Developing a mobile software with Android Studio

Bachelor's Thesis

Hämeenlinna University Centre
Degree Programme in Business Information Technology

Autumn 2020

Antti Kojo

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Tietojenkäsittelyn koulutusohjelma
Hämeenlinnan korkeakoulukeskus

Author Antti Kojo **Year** 2020

Title Developing a mobile software with Android Studio

Supervisor(s) Lasse Seppänen

TIIVISTELMÄ

Tämän opinnäytetyön tarkoitus on käydä läpi Android-sovelluskehityksen perusteita Android Studion avulla. Läpi käydään myös Java -ohjelmointikielen perusteita ja sen soveltuvuutta Android -laitteiden kanssa. Koska Androidille on mahdollista ohjelmoida myös Kotlin-ohjelmointikielellä, käydään työssä läpi myös Kotlinin perusasioita.

Työn käytännön osan tavoitteena on saada tehtyä valmis Android-sovellus. Sovelluksen tarkoituksena on olla kitaraharrastajille soveltuva tietopankki, johon käyttäjä voi tallentaa ja lukea kitarasointuja.

Työn tekemisen apuna ja tukena toimivat erilaiset tutoriaalit ja nettimateriaali aina erilaisia ongelmatilanteita kohdattaessa. Aiempaa kokemusta mobiilisovellusten kehittämisestä ei ollut ja työn tekeminen mahdollistikin itsenäisen mobiilisovelluskehityksen opiskelun.

Työn tulos oli toimiva sovellus, jolla käyttäjä voi tallentaa ja lukea kitarasointuja. Käyttäjäystävällisyydessä ja graafisessa puolessa riittää vielä hiottavaa ja mahdollinen jatkokehitys voisikin keskittyä tähän. Tarkoitus oli kuitenkin tutustua mobiilisovelluskehitykseen ja Android -laitteisiin ja tämä tavoite työssä saavutettiin.

Avainsanat Android, Android Studio, Mobiiliohjelmointi, Java, Kotlin

Sivut 30 sivua, joista liitteitä 1 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Hämeenlinna University Centre
Degree Programme in Business Information Technology

Author Antti Kojo **Year** 2020

Subject Developing a mobile software With Android Studio

Supervisor(s) Lasse Seppänen

ABSTRACT

The goal for this thesis is to go through the basics of Android application development using Android Studio as a tool. The basics of Java programming language and its compatibility with Android devices are also gone through. Since it is also possible to use Kotlin as a programming language when programming for Android – the basics of Kotlin are also covered.

The aim of the practical part of the work is to do a working Android application. The purpose of the application is to be a data bank suitable for guitar enthusiasts, where the user can store and later read the stored guitar chords.

Online tutorials and online material are utilized to support the progress of the practical part in different kinds of situations where help is needed. There was no prior experience in developing mobile applications and doing the thesis allowed for independent study of mobile application development.

The result of the work was a working application in which the user can store and read guitar chords. There is still room for future development in terms of user friendliness and graphic design – something to focus on with future development. However, the intention was to get acquainted with mobile application development and Android devices and this goal was achieved.

Keywords Android, Android Studio, Mobile development, Java, Kotlin

Pages 30 pages including appendices 1 pages

DICTIONARY

Operating system	Operating system or as sometimes referred OS is a software that acts as an interface between the user and the computer. It manages all basic tasks in a computer. Such as file-, memory- or process management.
Open source	Open source programs or software include the permission for anyone to use the source code behind the software. When software is released as open source – it makes it possible for programmers and enthusiasts to develop this software for their own projects.
IDE	Integrated Development Environment is a software or usually a collection of programs which offers comprehensive utilities to a computer programmer to produce code and develop software.
SQL & SQLite	Structured Query Language is a programming language designed to manage data, that is kept in a relational database.

CONTENTS

1 INTRODUCTION

Mobile and smart devices are very common today and becoming mandatory for everyone to possess if they are not already. One of the most common operating systems for these devices is Android and the official development language for Android is Java. This topic was chosen for the thesis since it is overall useful information and not just limited to mobile applications.

This thesis does not have a client and will be done as an individual project. The thesis will go over the process of what does one need to build an android application from scratch, what tools are needed and what steps must be taken to do so.

The goal of the thesis is to produce a working mobile software for android, that is useful for beginner guitar players, as it holds information about guitar chords. The application should show desired chords on command and how to play them – visualization is shown on a virtual fret board. The final application is a stripped version of a regular guitar application, which usually has a tuner. It was decided to not to include a tuner, since signal processing is not part of the thesis. The focus of the thesis lays on what basics are needed to produce a working application for an android device.

The programming of the application is done using Java programming language and Android Studio is used as a framework. The theoretical part of the thesis will go over Java programming language, Android Studio, guitar chords and the guitar fret board to some extent. Later chapters will tell about how to use these tools and the practical part will act as a tutorial, which should produce information about how to build a mobile application for an android device. Lastly the application is imported to an Android device and tested with it.

Following questions should be answered by the end of the thesis:

- What tools do you need to develop an Android application?

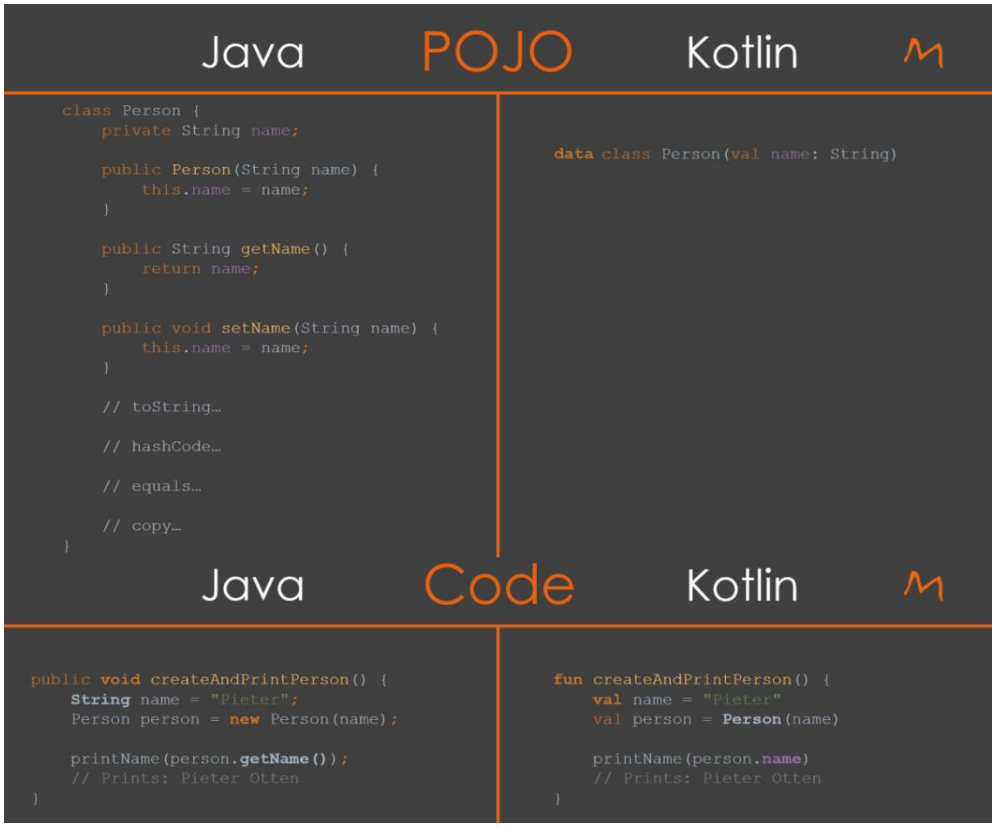
- What steps must be taken to develop a working Android application?

