Programming applications on mobile phones

Vezan Nicoleta-Teodora

30433

Table of contents:

1.Introduction

1.1 Main purpose

The aim of this project is to develop 3 application that use the Android operating system. The complexity of the application will increase, the first one being as simple as a “Hello world”, whereas the last one should be more interesting. All three will be making use of the facilities that the SO offers and will mainly access hardware components, such as: battery, WI-FI, Bluetooth, location. In order to do this, we need a good understanding of the Android OS, both software and hardware perspective.

1.2 General

The need of breaking the barrier when it comes to human interaction and passing information easy and fast creates each day a new problem. Therefore, the need of development and creating new applications that satisfy the needs in that moment is crucial. Every revolutionary idea started with a problem that needed solving as soon as possible.

When it comes to communication for example, the starting point was the computer then the internet. But that was not enough. People wanted something portable. The mobile phones come along. People was satisfying for a short period of time. After that they wanted more and more. Nowadays, if you have a smartphone, you are connected to everyone all over the world. But people want to be comfortable, so they look for an application when they have a problem.

Considering this, the Android platform was built to allow developers to create mobile applications that use all the resources that a phone has to offer. An app can call any of the basic features of the phone, such as making calls, sending text messages, or using the camera. Android does not differentiate between the basic applications of the phone and those created by the developers. They can be built to have equal access to the phone's capabilities to give users a wide range of applications and services. Being an open source platform, it will evolve continuously through incorporating the latest technologies.[1]

[1] Android Developers: < https://developer.android.com/about>.

1.3 History[2]

The android is software that was founded in Palo Alto of California in 2003. Its four founders were Rich Miner, Nick Sears, Chris White, and Andy Rubin from here, the operating system has developed a lot in the last 16 years.

Nowadays, the use of Android mobile phone is drastically increasing, and the people have an easy access to it. They are starting using phones for much more than a call, a text. This change in perspective when taking about phones was anticipated by Google as far as 2005. Therefore, starting from august 2005, Google bought the initial developer of telephones software, Android Inc. Google teamed up with other companies to form the Open Handset Alliance (OHA), which has become responsible for the continued development of the Android OS.
Since October 21, 2008, Android has been available as Open Source, so Google opened the entire source code under the Apache license. Under this license manufacturers are free to add proprietary extensions without making them available to the open source community while Google's contributions to this platform remain open source.

The firs version of the operating system of Android was launch in 2008. Android 1.0 was far less developed the ones today. However, there are a few similarities. Many say that Android pretty much nailed how to deal with notifications. The pull-down notification window, which can be found even nowadays in the smartphones, was far better than the notification system in IOS.

After this one may versions, with funny names were release, for example: Android 1.5 – Cupcake, Android 2.2 Froyo, Android 50 Lollipop, Android 8.0 Oreo. Each one of them has come up with something new that people loved.

Now, the Android ecosystem is an open source platform for mobile, embedded and wearable devices. Although google is the principle maintainer, anyone can contribute to the system. Android us customized by each device manufacturer to suits their needs.

[2] The history of Android OS: its name, origin and more https://www.androidauthority.com/history-android-os-name-789433/

2.Objective

This project tries to implement and develop three application based on the Android operating system. These will make use mainly on making the experience of a user more enjoyable and more useful. These will be done using Java as programming language. I will try to present next the three applications and their purpose.

2.1 “Hello world” application

This is the simple application. With this I intend to design a button, that when clicked will display a quote. The main objective here is to be familiarized with designing an Android application.

2.2 “Do not disturb” application

The main objective of this app will be to help the user to not worry about being disturb during an important meeting, when sleeping, or when something important is happening. The user will be able to set a period, when the sound, the wi-fi and Bluetooth will be stopped.

2.3 “Battery life” application

No one likes when you find your phone under 10% when you need to find a location for example, Therefore the aim of this application is to provide the user a way in which he or she can know when the battery is under a limit they will put. They can choose if they want their phone to ring or the light to turn on and off when this happens.

3. Theoretical Consideration

This chapter gives an overview of both the Android platform from the point of view hardware as well as software. Although some components vary from one manufacturer to another, the structure is common to each of these devices.

The phone can look like a minicomputer, featuring an ARM processor and two types of memory, combined into a single component, called a multi-chip package (MCP). The connection to the mobile network is made by the existing baseband modem and radio, connecting to the Internet by activating the Wireless option, and to other phones or external devices by enabling Bluetooth or USB. In addition to these we find a performance room, a touch screen and GPS tracking system.

