VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Kosti Harvisalo

INVENTAARIOSOVELLUS

Mäkelä Alu Oy

Tekniikka
2016

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Kosti Harvisalo
Opinnäytetyön nimi	Inventaariosovellus
Vuosi	2016
Kieli	suomi
Sivumäärä	33
Ohjaaja	Pirjo Prosi

Tämän opinnäytetyön tarkoituksena oli kehittää helppokäyttöinen inventaariosovellus Mäkelä Alu Oy:lle. Yrityksessä tehtiin aikaisemmin inventaariot manuaalisesti paperille kirjoittamalla. Yritys halusi hyödyntää uusia toteutustekniikoita inventaarion tekemiseen. Sovellusohjelmalla inventaarion tekeminen myös nopeutuisi ja virheidenkin määrä pienenisi.

Opinnäytetyö koostuu kahdesta pääosasta, puhelinsovelluksesta inventaarion tekemiseen ja työpöytäsovelluksesta tietojen tarkasteluun. Puhelinsovellus kehitettiin Android -käyttöjärjestelmälle ja se tehtiin B4A -ohjelmointityökalulla (aikaisemmin Basic4Android). Windowsissa toimivan työpöytäsovelluksen ohjelmointikielenä on Java ja se kehitettiin Netbeans-ohjelmointityökalulla. Android- ja Windows-sovellukset kommunikoivat keskenään XML-tiedostoilla. Itse kommunikointiyhteys toteutettiin Android-kehitysympäristön mukana tulevan ADB-sovelluksen avulla, joka on Google Inc–omaisuutta.

Sovelluksia testattiin itsenäisesti, mutta ne pääsevät myös käytännön testaukseen Mäkelä Alulla. Sovelluksia on myös mahdollista jatkokehittää tulevaisuudessa.

Avainsanat Java, Android, B4A, XML, inventaario

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Kosti Harvisalo
Title	Inventory Application
Year	2016
Language	Finnish
Pages	33
Name of Supervisor	Pirjo Prosi

The goal of this thesis was to develop a user friendly inventory application which makes inventory easier for an aluminum factory called Mäkelä Alu Oy. Before the company had to write inventory items onto paper. The company wanted to test new implementations when doing inventory. With an inventory application, doing the inventory is faster and it also lowers the change of errors.

This thesis consists of two main sections, a phone application for doing the inventory and a desktop application for examining the information made by the phone application. The phone application was developed for the Android operating system and it was made with a development tool called B4A (previously known as Basic4Android). The desktop application works in Windows operating system and it is written in Java programming language. The development tool used for the desktop application was Netbeans. Android and Windows applications communicate with each other by XML files, and the connection itself is established by ADB software, which is the property of Google Inc.

The applications were tested independently, but they will be tested at Mäkelä Alu company too. There is also a possibility for further development.

Keywords Java, Android, B4A, XML, inventory

SISÄLLYS

KUVIO-JA TAULUKKOLUETTELO

LYHENTEET JA TERMIT

Java	Java ohjelmointikieli.
Android	Käyttöjärjestelmä puhelimille ja tableteille.
B4A	Sovelluskehitystyökalu, joka kääntää Basic-kielellä kirjoitetun ohjelmakoodin Androidille sopivaksi.
XML	Extensible Markup Language, jonka tehtävä on tallentaa ja siirtää tietoa.
ADB	Android Debug Bridge, komentokehotetyökalu, joka luo yhteyden Android-laitteen ja tietokoneen välille.
IMEI	International Mobile Equipment Identity, uniikki sarjanumero, jonka avulla voidaan tunnistaa puhelin tai tabletti.

1 JOHDANTO

Suomen lain mukaan jokaisessa yrityksessä on tehtävä vuosittain ainakin yksi inventaario, että tiedetään varaston koko ja yrityksen materiaalien arvo. Riippuen yrityksen koosta, tämä voi olla suurikin urakka, ja usein siihen tarvitaan jotain työkalua, joka helpottaa inventaarion tekemistä.

Tämä opinnäytetyö tehtiin Mäkelä Alu Oy:lle. Yritys halusi testata uusia toteutustekniikoita inventoinnissa. Tehtyä käytännön sovellusta voidaan käyttää esimerkiksi tarvikeinventaariota tehtäessä. Sovellusta voidaan tarvittaessa myös jatkokehittää yrityksen muihin älypuhelimiin sekä viivakoodien lukukäyttöön.

Opinnäytetyö koostuu kahdesta osasta. Ensin tehtiin inventaarion tekemiseen sopiva Android-sovellus ja toiseksi tietojen keräämiseen sekä hyödyntämiseen tarkoitettu Windows-työpöytäsovellus.

2 TYÖN TAUSTA

2.1 Mäkelä Alu Oy

Mäkelä Alu Oy on alumiiniprofiileiden ja niiden jatkojalostukseen keskittynyt perheyritys. Toiminta on keskitetty alumiiniprofiilien puristamiseen ja pintakäsittelyyn. Mäkelä Alu Oy:llä on useita eri osastoja, kuten puristin, pakkaamo, maalaamo, anodisointi ja valimo.

2.2 Inventaario

Kirjanpitolaki velvoittaa yrityksen tekemään tilinpäätöksen tilikauden lopuksi. Tilikausi on kestoltaan pääsääntöisesti 12 kuukautta. /1/ Tilinpäätös pitää sisällään vaihto-omaisuuden tase-erittelyn, jolla tarkoitetaan inventaarion tekemistä. Tällöin varaston arvo lasketaan fyysisesti. Varaston arvon on tärkeää olla oikea, sillä varaston arvon muutosta verrataan edellisen tilinpäätöksen inventointiin. Tämä erotus kirjataan tuloslaskelmaan ja se joko suurentaa tai pienentää yrityksen tulosta. /2/

Varaston arvossa käytetään arvonlisäverottomia hankintahintoja tai alhaisimpia todennäköisiä hankintahintoja. Inventaarioarvona voidaan pitää myös kustannuslaskennalla saatua hyödykkeen hintaa, mikäli varastossa oleva tavara on itse valmistettu. /3/

Inventaarion tarkoituksena on saada varaston todellinen arvo selville taseeseen, jotta voidaan laskea varaston muutos verrattuna edelliseen inventaarioarvoon. Varaston muutoksella pystytään näin oikaisemaan tilikauden kirjanpitoon merkittyjä ostokuluja. Tällä tavalla saadaan kohdennettua tilikaudelle vain tilikauden myyntejä vastaavat ostot. /3/

2.3 Inventaariosovelluksen hankinta

Mäkelä Alu Oy halusi opinnäytetyön aiheeksi inventaariosovelluksen. Uuden sovelluksen avulla inventaarion tekemistä halutaan nopeuttaa ja tarkentaa. Opinnäytetyön tavoitteena oli räätälöidä heitä palveleva sovellus. Yrityksen toiveena oli

hyödyntää älypuhelimeen tehtävää sovellusta sekä viivakoodin lukumahdollisuutta. Viivakoodin lukua käytetään yrityksessä paljon, joten tämän ominaisuuden hyödyntäminen inventaariossa nähtiin tärkeänä. Puhelinsovellus tehtiin Android-laitteille. Tietojen keräämiseen ja hyödyntämiseen tarvittiin myös oma työpöytäsovellus, joka on toteutettu Java-ohjelmointikielellä.

3 TYÖSSÄ KÄYTETYT TEKNIIKAT JA TYÖKALUT

3.1 Java

Java on Sun Microsystemsin vuonna 1995 kehittämä ohjelmointikieli, joka toimii monilla eri alustoilla. Sen etu on juurikin helposti siirrettävät sovellukset alustoilta toiselle, ilman monen eri sovelluksen ja lähdekoodin muokkauksia. Javalla voi kehittää sulautettuja, puhelin-, peli-, web- ja yrityssovelluksia. /4/ Vuonna 2010 Oracle osti Sun Microsystemsin, jolloin Java siirtyi Oraclen omistukseen. /5/

Tässä opinnäytetyössä ohjelmointityökaluna oli käytössä Netbeans 8.0.2. Syy Netbeansin käyttöön on sen selkeä ja yksinkertainen käyttöliittymä. Lisäksi aikaisempi käyttö teki siitä valmiiksi tutun.