2 JAVA & KOTLIN

Java has been the official programming language for Android development for a long time now. Since in this thesis an Android application is developed, it was thought out to write the application in Java, because of the author's experience in Java and because of the good documentation Java has available to it. The other option was to write the application in Kotlin. Both options were good in their own way. Java has good documentation, but Kotlin seems to be to future language when it comes to Android development.

Java differs from Kotlin in the way that in Kotlin the programmer can write more compact code, but still retain the readability of the code. E.g. when creating classes: boilerplate code - such as getters and setters are not mandatory to be typed out. This is demonstrated below in Figure 1 taken from a website (mediaan.com, 2018). In this chapter both languages backgrounds are gone over. (Android.com, 2020)

Figure 1 Example of getters and setters in Java and in Kotlin taken from Mediaan.com



2.1 Java

Object-oriented programming languages started to take name for themselves in a larger scale during the mid-80's. This way of thinking in programming struck itself through in the early 90's and grew from a way of thinking to a hype-phenomenon although the history of object languages is older than that. The earliest object language was used by the state of Norway and in a calculating centre in the 60's. The language was called Simula. (Vesterholm & Kyppö, 2018, p.27)

Java is object-oriented programming language developed by Sun Microsystems. It has been built from the foundation of C++, which is also typically object-oriented language. According to Vesterholm & Kyppö "Java is a general-purpose concurrent class-based object-oriented programming language, specifically designed to have as few implementation dependencies as possible. Java allows application developers to write a program once and then be able to run it everywhere on the Internet" (Vesterholm & Kyppö, 2018, p.16)

In the background of Java being born, is a need for small electric device software. To fulfil this need, James Gosling created a programming language called Oak. Gosling was working for Sun Microsystems when this happened in 1991. Back then was customary to translate programs and software again for every processor model – this was a problem and Oaks purpose was to get rid of this. (Vesterholm & Kyppö, 2018, p.15)

When world wide web (www) started to gain popularity – a similar idea was planned to be executed in the web. Liveoak would make it possible for software to be ran on the web on different platforms. In 1994 HotJava browser was published, which also made the programming language called Java. Java was laid on the foundation of C and C++, from which some unnecessary things were removed. (Vesterholm & Kyppö, 2018, p.15-16)

Java is a developing language. It includes a big nomenclature and version history, which is hard to grasp and understand fully. The development environment which is available from Javas homepage is called JDK (Java SE Development Kit). JDK and SDK (Software Development Kit) are sometimes used to describe the same thing. Basically, they do mean the same thing. On top of JDK, there's JRE (Java SE Runtime Environment), which includes only virtual machine and standard class libraries to run Java-programs. (Vesterholm & Kyppö, 2018, p.19)

2.2 Kotlin

*"*Kotlin is a pragmatic programming language that runs on the Java virtual machine and Android. It combines object-oriented and functional features and, as it has 100% interoperability with Java, developers can use it to write a few files in an existing project or to write an application using Kotlin from scratch*."* (Mateus & Martinez, 2019)

Kotlin is one of the official programming languages for Android and Android Studio provides great support for Kotlin. Android Studio can be utilized to convert Java-based code to Kotlin with one of its built-in tools. (Mateus & Martinez, 2019)

When Google announced Kotlin as an official programming language for Android in 2017, developers then had an alternative programming language to develop applications. Before that only Java was used for this purpose and in some bases C++ also. (Mateus & Martinez, 2019)

*"*The announcement made by Google had a significant impact on the community of Android developers. Moreover, in the last years, the popularity of Kotlin has in- creased as the Stack Overflow Annual Developer Survey of 2018 and 2019 reported*."* (Mateus & Martinez, 2019)

Google has also announced that future Android development will be more and more Kotlin focused. “Which means that new APIs and features will be offered first in Kotlin. Consequently, they advise developers that are starting a new project to write it in Kotlin.” (Mateus & Martinez, 2019)

Kotlin combines object oriented and functional features, which offers an alternative approach to develop mobile applications. Some of which are not available for Java and Android Development. (Mateus & Martinez, 2019)

Although Kotlin is starting to have an increasing number of developers that use it as their Android programming language, research shows no studies of Kotlin and its features in the literature. This is probably due to Kotlin being so fresh and new language. (Mateus & Martinez, 2019)

With this information about Kotlin it was still decided to use Java as the programming language in the thesis. Kotlin seems to be a future language for Android, but the Author had more experience in working with Java. Kotlin may not have as good and comprehensive documentation as Java – it will undoubtedly be easier to research information and tutorials for Java than it would be for Kotlin.

3 OPERATING SYSTEM / ANDROID

Operating system is very central and vital software to run a computer. It makes it possible to run other software on top of itself. Operating system manages the resources available for that computer that it is run on.

Android is a mobile operating system and it is developed and owned by Google. Android operates on top of Linux Kernel, which offers the device developers some important security features. e.g. process differentiation (Väkiparta, 2017)

3.1 Android

"Android is the first truly open and comprehensive platform for mobile devices. It includes an operating system, user-interface and applications – all of the software to run a mobile phone, but without the proprietary obstacles that have hindered mobile innovation."
(Harju, 2016, p.11).

At the current moment Android is the most common OS among smartphones. The development of Android can be interpreted to have started back in 2005, when Google bought a company called Android Inc. Android Inc. was founded back in 2003 and its goal was to develop more advanced mobile devices. After Google had bought Android Inc., one of the founding members (Andy Rubin) started to lead a project within Google; to produce a Linux based mobile device platform.
(Harju, 2016, p.11)

From the late 2006 to 2007 it was rumoured that Google was interested in the mobile industry. In 2007 a group of phone industry companies founded Open Handset Alliance (OHA) and Google was one of the founders. Android was the first concrete result of OHA.