The operating system is based on the Linux 2.6 kernel and provides the necessary software to coordinate both the hardware and Android applications. Libraries are accessed by programmers through the existing API, and running the programs is performed on Dalvik virtual machines.

From a structural point of view, an Android application is composed of two main elements: classes describing the functionality of the program and resources consisting of aspects, images, strings characters or colors that describe the program interface. Defining the structure and metadata of the application, its components and requirements are specified in the manifest file of the application, file without which an application cannot run.[3]

3.1 Software

3.1.1 Kernel

Android is a Linux based operating system, mainly for touch screen mobile devices such as smart phones and tablet computers. This provides basic system functionalities, such as memory management, process management, device management like camera, keypad, display and so on. The kernel itself does not interact directly with the user but rather interacts with the shell and other programs as well as with the hardware devices on the system.

Android 8.0 introduced a modular kernel, splitting the device kernel into a system-on-chip (SoC), device, and board-specific deliverables. This change made it possible for original device manufacturers (ODMs) and original equipment manufacturers (OEMs) to work in isolated, board-specific trees for board-specific features and drivers, enabling them to override common kernel configurations, add new drivers in the form of kernel modules, etc.[3]

[3] Android (Operating System),Wikipedia<https://en.wikipedia.org/wiki/Android_(operating_system)>.

3.1.2 Libraries

The on top of Linux kennel there is a set of libraries and APIs written in C, as well as software applications running on platforms that include libraries compatible with Java. These libraries are used to play and record audio and video. Some examples of such libraries are: the SQLite is a data base which is useful for storage and sharing of application data, the SSL libraries are responsible for internet security. It is based on Apache, a free-software and open source license, which makes it attractive among developers.

When you develop an app on a latest version of android like 5.x and you also want it to run on those devices which are running older versions of android like 3.2 etc.. you can't do that until you add backward compatibility to your code.

To provide this backward compatibility android provides you the **Android Support Library** package. The Android Support Library package is a set of code libraries that provide backward-compatible versions of Android framework APIs as well as features that are only available through the library APIs. Each Support Library is backward-compatible to a specific Android API level.

Including the Support Libraries in your Android project is considered a best practice for application developers, depending on the range of platform versions your app is targeting and the APIs that it uses.[3]

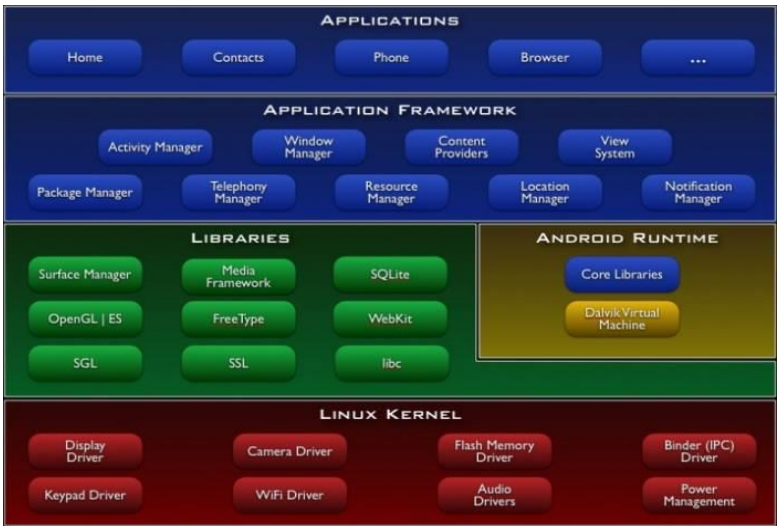
3.1.3 Android runtime

There is another important aspect when taking about the Android operating system, that is android runtime. The android runtime provides a key component called Dalvik Virtual Machine which is a kind of java virtual machine. It is specially designed and optimized for android. The Dalvik VM is the process virtual machine in the android operating system. This is a software that runs application on android devices. This VM makes use of Linux core features like memory management and multithreading that are characteristics of java language. Moreover, due to this Dalvik VM every android application runs its own process.

3.1.4 Application framework

The application framework layer provides many higher-level services to applications such as windows manager, view system, package manager, resource manager etc. The application developers can make use of these services in their application.

[3] Android (Operating System),Wikipedia<https://en.wikipedia.org/wiki/Android_(operating_system)>.