3.2 B4A

B4A (aikaisemmalta nimeltään Basic4android) on Android-sovelluskehitystyökalu. Kuten sanoista Basic4android voi päätellä, kirjoitettava koodi on Basic-kieltä, ja kun sovellus suoritetaan, B4A kääntää koodin suoraan Java-muotoon. Näin ollen B4A:lla tuotettu koodi on yhtä nopeaa kuin Javalla. /6/

Sovellus voidaan suorittaa joko Android-emulaattorin avulla, kytkemällä puhelin tietokoneen USB-paikkaan tai käyttäen B4A Bridge-osaa, jonka avulla sovellus lähetetään samassa lähiverkossa olevalle Android-puhelimelle.

B4A:n suurimpiin etuihin kuuluu graafinen ulkoasun luominen (Visual Designer), jolla voi nopeasti luoda ulkoasun sovellukseen. Myös kotisivujen laaja tuki ja suuri määrä eri kirjastoja helpottavat ohjelmiston kehitystä huomattavasti. /7/

B4A:ta voi kokeilla ilmaiseksi 30 päivän ajan ja siinä on projektin kokorajoitus, eli 30 päivän jälkeen tuote tulee ostaa. Opinnäytetyön kirjoitushetkellä B4A Standard -version hinta oli 59 USD ja siihen sisältyy kahden kuukauden ilmaiset päivitykset. Tähän opinnäytetyöhön Mäkelä Alu Oy hankki B4A-työkalun Standard-version. /8/

3.3 XML

XML (Extensible Markup Language) on merkintäkieli samoin kuin HTML. Sen tarkoitus on pitää tietoa sisällään ja siirtää sitä. XML tallentaa tiedon tekstimuotoon, joten se on ohjelmisto- ja laitteistoriippumaton tapa tallentaa ja jakaa tietoa. XML:n avulla monet ihmiset ja laitteet voivat yksinkertaisesti lukea näitä tietoja. /9/

3.4 ADB

ADB eli Android Debug Bridge on komentokehotetyökalu jonka avulla Android-laite tai emulaattori voi kommunikoida tietokoneen kanssa. /10/

3.5 MigLayout

MigLayout on ulkoasun luomiseen tarkoitettu kirjasto Javalle. Sen avulla voidaan toteuttaa ulkoasukokonaisuuksia, jotka olisivat vaikeita tai laitteistolle raskaita toteuttaa Javan omilla ulkoasukirjastoilla. /11/

4 SOVELLUSTEN MÄÄRITTELY JA SUUNNITTELU

4.1 Vaatimusmäärittely

Ennen sovelluksen valmistumista sille määriteltiin vaatimukset, jotka tulisi toteuttaa. Tässä opinnäytetyössä kaikki vaatimukset saatiin toteutettua. Taulukossa on esitelty työlle asetetut vaatimustenmäärittelyt (**Taulukko 1.**).

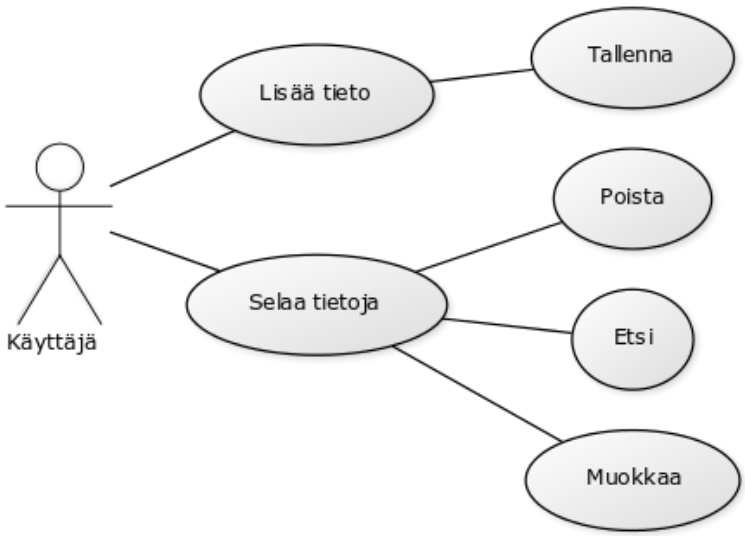
Taulukko 1. Vaatimusmäärittely.

Vaaditut
• Kaksi eri sovellusta: Android-sovellus B4A kehitystyökalulla ja Windows työpöytäsovellus Javalla
• Android-sovelluksella luetaan/kirjataan tuotteiden viivakoodit ja tiedot tallennetaan XML tiedostoon
• Windows sovelluksella siirretään puhelimesta XML tiedostot tietokoneelle
Odotetut
• Nopeat, toimivat ja yhtenäiset käyttöliittymät
• Viivakoodin luku puhelimen kameralla
• Kuvien ottaminen puhelimen kameralla
Lisätoiminnot
• Android: Jos lisätään jo olemassa oleva viivakoodi, ohjelma huomauttaa käyttäjää vain muokkaamaan jo lisätyn viivakoodin kappalemäärään oikeaksi

4.2 Käyttötapauskaavio

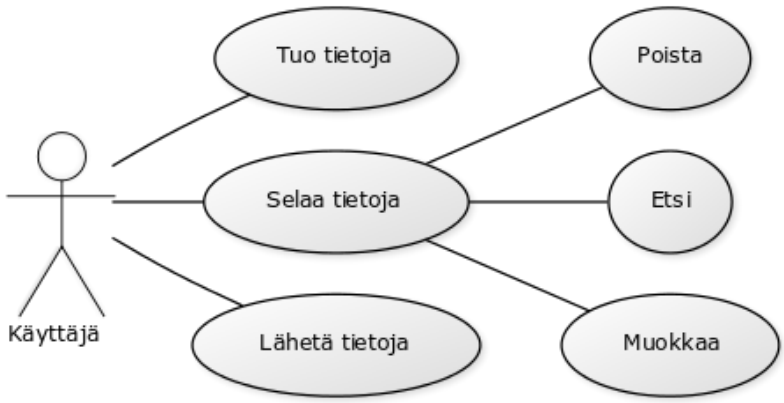
Käyttötapauskaavion tarkoitus on esittää sovelluksen keskeiset toiminnot helposti ymmärrettävässä muodossa. Koska tässä työssä on kaksi eri sovellusta, on myös käyttötapakaavioita kaksi.

Android-sovelluksessa toimijana on sovelluksen käyttäjä, jonka käyttötapaukset ovat tietojen lisääminen ja tietojen selaaminen. Tiedon lisääminen johtaa tiedon tallentamiseen, kun taas tietoja selatessa voi joko poistaa tiedon, etsiä haluamaansa tietoa tai muokata sitä (**Kuvio 1**.).



Kuvio 1. Android-sovelluksen käyttötapauskaavio.

Windows-sovellus toimii pitkälti samoin kuin Android-sovellus, siinäkin toimijana on käyttäjä, ja hänen käyttötapauksiinsa kuuluu tietojen muokkaaminen ja poistaminen, sekä tietojen tuominen puhelimesta Windows-sovellukseen ja päinvastoin, lähettäminen puhelimeen (**Kuvio 2**.).



Kuvio 2. Windows-sovelluksen käyttötapauskaavio.

4.3 Käytettyjen tekniikoiden valinta

Seuraavaksi käydään läpi millä perusteilla käytetyt tekniikat valittiin, sekä niiden hyötyjä ja haittoja.