In 2007 there was still no Android phones available. In 2008 T-Mobile operator published the first Android phone, which achieved popularity fast because of its cheap price and versatile operating system.
(Harju, 2016, p.11)

Open-source platforms and software usually have many versions and they update quickly. The following chart lists most important and early versions of Android

(Harju, 2016, p.12)

Version	API-level	Name	Publishing date	Some of the new features
1.0	1		23.09.2008	First version
1.1	2		09.02.2009	Debugging and small improvements
1.5	3	Cupcake	30.04.2009	Softkeyboard, videos, miniature programs
1.6	4	Donut	15.09.2009	Updated market and search, WVGA
2.0/2.1	5-7	Éclair	26.10.2009	HTML5, optimization
2.2.x	8	Froyo	20.05.2010	USB tethering, Flash support, optimization
2.3.x	9-10	Gingerbread	06.12.2010	UI-update, WXGA, NFC
3.x	11-13	Honeycomb	22.02.2011	Tablet version
4.0.x	14-15	Ice Cream Sandwich	19.10.2011	Updated UI, widget size customization, voice input engine
4.1/4.2	16-17	Jelly Bean	27.06.2012	Optimized to be faster, multiple user support

The first device running on Android was T-Mobiles G1 (Figure2), as mentioned. G1 had a follower a year later in 2009 when HTC published its own version called Magic, which also ran on Android. After this Android phones started to be more common and within the same year more than 10 different Android phones were published. In 2010 Android started to gain momentum and at least 44 different models were published from multiple different manufacturers. Popularity of Android kept rising and it quickly reached the top position among smart phones.
(Harju, 2016, p.13)

Figure 2 T-Mobiles G1 phone

The very first Android tablets were published in 2009, but they were not considered to be a challenger to Apples iPad until 2011. In February 2011 Motorola published its XOOM tablet and soon after Samsung released its Galaxy tablets.

Android is not just for phones and tablets. Android can be run on eReader devices, portable media players, fridges, TVs, wrist watches and many more types of devices thanks to its customizability.
(Harju, 2016, p14)

3.2 Advantages of Android

Android differs from other operating systems greatly in the manner that it offers an entirely open source code. Whereas other operating systems offer strictly closed or only limitedly open source code. This offers great customizability and even the freedom of making your own version of Android. (Harju, 2013, p.15)

Android software development does not require payments nor is it limited by approve processes. When publishing your software – the only payment that is required is the 25$ registration fee that Google Play™ -store requires. (Harju, 2013, p.15)

Play store being so open and effortless makes it easy for individual developers to publish their own applications. This has its up- and downsides. On the upside this broadens the spectrum of types of applications that are available for people to acquire, but it also can be a security risk. The end user of any application should practice caution when giving permissions for the application to use their phone.
(Harju, 2013, p.15)

3.3 Android studio

Android Studio is recommended and supported by Google. It has good materials provided by Google, which makes it a good platform to start learning Android development (Väkiparta 2017)

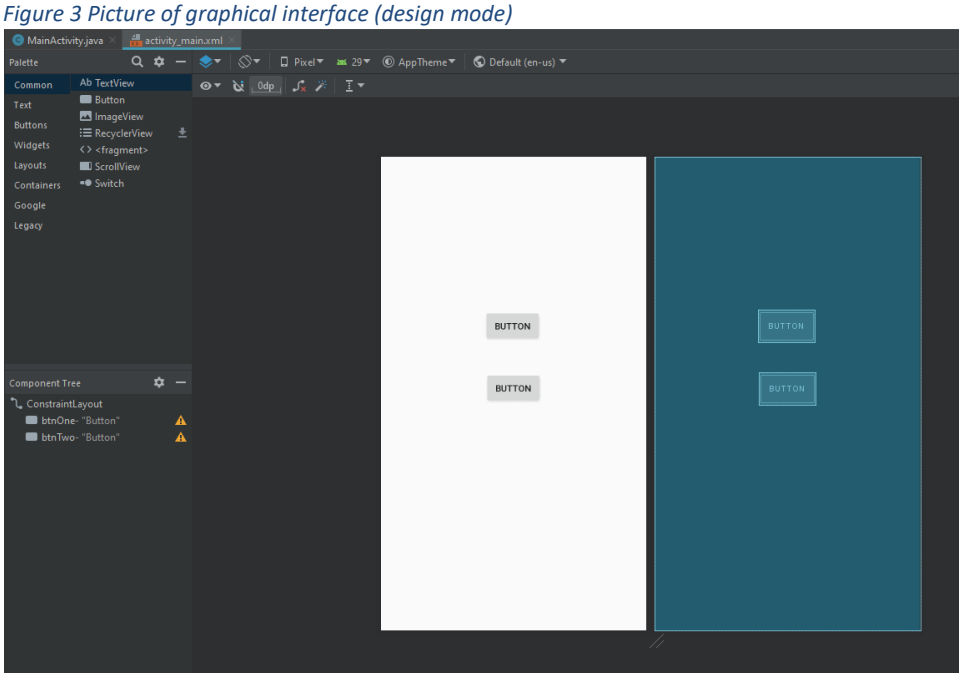
Google presented their new development environment for Android developers during their developers' conference in 2013. The new development environment is based on the "IntelliJ IDEA" Java IDE and it is intended to surpass and replace Eclipse. This new IDE is Android Studio. Android Studio comes with a powerful code-editor with integrated

functions such as “Smart Editing”. Smart Editing helps with ensuring that written code is legible. (Hohensee 2014)

Android Studio also has a built in “Gradle Build System” and it is going to be replacing the “Ant Build System”, which has been used before. (Hohensee 2014)

As is possible in Eclipse, it is also possible in Android Studio, that the layout of an app can be created in the text editor or alternatively in a graphical interface. This graphical interface is called “design mode” in Android Studio (Figure 3). Design mode is improved further in Android Studio. The app layout shows the layout for different Android versions, resolutions in the preview (Hohensee 2014)

Figure 3 Picture of graphical interface (design mode)

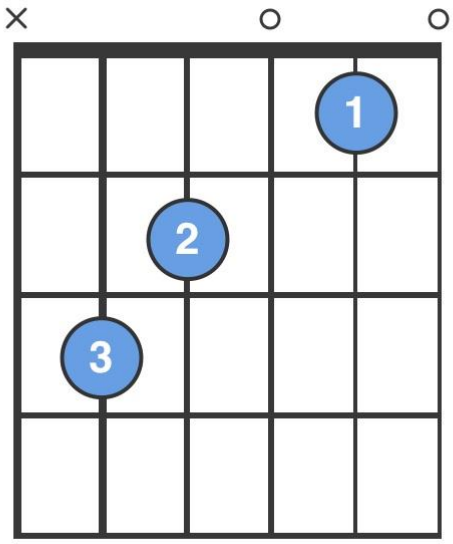


Like most of IDEs – Android Studios interface can be customized to your liking. By default, it holds all the file paths on the left, and they can be accessed by clicking. It is recommended not to change the default layout of the tools. This is mostly because almost every tutorial in the internet uses the default layout and this makes it easier to follow instructions. (Väkiparta 2017)

4 CHORDS AND THE FRETBOARD

This chapter discuss guitar chords in general, the guitar fretboard and how chords are marked in electronic form or tabbing in other words. This thesis will not go deep into music theory, but some of the basic principles of how chords are marked is necessary to be gone over. In figure 4 we have an example picture of a C major chord.