3.2 Hardware

Android is designed to be compatible with a wide range of hardware components. Due to the fact that is a platform for developing applications for mobile phones, it has been designed to have a efficient management of resources and existing memory. Most Android devices have been using processors ARM since the beginning. They are known to be quite powerful, when we refer to the low energy consumption. However, some corporations that have ported Android to other platforms have chosen to use their own CPUs, such as Intel who uses Atom processor.
There are two types of memory in smart phones: RAM (random access memory) and NAND (non-volatile flash memory). RAM is used by system to load, execute and manipulate different parts of the system OS. Whereas, NAND mainly stands for storing the OS as well as the data desired by users. Due to the limited space, the two types of memory are combined in a single component known as Multi-Chip Package (MCP).[4]

One of the most interesting hardware components is the GPS (Global Positioning System). This does not just identify the location of the phone, using GPS satellites, but it also offers applications the ability to access and use this information. In addition to this, Android allows internet connection through activation of Wireless option, as well as connecting to other phones or external devices, by activating the option Bluetooth.

The screen is the interface of the OS for interaction with the user. It is in continuous development, with the aim of increasing the resolution, brightness, also the ability to respond to external stimuli. Another hardware component is the camera. Most devices combine its functionality to record next to the data and time at which a photo or video was taken and GPS coordinates.[4]

[4] Android hardware components<https://subscription.packtpub.com/book/networking_and_servers/9781789131017/1/ ch01lvl1sec15/android-hardware-components>

Battery life is still a problem, due to the large existing apps energy consumers. This is desired to be overcome as soon as possible, the research is aiming at making stronger batteries, or charging during movement.[4]

Android can detect and change the position of the screen, depending on how the phone is held or rotated. This is possible due to the existence of an accelerometer that detects these changes. The new versions also support a gyroscope, which is much more sensitive.[4]

More hardware components are the speaker and microphone of the phone. Some devices have two or three microphones, which combined with Android software can detect and eliminate background noise, in order to provide better sound quality.

3.3 The components of an Android application

An Android application is an installable unit that can be started and used independently of other applications. It can have only one class that is instantiated as soon as it is started application and is the last component that is executed when the application is stopped.

An Android application is made up of software and resource files. Components of an Android application can access the components of another application based on one task descriptions (Intent). This way you can create tasks executed between applications. Integration of these components can be made so that the application runs flawlessly even if the components additional components are not installed or there are different components that perform the same task.[5]

Here are the basic components that are used in an application:

1. *Activities* - Activity-type components are components responsible for presenting a graphical user interfaces, registration and processing of user orders. An application can have several Activity components, one of them being considered a main activity.[5]

   Each activity type object has a life cycle that describes its state activity at any given time:

   - Active state (Running). The activity is displayed on the screen of the phone, the user interacts directly with the activity via the device interface.

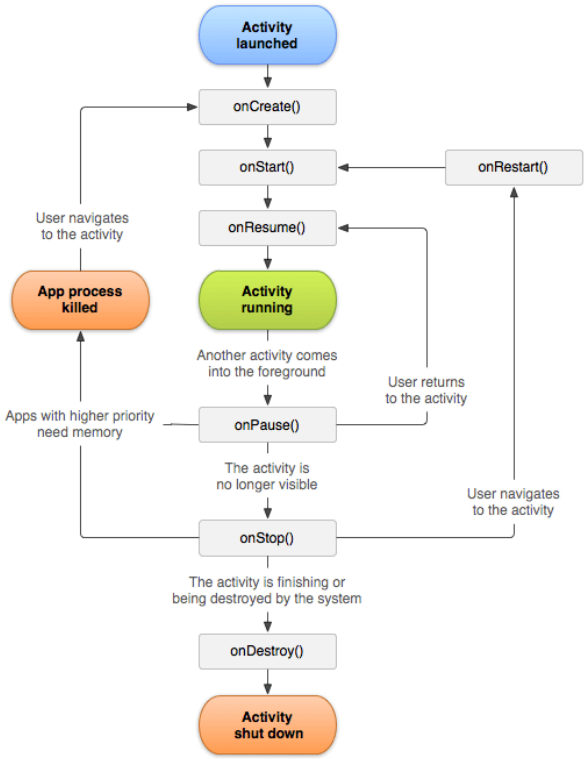
   -pause (Paused). The activity is no longer in the foreground, the user is gone interact with the application.

   - Stopped. The activity is no longer used and is no longer visible. The activity must be restarted before it can be reactivated.

   - Destroyed. The activity is destroyed, and memory is released because it is no longer needed.

