

Nguyen Viet Khoa

WAREHOUSE MANAGEMENT APPLICATION FOR ANDROID

WAREHOUSE MANAGEMENT APPLICATION FOR ANDROID

Nguyen Viet Khoa
Bachelor's Thesis
Spring 2017
Information Technology
Oulu University of Applied
Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Nguyen Viet Khoa

Title of Bachelor's thesis: Warehouse Management Application for Android

Supervisor: Kari Laitinen

Term and year of completion: Spring 2017

Number of pages: 50

This Bachelor's Thesis was about the process of fully developing a native application that runs on the Android platform. The objective of this Thesis was to demonstrate the steps, tools, and Android fundamentals that are needed to make an idea of an Android application become reality. After the theory about the Android development was covered, it was put into practice by implementing a fully functional application. This was a warehouse management system (WMS) which runs on the Android platform. The aim of this application was not only to speed up the work of a warehouse manager, but also to minimize their mistakes by replacing pen and paper with an application that runs on a handheld device such as a tablet.

This application was built by the author and one of his friends when they worked for a company for their training project. This app was built using the knowledge the author acquired from his degree together with his own research about the Android development. The task is to build the front-end of the application as well as the logic behind it. The server side part of this application was made by the author's friend. Therefore, only the developing process of front-end functions of the application is described in this Thesis. This application was built using Android Studio.

As a result, this application works very well and fulfills every requirement. It still has many potential functions that could be implemented to be more effective and up to date, such as the UI customization and performance optimization.

Keywords:

Android, mobile application, warehouse management system, java, xml, wms, front-end, Android development

TERMS AND ABBREVIATIONS

WMS	Warehouse Management System
XML	Extensible Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
OOP	Object Oriented Programming
UI	User Interface
SDK	Software Development Kit

CONTENTS

ABSTRACT.....	3
TERMS AND ABBREVIATIONS.....	4
CONTENTS.....	5
1 INTRODUCTION.....	7
2 PART 1: ANDROID DEVELOPMENT.....	8
2.1 Tools.....	8
2.1.1 Android Studio.....	8
2.1.2 Cordova.....	10
2.1.3 GameMaker: Studio.....	11
2.2 Android Application Fundamentals.....	11
2.2.1 App Components.....	12
2.2.2 Manifest File.....	13
2.2.3 App Resources.....	14
2.3 Designing application features.....	14
2.4 Implementing and Testing.....	15
2.5 APK file.....	17
3 PART 2: WAREHOUSE MANAGEMENT APPLICATION.....	18
3.1 Introduction.....	18
3.2 Data Classes.....	21
3.3 Activities.....	23
3.3.1 Log in / Register.....	23
3.3.2 Main Activity.....	24
3.3.3 Customer.....	26
3.3.4 Quotation.....	32
3.3.5 Order.....	34
3.3.6 Warehouse.....	36
3.3.7 Sale report.....	38
3.4 Array Adapters.....	40
3.5 Data Interface Class.....	44
3.6 Network class.....	45

3.7	Message box and Loading Popup	46
3.8	Testing.....	48
3.9	Publishing.....	49
4	CONCLUSION	50
	REFERENCES.....	51

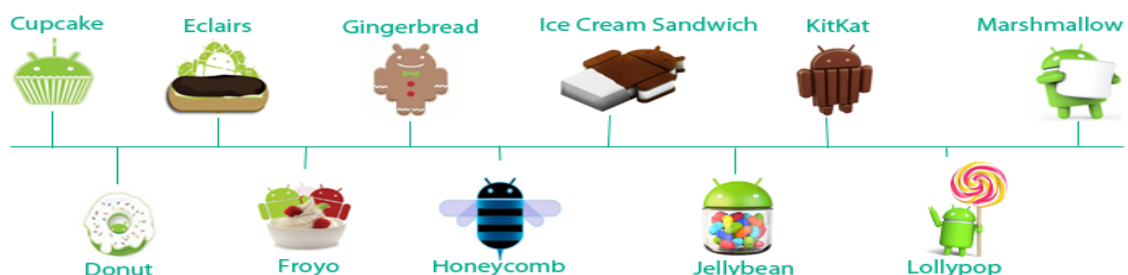
1 INTRODUCTION

Android is a mobile operating system developed by Google in the Android Open Source Project. Android is open source and it is based on the Linux kernel. Android was first developed by Android, Inc., which was bought by Google in 2005.

Android was first designed primarily for mobile phones. Later on, Android was also developed for a number of other electronic devices with a specialized UI. For example: Android Auto for cars, Android TV for television and Android Wear for wrist watch... (1)

Android is the best selling operating system on smartphones and tablets, although it has always been trailing by IOS from Apple. The highly customizable nature gives Android an edge over its competitors. Android allows community developers to freely display their creativity. It encouraged a lot of community driven projects to not only offer new updates to older devices that were no longer supported by the original manufacturer; but also added many new features for high-end users.

Android applications are written in the Java programming language using the Android software development kit (SDK)



History of Android Operating System

FIGURE 1. Android operating system versions (2)

2 PART 1: ANDROID DEVELOPMENT

This part is going to demonstrate the process of developing a fully functional application that runs natively on the Android operating system.

2.1 Tools

The first thing to do in order to develop any program is choosing the working environment, which in this case is the Android software development kit (SDK). There are various options for Android developers to choose from. From Android Studio, the official Intergrated Development Environment made by Google, to interesting tools from other third party companies, such as GameMaker:Studio, Unity3D and Cordova. These tools allow developers to do almost anything they can imagine.

2.1.1 Android Studio

(See reference 3)

Android Studio is the official Android Intergrated Development Environment (IDE) from Google. Succeeding Eclipse in 2014, Android Studio is an all around IDE that allows developers to do everything they need without leaving the environment.

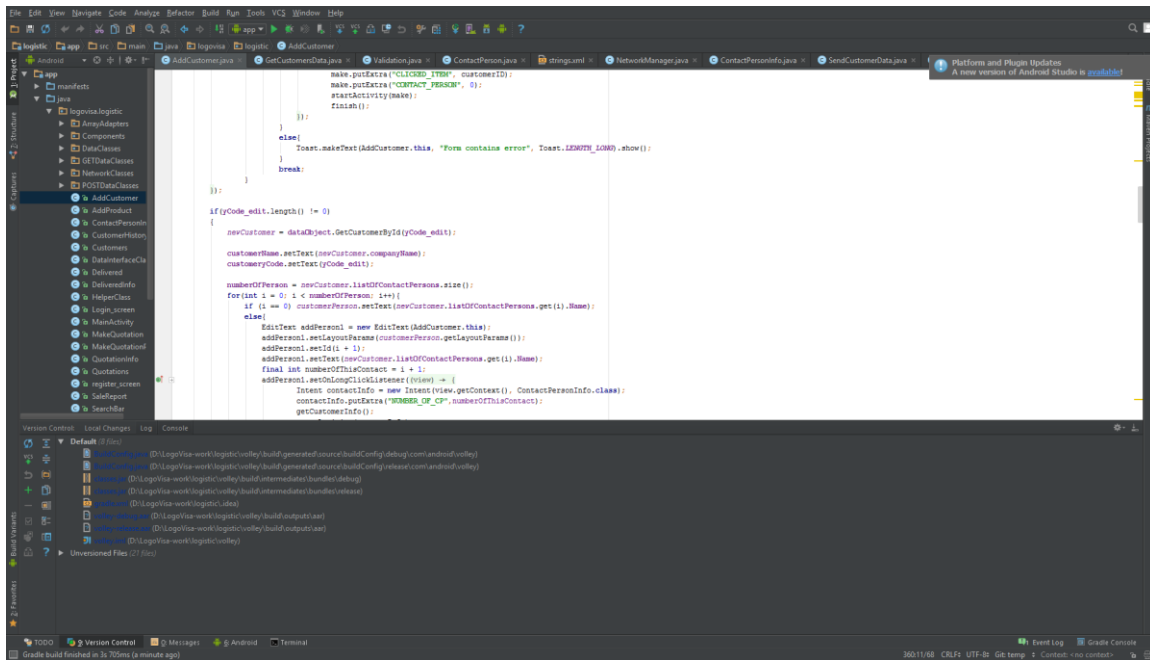


FIGURE 2. Android Studio

Android Studio (FIGURE 2) is based on IntelliJ IDEA, which is a powerful Java code editor. In addition to that, Android Studio also offers several features that a developer would need when building an application, such as:

- Testing tools and frameworks.
- Integrated version control system.
- Lint tools to measure performance, usability and version compatibility.

Android Studio also offers a feature rich simulator to test applications on a virtual device. A developer can customize every aspect of the device, such as Android version, ram and screen size.

Another very useful feature of Android Studio is Android Device Monitor. This allows a developer to monitor plugged in devices and also virtual devices running in the simulator to see how their apps affect the system.

Overall, Android Studio is a very powerful all around IDE to build and test Android apps. It has the best code editor the author has used and very effective tools to test developing apps live on a virtual device or plugged-in device. Android Studio includes everything that is needed to build a fully functional Android application.

2.1.2 Cordova

Cordova was originally a project named Phonegap developed by a startup called Nitobi in 2009. In 2011, the company was acquired by Adobe and Phonegap was donated to Apache Software Foundation. (4)

Cordova is the best mobile application development framework to build a hybrid app. This enables web developers to create a mobile application using web technologies such as HTML, CSS and JavaScript.

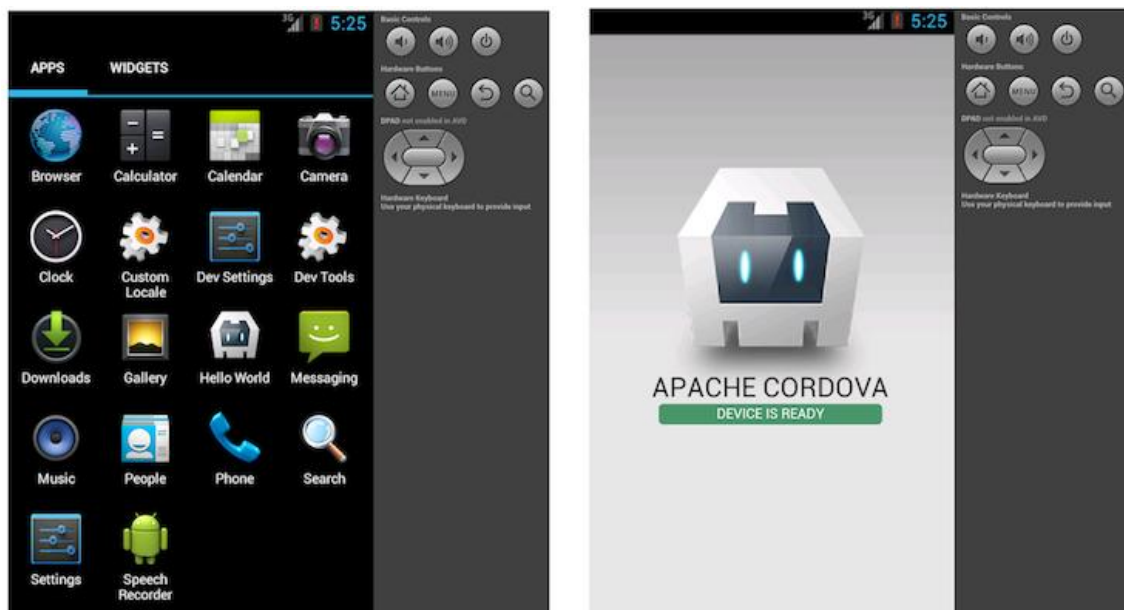


FIGURE 3. Cordova's Android Simulator (5).

Cordova acts like an Android webview. It takes the original code which was made for a web interface and renders it for a mobile user, e.g. a browser. Therefore, Android apps made using Cordova are hybrid apps. Although this will slow down the apps compared to a native development, it enables a developer to build web apps, Android and iOS app, at the same time without putting much effort on migrating their codes.

2.1.3 GameMaker: Studio

GameMaker: Studio is a game engine created by Mark Overmars and developed by YoYo games. (6)

GameMaker: Studio offers people a simple way to make fully customizable Android games. It allows users to create games through its proprietary drag and drop system together with the GameMaker language, which is a very easy way to learn a programming language compared to Java or C++.