Android-puolen inventaariosovellus on käyttäjälle työkalu, eli siltä vaaditaan nopeaa toimintaa ja yksinkertaista ulkoasua, jotta maksimaalinen hyöty saavutetaan. Sovelluskehitystyökaluksi valittiin B4A, koska sillä on helppo ja nopea luoda sovelluksia. Erityisesti graafinen ulkoasun luominen ja B4A Bridgen käyttö nopeuttavat sovelluksen tekemistä huomattavasti.

Windows-sovellus on monipuolisempi kuin Android-sovellus, eli se on raskaampi ja hiukan vaikeampi käyttää. Ohjelmointikieleksi valittiin Java, koska siitä on aikaisempaa kokemusta ja tukea löytyy paljon monista eri lähteistä.

Sovellukset kommunikoivat keskenään käyttäen ADB-yhteyttä, ja se toimii tässä sovelluksessa vain USB-kaapelilla. Sen hyvä puoli on varmuus, verrattuna vaikkapa langattomaan tiedonsiirtoon, ja haittapuolena on käyttömukavuuden laskeminen.

Itse tiedot, joita siirretään, ovat XML-tiedostoja. XML:n etu on helppo luettavuus, eli kuka tahansa voi lukea ja ymmärtää XML-tiedostoja. Suurin syy XML-tiedostorakenteen valintaan on kuitenkin mahdollinen myöhempi sovelluksen laa-

jentaminen yrityksessä jo olemassa oleviin tiedostoihin, ja juuri siksi XML-tiedostot sopivat tähän työhön paremmin kuin vaikkapa SQL-tietokanta.

4.4 Käyttöliittymän suunnittelu

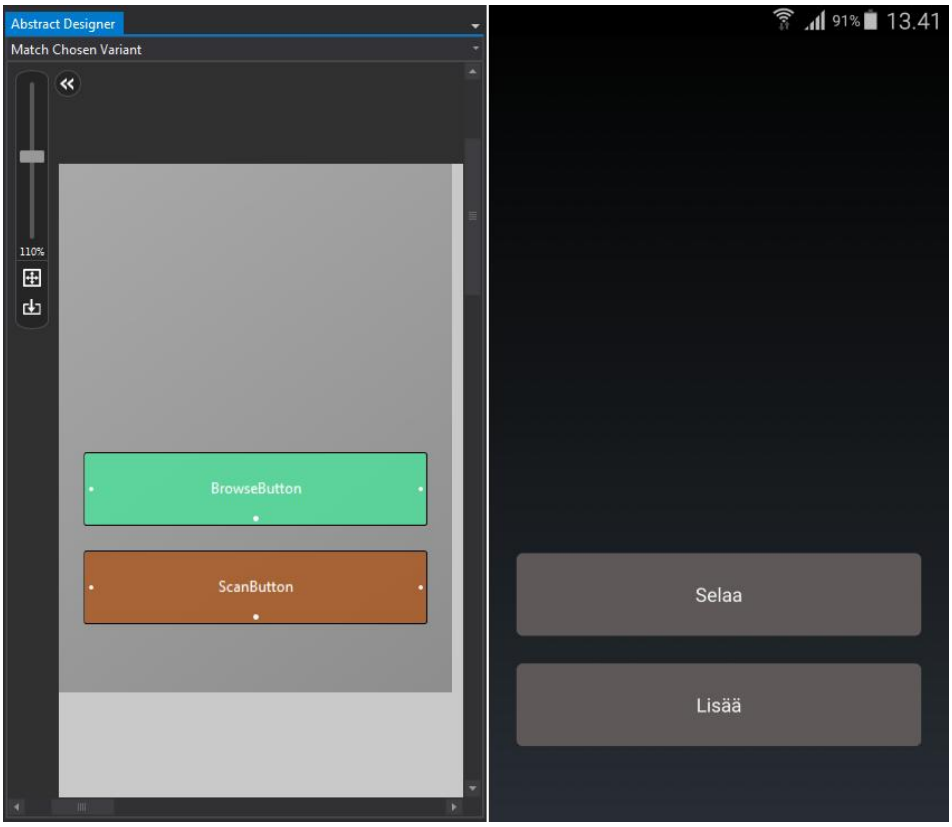
Käyttöliittymän suunnittelu aloitettiin määrittelemällä, mitä tarkalleen sovelluksilta vaaditaan. Android-sovellukselta vaaditaan helppokäyttöisyyttä ja nopeutta, sama pätee Windows-sovellukseen, tosin siinä on enemmän tietojen muokkausmahdollisuuksia.

4.5 Android-sovelluksen suunnittelu

Suunnitteluvaiheessa sovelluksen tärkein osa oli yksinkertainen käyttöliittymä, joka on myös nopea. Kun sovellus käynnistetään, avautuu päävalikko. Siinä on yksinkertaisesti Selaa- ja Lisää -napit. Valikoiden suunnittelu eli mockup tehtiin B4A:n Visual Designerilla.

4.5.1 Päävalikko

Android-sovelluksen päävalikossa käyttäjä pystyy päättämään, haluaako hän selata olemassa olevia tuotteita vai lisätä uuden tuotteen.

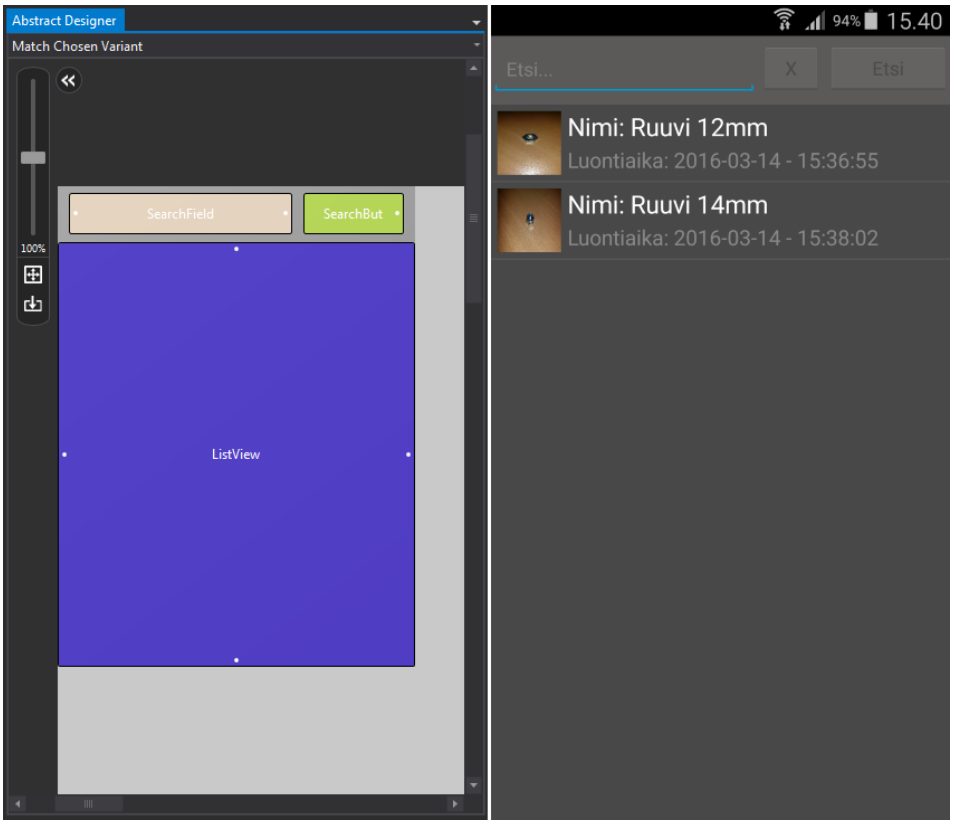


Kuvio 3. Android-sovelluksen päävalikko.

Vasemmalla on päävalikon alkuperäinen suunnitelma ja oikealla itse lopullinen näkymä (**Kuvio 3.**). Lopullinen toteutus ei siis ole muuttunut suunnitelmasta ollenkaan. Päävalikon yläosaan on jätetty tyhjää tilaa mahdollista sovelluksen laajentamista varten.