Figure 4 C major chord



The grid represents the guitar neck and can be read, so that the vertical lines represent the guitar strings. Low E being the left-most one and high e being the right-most one. E, A, D, G, b, e from left to right. The horizontal lines represent the guitar frets. The up-most one being the guitars nut – this can be interpreted by the thicker horizontal line, which is usually made thicker to represent the guitars nut. The one below that being the first fret and the one below that being the second, and so on. Usually if a chord is played from a different position, the fret number is usually marked with a number outside of the grid or the fretboard. In this thesis the focus is only on chords that are played from the first fret or the “default position.”

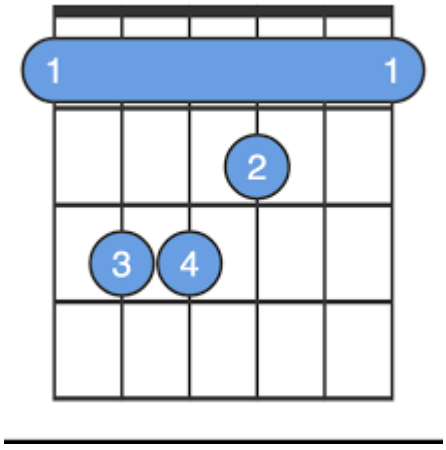
The X's and O's above the nut marker can be interpreted that an X means that the string is not played in the desired chord and thus is muted. The O symbols that the string is played open, meaning that no frets are pressed on that particular string.

The big blue dots signal that in that chord those frets are being pressed. E.g. in the C major chord, the A string is being pressed from the third fret, the D string is being pressed from the second fret and the b string is being pressed from the first fret. The number markings inside the dots do not mean the fret number, but they indicate the finger which is preferred to press the string. These numbers being 1 for index finger, 2 for middle finger and so forth.

If a chord needs every string to be pressed from one particular fret, it can be tabbed as a one line across all the strings as shown in the figure 5.

Tabbing visualization resembles an array quite closely. This is taken into consideration when making a visualization and the database for the application.

Figure 5 F chord



5 DEVELOPMENT

This chapter will go over the procedures that are needed to complete to create a new Android Studio project. This chapter will also tell a little about the activity windows that are the different pages of the application. To be able to save data, or chords in this case - a database is needed, so creating a database is gone over as well. What the database holds is a chord object, that is a set of values. Creating this object class is also gone over. Lastly adding the buttons to save and review the saved data is talked about and the UI of the application is shown.

5.1 Creating the project

When launching the Android Studio; a welcome window (Figure 7) should appear. From there the work is continued by clicking the “Start a new Android Studio project” button.

After that, a project template must be selected (Figure 7). For this application, the “basic activity” template was selected from under the phones and tablet tab. This template defines some basic attributes for your application like what basic functions the main window is going to have.

Figure 6 Welcome window

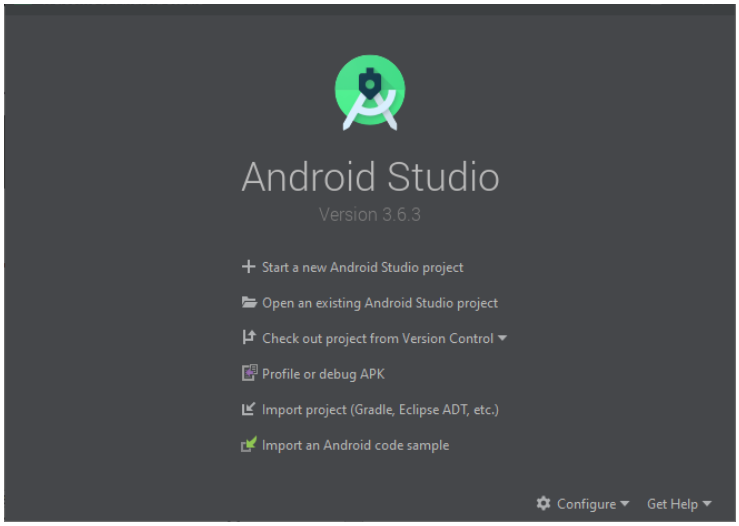
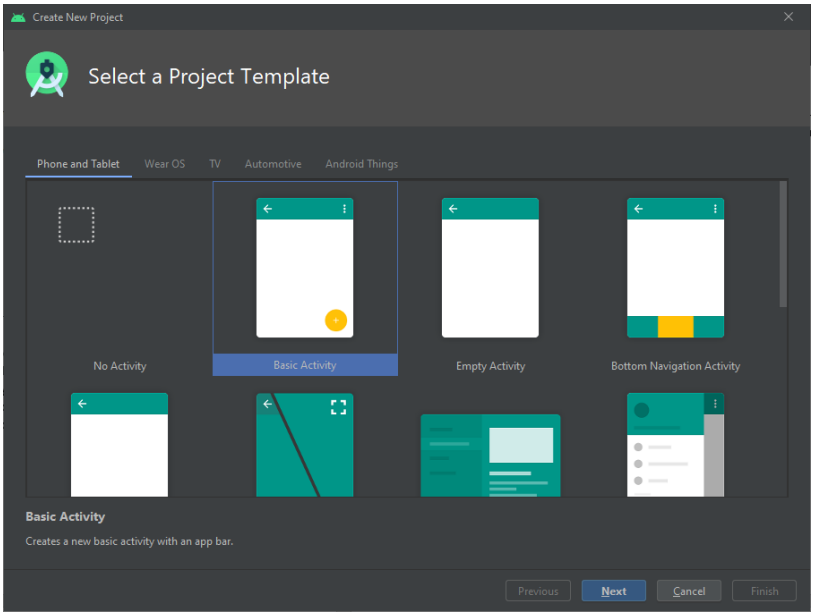
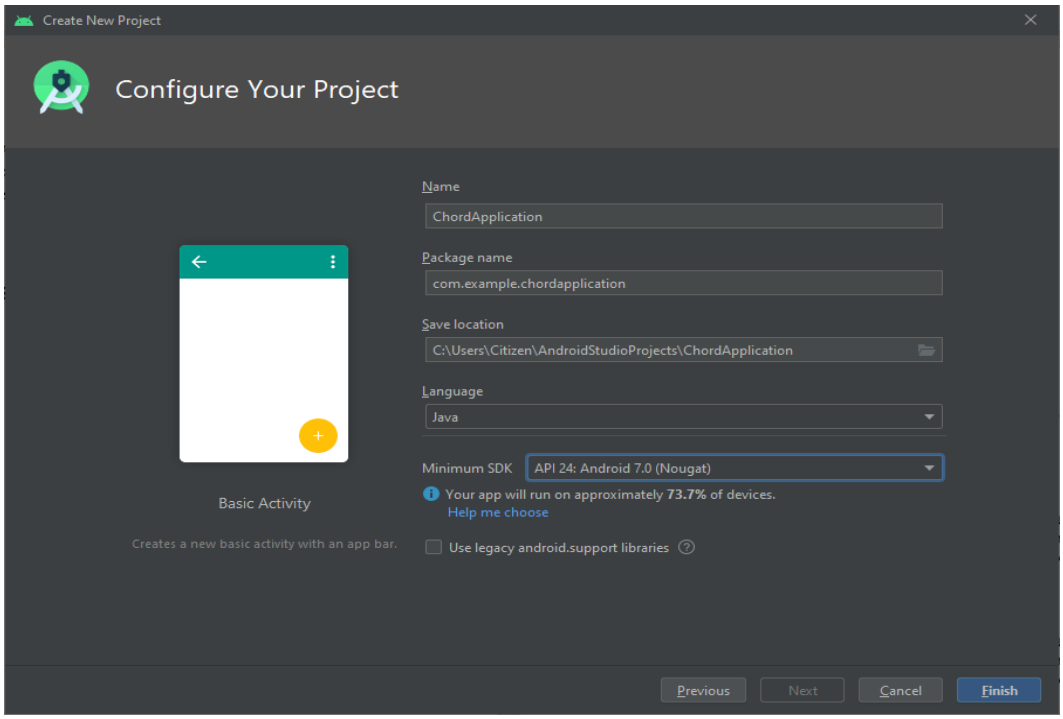