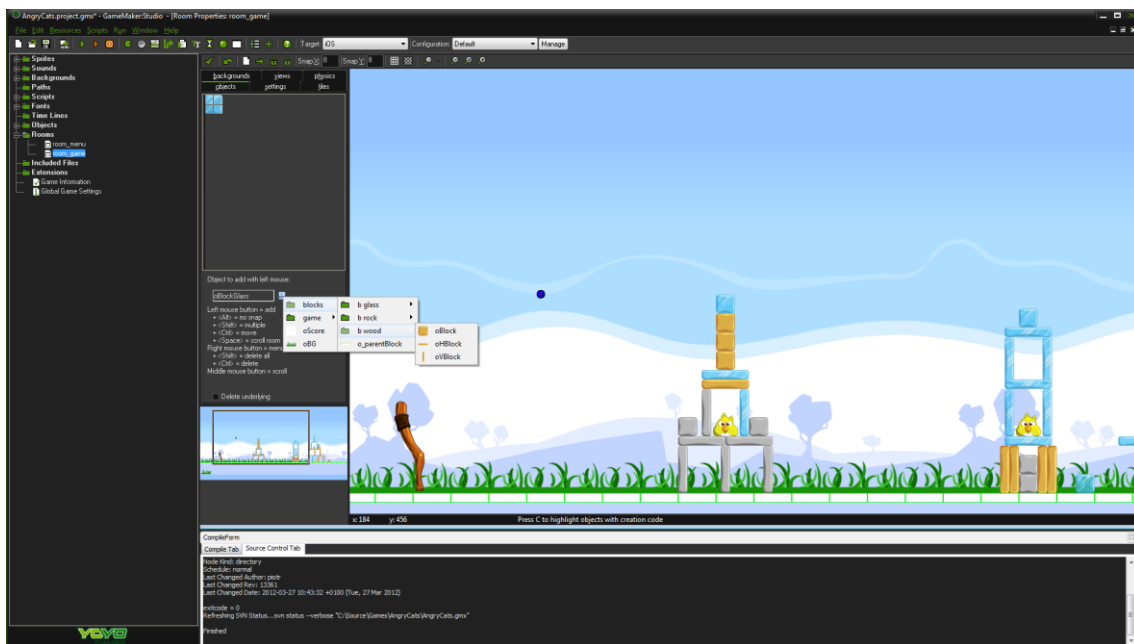


FIGURE 4. GameMaker: Studio (7)

GameMaker: Studio is the easiest way to make an Android 2D game. The tool is not free. The price is \$150 for the tool and \$299 for the Android export module.

2.2 Android Application Fundamentals

After choosing a suitable developing tool, the next thing one has to do to start building an Android application is to understand the way it works.

In order to be fully functional, every Android application has to include 3 parts, which are:

- The core framework components that contain the code of an application.
- The manifest file which contains the declaration of the components and the required device features that the app needs.
- Resources which are separated from the code that the app needs, such as images, videos and audios.

2.2.1 App Components

An Android application is always made of components. They are the essential parts that define an application. There are four types of components:

- **Activities**
Activities are where the user interacts with an application. One activity is one single screen that is showed to the user. There is only 1 activity running at a time in an application. As the system will also keep track of the previous used activity, the user may return to it. An activity is implemented as a subclass of the Activity class.
- **Services**
Services are components that run in the background without the user interface. For example, a service might play music or fetch data over the network in the background while the user is in a different app. A service can be started by an activity and it will keep running until its work is completed. A service is implemented as a subclass of the Service class.
- **Broadcast receivers**
Broadcast receivers are the components that allow applications to receive event announcements from the system. There are several kinds of broadcasts from the system, such as a low battery, time, a service is finished. This component enables the system to send an event broadcast to apps that are not even running. A broadcast receiver may display a status bar to notify the user about the event. A broadcast receiver is implemented as a subclass of the BroadcastReceiver class.
- **Content providers**

Content providers manage the app's data that is stored into the file system or other storages that the app has access to. Content providers can be used as an entry point to share data between apps or to read and write private data of an app. The content provider is implemented as a subclass of the ContentProvider class. (FIGURE 5)

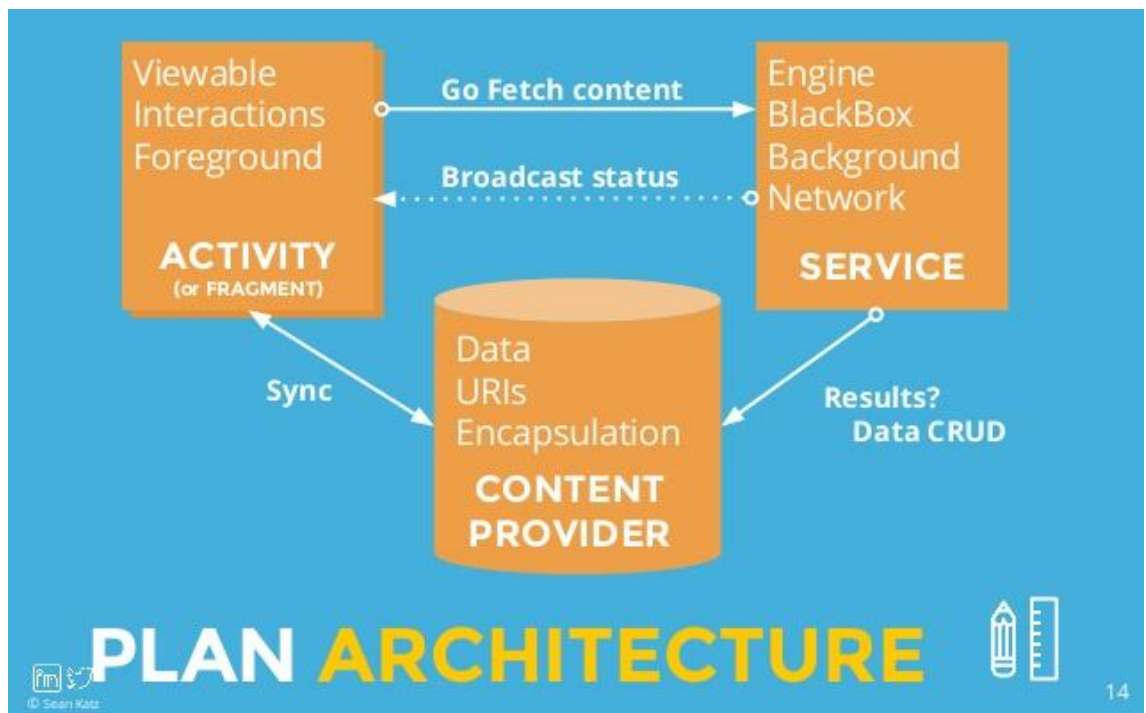


FIGURE 5. Relations between Android Fundamentals

2.2.2 Manifest File

The manifest file is always located at the root of an app directory. It contains the declaration of components of the application. In other words, a component must exist in the manifest file in order to be started by the system.

The manifest file also includes some other important things:

- The minimum version of Android OS that the app requires to run.
- Device features that the app is going to use, such as camera and bluetooth.
- User permissions that the app needs, such as user contacts and internet access.

- API libraries that the app needs to be linked other than the Android framework API.

2.2.3 App Resources

App resources are all other things that an Android application needs other than its code, such as an audio file, images and videos.

App resources also contain xml files that define the look of an application. Because resources are separated from the source code, it can be dynamically changed depending on the device configuration. For example, UI strings can be defined in several XML files in several different languages. The system will decide which language to use depending on the device's user configuration.

A unique ID will be assigned to every resource included in a project. Therefore, instead of putting a string in the source code, a string in XML file can be defined, and then that string can be referred to using its id, e.g. `R.string.nameofthestring`.

App resources give developers the ability to change the visual presentation of an application dynamically without actual changes to the source code. This also enables 2 developers to work on 2 different parts of a project without worrying of violating each other's work. (8)

2.3 Designing application features

To effectively implement an idea, first the app must be designed prototypically.

A design of an application means defining the app's features, the workflow of those features and their requirements.

FIGURE 6 shows an example design for a simple mobile game named Chippy's Revenge. (9)

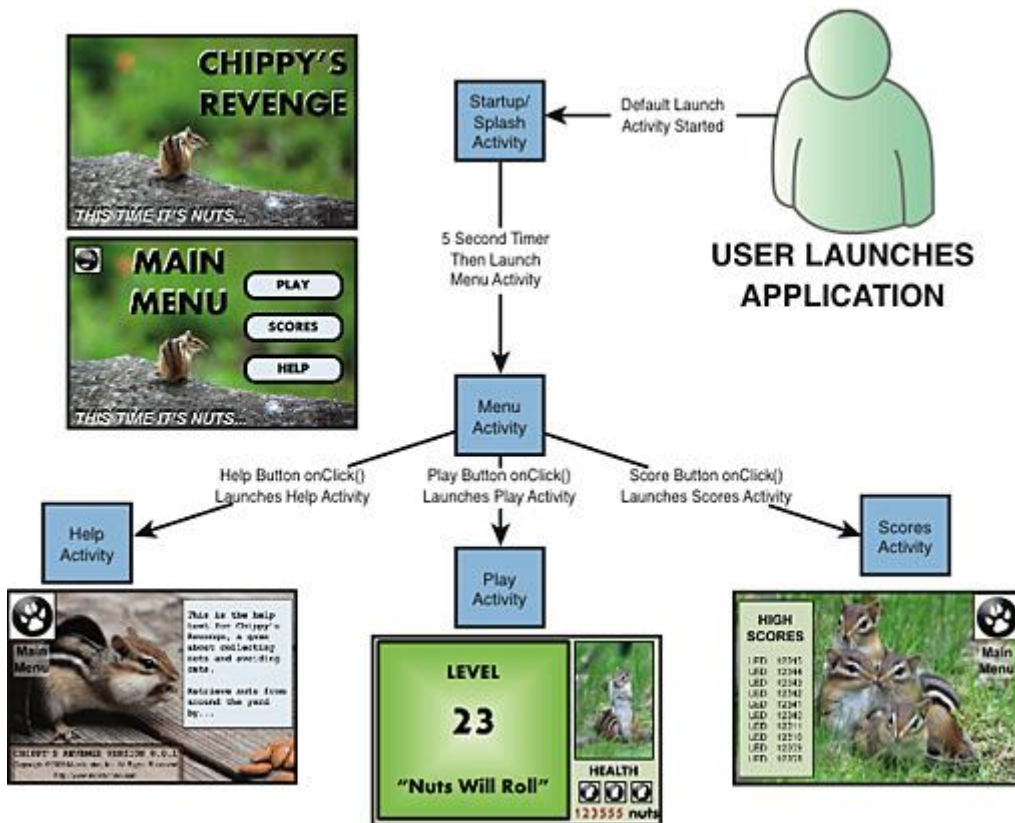


FIGURE 6. Chippy's Revenge design

Doing this kind of planning helps developers to implement an application more efficient and also to avoid possible complication in the future when there are more features to be added.

2.4 Implementing and Testing

After designing the workflow of the app, there is the process of coding, testing and repeating.

Most of the time an activity for each screen of the app will be implemented. Activities should be implemented in the order they are accessed in the design. An activity has a number of callbacks for events that will happen when an activity has been created until it has destroyed.