4.5.2 Selaa-valikko

Selaa-valikon tarkoitus on näyttää käyttäjälle puhelimessa olevat tuotteet listassa. Käyttäjä voi myös hakea, muokata tai poistaa lisäämiään tuotetietoja.

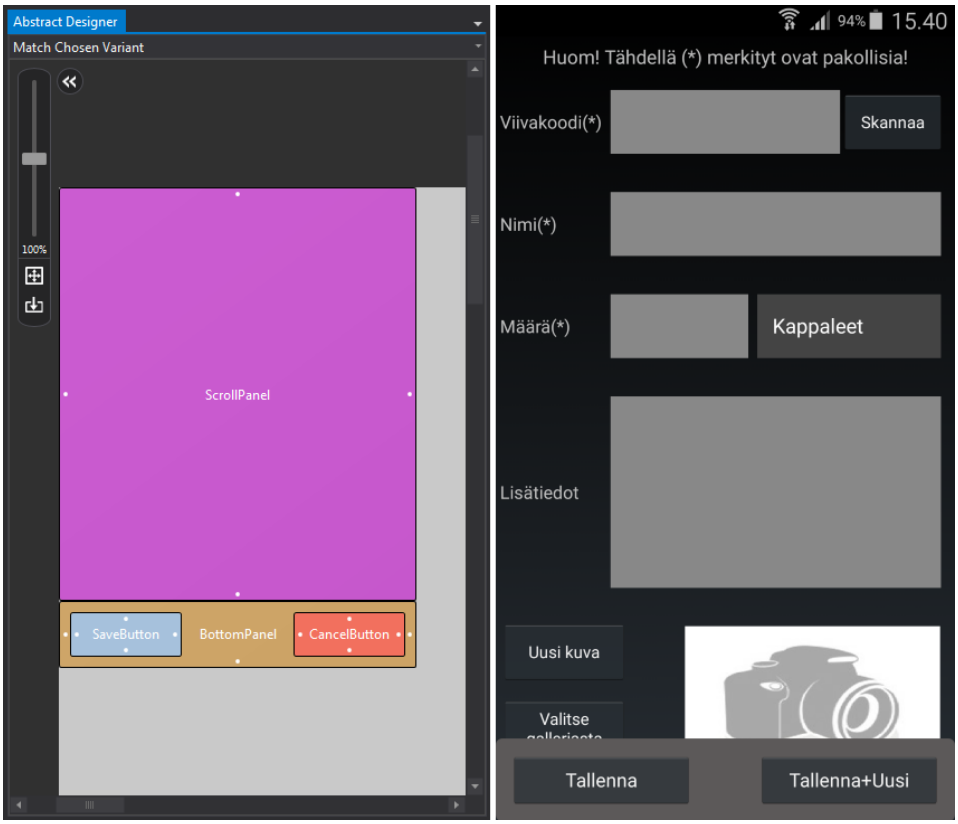


Kuvio 4. Android-sovelluksen Selaa-valikko.

Vasemmalla puolella on alkuperäinen suunnitelma siitä, miltä Selaa-valikko voisi näyttää, ja oikealla on itse valmis sovellus (**Kuvio 4.**). Lopullinen versio säilyi hyvin pienin muutoksin, ainoastaan yläosan etsintäpaneeliin on lisätty X-painike, jolla käyttäjä voi tyhjätä nopeasti tekemänsä haun.

4.5.3 Lisää-valikko

Lisää-valikossa käyttäjä voi lisätä uusia tietoja. Tässä ikkunassa on kaksi pakollista kenttää, Nimi- ja Määrä -kentät. Käyttäjä voi lisätä määrän kappaleina, kiloina, tilavuutena tai pituutena. Käyttäjä voi myös halutessaan ottaa kuvan lisäämälleen tiedolle, valita kuvan puhelimen galleriasta tai poistaa olemassa olevan kuvan.



Kuvio 5. Android-sovelluksen Lisää-valikko.

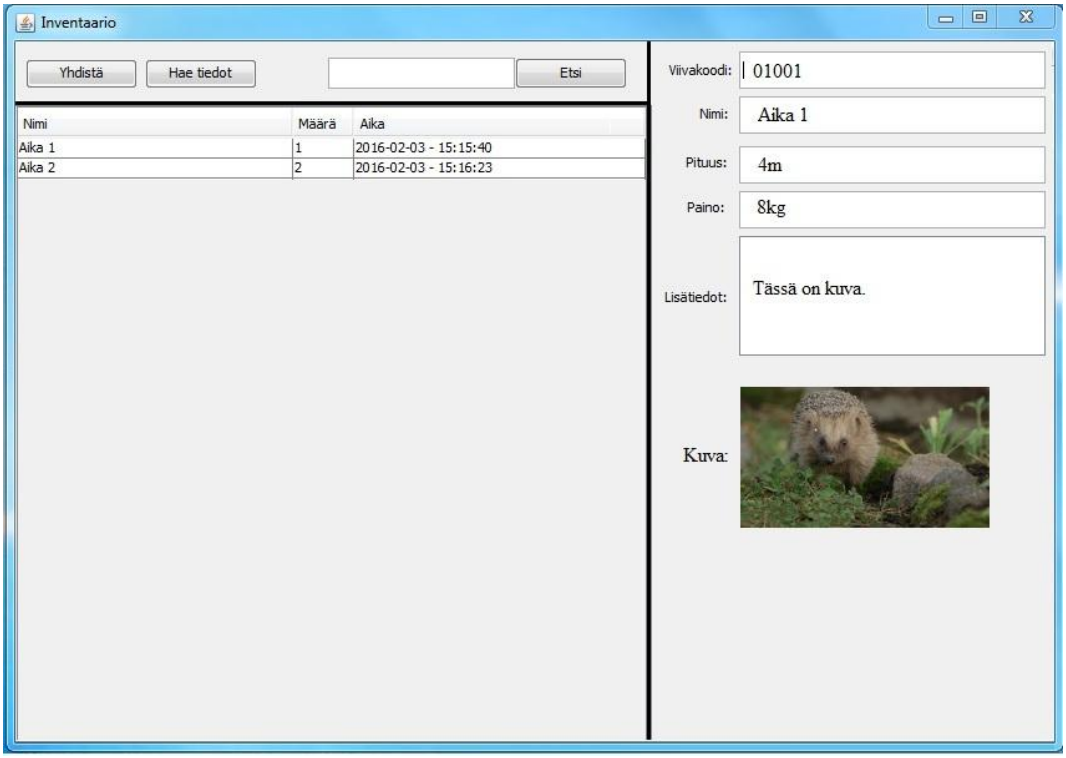
Vasemmalla puolella on alkuperäinen suunnitelma sovelluksen ulkoasusta ja oikealla puolella itse valmis sovellus, lopullinen ulkoasu ei muuttunut paljon suunnitellusta (**Kuvio 5.**). Ainoastaan yläpaneeliin lisättiin tekstiä ja Peruuta-painike vaihdettiin Tallenna+Uusi-painikkeeksi, joka siis tallentaa tiedon jättäen käyttäjän Lisää-valikkoon.

4.6 Windows-sovelluksen suunnittelu

Windows-sovellusta suunniteltaessa päätettiin valita yksinkertainen päänäkymä, jossa suurin osa tiedoista näkyy samassa ikkunassa. Näin käyttäjän ei tarvitse avata useita ikkunoita nähdäkseen haluamansa tiedot, ja sovellusta on huomattavasti mukavampi ja tehokkaampi käyttää.