Only one activity can be displayed in the foreground at one time. It is the system that manages the states and transitions. It will be announcing when the status of the current activity changes or another application is launched. The following methods are used for transition events:

- **onCreate (Bundle)** - is called when the activity is created. Using the argument, The Bundle offers the ability to restore the saved state in a previous session. .[6]
- **onStart ()** - the method is called when the activity is to be displayed in the foreground.
- **onResume ()** - is called when the activity is visible, the user can interact with it. .[6]
- **onRestart ()** - is called when the activity returns to the foreground from a stopped state.
- **onPause ()** - the method is called when another activity is brought to the foreground, the current activity being moved to the background. .[6]
- **onStop ()** - the method is called when the activity is no longer used, the user interacting with another activity. .[6]
- **onDestroy ()** - called when the activity is destroyed, the memory is released. .[6]
- **onRestoreInstanceState (Bundle) -** called if the activity is started with the data from a previous state. .[6]
- **onSaveInstanceState (Bundle)** - method is called to save the current state of activity.[6]



[6] Understand the Activity Lifecycle <https://developer.android.com/guide/components/activities/ activity-lifecycle>
[5]Introduction to Activities <https://developer.android.com/guide/components/activities/intro-activities>

2. *Intent* - Intent objects are asynchronous messages that allow applications to request features from other Android components. With the help of Intent objects, it is possible communication during running with various components either inside the application or localized in other applications. Among the components that can be called through Intent objects counts services, activities, etc. An activity can call a component directly (Explicit intent) or ask the system to evaluate the components recorded based on data from Intent (Default Intent).

3. *Services*- A Service component is a component that runs in the background, without interaction direct with the user and whose life cycle is independent of that of other components. Once it started, the service implicitly executes its tasks within the main thread that it must do even if the component that initially started it is destroyed. The service is used when the application must perform a long-lasting operation that does not interact with or to provide functionality for other applications.[7]
4. *Content Provider* - A Content Provider component is an object within an application that offers structured interface to the application data. With this help the application can share data with other applications. The Android system contains an SQLite database where the data that will be stored can be stored accessed using Content Provider components. Shared data can be images, text files, video, audio.[7]

5. *Broadcast Receivers* - An ad recipient type object is an Android component that receives type messages broadcast. These messages can be transmitted either by other applications to announce the completion / start an operation, either system to announce the change of system parameters (battery, memory, signal, etc.). Broadcast messages are usually Intent objects.[7]

   There are two major classes of messages that can be received:

- **Normal messages** (sent with context.sendBroadcast). These are completely asynchronous, and they are transmitted in a random order, often at the same time. This is more effective though the results cannot be used by the receivers.

- **Ordered messages** (transmitted with Context.sendOrderedBroadcast). They are delivered one to one receiver. As the receiver executes the code, the result can be propagated to the next receiver or the Broadcast can be completely dropped and thus the result cannot be transmitted to another receiver. The order in which they are transmitted is controlled by a named attribute priority (android: priority attributes). Receivers with the same priority will be run in an order arbitrary.

6. *Widgets* - A snippet represents a behavior or portion of the user interface in the framework of an activity. Multiple fragments can be combined into one activity to build a multi-panel interface or you can reuse a fragment in multiple activities.[7]

   Fragment type objects have their own life cycle. A fragment must always be incorporated into an activity, the life cycle of the fragment being directly affected by the cycle of life of the host activity. For example, if the activity is stopped then the included fragments in it are interrupted. The life cycle of the fragment differs from that of the activity when the activity is running, each fragment can be manipulated independently (add, delete, fragment modification, etc.).

   As a structure, the fragment type component is very similar to the Activity type component in their framework being found in addition to specific methods and onCreate (Bundle), onStart () methods, onResume (), etc.

7. *Manifest File* - Each application must have an AndroidManifest.xml file in the root directory. This file contains information about all the components, permissions, services and libraries used in-app. [7]

   A permission is a restriction that limits access to data or part of the code to protect the user's confidential data. Each permission is identified by one unique label. Often the label indicates the action that is limited.

   The following list provides a perspective on the nodes that can be specified in the manifest file and their role in the construction of the application.

   - uses-sdk marks the SDK version that must be available on an Android device for the application to work as well as the version on which the application was made and tested
   - uses-configuration specifies a combination of the input mechanisms supported by application, referring to the available keyboard, navigation mode in a window, and in touch mode.