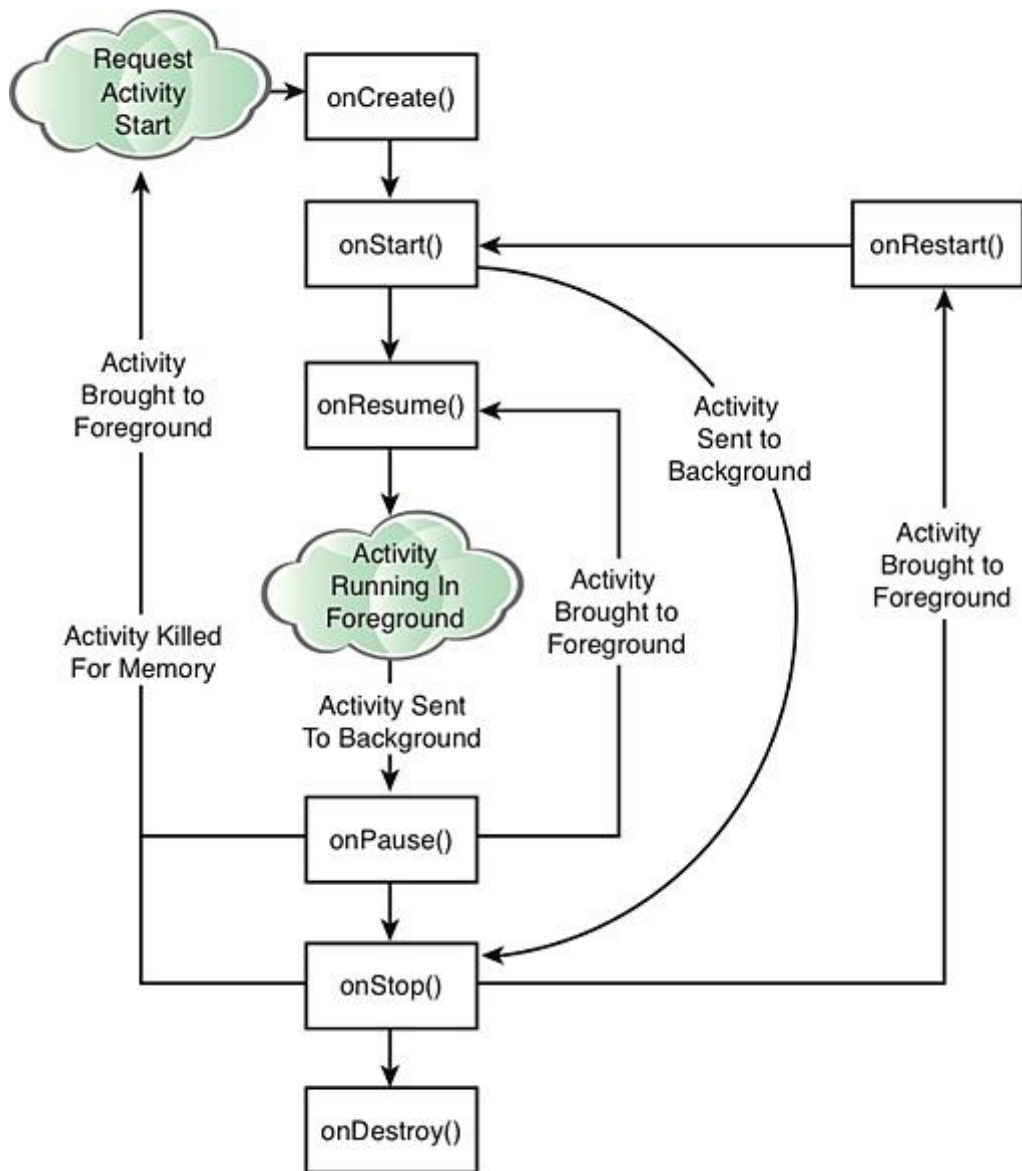


FIGURE 7. Lifecycle of an Activity (9).

Normally the system will decide when to pause or destroy activities. It is possible to modify the source code to do this but it is not recommended.

Testing should be done along with implementing the code. Android IDE such as Android Studio provides very useful ways to test the code on the fly, either with a plugged in device or a virtual machine. It also points out the bug which is preventing the app from running correctly.

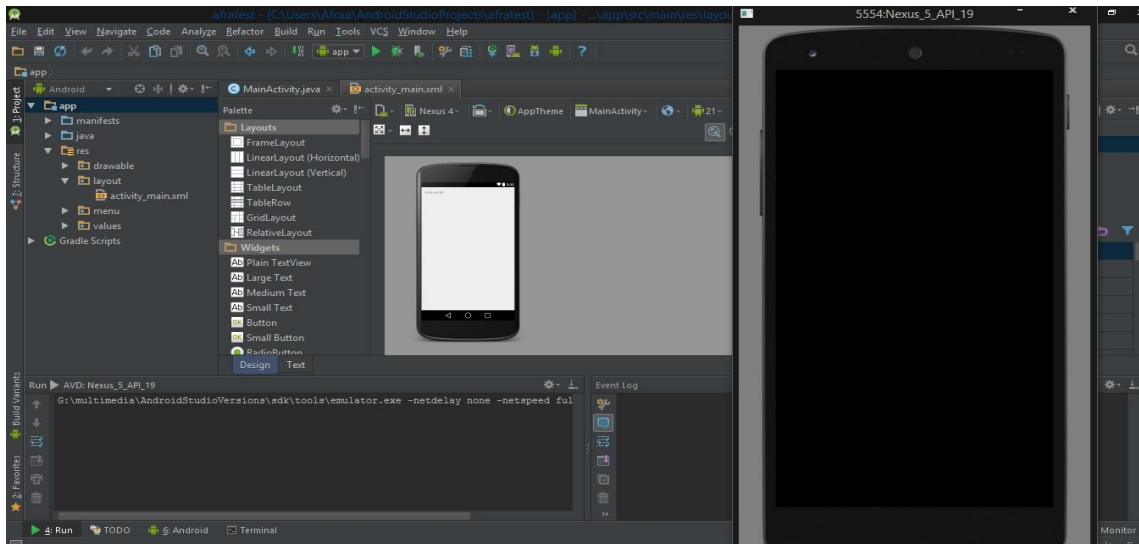


FIGURE 8. Android Studio Emulator

2.5 APK file

When an Android application is completed, it is compiled into an APK file in order to be published on the Google Play Store or to be downloaded and installed by the user.

APK files must be digitally signed before they can be installed in devices. When signing an APK file, the owner of the file keeps a private key. The corresponding public key that was paired to that private key is then attached to the APK file. Any future update or changes to the APK file must be done using the correct private key. The public-key certificate contains the pair of public and private keys also including some other metadata of the owner, such as the name and location.

If an app is published on the Google Play store, the same certificate must be used throughout the lifespan of the app enabling the user to install updates of it.

3 PART 2: WAREHOUSE MANAGEMENT APPLICATION

Part 1 I introduced the basic things needed to start developing an Android application. In this Thesis the author built a native Android application for the Warehouse management system (WMS) to further describe an Android development process. This part is going to be about this warehouse management application.

3.1 Introduction

This app runs on handheld devices, such as tablets or phones. It will act as a mobile UI for a warehouse management system. Its users will be warehouse managers and their employees.

Managers can use this app to monitor everything going on in their warehouse. For example, the number of products or goods they have in their warehouse, the components needed to make the products, the warehouse's customers and their quotations, orders and invoices.

Employees in the warehouse can use this app to create quotations or orders for their customers, put orders on production when they are being produced, or deliver orders.

FIGURE 9 is the Usecase diagram of this application.

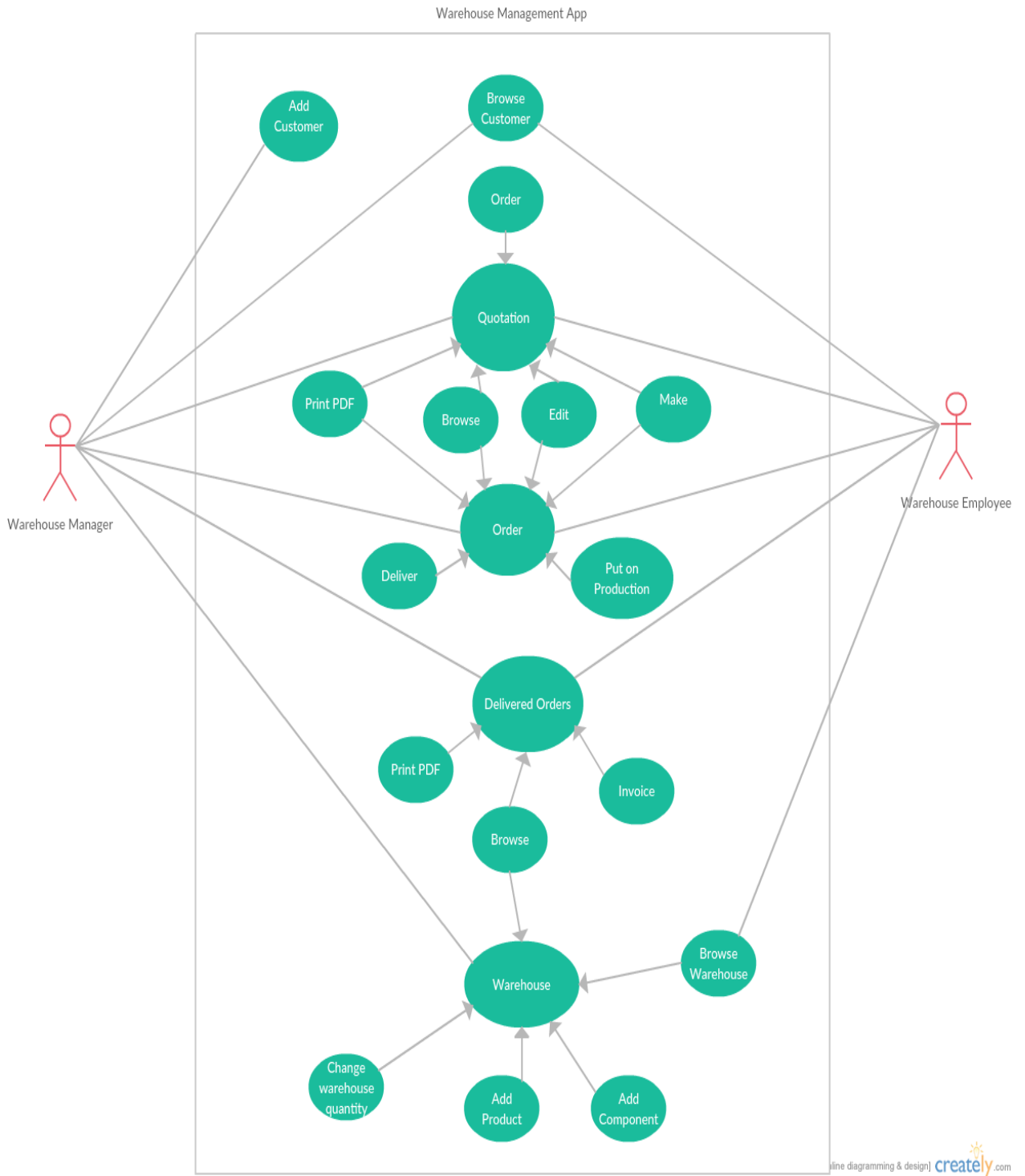


FIGURE 9. WMS Usecase Diagram

Users will need to log in before using the app because there are 2 user types, a manager and a normal user. A user account is also needed because, for example, the manager will need to know which of their employees made a quotation, or delivered an order, in case there is something wrong going on.

A customer list can be browsed and used to create quotations or orders by both users. Only managers can add new customers.

A quotation is made for a specific customer. Every quotation has its unique id and it can only be assigned to one customer. A quotation can be ordered. When ordered, a quotation will become an order with the same information but a date will be added to show when a product will need to be delivered.

An order can be put on production to mark that these products in this order are being produced and the corresponding component will be deducted from the warehouse. When all the products are completed and delivered to the customer, the order can be marked as delivered, and it becomes a delivered order with the same information plus a delivery date.

A delivered order can be browsed and used to make invoices.

Both users have the same privileges for a quotation, an order and a delivered order.

A warehouse interface is where the manager keeps track of the quantity of products and components available in the warehouse. The manager can also add new product, or a component or change their quantity. Other employees can only browse the product and component list to see the availability of a product or a component but they cannot change anything.

This app is developed using Android Studio. Java is the language of development and Object Oriented Programming is the model.

3.2 Data Classes

Android applications are based on Object Oriented Programming. This is a programming model organized around objects rather than "actions" and data rather than logic.

There are 8 data classes are used. Their attributes and relations are described in the figure 10.