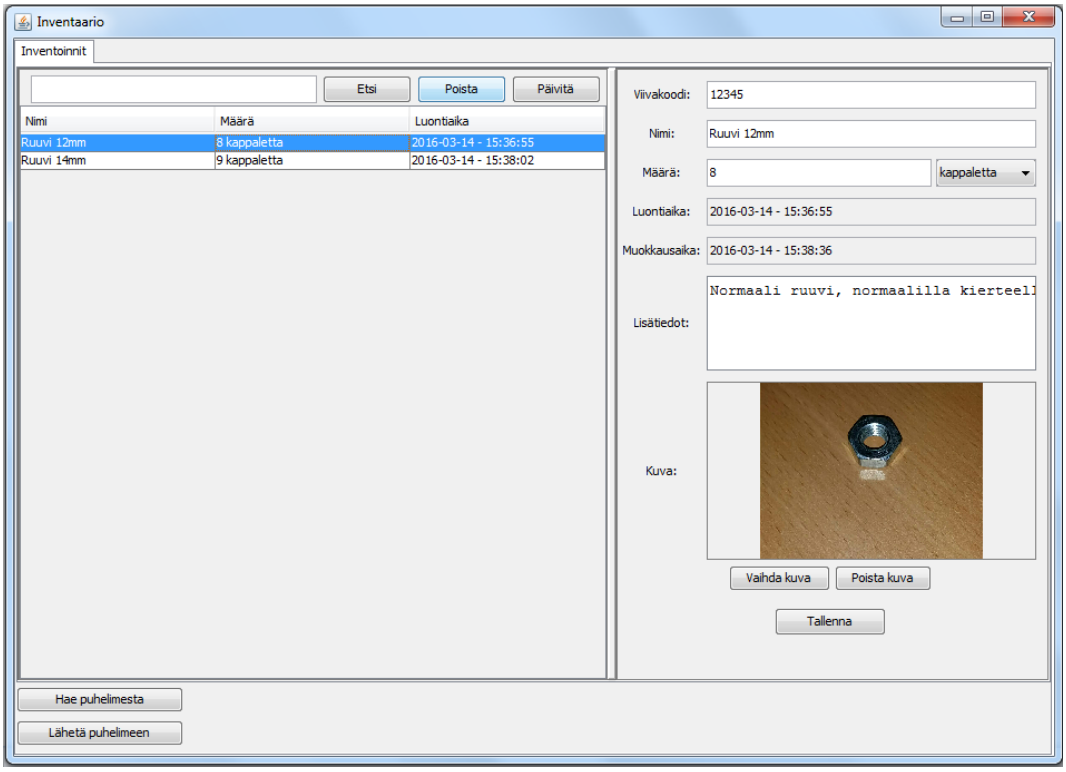
4.6.1 Windows - sovelluksen päänäkymän suunnittelu

Windows-sovelluksen suunnitelma eli mockup tehtiin Netbeansin graafisen ulkoasun luomiseen tarkoitetun JFrame Form -luokan avulla. Suunnitelmaa muokattiin lopuksi Windows Paintin avulla.



Kuvio 6. Windows-sovelluksen suunnitelma.

Alkuperäisen suunnitelman mukaan vasen osa listaa kaikki lisätyt tuotteet ja oikea puoli näyttää valitun tuotteen tarkat tiedot. Tuotetietojen kopioiminen on ikkunassa vasemmalla ylhäällä, jonka oikealla puolella on tuotelistauksesta tuotteiden etsiminen (**Kuvio 6.**).



Kuvio7. Windows-sovelluksen lopputulos.

Sovelluksen lopullinen ulkoasu ei paljon muuttunut suunnitelmasta (**Kuvio 7.**). Suurin muutos suunnitelman ja toteutuksen välillä on tiedonsiirtopainikkeiden vaihto sovelluksen yläosasta sovelluksen alaosaan. Sovellukseen tehtiin myös joitakin lisäyksiä, kuten sovelluksen oikealla puolella sijaitsevaan tuotetietojen tarkastelu -osaan lisättiin Vaihda kuva ja Poista kuva -painikkeet. Myös Tallenna-painike lisättiin mahdollisten muutosten tallennusta varten.

5 SOVELLUSTEN TOTEUTUS

Seuraavaksi käydään läpi molempien sovellusten toteutus ja niiden tärkeimmät osat.

5.1 Windows-sovelluksen toteutus

Sovellusten tekemistä aloitettaessa ensimmäinen ja tärkein asia oli tiedonsiirtoyhteyden muodostaminen. Tämän takia ensimmäisenä tehtiin hyvin pelkistetty Windows-sovellus, joka pystyi vastaanottamaan ja lähettämään tiedostoja puhelimesta ADB-yhteyden avulla. Tämän jälkeen sovelluksen ulkoasua ja käytettävyyttä parannettiin sekä lisättiin toimintoja. Lopulliseen sovellukseen tuli viisi luokkaa (**Taulukko 2.**).

Taulukko 2. Windows-sovelluksen luokat ja niiden tehtävät.

Luokka:	**Luokan tehtävä:**
ADB.java	Käynnistää-ja resetoi -ADB yhteydet.
Communication.java	Kommunikointi Windows-ja Android-sovellusten välillä käyttäen ADB-sovellusta.
GUI.java	Sovelluksen ulkoasu ja muiden luokkien hallinta.
Main.java	Pääluokka, ainoa tehtävä on suorittaa GUI-luokka.
XML.java	XML-tiedostojen luku ja niistä hakeminen.

Windows-sovelluksessa käytettiin vain MigLayout-kirjastoa, jonka avulla luotiin lähes koko sovellus.

5.1.1 Tiedonsiirtoyhteyden toteutus

Windows-sovelluksen keskeinen osa on puhelimen ja tietokoneen välinen tiedonsiirtoyhteys. Kaapelilla toimiva tiedonsiirtoyhteys valittiin, koska se on luotettava ja nopea tapa siirtää tiedot. Tämän sovelluksen laitteiden kommunikointiin valittiin ADB-työkalu, joka on Googlen kehittämä, ja näin ollen tarjoaa laajan yhteensopivuuden Android-laitteille. ADB-yhteyden muodostaminen vaatii puhelimesta USB-virheenkorjauksen päälle asettamisen /10/. Toimiakseen ADB-sovellus vaatii kolme tiedostoa, *adb.exe*, *AdbWinApi.dll* ja *AdbWinUsbApi.dll*, joiden on oltava samassa kansiossa.

Android-laitteiden etsintä tapahtuu komennolla "adb devices", joka tulostaa ADB-sovelluksen luoman sarjanumeron yhdistetyille Android-laitteille. Sen jälkeen on joko laitteet, joissa on hyväksytty USB-virheenkorjaus (device), tai laitteet, joissa USB-virheenkorjausta ei ole hyväksytty (unauthorized). /12/ Kuviossa 8 on kaksi tietokoneeseen yhdistettyä laitetta, joista ylemmässä on hyväksytty USB-virheenkorjaus ja alemmassa ei.

Kuvio 8. Tietokoneeseen yhdistetyt Android-laitteet.

Koska ADB-työkalu on komentokehote-pohjainen, tämän opinnäytetyön Windows-sovellus kommunikoi Windowsin komentokehotteen kautta.

```
String[] command = {"adb", "devices"};

ProcessBuilder probuilder = new ProcessBuilder(command);
Process process = probuilder.start();

InputStream is = process.getInputStream();
InputStreamReader isr = new InputStreamReader(is);
BufferedReader br = new BufferedReader(isr);

//2.Skannataan yhdistetyt tai ei yhdistetyt laitteet
List DevicesList = new List();
String LineRead;
while((LineRead = br.readLine()) != null) {
    if(LineRead.contains("daemon not running")||LineRead.contains("daemon started")
            ||LineRead.contains("List of devices attached")) {
        //Ei lisätä listaan, jos tämä toteutuu.
        //"daemon not running" ja "daemon started" tulevat kun adb käynnistyy ja/tai resetoituu.
        //"List of devices attached" tulostuu aina, sekin ohitetaan.
    } else {
        if(LineRead.trim().contains("unauthorized")) {
            //Lisätään listaan laitteet, joissa EI ole hyväksytty USB-virheenkorjausta.
            DevicesList.add("-Denied-"+LineRead.replace("unauthorized", "").trim());
        }
        if(LineRead.trim().contains("device")) {
            //Lisätään listaan laitteet, joissa ON hyväksytty USB-virheenkorjaus.
            DevicesList.add("-Accepted-"+LineRead.replace("device", "").trim());
        }
    }
}
br.close();
```

Kuvio 9. Laitteiden etsintä ADB-sovelluksen avulla.