Figure 7 Select a project template -window



Next window is for configuring the project (Figure 9). Project name, save location, programming language and android version must be chosen. The project in the thesis was named descriptively as ChordApplication. Programming language was chosen as Kotlin. Other programming language option was Java, but it was decided not to be used in the thesis project. Android version 7.0 (Nougat) was chosen as the android version, since it was the same version that the physical phone had that would be used to test the application with.

Figure 8 Configure window



5.2 Creating the application

Creating the application was done step-by-step with looking up tutorials and googling solutions to problems.

First step was to create a few activity windows in which the few different functionalities would be placed. Three activity windows were created overall. A start screen with buttons to go to the other activity windows, which were an activity window for saving new chords and another activity window where the user can see all the saved chords. All this could have been done in one activity window, but then that window or page would have been very cluttered and not tidy. Very little planning went into the design of the activity windows as they did not include many functions.

After the activity windows were done it was important to create a chord class. It was important to know what attributes a chord class object should hold before creating any database functions. A chord class was given a

name and integer values to represent the frets where each string should be pressed to play the chord correctly. It was considered that the chord class should have a boolean value. This boolean value would have communicated whether the chord is a barre chord or not, but it was decided not to be include as an attribute since it would have brought very little extra value to the project.

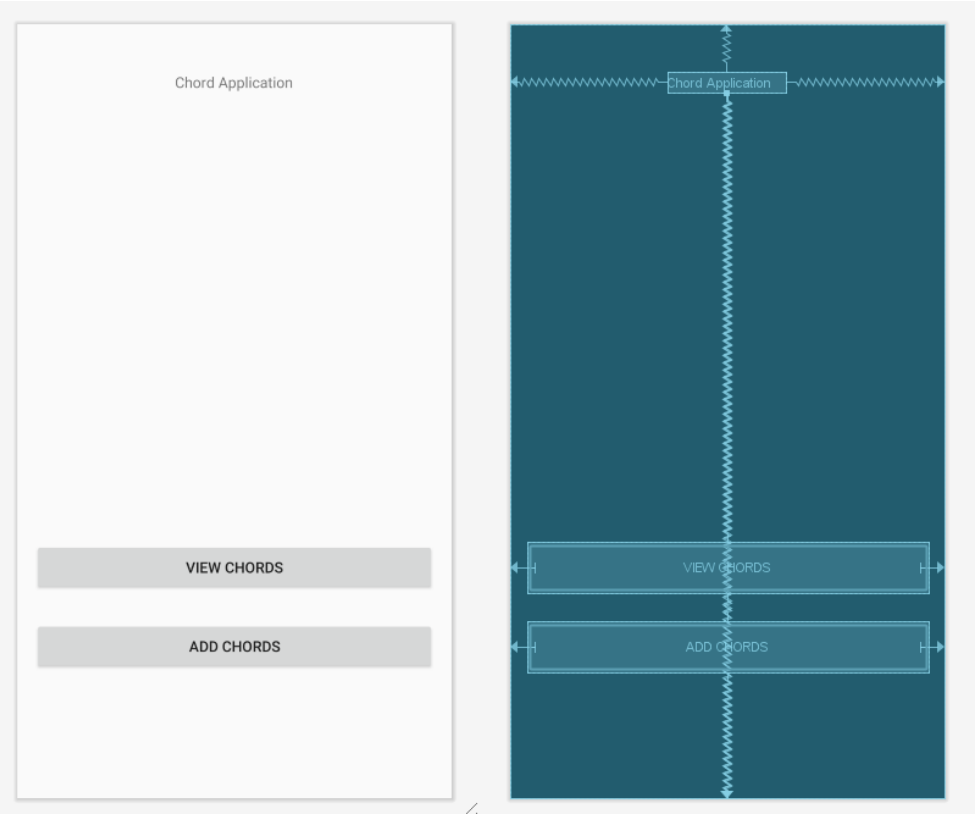
When the chord class and activity windows were done, it was only then left to do the database class. This was the most problematic part of the project since the author had only some experience working with SQL and no experience at all with SQLite with Android as an environment. Luckily, it turned out that SQLite worked very similarly to SQL.

5.3 Main Activity

First when creating the application, it was necessary to add buttons that lead to different windows with different functionalities. These buttons are shown below in Figure 10. Adding buttons was not necessary and the starting window or “main activity” -window could have included some functionalities, but to keep the windows tidy and the application clearer it was chosen to keep different functionalities separate from each other. Different functionalities being an activity window from which user can add chords to the database and a different activity window for viewing and deleting the saved chords.

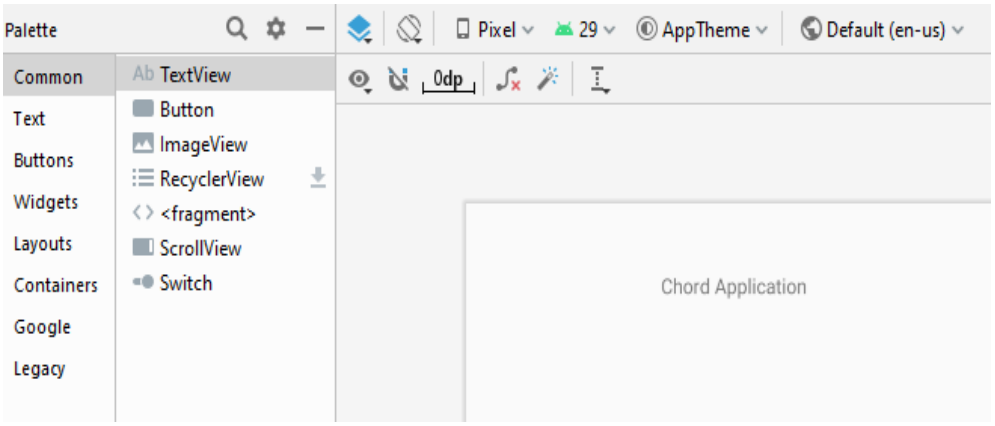
At the starting stages of development there was minimum effort towards the design of the application and so the layout looks rather bland.