   - uses-feature illustrates the hardware requirements of the application and prevents its installation on a device that does not meet these conditions

   - supports-screens lists the screen sizes that the application supports

[7] Application Fundamentals<https://developer.android.com/guide/components/fundamentals>

- application is a unique node that specifies the meta date of the application as well as the components it constitutes a container for nodes such as activity, service, receiver or
- activity is a necessary node for each activity running inside the application, each node of this type having the possibility to specify through the son node

- service designates a service from the application

- Provider specifies each of the application's data providers

- The receiver has the role of recording a Broadcast Receiver, without the need to launch application first

8. *Processes and threads* - When starting an application, a single-threaded Linux process is automatically created, and it is called main thread. In this process all are executed the components and instructions associated with them. If a component is started when there is already a main process, then the component will run in the already existing process. In case of the application must perform a long-lasting operation or an operation that would affect the performance it is associated with a new process that meets the respective set of instructions, thus creating additional threads for any process. [7]

   Processes are executed over a longer period; they are stopped when the system needs memory or has higher priority processes to execute. To determine which process needs to be stopped, the system organizes all processes according to importance.[7]

9. Android resources - An Android application is made up of files containing source code and resource files. The resources are separated by the source code and represent a collection of videos, audio, images, text used to create the richest visual interface. Their access is through source code. The separation of resources allows the developer to create interfaces adapted to the different device configurations.[7]

[7] Application Fundamentals<https://developer.android.com/guide/components/fundamentals>

4.Design and implementation

4.1 Setup the environment

1. Install Android Studio - https://developer.android.com/studio/install

Android Studio, it is a free, simple and open source IDE for developing android applications.
It is the official one and it is written in Java and has an integrated gradle build system.

2. Download oracle JAVA JDK - https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

After installing, you must add the system environment variables in order to work properly.
Go to System Properties → Environment Variables → System Variables → look for Path and add here the memory location for the Java JDK

3. Check if the emulator is working

Go to Android Studio → Tools → Android → SDK Manager → Appearance & Behavior → System Settings → Android SDK → SDK Tools → check if Intel x86 Emulator Accelerator is clicked
Now that this is setup is finished and you can create a virtual device. To do this choose an Android Version, then choose a phone → Edit the details → Finish.

4. Test the application

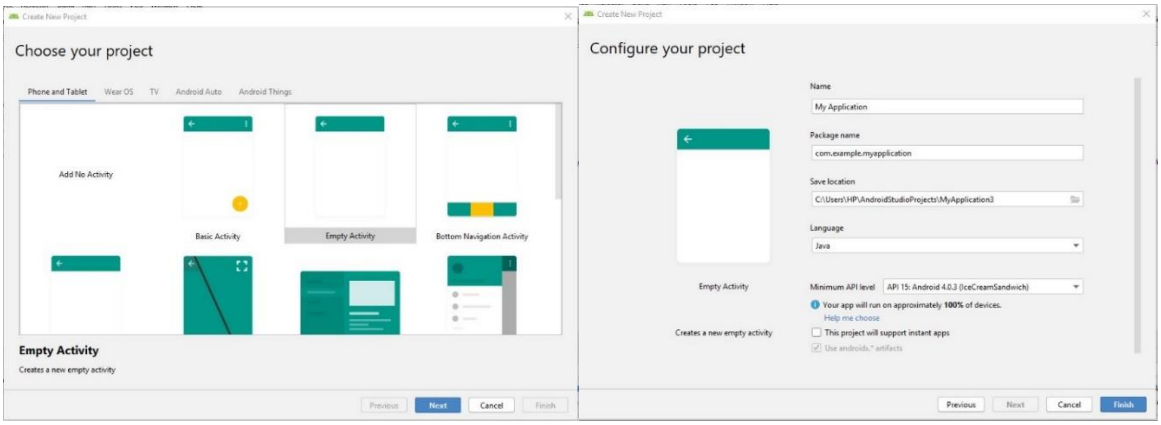
In order to test your application on your physical device, you need to set your phone on Developer Mode.

Go to Settings → About Phone → Build Number → click on it seven times, also you have to select the USB debugging), then plug in the phone into your computer with the property to transfer files and all good.

4.2 “Hello world” application

In order to make your very first Android application, we need create a simple project in Android Studio.

File → New → New Project → Next → Finish



On the screen we want to have a field where the user can introduce his/her name, and after pressing the button, the application will greet the user with a message. In order to create the Text View where the name should be introduced and the button that is needed to pe pressed we need to edit the “activity_main.xml” file. Here it’s described how the first view looks like; we’ll be using a constrained Layout. [8]

activity_main.xml | MainActivity.java

```
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="100dp"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.811"
        app:layout_constraintStart_toStartOf="parent" />

    <EditText
        android:id="@+id/nameBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="124dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.297"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />
```

[8] Build a simple user interface <https://developer.android.com/training/basics/firstapp/building-ui>.