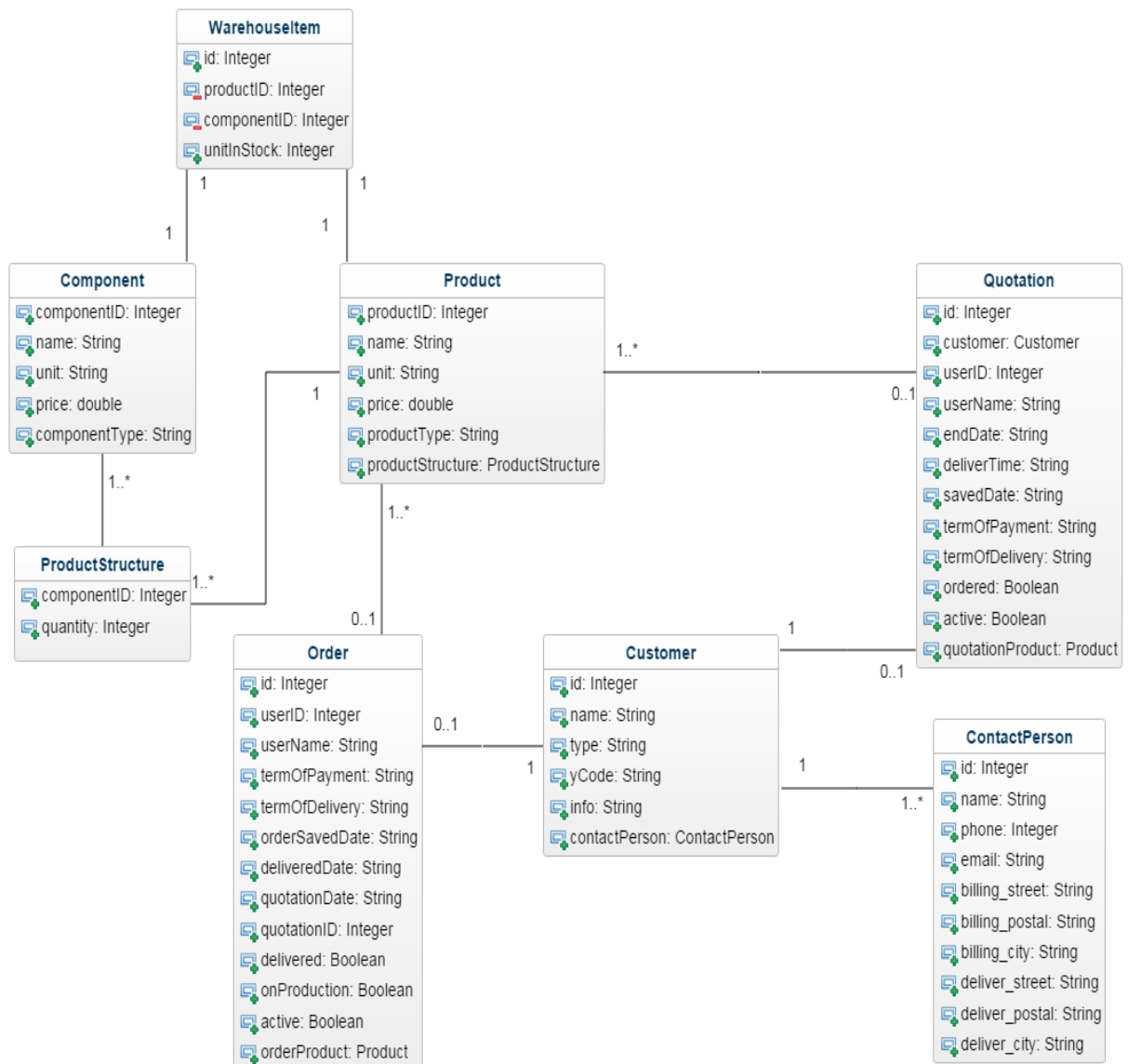


FIGURE 10. Data Classes

WarehouseItem is a data class to hold information about the number of products or components in the warehouse. A WarehouseItem object can only have one productID or componentID at a time. The unitInStock attribute is used to hold the quantity of that product or component.

The productStructure class is used to store the recipe to produce a product. Every product object has a product structure attribute, which is a productStructure object holding information about what and how many components are needed to make that product. Therefore, when a product is produced and delivered, the corresponding components which were used to make that product will be deducted from the warehouse.

Customer objects will hold information on the customer of that warehouse storing or ordering goods from the warehouse. The yCode attribute is unique for every customer. Every order and quotation will have their own customer and their contact person to define the address they should be delivered to and who the bill should be sent to. A customer also has a contact person list holding the contact information of the people corresponding to that company.

The quotation class is used to store information about quotations. Every quotation will have a list of products that are needed to be delivered and only one customer object showing to whom this quotation belongs. The userID and userName attribute is to show the employee who made this quotation. The active Boolean will turn to False when that quotation expires. The order Boolean will turn to True when that quotation is ordered and then an Order object will be created to hold that quotation's information additional to some new informations.

Order objects will hold information about orders that were created. It will have a quotation ID and a date if it was ordered from a quotation. Other info is the same as that of quotation objects. There is the deliveredDate attribute showing the date this order was delivered. This attribute will be null until the order has been delivered.

3.3 Activities

Every screen the user interacts with, when using the app, is an activity. An activity is made of 2 parts, an xml file to define the look of it and a Java file containing the source code.

This WMS has 19 activities. There are 19 screens which are shown to the user when required.

3.3.1 Log in / Register

This is the first activity that the user will use when they open the app for the first time. The register activity is only available to the warehouse manager, who has an account with the admin privilege. Other users must log in with a registered account in order to use the app.

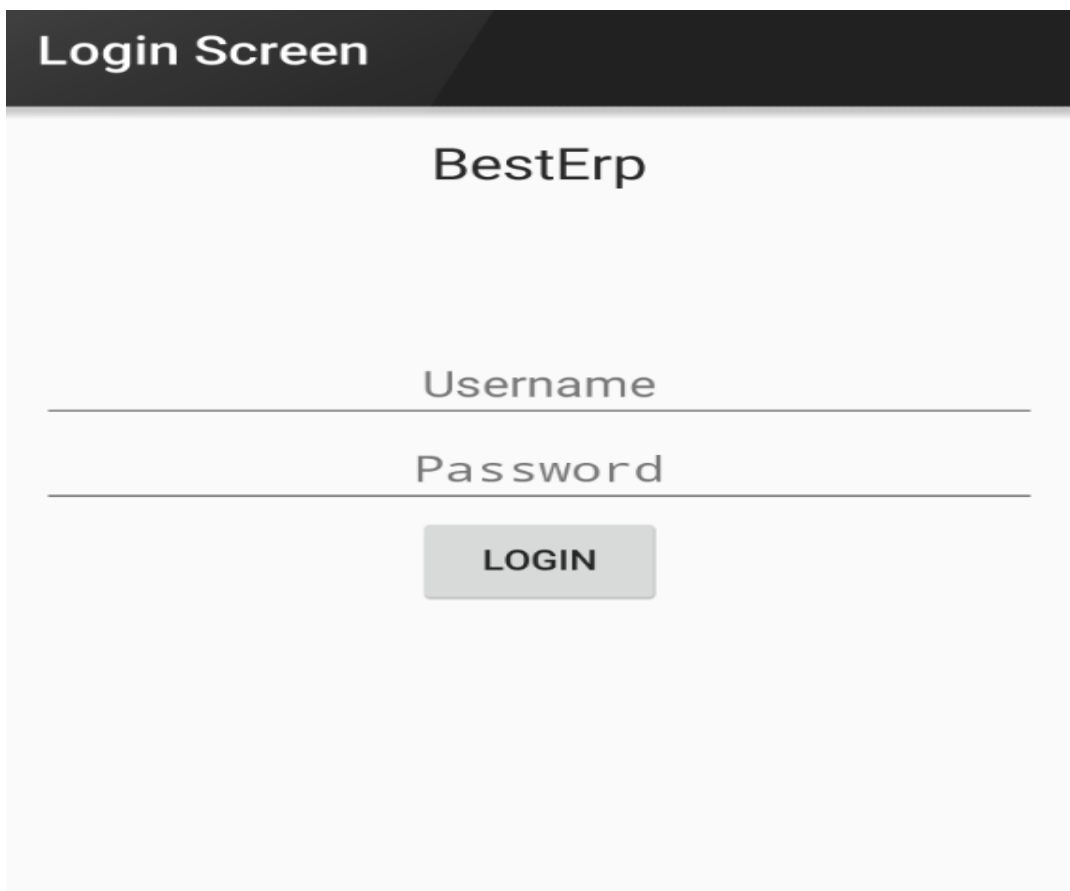


FIGURE 11. Log in screen

3.3.2 Main Activity

The main activity is the one that logged in users will first see when they open the app. It contains the main menu which has a list of the app's features that the user can access.

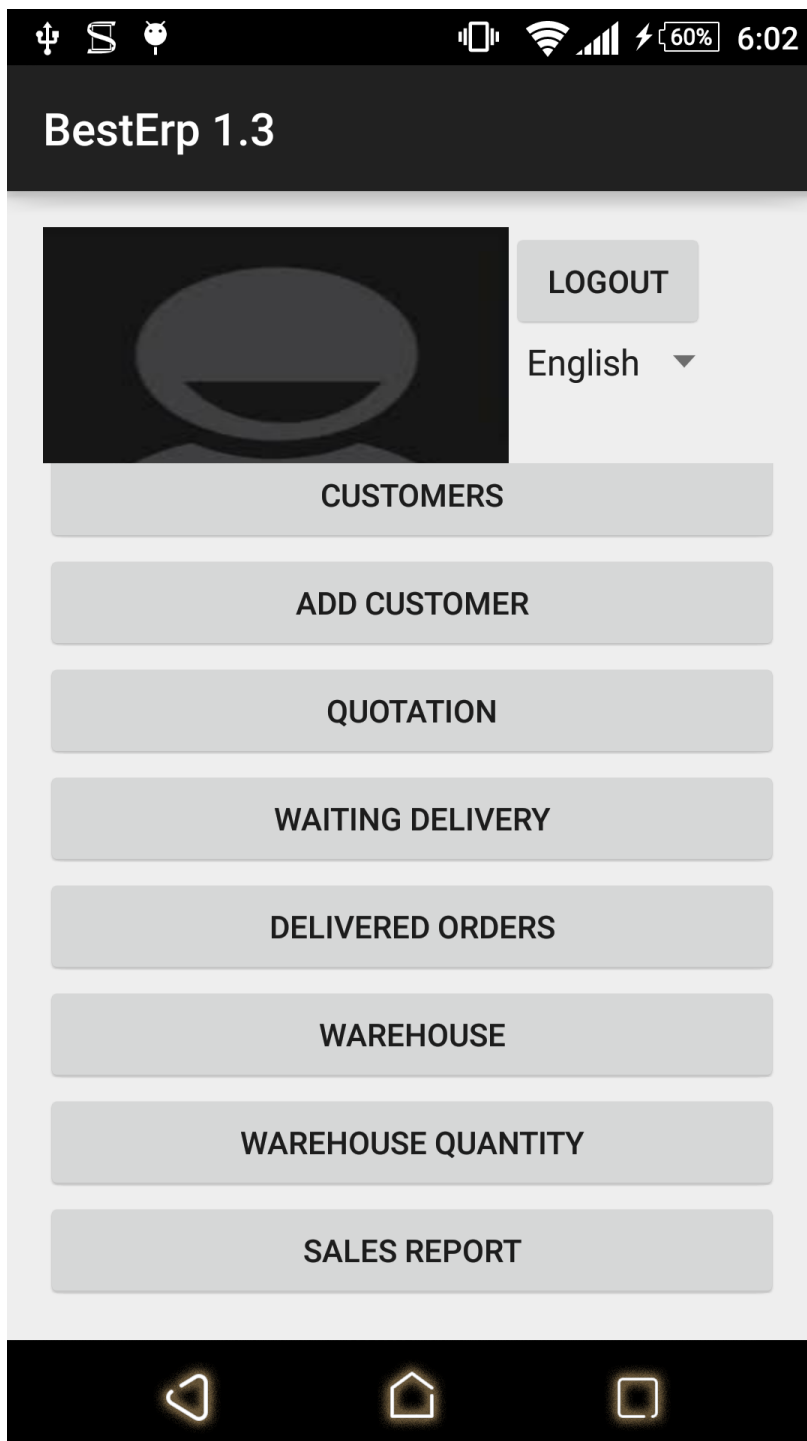


FIGURE 12. Main Activity

There is also a drop down menu on the top right corner for the user to change the language of the app between Finnish and English. This is an Android Spinner. This multi-language function is done using the App resource files feature of Android. By making 2 strings.xml files, one in English located in the folder en, the other in Finnish, located in the folder fi, every string used in the app is referred to as @string/<name> in the xml file and R.string.<name> in the Java code, the system can decide which language to use depending on the user setting on the device.

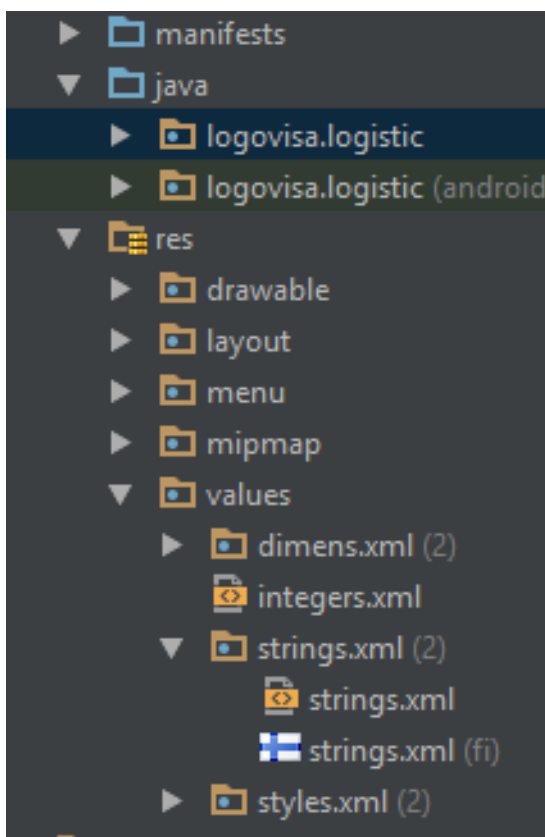


FIGURE 13.strings.xml

Every button seen in the main activity is a feature of the app. The warehouse quantity, an activity to manually adjust the quantity of products and components in the warehouse, and add a customer activity, an activity to add a new customer are only available to users who have the admin privilege.

