
227
228 private void updateListView()
229 {
230     // Jos luokalle on annettu käyttöliittymäpäivitysmetodi, kutsutaan sitä.
231     if (this.listViewUpdater != null)
232     {
233         listViewUpdater(true);
234     }
235 }
236
237 }
238
239 }
240 }
241 }
```

```

1 <phone:PhoneApplicationPage
2   x:Class="Pakettivahti.Views.ShowCode"
3   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5   xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
6   xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
7   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
8   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
9   FontFamily="{StaticResource PhoneFontFamilyNormal}"
10  FontSize="{StaticResource PhoneFontSizeNormal}"
11  Foreground="{StaticResource PhoneForegroundBrush}"
12  SupportedOrientations="Landscape" Orientation="Landscape"
13  mc:Ignorable="d"
14  shell:SystemTray.IsVisible="True">
15
16  <!--LayoutRoot is the root grid where all page content is placed-->
17  <Grid x:Name="LayoutRoot" Background="Transparent" DoubleTap="LayoutRoot_DoubleTap">
18    <Grid.RowDefinitions>
19      <RowDefinition Height="Auto"/>
20      <RowDefinition Height="*/>
21    </Grid.RowDefinitions>
22
23    <!--TitlePanel contains the name of the application and page title-->
24    <StackPanel Grid.Row="0" Margin="12,17,0,28">
25      <TextBlock Text="{Binding Path=LocalizedResources.ApplicationTitle,
26        Source={StaticResource LocalizedStrings}}" Style="{StaticResource PhoneTextNormalStyle}"/>
27      <TextBlock Text="{Binding Path=LocalizedResources.ShowCodePage,
28        Source={StaticResource LocalizedStrings}}" Margin="9,-7,0,0"
29        Style="{StaticResource PhoneTextTitle1Style}"/>
30    </StackPanel>
31
32    <!--ContentPanel - place additional content here-->
33    <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
34
35      <!-- Seurantakoodin näyttökenttä -->
36
37      <TextBlock x:Name="tbCode" HorizontalAlignment="Left" Margin="10,30,0,0" TextWrapping="Wrap"
38        Text="FI1234567890123456789" VerticalAlignment="Top" Width="684" Height="199"
39        Style="{StaticResource PhoneTextTitle1Style}" TextAlignment="Center"/>
40
41    </Grid>
42  </Grid>
43
44 </phone:PhoneApplicationPage>

```

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Net;
5 using System.Windows;
6 using System.Windows.Controls;
7 using System.Windows.Navigation;
8 using Microsoft.Phone.Controls;
9 using Microsoft.Phone.Shell;
10
11 namespace Pakettivahti.Views
12 {
13     public partial class ShowCode : PhoneApplicationPage
14     {
15         public ShowCode()
16         {
17             InitializeComponent();
18         }
19
20         protected override void OnNavigatedTo(NavigationEventArgs e)
21         {
22             base.OnNavigatedTo(e);
23
24             // Luetaan koodi Ursta.
25             tbCode.Text = this.NavigationContext.QueryString["code"];
26         }
27
28         private void LayoutRoot_DoubleTap(object sender, System.Windows.Input.GestureEventArgs e)
29         {
30             // Palataan takaisin pääsivulle.
31             this.NavigationService.GoBack();
32         }
33     }
34 }