Figure 9 Main Activity window in the designer view



Buttons are easily added to the window layout from Android Studios designer view by dragging them from the “Palette” tab to the window (Figure 11). Each button must be given restrictions. This is how they stay in their desired position in relation to other buttons and list views etc.

Figure 10 Palette tab in designer view



Each button is given an ID. By using this ID, they can be assigned to a variable inside the code. The variable is then given an OnClickListener method, which in a sense listens if the user clicks or pushes on the button. These OnClickListeners are shown in the Figure 12. When the button is pressed a new activity window is then called and set visible on the screen.

Figure 11 Code for the Main Activity window

```
public class MainActivity extends AppCompatActivity {

    Button btn_addChords, btn_viewChords;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btn_addChords = findViewById(R.id.btn_addChords);
        btn_viewChords = findViewById(R.id.btn_viewChords);

        btn_addChords.setOnClickListener((v) -> {
                Intent intent = new Intent( packageContext: MainActivity.this, AddChordsActivity.class);
                MainActivity.this.startActivity(intent);
        });

        btn_viewChords.setOnClickListener((v) -> {
                Intent intent = new Intent( packageContext: MainActivity.this, ViewChordsActivity.class);
                MainActivity.this.startActivity(intent);
        });
    }
}
```

5.4 Chord Class

In order to add chord objects to a database, it was necessary to build a chord class that defined what values different chords hold. Every chord object was given an ID, name and each string was named as if the imaginary guitar was in standard E tuning. These attributes are shown in the Figure 13.

Each string was given an integer value to point out the fret number where the string is supposed to be pressed. This was not the best solution because strings are not always played at all – depending on the chord. String can be given a 0 value which means that the string is played open, but the strings cannot be given an x value which would communicate that the string is not played at all. This is due to the integer value type.

Figure 12 Chord Class attributes and constructors

```
public class ChordClass {

    private int id;
    private String name;
    private int E;
    private int A;
    private int D;
    private int G;
    private int b;
    private int hE;

    //Constructor
    public ChordClass(int id, String name, int e, int a, int d, int g, int b, int hE) {
        this.id = id;
        this.name = name;
        this.E = e;
        this.A = a;
        this.D = d;
        this.G = g;
        this.b = b;
        this.hE = hE;
    }

    //Empty Constructor
    public ChordClass() {
    }
```

Chord Class was also given getters and setters so the attribute values could be set and fetched whenever necessary. A ToString method (Figure 14) was given to the class so the values could be printed out to the screen. This method was generated automatically and using Android Studios' features. It was necessary to manually modify it, so that the output was clearer and user friendly.

Figure 13 ToString method

```
@Override
public String toString() {
    return "ChordClass{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", E=" + E +
            ", A=" + A +
            ", D=" + D +
            ", G=" + G +
            ", b=" + b +
            ", hE=" + hE +
            '}';
}
```

5.5 Creating the database

SQLite database was used for this project mostly because Android Studio has built-in libraries to use SQLite and for its simplicity.

Each of the Chord Class attributes are given a column in the database where to store that attributes information when creating chord objects.

Database Class was given methods for when first time creating the database – a basic create statement. A method for upgrading the database version, but this method was not used in this thesis. And methods for adding a chord, a select all method and a delete method.

onCreate method, where the database is first created had basic SQL statement in which the table is created using predefined variables for each of the columns. These variables came handy later anytime when a column needed to be accessed.

addOne method, where a chord is added to the database used ChordCLasses get methods to add content values to each of the columns. A nullColumnHack (Figure 15) value is available for this method, which prevents totally empty rows to be inserted into the database. This is where it could be defined what to insert instead of an empty value. This could have been utilized to give “x” values to strings that are not played at all in a given chord. Alas it proved to be difficult to use and stays as a thing for future development. Biggest problems were to make the chords to their most readable form and to have a user-friendly delete function.

Figure 14 Function for adding a chord into the database

```
//Method for adding a new chord into database
public boolean addOne(ChordClass chordClass){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();

    cv.put(COLUMN_CHORD_NAME, chordClass.getName());
    cv.put(COLUMN_E, chordClass.getE());
    cv.put(COLUMN_A, chordClass.getA());
    cv.put(COLUMN_D, chordClass.getD());
    cv.put(COLUMN_G, chordClass.getG());
    cv.put(COLUMN_b, chordClass.getB());
    cv.put(COLUMN_hE, chordClass.gethE());

    long insert = db.insert(CHORD_TABLE,  nullColumnHack: null, cv);
    if(insert == -1)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Figure 15 Select All -function in code

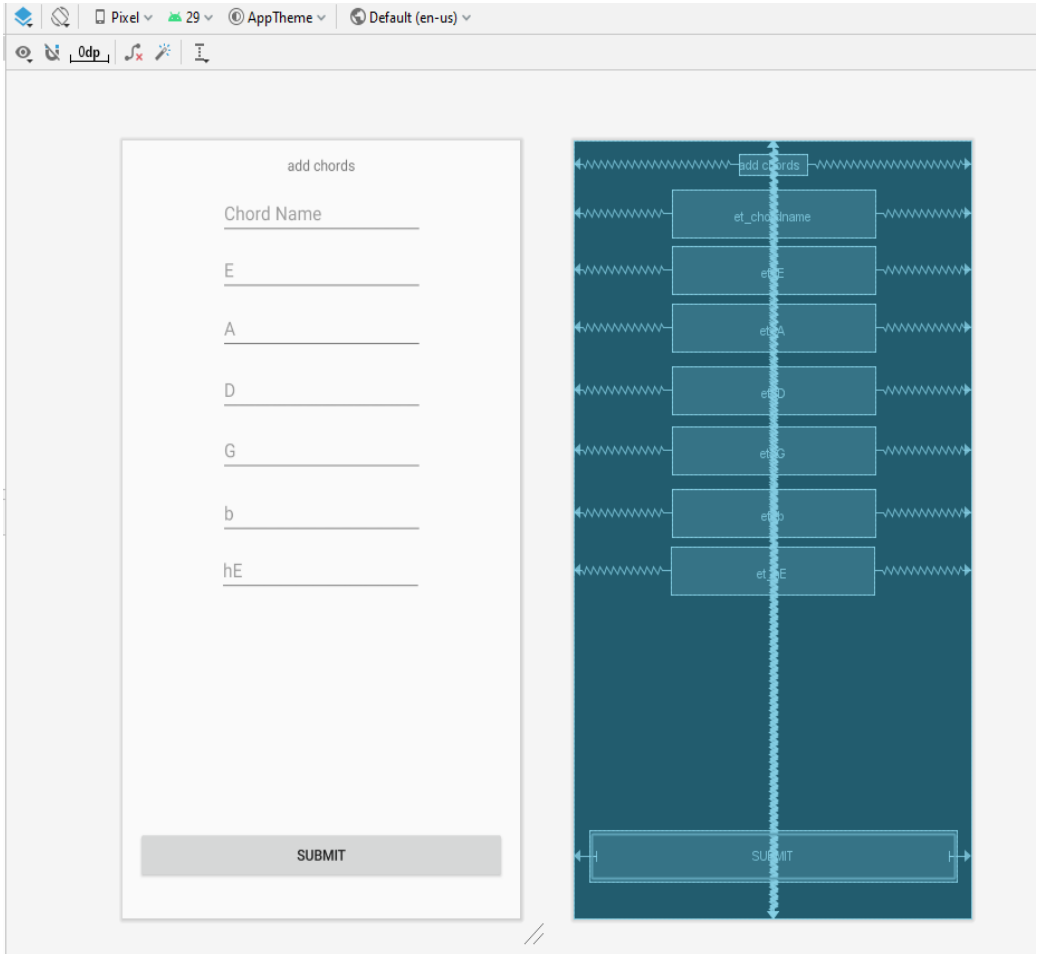
```
//Method for selecting * from database
public List<ChordClass> selectAll() {
    List<ChordClass> returnList = new ArrayList<>();