3.3.3 Customer

Customer can be added using the Add Customer activity, which is only available to warehouse managers (a user with the admin privilege).

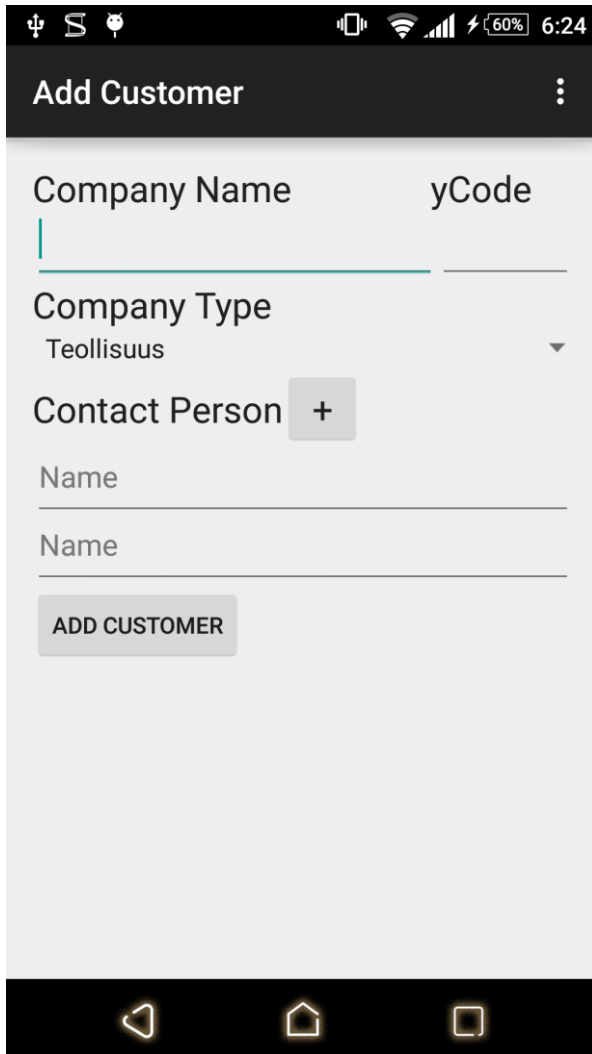


FIGURE 14. Add customer activity

Every customer has their list of people to contact. A contact person is used when making a new quotation, or an order to define its delivery address and contact information. By holding on the input name of the contact person, the user can open a pop up activity to enter that person's info.

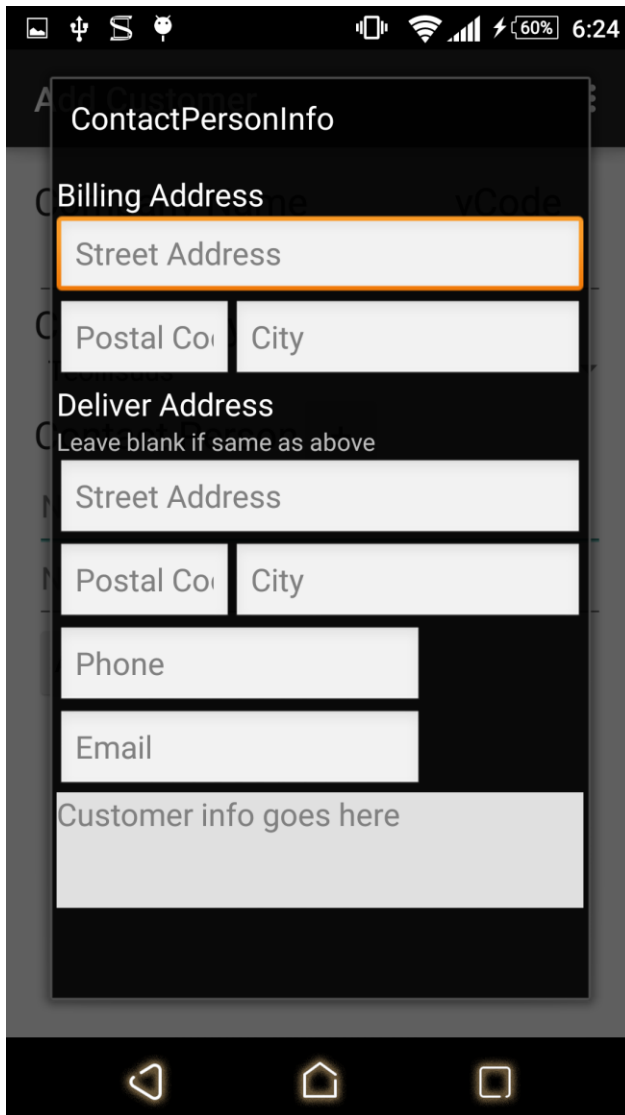


FIGURE 15. Contact person info

When the delivery address is left blank, the system will automatically take the billing address input above to use it as the delivery address.

Multiple contact people can be added for one customer. Although, when making a quotation and an order, the user will have to choose only one contact person for that quotation/order, and that person's address and info will be used as the delivery address for that quotation/order.

The main Customer activity is a list of registered customers of the warehouse. This activity is used to browse customers and to create a new quotation/order

for them, to edit their information or to look at the quotations/orders associated with them.

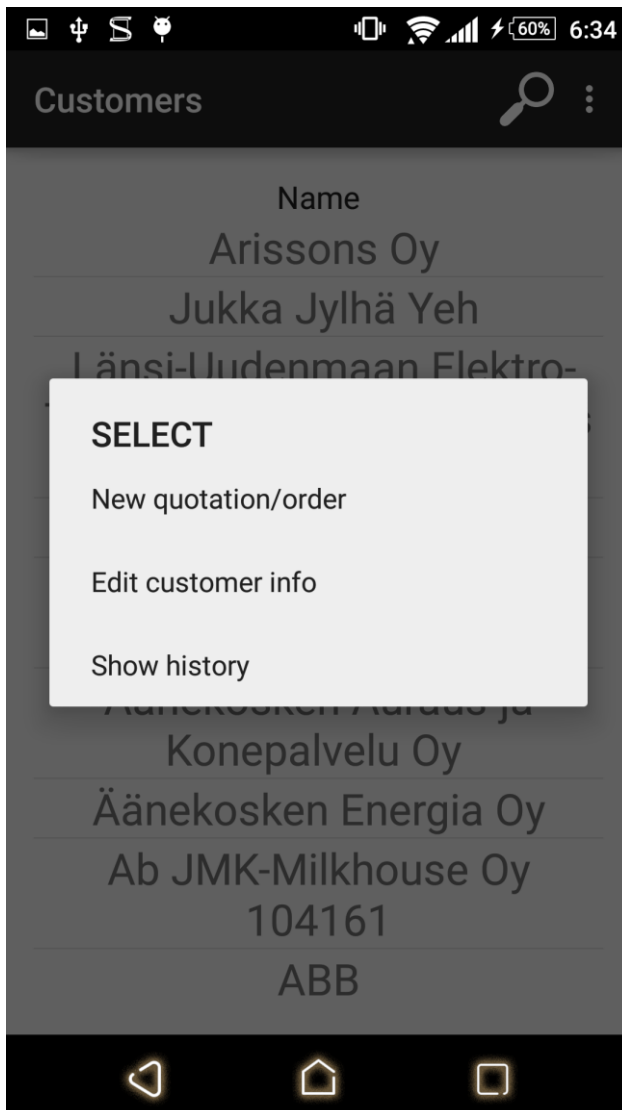


FIGURE 16. Mini menu when clicked on a customer

There is a search button on the menu bar for the user to quickly filter out the customer they want. This filter function can filter by name, ycode and contact person's name.

When the user chooses to edit the customer info, it will be brought back to the add customer activity with that customer's information pre-entered, so that they can edit and save new information of that customer.

When they choose to show the history, they will see the list of every quotation and order associated with that customer.

Choosing new quotation/order, the user will then be asked to select a contact person from that customer's contact people list. After that, the user can start making a quotation or an order for that customer in the Make Quotation activity.

Länsi-Uudenmaan Elektro-Tea...

Company Type: Sähköurakoitsija
Billing Address: Industrigatan 3 10600 Ekenäs
Deliver Address: Industrigatan 3 EKENÄS
Contact Person: Berndt Mannström
Phone: 0400 473 182

CALL EDIT CONTACT PERSON

Quotation/Order QUOTATION

Deliver Time WEEK

Quotation end date 22.05.2017

Term of payment 14 pv netto

Term of delivery EXW

Additional info goes here

SAVE ADD PRODUCT

FIGURE 17. Making quotation/order

There is a switch in the middle for the user to switch between making a quotation and an order. When switching to an order, the delivery time will be an exact date instead of a time period and the quotation end date will disappear.

The user can also edit that customer's information or choose another contact person for the quotation/order using two buttons on top (FIGURE 17). The call button is used to dial the phone number of the contact person who is currently responsible for that quotation/order.

The add product button is used to make the list of products for that quotation/order. After clicking that button, a popup activity named Add Product activity will appear.

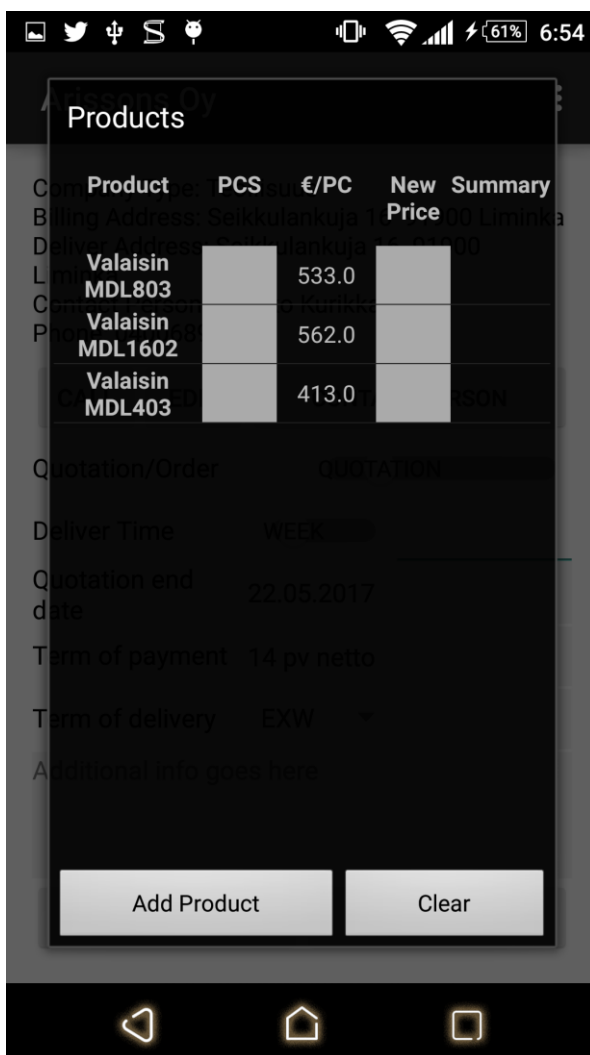


FIGURE 18. Add Product activity

The user will be able to choose a product from the list of products to add it to the quotation when they click the Add Product button. There is also a search button to help the user to filter the product they want.