Sovellus lukee komentokehotteesta saadut vastaukset rivi kerrallaan ja vain olennaiset tiedot lisätään listaan *DevicesList* (**Kuvio 9.**). *DevicesList*-listaan lisätyt laitteet, joissa on hyväksytty USB-virheenkorjaus, tulostetaan käyttäjälle ja käyttäjä saa valita haluamansa laitteen. Mikäli laitteesta ei ole hyväksytty USB-virheenkorjausta, käyttäjälle aukeaa uusi ikkuna, jossa kerrotaan toimenpiteet mitä hänen tulee tehdä saadakseen yhteyden toimimaan.

5.1.2 Tiedostojen kopioiminen laitteiden välillä

Tiedoston kopioiminen Android-laitteesta tietokoneelle tapahtuu komennolla "adb -s <laitteen sarjanumero> pull <mistä kansio> <mihin kansio>". Toisinpäin tapahtuvan siirron komento on muuten sama, mutta "pull" komennon sijasta käytetään "push" komentoa. /13/

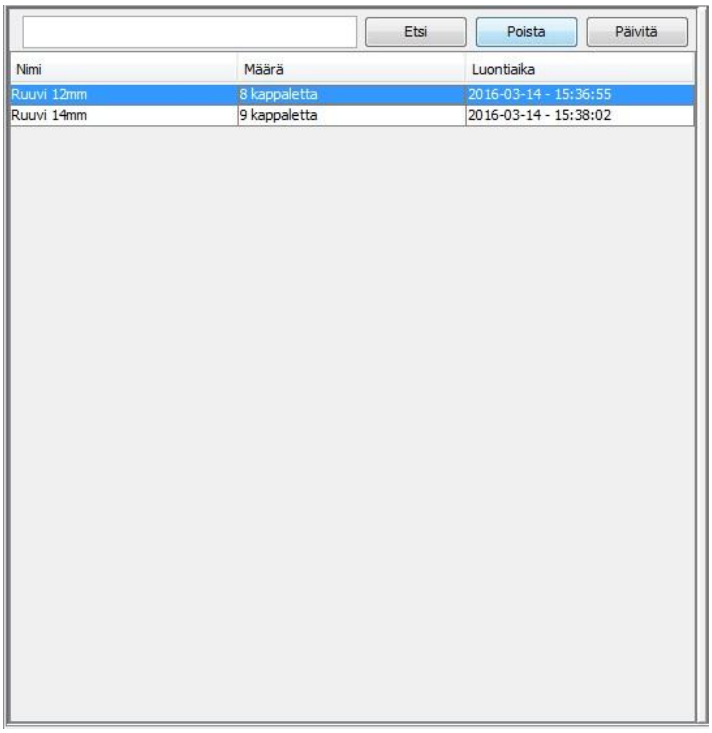
```
String InputFolder = "/sdcard/Android/data/com.b4a.inventaario/files/Barcodes";
String OutputFolder = System.getProperty("user.dir")+"/Puhelimesta/"+cmdReturn.toString().substring(20, 35);

//3. Kopioi tiedostot puhelimesta
ProcessBuilder pb = new ProcessBuilder("adb", "-s", SelectedDeviceID, "pull", InputFolder, OutputFolder);
Process pc = pb.start();
pc.waitFor();
```

Kuvio 10. Tiedoston kopiointi puhelimesta tietokoneelle.

Sovelluksen kaikki komennot ohjataan ADB-komentokehotteeseen *ProcessBuilder* -luokkaa käyttäen, joka tässä tapauksessa kopioi tiedoston puhelimesta tietokoneelle (**Kuvio 10.**).

5.1.3 Sovelluksen ulkoasun luominen

Seuraavaksi käydään läpi MigLayout-kirjaston toimintaperiaate. MigLayout-kirjaston avulla voidaan luoda erilaisia ulkoasuja sovelluksiin.



Kuvio 11. Windows-sovelluksen vasemman paneelin ulkoasu.

Esimerkkinä seuraavaksi tarkastellaan Windows-sovelluksen vasenta paneelia, jossa näkyy kaikki puhelimesta tietokoneelle kopioidut tuotteet (**Kuvio 11.**).

Sovelluksen osat voidaan asettaa joko String-tai API -tyyppisesti koodiin. Tässä sovelluksessa on käytetty String -tapaa.

```
CenterLeftPanel.setLayout(new MigLayout(
        "insets 0, fillx",
        "[fill,grow]5[]",
        "5[]2[fill,grow]"));
```

Kuvio 12. Windows-sovellus, MigLayout-koodi.

Tarkastellaan MigLayout-kirjaston käyttöä rivi kerrallaan (**Kuvio 12.**).

Ensimmäisellä rivillä otetaan käyttöön MigLayout *CenterLeftPanel*-paneelissa eli sovelluksen vasemman puoleisessa osassa.

Toisella rivillä asetetaan ulkoasun yleiset asetukset, **insets 0** poistaa kaikki mahdolliset välit, ja **fillx** täyttää paneelin vaakasuunnassa.

Kolmannella rivillä asetetaan vaakarivien asetukset, tässä tapauksessa [**fill, grow]** täyttää ja kasvattaa ensimmäisen lisätyn osan (*SearchField*-kentän sekä *CenterPanel_Scrollpane*-paneelin) vaakasuunnassa. Numero **5** lisää viiden pikselin kokoisen välin ennen viimeistä osaa **[]**, jossa on kolme painiketta.

Neljännellä rivillä on pystyrivien asetukset. Se lähtee ylhäältä **viiden** pikselin välillä, jonka jälkeen tulee **[],** tässä tyhjässä osassa on *SearchField*, *SearchTableButton*, *RemoveFromTableButton* ja *UpdateTableButton*. Tämä osa voi olla tyhjä, koska näiden neljän osan tarkemmat arvot asetetaan vasta, kun osat lisätään paneeliin (**Kuvio 13**). Seuraavaksi tulee **kahden** pikselin kokoinen väli, jonka jälkeen viimeinen osa [**fill, grow]** täyttää ja kasvattaa osan *CenterPanel_Scrollpane*-paneelin pysty- ja vaakasuuntaisesti.

```
CenterLeftPanel.add(SearchField, "gapleft 10, grow, height 25:25:25");
CenterLeftPanel.add(SearchTableButton, "width 80:80:80, height 25:25:25");
CenterLeftPanel.add(RemoveFromTableButton, "width 80:80:80, height 25:25:25");
CenterLeftPanel.add(UpdateTableButton, "width 80:80:80, height 25:25:25, gapright 5, wrap");
CenterLeftPanel.add(CenterPanel_Scrollpane, "grow, spanx 4");
```

Kuvio 13. Windows-sovellus, MigLayout-osien lisäys.

Seuraavaksi lisätään osat *CenterLeftPanel*-paneeliin eli sovelluksen vasemman puoleiseen osaan. *SearchField* on ylhäällä vasemmalla oleva JTextField-kenttä, jonne käyttäjä voi syöttää haluamansa hakusanan tai kirjaimet. **Gapleft 10** tekee 10 pikselin kokoisen välin ennen *SearchField*-kenttää, **grow** taas kasvattaa kentän sovelluksen ikkunan kokoa muutettaessa.

MigLayout voi pakottaa valitun kentän leveyden, esimerkiksi komennolla **width 80:80:80**, jossa numerot 80:80:80 ovat minimi:suositeltu:maksimi. Koska nämä kaikki arvot ovat samat, on kyseisen kentän koko pakotettu olemaan aina 80 pikseliä.

Neljännellä rivillä oleva *UpdateTableButton*-painikeessa oleva **gapright 5** tekee viiden pikselin kokoisen välin painikkeen jälkeen, ja komento **wrap** aloittaa uuden rivin.