    //Get data from the database
    //Cursos variable type is equivalent for result set
    String queryString = "SELECT * FROM " + CHORD_TABLE;
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.rawQuery(queryString,  selectionArgs: null);

    if(cursor.moveToFirst()){
        //Loop through the cursor (result set) and create new chord object.
        do {
            int chordID = cursor.getInt( columnIndex: 0);
            String chordName = cursor.getString( columnIndex: 1);
            int chordEPosition = cursor.getInt( columnIndex: 2);
            int chordAPosition = cursor.getInt( columnIndex: 3);
            int chordDPosition = cursor.getInt( columnIndex: 4);
            int chordGPosition = cursor.getInt( columnIndex: 5);
            int chordbPosition = cursor.getInt( columnIndex: 6);
            int chordhEPosition = cursor.getInt( columnIndex: 7);

            ChordClass newChord = new ChordClass(chordID, chordName, chordEF
            returnList.add(newChord);
        } while(cursor.moveToNext());

    }
    else{
        //error. no actions
    }

    //close cursor and db when actions are complete

    return returnList;
}
```

The selectAll function (Figure 16) was made for getting all the saved chords from the database. It loops through the query result set which is called “cursor” in Android Studio using do while. While the cursor has a next result to move to it keeps going. A new chord object is then made out of these results and it is put to an array list, which is then returned at the end of the function.

5.6 Add Chords activity

Creating the addChordsActivity window (Figure 17) included first to add button for submitting the chord information and text fields where the information would be written down. The button and the text fields were then given restrictions so they would not wonder off but would rather stay in their desired positions.

Figure 16 addChordsActivity window in designer view



In the code the submit button and the text fields were then given references (Figure 18). The references would be named after their ID. The ID would then be used to find the button and the text fields from the activity window and the ID would be then attached to the corresponding reference.

Figure 17 button and text field references

```
//references to text fields and list view
EditText et_chordName, et_E, et_A, et_D, et_G, et_b, et_hE;

Button btn_submit;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_addchords);

    //Initializing buttons and text fields using references
    et_chordName = findViewById(R.id.et_chordname);
    et_E = findViewById(R.id.et_E);
    et_A = findViewById(R.id.et_A);
    et_D = findViewById(R.id.et_D);
    et_G = findViewById(R.id.et_G);
    et_b = findViewById(R.id.et_b);
    et_hE = findViewById(R.id.et_hE);
    btn_submit = findViewById(R.id.btn_submit);
```

Now each reference knows to what ID they are supposed to respond to. After this the button was given an onClickListener. The onClickListener responds to the submit button clicks. After each click a try / catch is called. Inside the try a new ChordClass object is created getting information from the text fields and then parsing it to a string in order to make it a similar type as the ChordClass requires.

If the creating fails, then the catch part of the phrase then makes a new ChordClass object with hard coded “error” values and informs the user that an error had occurred. An example why such an error would occur would be that one of the text fields was empty.

After all this a DataBaseHelper (Figure 19) is then used to call the addOne function from the Database class and it is passed the newly created chord object as a parameter.

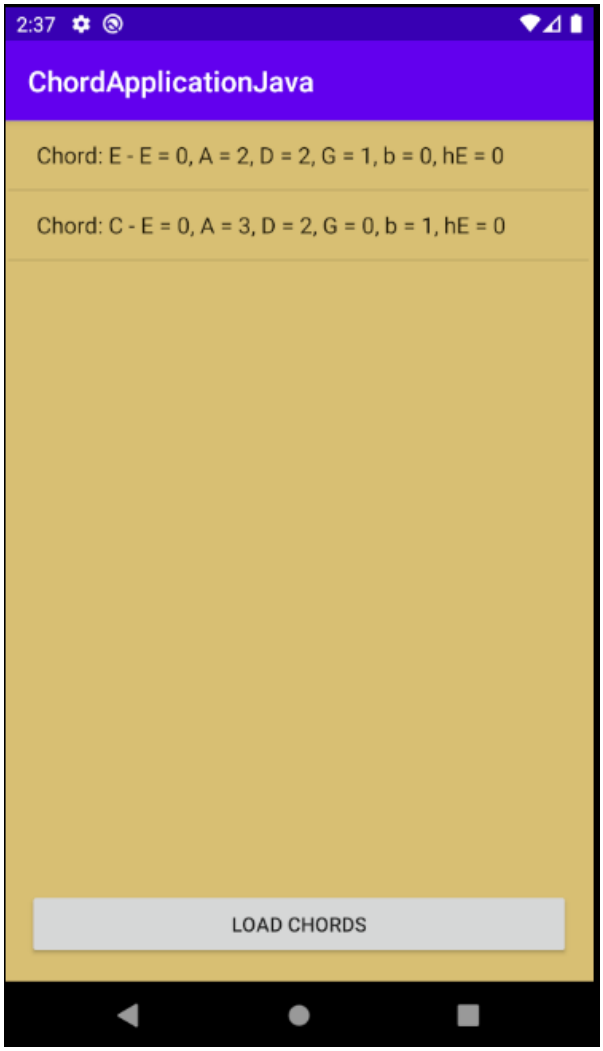
Figure 18 Try / Catch inside the onClickListener