FIGURE 19. List of product to choose from

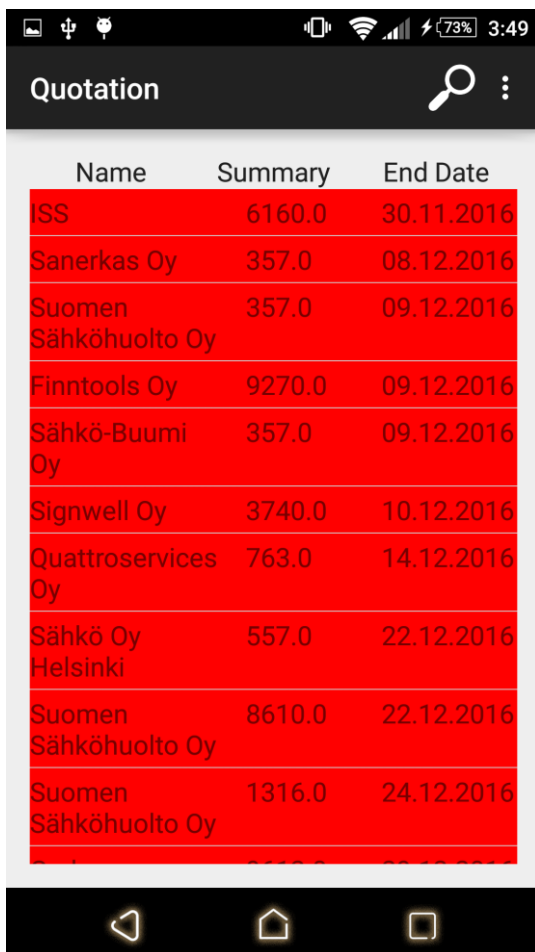
This product list will only load the first 50 results from the server. There is an On Scroll Listener implemented so that when the user scrolls to the bottom of the

list, it will automatically load 50 more results. This is to prevent the initial lag when navigating to this activity when there are too many products in database.

3.3.4 Quotation

The quotation activity (FIGURE 20) is where users will browse and manage every quotation. It is basically a Listview of all quotations belonging to that user. If that user has the admin privilege, they can see all quotations belonging to other users too. There is also a search function to help users filter out the quotation they want.

The quotations that have been ordered will appear in green, the quotations that have not been ordered will appear in red and they will always be listed first.



Name	Summary	End Date
ISS	6160.0	30.11.2016
Sanerkas Oy	357.0	08.12.2016
Suomen Sähköhuolto Oy	357.0	09.12.2016
Finntools Oy	9270.0	09.12.2016
Sähkö-Buumi Oy	357.0	09.12.2016
Signwell Oy	3740.0	10.12.2016
Quattroservices Oy	763.0	14.12.2016
Sähkö Oy Helsinki	557.0	22.12.2016
Suomen Sähköhuolto Oy	8610.0	22.12.2016
Suomen Sähköhuolto Oy	1316.0	24.12.2016

FIGURE 20. Quotation activity

The user can click and hold on a quotation to delete it. There will be a confirmation message so that the user will not delete quotations by mistake. After simply clicking on a quotation, the user will be brought to the quotation info activity.

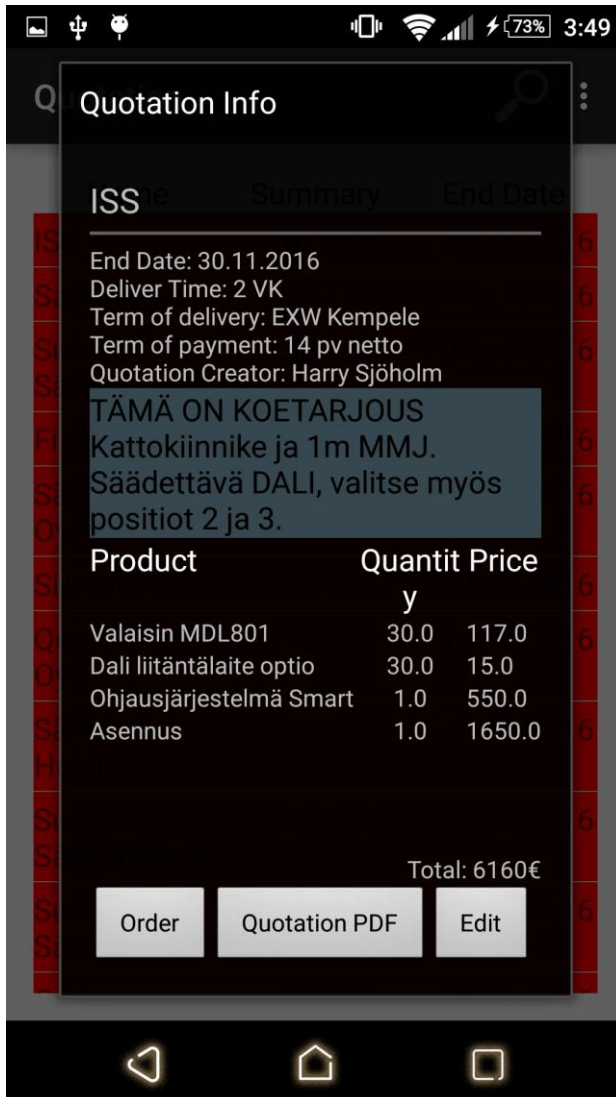


FIGURE 21. Quotation Info activity

The quotation Info activity (FIGURE 21) is where all the quotation's information will be. The user will be given the options to edit, order or print the current quotation's PDF. If the user chooses to edit a quotation, they will be brought back to the MakeQuotation activity with all the information of the current quotation pre-entered. Choosing to order a quotation, the user will be prompted to choose a delivery date. After that, a new order will be created according to that quotation and the quotation will be marked as ordered.

3.3.5 Order

There are two activities to show orders: Waiting Delivery and Delivered Orders. They are both similar to the Quotation activity, where there is a List View of every orders belonging to that user, the admin user can see everyone else's orders and there is also a filter function.

The waiting delivery activity (FIGURE 22) has a list of orders which were ordered from a quotation or made directly for a customer. The delivered orders will appear at green in the bottom of the list. In the middle the orders that have been put on production appear in yellow and on top the active orders appear in red.



Name	Summary	Deliver Date
Leinolan Maito	19947.0	22.06.2017
Best-Hall Oy	27400.0	15.05.2017
MTY Rauhala	6894.0	28.08.2017
CNC Maint-Tech Oy	14820.0	11.05.2017
Mikko Jokisaari	15000.0	26.06.2017
Lehtola Seppo	4511.0	20.06.2017
UpeSystems Oy	7250.0	08.05.2017
Töpselituote Oy	13500.0	01.06.2017
Best-Hall Oy	3590.0	23.05.2017
RKL Poukkula	10500.0	15.05.2017
Sähköliike Kuusisalo Oy	4257.0	30.05.2017
Rapis Oy	6500.0	09.05.2017
ABB	28133.0	28.04.2017

FIGURE 22. Waiting delivery activity

The user can also delete orders by clicking and holding on the order. If they choose one order to manage, there will also be the Order Info activity pop up for them similar to that of with quotation. Although, there will be different operations than in the quotation.

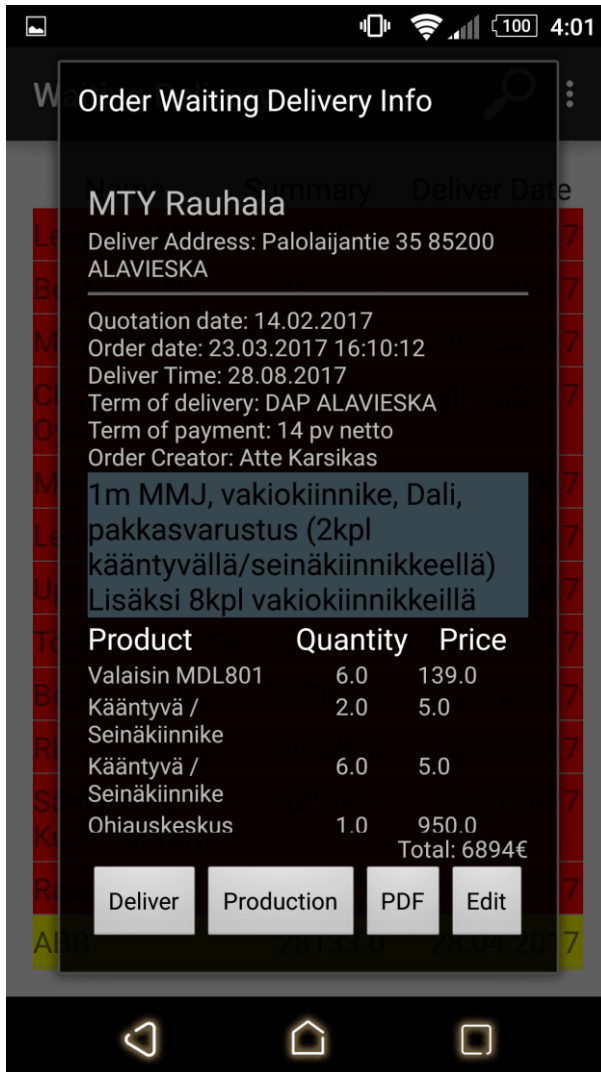


FIGURE 23. Orders info activity

Although the Delivered Order activity (FIGURE 24) is very similar to the Waiting delivery and the Quotation activity, it contains a List View of orders that have been delivered instead. It also has a pop up activity to show delivered orders info. The main operation that the users will perform in here is to manage the delivered orders and print an invoice for their customer.

Name	Summary	Delivered date
Infotark AS	1426.0	19.04.2017
Raahen Konepajatyö Oy	16020.0	25.01.2016
Raahen Konepajatyö Oy	480.0	25.01.2016
Jokilaakson sähkö oy	630.0	03.12.2015
Sähköliike Haapasähkö Oy	5588.0	25.12.2015
Pellon Group Oy	390.0	25.01.2016
Ponsse oyj	2625.0	15.01.2016
presteel oy	8760.0	29.01.2016
jani härkänen	1080.0	11.12.2015
Nivalan Tervas Oy	6723.0	21.01.2016

FIGURE 24. Delivered Orders activity.

3.3.6 Warehouse

The warehouse activity (FIGURE 25) contains a list of products and the components to manufacture them. The app will also calculate the total value of the product and component based on their price and unit in stock. There is also a filter function to help the user finding the product or component they want.

Name	Code	Unit Type	In stock PCS	Unit Price	Value
Product					
Total value: -81821.00					
Valaisin MDL401	MDL401 4322408	kpl	-66.0	138.0	-9108.0
Valaisin MDL402	MDL402 4322409	kpl	0.0	275.0	0.0
Valaisin MDL403	MDL403 4322410	kpl	0.0	413.0	0.0
Valaisin MDL801	MDL801 4322411	kpl	-42.0	184.0	-7728.0
Valaisin MDL802	MDL802 4322412	kpl	0.0	362.0	0.0
Valaisin MDL803	MDL803 4322413	kpl	0.0	533.0	0.0
Valaisin MDL120 1	MDL120 1 4322414	kpl	-8.0	239.0	-1912.0

FIGURE 25. Warehouse activity

The components button is used to switch between the list of components and products. This List View of products and components also has an infinite scroll function similar to the one in the Add Product activity, where only 50 results will be loaded at a time and more will be loaded when the user scrolls down to the bottom of the list.

By clicking on a product or component, the user will have the ability to manually modify the quantity of it in the Warehouse quantity activity. Although, this activity is only available to users with the admin privilege.

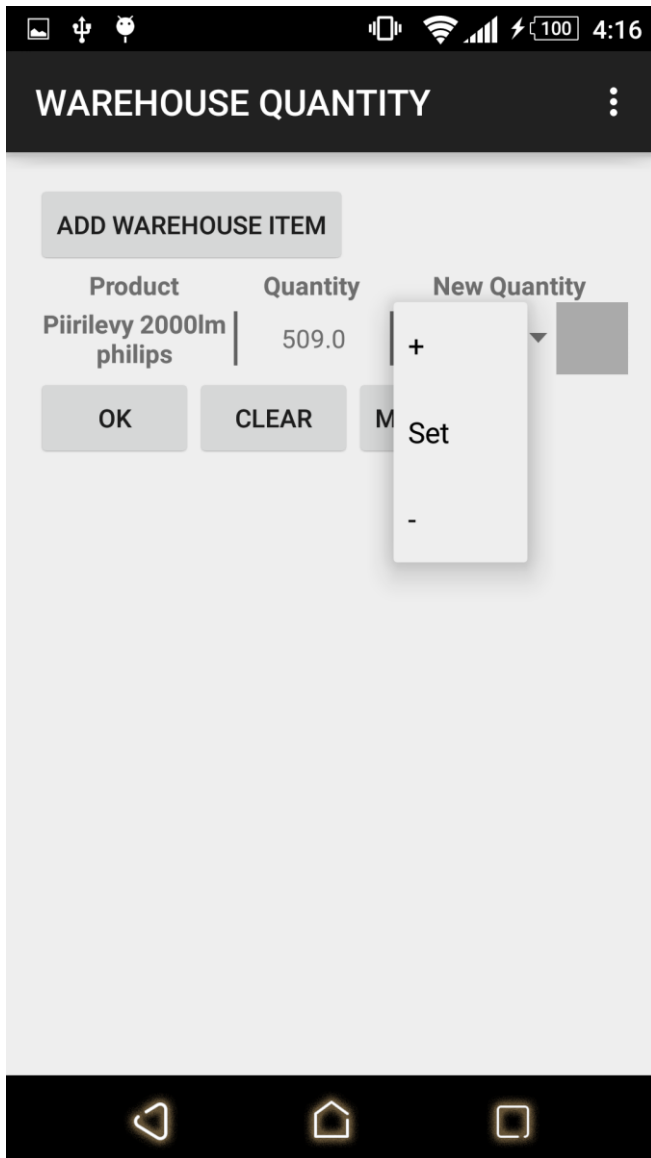


FIGURE 26. Warehouse quantity activity

The add warehouse item button is used to enable the user to add multiple products or components to the list so that they can modify their quantity at the same time.