Viidennellä rivillä oleva *CenterPanel_Scrollpane*-paneeli alkaa siis uudelta riviltä, komento **grow** kasvattaa tämän osan vaaka-ja pystysuuntaisesti aina paneelin reunoihin asti. Komento **spanx 4** kasvattaa *CenterPanel_Scrollpane*-paneelin vaakasuunnassa neljän osan mittaiseksi. Mikäli spanx 4 komentoa ei olisi, *CenterPanel_Scrollpane*-paneeli olisi vain *SearchField*-kentän kokoinen vaakasuunnassa.

5.2 Android-sovelluksen toteutus

Kun Windows-sovelluksessa saatiin tiedonsiirtoyhteys toimimaan, oli Android-sovelluksen vuoro, jossa tärkeintä oli tuotetietojen lisäys puhelimen muistiin. Näiden perusasioiden jälkeen alkoi ominaisuuksien lisäys, aluksi Android-sovellukseen ja myöhemmin Windows-sovellukseen. Android-sovelluksen kehitysvaiheen muutoksia testattiin fyysisellä puhelimella.

Android-sovelluksessa on neljä eri luokkaa ja viisi eri ulkoasutiedostoa, jotka on luotu B4A Visual Designerilla (**Taulukko 3.**).

Taulukko 3. Android-sovelluksen luokat ja niiden tehtävät.

Luokka:	**Luokan tehtävä:**
AddProduct	Lisää-valikko, käyttäjä voi lisätä uuden tuotetiedon.
Main	Päänäkymä, vaihtoehdot siirtyä Selaa (ShowProducts) tai Lisää (AddProduct) luokkaan.
ShowProducts	Selaa-valikko, näyttää lisättyjen tuotteiden tiedot.
TakePicture	Lisää-valikossa oleva kuvanotto luokka.

Ulkoasutiedosto:	**Tiedoston tehtävä:**
addproductdesign.bal	Lisää-valikon ulkoasu.
checkpicture.bal	Varmistaa käyttäjältä onko otettu kuva onnistunut.
main.bal	Päänäkymän ulkoasu.
showproducts-design.bal	Selaa-valikon ulkoasu.
takepicture.bal	Kuvan ottamisen ulkoasu.

B4A tarjoaa useita hyödyllisiä kirjastoja, ja tässä työssä niitä oli käytössä kymmenen (**Taulukko 4.**). Kirjastot ovat saatavilla vain B4A:n ostaneille. /14/

Taulukko 4. B4A-kirjastot.

Nimi:	**Käyttö tässä työssä:**
ABZing	Viivakoodien luku
ACL	Kamera, kuvien ottaminen
BitmapExtended	Bitmapin kääntäminen, jos kuva otettu pysty asennossa
ContentResolver	Selkeyttää polun kuvan valinnan yhteydessä
IME	Näppäimistön piilottaminen
Phone	Näytön orientaation pakotus, sekä tiedoston uudelleen nimeäminen
RSImageProcessing	Pikkukuvien luominen
SQL	Kursorin käyttö polun selvittämisessä kuvan valinnan yhteydessä
XMLBuilder	XML-tiedostojen kirjoitus
XmlSax	XML-tiedostojen luku

5.2.1 XML-tiedostorakenne ja tiedoston luonti

Keskeisin osa Android-sovelluksella on lisätä uusi tuote ja käyttäjän mahdollisesti haluama kuva tuotteelle. Lisätylle tuotteelle luodaan aina oma XML-tiedosto, jossa on käyttäjän syöttämät tiedot, aikaleimat, sekä puhelimen IMEI-laitetunnus. IMEI (International Mobile Equipment Identity) on Android-laitteen uniikki sarjanumero /15/. Tässä työssä IMEI-tunnuksen avulla saadaan valitut tiedot kopioitua valittuun laitteeseen.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Information>
    <Barcode>12345</Barcode>
    <Name>Ruuvi 12mm</Name>
    <Amount unit="kpl" value="8" />
    <AdditionalInfo>Normaali ruuvi, normaalilla kierteellä.</AdditionalInfo>
    <CreatedTime>2016-03-14 - 15:36:55</CreatedTime>
    <ModifiedTime>2016-03-23 - 16:06:59</ModifiedTime>
    <Filename>2016-03-14_15-36-55_123456789012345</Filename>
    <PhotoTime>2016-03-14 - 15:36:55</PhotoTime>
  - <PhoneID>
      <IMEI>358672059533426</IMEI>
    </PhoneID>
  </Information>
```

Kuvio 14. XML-tiedostorakenne.

XML-tiedoston tiedostonimi on aina lisäysaika_IMEI-tunnus, tässä esimerkissä se on *2016-03-14_15-36-55_123456789012345.xml* (**Kuvio 14.**).

```
'Luodaan XML tiedosto
XmlBuilder = XmlBuilder.create("Information")
XmlBuilder = XmlBuilder.element("PhoneID") _
.element("IMEI").text(PhoneIMEI.GetDeviceId).up()

props.Put("{http://xml.apache.org/xslt}indent-amount", "4")
props.Put("indent", "yes")
File.WriteString(File.DirDefaultExternal & "/Barcodes", _
TimeNowFilename&"_"&PhoneIMEI.GetDeviceId&".xml", XmlBuilder.asString2(props))
```

Kuvio 15. XML-tallennus.

Esimerkkinä XML-tiedoston luomisesta käytetään IMEI-koodin tallentamista XML-tiedostoon (**Kuvio 15.**). *XmlBuilder.create("Information")* luo tiedoston pääelementin, jonka sisälle loput elementit tulevat. *XmlBuilder.element("PhoneID")* taas luo PhoneID-elementin.

XML-tiedoston ulkoasu annetaan properties-luokan avulla, eli riveillä *props.Put("{http://xml.apache.org/xslt}indent-amount", "4")* ja *props.Put("indent", "yes")*, nämä luovat XML-tiedostoille ominaisen helposti luettavan ulkoasun (**Kuvio 14.**).

Lopuksi tiedostolle annetaan nimi ja se kirjoitetaan puhelimen muistiin käyttäen *File.WriteString*-luokkaa.

5.2.2 B4A pikkukuvien luonti

Android-sovelluksella otetut tai lisätyt kuvat ovat usein kooltaan usean megatavun kokoisia, ja esimerkiksi Selaa -valikossa useamman suuren kuvan lisääminen tiedon vasempaan reunaan täyttää puhelimen muistia, ja lopulta tuloksena voi olla sovelluksen kaatuminen.

Tähän ongelmaan ratkaisuksi tuli pikkukuvien eli tuttavallisemmin thumbnailien teko. Pikkukuvat luodaan vasta kun käyttäjä painaa Tallenna-painiketta tiedon lisäys- tai muokkausvaiheessa.

B4A:sta ei kuitenkaan raportin kirjoitushetkellä löytynyt valmista kirjastoa, jolla pikkukuvien luonti onnistuisi helposti, joten se täytyi tehdä itse.

Pikkukuville valittiin oletusleveydeksi 200 pikseliä, mikä tarkoittaa, että korkeus täytyi laskea alkuperäisestä kuvasta, jotta pikkukuvien kuvasuhde pysyisi oikeana.

Käytetään esimerkkinä kuvaa, jonka leveys on 1920 pikseliä ja korkeus 1080 pikseliä, tarkoituksena on siis saada pikkukuva, jonka kuvasuhde on oikea. Ensimmäisenä täytyy laskea alkuperäisen kuvan kuvasuhde, joka saadaan kaavalla (1)

$$Kuvasuhde = \frac{1920\,(\text{alkuper äinen leveys})}{1080\,(\text{alkuper äinen korkeus})} = 1{,}77 \qquad (1)$$

Tästä voidaan laskea kuvan korkeus kaavalla (2)

$$Pikkukuvan\ korkeus = \frac{200\,(\text{pikkukuvan leveys})}{1{,}77\,(\text{kuvasuhde})} = 113 \qquad (2)$$

Näin ollen pikkukuvan kooksi tulee 200 pikseliä * 113 pikseliä.

Pikkukuvien luomiseen käytetään kirjastoa nimeltä RSImageProcessing. Tässä kirjastossa on komento writeBitmapToFile, joka tallentaa Bitmap muodossa olevan kuvan uudeksi tiedostoksi ja asettaa sille määrätyn leveyden ja korkeuden sekä kuvan laadun 0 % - 100 %. Uuden kuvan nimeksi tulee TempImage_small.png ja laaduksi valittiin 30 %, alkuperäisen kuvan laatu on luonnollisesti 100 % (**Kuvio 16**).