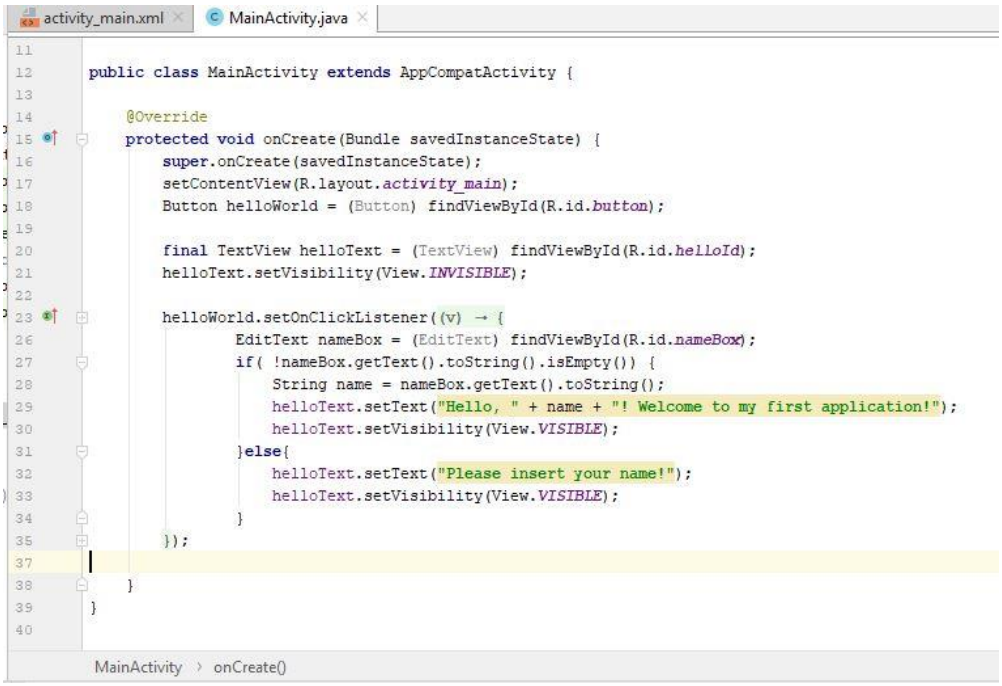
Here we have specified the type of the component, and the constraint that are necessary in order to specify the position in the page. Each component must have a unique id.

Now, in our “MainActivity.java” we have a “*onCreate*” method, where we initialize the activity of this view, the TextView and the Button. To create them I have used *findViewById();*

Followe by this is the method *setContentView(int layoutid)* which is used to show the layout on screen. Its argument R.*layout.activity_main* is an integer number implemented in nested layout class of R.java class file. At the run time device will pick up their layout based on the id given in setContentView() method.

After we initialized them, we set the visibility to the message to false, so that it will appear only after the button is pressed. We did that using the method → *setVisibility();*

Now that we set everything, we will implement the method which will be activated when pressing the button; *setOnClickListener(lambda expression);* In this method we will specify the behavior of the view, and that is, if ta name is introduced in the field and the button is pressed, the message will appear, if the name is not introduced a message saying “ Please insert your name” will appear.



Now, all we must do is connect a phone and press the run button.

4.3 “Battery life” application

In this application we take the level of battery and the status of the phone(charging, not charging, full) and we print it on the screen. There is also an imagine that will change depending on the level of the battery.

In the “activity_main.xml” file I will add the main changes in order to create a certain view for this application, I will two TextViews and an ImageView in which I will display a battery picture. I have defined the constrains regarding the top, bottom, left and right part of the view. And I have also define the size of the image that will be displayed on the screen.

activity_main.xml | AndroidManifest.xml | BatteryReceiver.java | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/fullB"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_marginTop="100dp"
        android:layout_marginEnd="76dp"
        android:layout_marginRight="76dp"
        android:scaleType="centerInside"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.866"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:srcCompat="@drawable/battery1" />
```

I have implemented two classes; one is the main class and one is the one where I have implemented the methods for the battery management.

I will start with the BatteryReceiver. I have defined the text views and the image. Than using the method “onReceive”, this method is called when the BroadcastReceiver is receiving an Intent broadcast. In has as arguments the context in which the application is running and the intend that is received.

Now we must write the code which will take the level of the battery and the status of the phone. For the battery level I have used the intend and the static class BatteryManager. I have got the level and the scale. There are some macros defined in this class ***EXTRA_LEVEL, EXTRA_SCALE. [9]***

[9] Monitor the Battery Level and Charging State<https://developer.android.com/training/monitoring-device-state/battery-monitoring