```

```
1 using Pakettivahti.Core.Model;
2 using System;
3 using System.Collections.Generic;
4 using System.Diagnostics;
5 using System.Globalization;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Data;
10
11 namespace Pakettivahti.ViewModel
12 {
13
14     //[ValueConversion(typeof(Int), typeof(String))]
15     public class StatusColorConverter : IValueConverter
16     {
17         // Muunnetaan lähetysolion statustieto sopimaan luettelosolun taustaväriin nimeen.
18
19         public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
20         {
21             // Palautetaan solun korostustaustaväriin nimi, jos lähetys on noudettavissa.
22             if ((int)value == StatusNames.READY)
23             {
24                 return "DarkGreen";
25             }
26             else
27             {
28                 // Ei anneta ollenkaan väriä, jolloin käytetään oletusväriä.
29                 return null;
30             }
31         }
32
33         public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
34         {
35             // Muunnos suoritetaan vain yhteen suuntaa, statustiedosta väriksi.
36             return null;
37         }
38     }
39 }
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Pakettivahti.Core.Model
8 {
9     public class StatusNames
10     {
11         // Taulukko sisältää lähetysseurannan tilan merkkijonona.
12         public static readonly string[] Names = { "----- Haetaan tietoja. -----", "Ei tietoja.",
13             "Rekisteröity", "Noudettavissa.", "Noudettu", "Arkistoitu" };
14
15         // Tila-arvon raja-arvot.
16         public static readonly int STATUS_MIN = 0;
17         public static readonly int STATUS_MAX = Names.Length - 1; // Indeksointi alkaa nolasta.
18
19         // Lähetysseurannan tilavakiot.
20         public static readonly int DEFAULT = 0;
21         public static readonly int UNKNOWN = 1;
22         public static readonly int REGISTERED = 2;
23         public static readonly int READY = 3;
24         public static readonly int FINISHED = 4;
25         public static readonly int ARCHIVED = 5;
26     }
27 }
```

```
1 using Microsoft.Phone.Shell;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace Pakettivahti.Core.Service
9 {
10     public class TileUpdater
11     {
12
13         public void UpdateDefaultTile(int count)
14         {
15             // Päivitetään sovelluksen oletus-Tileen noutoa odottavien lähetysten lukumäärä ja aikaleima.
16
17             // Muodostetaan aikaleima merkkijonoksi.
18             DateTime now = DateTime.Now;
19             string text = "" + now;
20
21             // Muotoillaan suomenkielinen teksti lukumäärän mukaan.
22             switch (count)
23             {
24                 case 0:
25                     text += ". Ei saapuneita lähetyksiä.";
26                     break;
27
28                 case 1:
29                     text += ". Yksi saapunut lähetys!";
30                     break;
31
32                 default:
33                     text += ". " + count + " saapunutta lähetystä!";
34                     break;
35             }
36
37             // Haetaan sovelluksen oletus-Tile ja lisätään siihen päivitettyt tiedot.
38             ShellTile defaultTile = ShellTile.ActiveTiles.First();
39             StandardTileData tileData = new StandardTileData { Count = count, BackContent = text };
40
41             // Päivitetään tile.
42             defaultTile.Update(tileData);
43
44         }
45     }
46 }
47
```



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <Deployment xmlns="http://schemas.microsoft.com/windowsphone/2012/deployment" AppPlatformVersion="8.0">
3   <DefaultLanguage xmlns="" code="fi-FI" />
4   <Languages xmlns="">
5     <Language code="fi-FI" />
6   </Languages>
7   <App xmlns="" ProductID="{4773177e-bf8e-4fb8-b09a-7bda318660c7}" Title="Pakettivahti"
8     RuntimeType="Silverlight" Version="1.0.0.0" Genre="apps.normal" Author="Pakettivahti author"
9     Description="Sample description" Publisher="Pakettivahti"
10    PublisherID="{5eda2da1-a8aa-4fe8-848d-0b77bb081a50}">
11     <IconPath IsRelative="true" IsResource="false">Assets\ApplicationIcon.png</IconPath>
12     <Capabilities>
13       <Capability Name="ID_CAP_NETWORKING" />
14       <Capability Name="ID_CAP_SENSORS" />
15       <Capability Name="ID_CAP_WEBBROWSERCOMPONENT" />
16     </Capabilities>
17     <Tasks>
18       <DefaultTask Name="_default" NavigationPage="Views/MainPage.xaml" />
19
20     <!-- Taustatoiminto otetaan käyttöön. -->
21     <ExtendedTask Name="BackgroundTask">
22       <BackgroundServiceAgent Specifier="ScheduledTaskAgent" Name="BackgroundAgent"
23         Source="BackgroundAgent" Type="BackgroundAgent.ScheduledAgent" />
24     </ExtendedTask>
25
26     </Tasks>
27     <Tokens>
28       <PrimaryToken TokenID="PakettivahtiToken" TaskName="_default">
29         <TemplateFlip>
30           <SmallImageURI IsRelative="true"
31             IsResource="false">Assets\Tiles\FlipCycleTileSmall.png</SmallImageURI>
32           <Count>0</Count>
33           <BackgroundImageURI IsRelative="true"
34             IsResource="false">Assets\Tiles\FlipCycleTileMedium.png</BackgroundImageURI>
35           <Title>Pakettivahti</Title>
36           <BackContent></BackContent>
37           <BackBackgroundImageURI></BackBackgroundImageURI>
38           <BackTitle></BackTitle>
39           <DeviceLockImageURI></DeviceLockImageURI>
40           <HasLarge></HasLarge>
41         </TemplateFlip>
42       </PrimaryToken>
43     </Tokens>
44     <ScreenResolutions>
45       <ScreenResolution Name="ID_RESOLUTION_WVGA" />
46       <ScreenResolution Name="ID_RESOLUTION_WXGA" />
47       <ScreenResolution Name="ID_RESOLUTION_HD720P" />
48     </ScreenResolutions>
49   </App>
50 </Deployment>
```