3.3.7 Sale report

The sale report activity helps warehouse workers to manage quotations and orders available to them in the system.

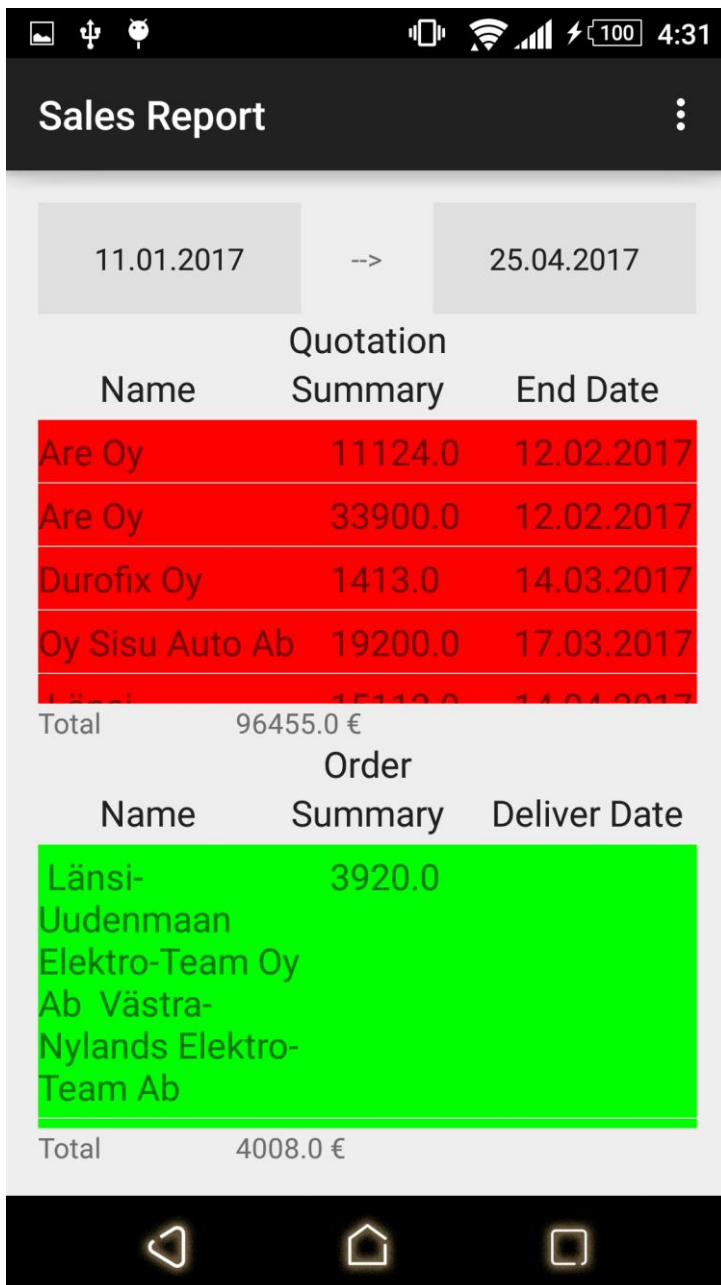


FIGURE 27. Sale report activity

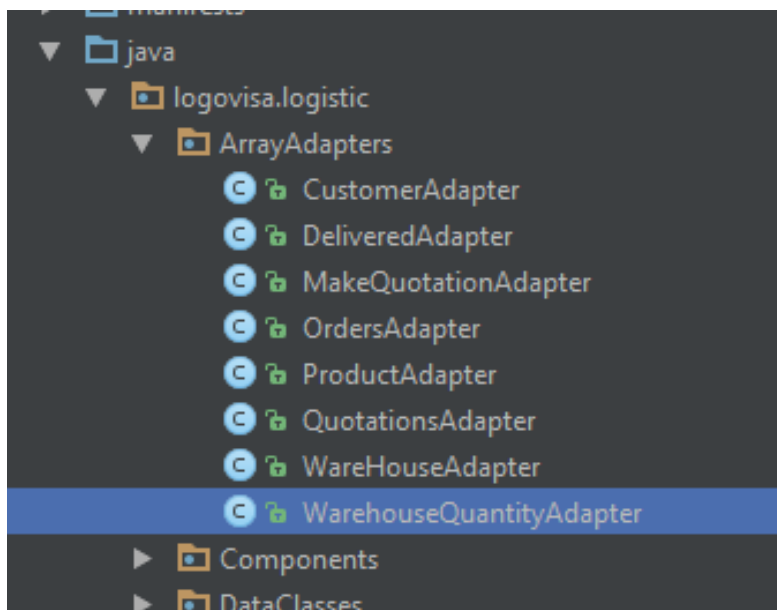
In this activity, users will be able to filter out quotations and orders that were created or ordered in a specific period of time. The app will also calculate the total value of those quotations and orders that were filtered out. The coloring rule follows the same rules as in the quotation and order activities.

3.4 Array Adapters

In the Activities mentioned above, the use of a couple of Android ListViews can be seen.

"ListView is a view group of Android to display a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list." (10)

In this project, customized ArrayAdapter classes were used to populate data into the ListView. There are 8 different adapter classes to convert data for different lists.



Each of these adapters contains a getView method, in which there are codes to get data from the Data Interface Class and convert it into views and put it in the ListView. All the customizations to the views that depend on the data are also done in this method. For example, quotations that are ordered are displayed in green, orders that are not delivered are displayed in red.


```

@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    LayoutInflater quotationInflater = LayoutInflater.from(getContext());
    View customView = quotationInflater.inflate(R.layout.list_view_items, parent, false);
    Customer filteredCustomer = filteredCustomerList.get(position);
    //Get needed UI components
    TextView customerNameText = (TextView) customView.findViewById(R.id.customerNameText);
    TextView date = (TextView) customView.findViewById(R.id.quotationEndDateText);
    TextView summary = (TextView) customView.findViewById(R.id.summaryText);
    summary.setHeight(0);
    summary.setWidth(0);
    date.setWidth(0);
    date.setHeight(0);
    //customerNameText.setLayoutParams(new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MATCH_PARENT,
    customerNameText.setLayoutParams(new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
    customerNameText.setTextSize(25);
    customerNameText.setGravity(Gravity.CENTER_HORIZONTAL);
    //Set data to the components
    customerNameText.setText(filteredCustomer.companyName);
    return customView;
}

```

FIGURE 28. Array Adapter for Customer

```

private class ItemFilter extends Filter {
    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        String filterString = constraint.toString().toLowerCase();

        FilterResults results = new FilterResults();
        List<Quotation> nlist = new ArrayList<>();

        String filterableString ;
        int c = 0;
        for (int i = 0; i < count; i++) {
            Quotation filterQuotation = getItem(i);
            filterableString = filterQuotation.quotationCustomer.companyName;
            if (filterableString.toLowerCase().contains(filterString)) {
                nlist.add(c, filterQuotation);
                c++;
            }
        }

        results.values = nlist;
        results.count = nlist.size();

        return results;
    }

    @SuppressWarnings("unchecked")
    @Override
    protected void publishResults(CharSequence constraint, FilterResults results) {
        filteredQuotationList = (List<Quotation>) results.values;
        notifyDataSetChanged();
    }
}

```

FIGURE 29. Item Filter class of quotation adapter

Another class, which was implemented in most of these adapters, is the ItemFilter class, which enables the user to search for their desired items. This class takes in user's input string and finds entries to contain it in the array list of data which is currently being used. Then it modifies the array list so that it only includes the matching entries it found above and it populates it back to the adapter.

There is also a sorting function implemented in these adapters to help the user to sort the list by Name, Date and Summary value.

```
public void SortByName(boolean click){
    orderedQuotationList.clear();
    NOTorderedQuotationList.clear();
    for(Quotation q : filteredQuotationList){
        if(q.ordered) orderedQuotationList.add(q);
        if(!q.ordered) NOTorderedQuotationList.add(q);
    }
    filteredQuotationList.clear();
    if(click == true) {
        Collections.sort(orderedQuotationList, Quotation.customerNameComparator);
        Collections.sort(NOTorderedQuotationList, Quotation.customerNameComparator);
    }
    else {
        Collections.sort(orderedQuotationList, Quotation.customerNameComparator_DE);
        Collections.sort(NOTorderedQuotationList, Quotation.customerNameComparator_DE);
    }
    for(Quotation q : NOTorderedQuotationList) filteredQuotationList.add(q);
    for(Quotation q : orderedQuotationList) filteredQuotationList.add(q);
    notifyDataSetChanged();
}
```

FIGURE 30. Quotation Sort by Customer Name method

The sorting method simply sorts the ArrayList that is currently being used using the comparators implemented in the Data classes. Then it calls the notifyDataSetChanged method, which tells the system that the data in ListView has been changed and the list needs to be updated.

```

public static Comparator<Quotation> customerNameComparator = (lhs, rhs) → {
    String customerName1 = lhs.quotationCustomer.companyName.toLowerCase();
    String customerName2 = rhs.quotationCustomer.companyName.toLowerCase();

    return customerName1.compareTo(customerName2);
};

public static Comparator<Quotation> customerNameComparator_DE = (lhs, rhs) → {
    String customerName1 = lhs.quotationCustomer.companyName.toLowerCase();
    String customerName2 = rhs.quotationCustomer.companyName.toLowerCase();

    return customerName2.compareTo(customerName1);
};

```

FIGURE 31. Customer Name comparator in Quotation data class

This sorting method is then called when the user clicks on the list headers. If clicked again, the sorting order will be inverted.

```

//Name Header onclick handling
TextView companyName = (TextView)findViewById(R.id.textName);
companyName.setOnClickListener((v) → {
    if (nameHeaderClick == true) {
        arrayAdapter.SortByName(true);
        nameHeaderClick = false;
    } else {
        arrayAdapter.SortByName(false);
        nameHeaderClick = true;
    }
});

```

This array adapter then needs to be declared and set to the ListView that will display the view they converted.

```

private void showOrders() {
    ordersArray = dataInstance.GetAllOrders();
    ArrayList<Order> thisUserOrders = new ArrayList<>();
    for(Order o : ordersArray){
        if(o.userName.equals(dataInstance.getUserName())) thisUserOrders.add(o);
    }
    arrayAdapter = new OrdersAdapter(this, thisUserOrders);
    ordersListView.setAdapter(arrayAdapter);
}

```

FIGURE 32. Orders ListView adapter

3.5 Data Interface Class

In this project, a lot of data needs to be stored, used and updated throughout the whole user session, e.g. user's name, type or customer list. Therefore, a Data Interface class has been implemented to take care of this. This class contains every data that has been fetched from the server and a lot of methods to change and get them.

```
public class DataInterfaceClass {
    private static DataInterfaceClass instance;
    private static ArrayList<Component> componentList;
    private static ArrayList<Customer> customerList;
    private static ArrayList<Quotation> quotationList;
    private static ArrayList<Order> orderList;
    private static ArrayList<Order> deliveredOrders;
    private static ArrayList<DeliveredOrder> deliveredOrderList;
    private static ArrayList<Product> productList;
    private static ArrayList<WareHouseItem> warehouseItemList;
    private static ArrayList<Product> chosenProductList;
    private static ArrayList<OrderProduct> tempProductList;
    private static ArrayList<WareHouseItem> chosenWarehouseList;
    private static ArrayList<String> userQuantity;
    private static ArrayList<String> userPrice;
    private static ArrayList<String> deliveryTerms;
    private static ArrayList<String> customerTypeList;
    private static String userType = "";
    private static String userName = "";
    private static String lastIdCustomerList = "";
    private static ArrayList<ContactPerson> contactPersonsList;
    private boolean isFirst = true;
    private boolean TestDataCreated = false;
}
```

FIGURE 33. Data Interface class property declaration

By using this class, data can be stored and fetched at any time without keeping activities running. It also helps to organize the data flow in the app better and eliminates repetitive codes. For example, by simply declaring an instance of this class and using it's method, the needed data can be retrieved in just 2 line of codes.

```
dataInstance = DataInterfaceClass.getInstance();  
arrayAdapter = new CustomerAdapter(this, dataInstance.GetAllCustomers());
```

3.6 Network classes

The network classes is where the connection information to the database is kept and the methods to get data from the server are done.