```
//button listener for submitting chord information to database
btn_submit.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        ChordClass chordClass;

        try {
            chordClass = new ChordClass( id: -1, et_chordName.getText().toString(), Integer.parseInt(et_E.getText().to
                    Integer.parseInt(et_D.getText().toString()), Integer.parseInt(et_G.getText().toString()), Integer
            Toast.makeText( context: AddChordsActivity.this,  text: "Submitted", Toast.LENGTH_SHORT).show();
        }
        catch (Exception e){
            Toast.makeText( context: AddChordsActivity.this,  text: "Error, Check fields", Toast.LENGTH_SHORT).show();
            chordClass = new ChordClass( id: -1, name: "error",  e: 0,  a: 0,  d: 0,  g: 0,  b: 0,  hE: 0);
        }

        DataBaseHelper dataBaseHelper = new DataBaseHelper( context: AddChordsActivity.this);
        dataBaseHelper.addOne(chordClass);
    }
});
```

5.7 View Chords activity

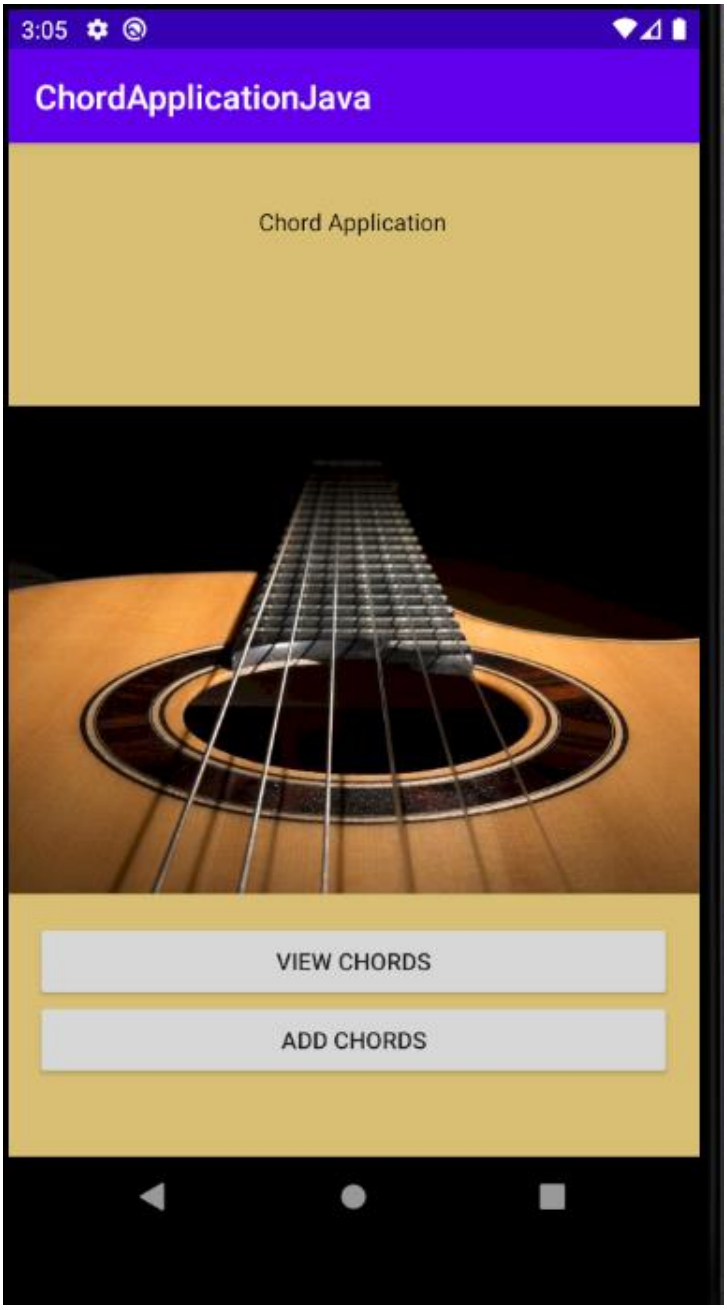
ViewChords activity window (Figure 20) is where the saved chords can be viewed by the user. This activity window also includes a delete function for when the user clicks on a chord – a delete function is called and the clicked chord is then deleted. As of this moment there is no text or a hint to inform the user of this function. This is something that should be added in future development for more user-friendly experience. The process of deleting a chord should also be made more complex rather than just clicking. For limiting accidental deleting.

Figure 19 View Chords activity window



Putting finishing touches on the activity windows to get them looking unified was the next step. Activity windows' colour was picked to match the background picture in the main activity window (Figure 21).

Figure 20 Main activity window run on virtual device



6 CONCLUSION / SUMMARY

The goal of the thesis was to get familiar with and learn about Android Studio and Android programming in general. A goal was also to produce a working application that can be ported to an android phone.

Project started somewhat slowly since almost every aspect was new and had to be studied. The only things that the author had previous experience with were Java programming language and virtual devices. Making the database was the most complex part of the project code-wise. This was overcome with the help of internet tutorials and previous experience in database programming. In terms of graphical design there was little effort, and this could have been something to have more focus on to make the application more user friendly and legitimate.

Before starting the project there were plans to make a graphical interface to display the chords. Unfortunately, this fell short due to lack of time and Android Studio knowledge. Graphical design could have been made an emphasis in the planning phase. A graphical layout planning could have been a part of the thesis and a software like figma could have been utilized to do so. Alas it was not to be and remains as something that is left for future development.

The thesis was a success in terms of developing a working application even though it is not ported on any phones and is only ran on a virtual device. A lot of knowledge and information about Android programming was gathered while working on the thesis.

7 REFERENCES AND APPENDICES

REFERENCES

Adelekan I (2018) Kotlin Programming By Example : Build real-wolrd Android and web applications the Kotlin way. Packt Publishing, Limited

Guitar-Chord.org. N.d. Viitattu 14.4.2020.
https://www.guitar-chord.org/chord-theory.html

Haavisto, H. (2017). Android-sovellus auton polttoainetalouden tarkkailuun.
https://www.theseus.fi/bitstream/handle/10024/124823/Haavisto_Henri.pdf?sequence=1&isAllowed=y //ONKO KÄYTETTY??

Harju, J. (2013). *Android-ohjelmoinnin perusteet.* Helsinki, Suomi: Books on demand GmbH.

Hohensee B (2014) Android for Beginners - Developing apps using Android Studio

Väkiparta, M (2017). Android sovelluskehityksen perusteet Android Studiolla.
https://www.theseus.fi/bitstream/handle/10024/123499/Matti_Vakiparta_Opinnaytetyo.pdf?sequence=1&isAllowed=y

Vesterholm, M & Kyppö. (2018). Java -ohjelmointi. Liettua, BALTO print.

Sqlite-org/about.html Viitattu 17.6.2020
https://www.sqlite.org/about.html

android.com Viitattu 24.8.2020
https://developer.android.com/kotlin

mediaan.com Viitattu 4.12.2020
https://www.mediaan.com/mediaan-blog/kotlin-vs-java

Appendix 1

APPENDIX HEADING