```
Sub CreateThumbnail
    Dim PictureRatio As Double
    Dim ThumbBitmap As Bitmap
    Dim ThumbWidth As Double = 200
    Dim ThumbHeight As Double
    Dim r As RSImageProcessing

    r.Initialize
    ThumbBitmap = LoadBitmap(File.DirDefaultExternal, "Pictures/TempImage.jpg")

    PictureRatio = NumberFormat(ThumbBitmap.Width/ThumbBitmap.Height, 0, 2)

    ThumbHeight = NumberFormat(ThumbWidth/PictureRatio,0,0)
    r.writeBitmapToFile(r.createScaledBitmap(ThumbBitmap,ThumbWidth,ThumbHeight,False), _
    File.DirDefaultExternal&"/Pictures", "TempImage_small.png",30)
End Sub
```

Kuvio 16. Android-sovelluksen pikkukuvien luonti.

6 OHJELMISTOJEN TESTAUS

Sovellusten testauksella on tarkoitus selvittää täyttääkö sovellukset niille asetetut kriteerit sekä mahdollisten virheiden paikallistaminen ja niiden korjaaminen. Tässä opinnäytetyössä testausta tehtiin koko kehityksen ajan, ja löydetyt virheet korjattiin välittömästi. Aina kun uusi ominaisuus lisättiin, sitä testattiin useaan otteeseen. Tärkein testaaja oli opinnäytetyön antaja, tietohallintopäällikkö Risto Niskakangas Mäkelä Alulta, joka myös määritteli sen, mitä sovellukselta vaaditaan. Lisäksi testauksessa oli apuna myös kaksi henkilöä, joille tietotekniikka ei ole kovinkaan tuttua, näin saatiin hyvää käytännön palautetta ja parannusehdotuksia.

Windows-sovellusta testattiin kahdella eri tietokoneella, ensimmäinen oli pöytäkone, jolla myös sovelluksen kehitys tapahtui ja toinen oli kannettava tietokone. Molemmissa koneissa oli asennettuna Windows 7-käyttöjärjestelmä.

Android-sovellusta testattiin kolmella fyysisellä Android-laitteella, joista kaksi oli 5.2" puhelimia ja yksi 10.1" tabletti. Puhelimissa käyttöjärjestelmän versio oli Android 5.1, ja tabletissa versio 4.4.

Sovellusten ominaisuudet saatiin testattua hyvin, ja viimeisellä testauskerralla virheitä ei enää löytynyt. Testaajien palaute oli pääosin positiivista, vaikkakin joitakin osia olisi voinut yksinkertaistaa käyttäjien näkökulmasta. Nämä parannukset voidaan tehdä mahdollisessa sovellusten jatkokehityksessä.

Sovelluksia tullaan vielä testaamaan Mäkelä Alu Oy:ssä ja siellä päätetään lopullisesti otetaanko sovellukset käyttöön vai ei.

7 YHTEENVETO

Tämän opinnäytetyön tuloksena syntyi kaksi toimivaa sovellusta, B4A -ohjelmointityökalulla tehty Android-sovellus, sekä Javalla tehty Windows-sovellus, jotka kommunikoivat keskenään ja ovat helppoja käyttää. Sovelluksille asetetut pakolliset ominaisuudet saatiin toteutettua, myös muutama matalamman prioriteetin ominaisuus.

Haastetta päättötyön tekemiseen toi kahden sovelluksen valmistaminen ja yhdistäminen aikataulun puitteissa. Huomattavasti enemmän aikaa vei Android-sovelluksen tekeminen, koska B4A-kehitystyökalu oli entuudestaan vieras, samoin sen käyttämä Basic-ohjelmointikieli. Lisäksi Android-sovellus oli laajempi sisällöltään. B4A käyttö tuli kuitenkin nopeaa tutuksi, samoin kuin Basic-kieli, mutta aika-ajoin eteen tuli haastavia ongelmia, joiden ratkaisussa vierähti rutkasti aikaa. Kokonaisuudessaan työn tekeminen oli siis opettavainen kokemus, jonka aikana omatkin taidot paranivat.

Windows-sovelluksen kehittämisessä ei suurempia haasteita ollut, koska Java-ohjelmointikieli ja käytössä ollut Netbeans-kehitystyökalu olivat jo entuudestaan tuttuja.

Sovelluksista tuli toimivat, vaikkakin todelliseen käyttöönottoon vaaditaan vielä jatkokehitystä. Varsinkin Windows-sovellukseen olisi tarkoitus lisätä ominaisuuksia, tärkeimpänä tietojen tulostus paperille tai vaihtoehtoisesti vienti Word-tiedostoksi, josta käyttäjän olisi näppärä tulostaa haluamansa raportti tuotteista. Myös Android-sovellukseen olisi hyvä saada joustavuutta, esimerkiksi sovellukseen osien lisääminen tai käyttötarkoituksen muokkaus vaatii koodiin paikoitellen suuriakin muutoksia. Jatkokehitys riippuu kuitenkin siitä, missä laajuudessa sovellukset otetaan käyttöön Mäkelä Alu Oy:ssa.

LÄHTEET

/1/ L 30.12.1997/1336. Kirjanpitolaki. Säädös säädöstietopankki Finlexin sivuilla. Viitattu 6.4.2016. http://www.finlex.fi/fi/laki/ajantasa/1997/19971336#L1

/2/ Yleistä tietoa kirjanpitoon liittyvistä termeistä. Viitattu 14.11.2015. http://kirjanpitosi.fi/hyva-tietaa

/3/ Varaston inventointi. Taloushallintoliitto. Viitattu 18.11.2015. https://taloushallintoliitto.fi/kirjanpidon-abc-mita-jokaisen-tulisi-tietaa-kirjanpidosta/tilikausi-ja-tilinpaatos/varaston

/4/ Learn About Java Technology. Viitattu 2.1.2016. https://www.java.com/en/about/

/5/ Oracle and Java. Viitattu 2.1.2016. https://www.oracle.com/sun/index.html

/6/ B4A, General Features. Viitattu 2.2.2016. http://www.basic4ppc.com/android/why.html

/7/ B4A, The simplest way to develop real-world, native Android apps! Viitattu 16.2.2016. https://www.b4x.com/b4a.html

/8/ B4A store. Viitattu 19.3.2016. https://www.b4x.com/store.html

/9/ Introduction to XML. Viitattu 5.2.2016.http://www.w3schools.com/xml/xml_whatis.asp

/10/ ADB, Android Debug Bridge. Viitattu 28.3.2016.http://developer.android.com/tools/help/adb.html

/11/ MigLayout, Java Layout Manager for Swing, SWT and JavaFX 2. Viitattu 28.3.2016.http://www.miglayout.com/

/12/ ADB, Directing Commands to a Specific Emulator/Device Instance. Viitattu 5.4.2016. http://developer.android.com/tools/help/adb.html#directingcommands

/13/ ADB, Commands. Viitattu 5.4.2016. http://developer.android.com/tools/help/adb.html#commandsummary

/14/ B4A, Can't download libraries. Viitattu 5.4.2016. https://www.b4x.com/android/forum/threads/cant-download-libraries.6826/

/15/ What is IMEI? Viitattu 6.4.2016. http://www.imei.info/faq-what-is-IMEI/