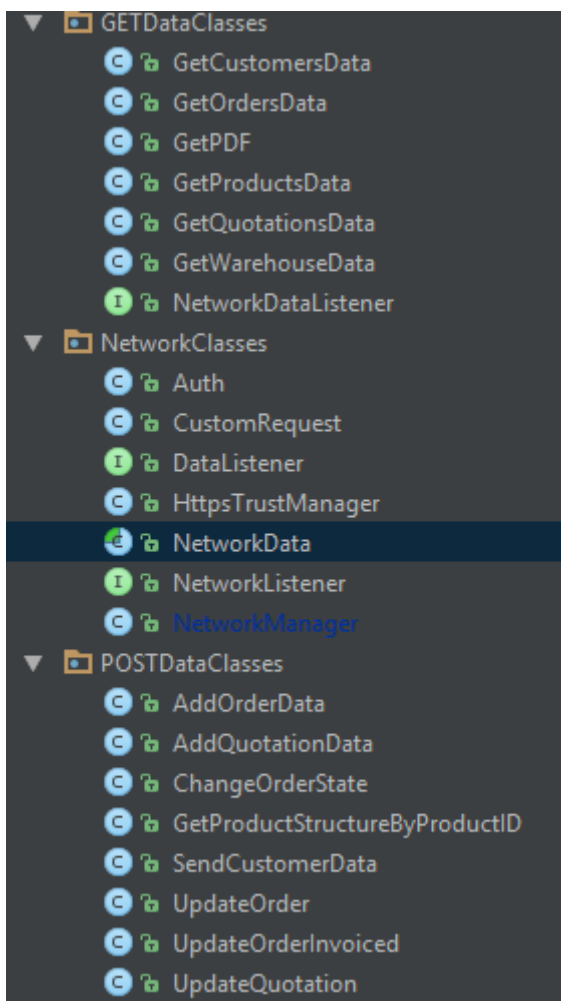


FIGURE 34. Network classes

The network classes contain all the methods to connect to the server, to get JsonObject from the server and to post data to the server. The get data classes call the get method from the network class and receives json data from the server. Then converts and populates data to the Data Interface for a later use.

The post data classes convert data to json objects and call post methods from the network classes to send data to the database to be stored.

3.7 Message box and Loading Popup

There is a message box class to handle a confirmation message to prompt the user when they are going to do some important operation, such as exiting the app or making a quotation. This class defines the layout of the message popup, which the user will see, and the information in it. This message popup is also used to show the user when there is some error that is preventing the app from working properly, for example no network connection detected.

```

public void quotationDoneBox() {
    final AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder
        .setTitle(title)
        .setMessage(info)
        .setIcon(android.R.drawable.ic_dialog_alert)
        .setPositiveButton("OK", (dialog, which) -> {
            GetQuotationsData.getInstance().initData(context, () -> {
                Intent info = new Intent(context, QuotationInfo.c
                if(currentQuotation.quotationId != null) info.put
                else info.putExtra("CLICKED_ITEM", "last");
                info.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
                context.startActivity(info);
            });
        });
    builder
        .show();
}
}

```

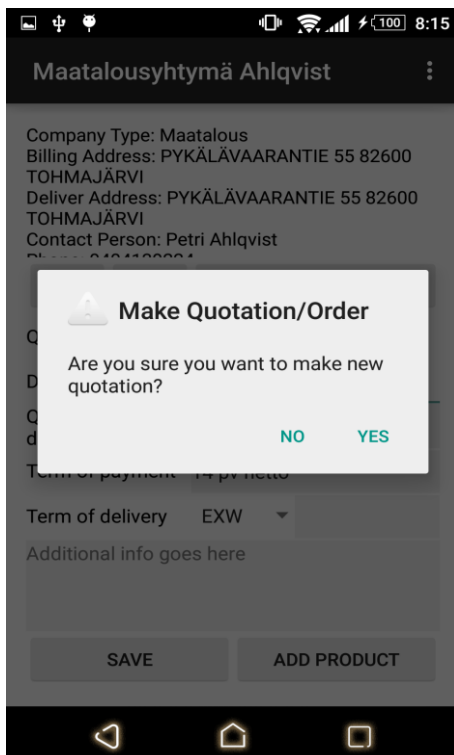


FIGURE 35. Make quotation confirmation message

Another kind of popup message is also used very often in this project. It is a loading popup to tell the user what is going on and why they have to wait. This popup appears every time the app is fetching data from the server and populating it for the user to be viewed.

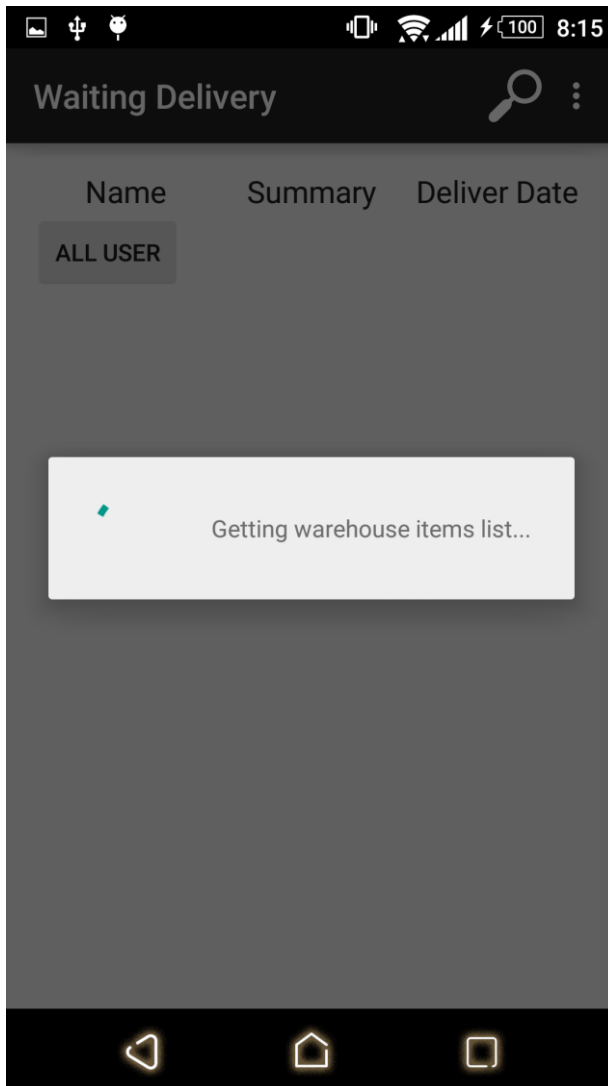


FIGURE 36. Loading popup

This is a simple Android Progress Dialog with a title telling the user what is happening. It will automatically disappear when the operation has been completed.

3.8 Testing

Testing is carried out alongside with coding using an Android device and the Android Studio live logcat feature, which is a very useful function that points out everything from bugs, performance issues or coding logs.

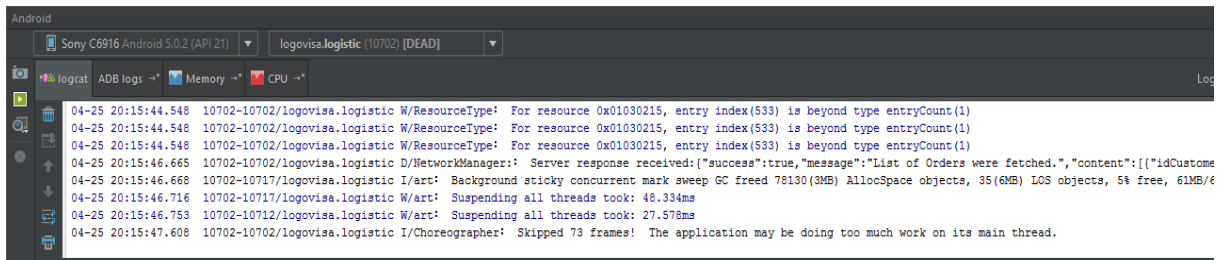


FIGURE 37. Android Studio testing feature

3.9 Publishing

After every feature of the app has met the requirements and testing has been carried out, the app can be signed and Android Studio will generate an apk file which can be installed on other devices which can be published to the app store.

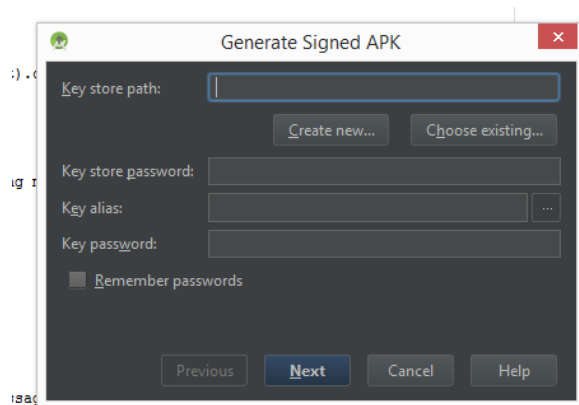


FIGURE 38. Generating APK file in Android Studio

A personal key can be created containing the creator's personal information, it can be assigned to the apk file of the app so that only the person with the right key can make updates to the apk file. If the key is not the right one, the one signed to an apk, the Android system will consider the new apk a different app and it will delete the older app to install a new one instead of updating it.

4 CONCLUSION

Android development is very beginner friendly with useful tools and framework for developers to maximize the effectiveness of their work. Especially Android Studio is a very effective development environment, which includes everything a developer can wish for, from a smart code editor to a super detailed and fast debugger, an easy-to-use version control and many other amazing features.

With that said, Android development is still very challenging in many different ways. One of them is software fragmentation. Because there are so many Android OS versions, it is very hard for developers to cover all of them. Thus the app might work well on the most recent OS but cannot start on an older one. Another challenge is the hardware variation. There are many devices running Android from many manufacturers. This makes it very hard to optimize an application to run well on all of them. This features various occasions where an app crashes on specific devices while works perfectly on others. Those are the challenges I have faced personally during my time in developing the WMS application. Although, those challenges were also very captivating for me to try and overcome.

After I completed working on this application, I have gained plenty of knowledge about Android development and I have improved my Java skills dramatically. This motivates me to pursuit my career as an Android developer after graduation.

REFERENCES

1. Wikipedia. Android (operating system). Date of retrieval: 01.02.2017
[https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))
2. Techtrickle. History of Android OS. Date of retrieval: 04.02.2017
<http://techtrickle.com/history-of-android-operating-system/>
3. Android developer document. Meet Android Studio. Date of retrieval: 10.02.2017
<https://developer.android.com/studio/intro/index.html>
4. Wikipedia. Apache Cordova. Date of retrieval: 14.02.2017
https://en.wikipedia.org/wiki/Apache_Cordova
5. Apache Dordova. Create your first Cordova App. Date of retrieval: 15.02.2017
<https://cordova.apache.org/docs/en/latest/guide/cli/>
6. Wikipedia. GameMaker: Studio. Date of retrieval: 16.02.2017
https://en.wikipedia.org/wiki/GameMaker:_Studio
7. Gadget Daily. GameMaker: Studio gives Android developers chance to create games in record times. Date of retrieval: 20.02.2017
<https://www.gadgetdaily.xyz/gamemaker-studio-gives-android-developers-chance-to-create-games-in-record-times/>
8. Android developer document. Application Fundamentals. Date of retrieval: 27.02.2017
<https://developer.android.com/guide/components/fundamentals.html>
9. Computer World. How to build an Android app. Date of retrieval: 10.03.2017
<http://www.computerworld.com/article/2514892/application-development/app-development-how-to-build-an-Android-application-step-by-step.html>
10. Android developer document. List View. Date of retrieval: 16.04.2017
<https://developer.android.com/guide/topics/ui/layout/listview.html>