```
public int bateryLevel(Intent batteryIntent) {
    int level = batteryIntent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
    int scale = batteryIntent.getIntExtra(BatteryManager.EXTRA_SCALE, -1);

    return (level * 100 / scale);
}
```

Start by determining the current charge status. The BatteryManager broadcasts all battery and charging details in a sticky Intent that includes the charging status.[9]

First we take the action from the intend with the *getAction().* If the action that results from this is not null and is part of the field then we take the status from the intent using the values for "status" field in the ACTION_BATTERY_CHANGED Intent: [9]

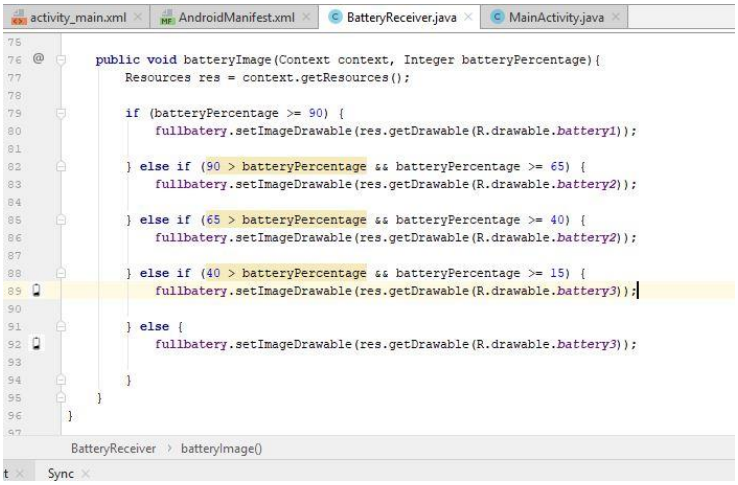
BATTERY_STATUS_UNKNOWN, BATTERY_STATUS_CHARGING, BATTERY_STATUS_DISCHARGING , BATTERY_STATUS_NOT_CHARGING, BATTERY_STATUS.

Then with a case we set the message.

```
if (action != null && action.equals(Intent.ACTION_BATTERY_CHANGED)) {
    int status = batteryIntent.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
    switch (status) { // case for message

    }
```

In order to set the image according to the level of the battery, I have saved in the drowable folder three different picture. Than I have set the ImageDrawable, using *setImageDrawable(),* according to the level of battery and the three picture that I have saved. Depending on the level the battery is I have defined some intervals: higher than 90, between 90 and 65, less than 65 → 40, less than 40 → 15 and lower than 15. To access the images I have use the *getResources()* method from the context. To get the actual image I have used the *getDrawable()* method.

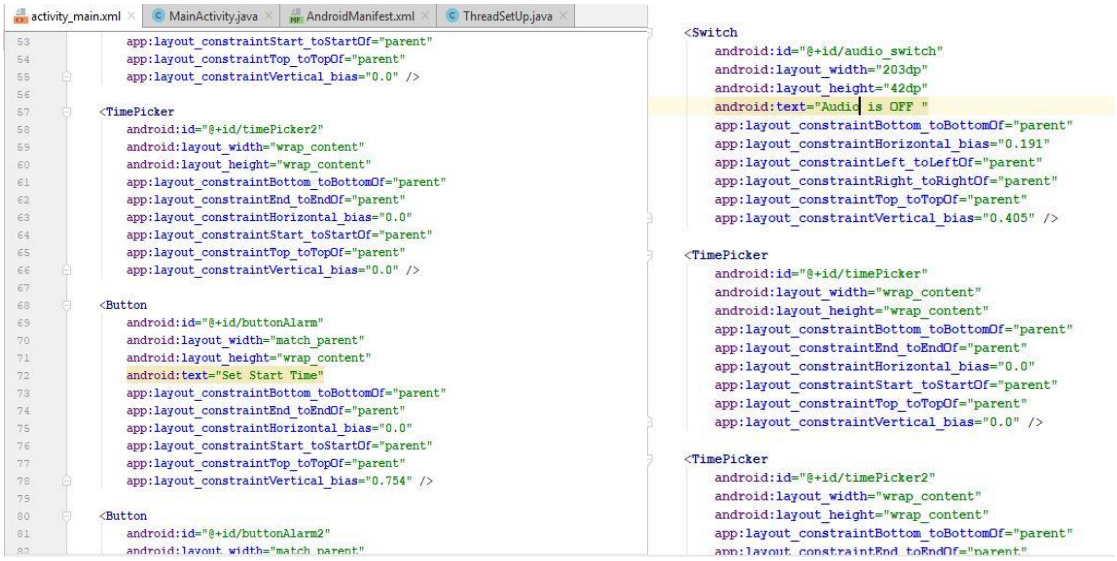


After this, in the main class I have implemented the three overwrriten methodes: onCreate(), onPause(), onResume(). In the method “onCreate” I have initialize the view. After that “onResume”,I register a BroadcastReceiver, which is the BatteryReciver, to be run in the main activity thread. And “onPause” I have unregister the previously registered BroadcastReceiver, the BatteryReciver and all filters that have been registered for this BroadcastReceiver will have been also removed.

4.4 “Do not disturb” application

In this application the user can select an interval of time in which the Bluetooth, Wi-Fi and sound will be turned off, after the interval has passed, they will be turned on again. The user can also separately change the state of the Bluetooth, Wi-Fi and sound as he/she wants.

In the “activity_main.xml” file I will add the main changes in order to create a certain view for this application, I will three Buttons, three Switches and two TimePickers. The Switches will be used to change the states of the Bluetooth, wi-fi and sound. The TimePickers are used to help the user select an interval. I have defined the constrains regarding the top, bottom, left and right part of the view. And I have also define the size of the image that will be displayed on the screen.



In order to make use of the Bluetooth, wi-fi and sound, I have access some classes that are already implemented in Android Studio, that is:

```
protected static WifiManager wifiManager;
protected static BluetoothAdapter bluetoothAdapter;
protected static AudioManager audioManager;
```

I have made them protected and static because I wanted to access the same object from another classs in the same package.

I have implemented two classes, one is the MainActivity, where I have Overwrite the onCreate(), onStart(), onStop(), and the BroadcastReceiver for the three components. In the other class that I have named ThredSetUp, I have created a thread by implemteding Runnable and then Overwriting the method run().

```
public class ThreadSetUp implements Runnable {
```

The run() method keeps track of the interval of time and changes the settings of Bluetooth, wi-fi and sound, whether we are in the interval or not. The run method will terminate when the interval is over.

```
@Override
public void run() {
    while(enable) {
        actualDate = new Date();
        if ((actualDate.after(startDate) || actualDate.equals(startDate)) &&
actualDate.before(endDate)){
            MainActivity.wifiManager.setWifiEnabled(false);
            MainActivity.wifiSwitch.setText("WiFi is OFF");
            MainActivity.bluetoothSwitch.setText("Bluetooth is OFF");
            MainActivity.bluetoothAdapter.disable();
            MainActivity.audioManager.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
            ok=0;
```

```
} else {
    MainActivity.wifiManager.setWifiEnabled(true);
    MainActivity.wifiSwitch.setText("WiFi is ON");
    MainActivity.bluetoothSwitch.setText("Bluetooth is ON");
    MainActivity.bluetoothAdapter.enable();
    MainActivity.audioManager.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
    MainActivity.audioSwitch.setText("Auto is ON")
    if(ok !=1)
    enable = false;}
try {
    Thread.sleep(3000);
} catch (InterruptedException e) {
    e.printStackTrace(); }}}
```

The Thread.sleep(1000) – puts on hold the thread for 1 second, then goes into the loop again and checks if we are on the interval or not.[10]

The thread will start when the user finishes choses an interval of time.

```
ThreadSetUp componentSetUp = new ThreadSetUp(startDate, endDate, enable);
new Thread(componentSetUp).start();[10]
```

In the MainActivity class, I have the method onCreate(); here I have initializes the swiches and the TimePickers. But Also I have used the BluetoothAdaptor class. This class represents the local device Bluetooth adapter. The BluetoothAdapter lets you perform fundamental Bluetooth tasks, such as initiate device discovery, query a list of bonded (paired) devices, instantiate a BluetoothDevice using a known MAC address, and create a BluetoothServerSocket to listen for connection requests from other devices, and start a scan for Bluetooth LE devices.[11]

In order to turn on and off the Bluetooth I have used the following methods:

```
bluetoothAdapter.enable(); bluetoothAdapter.disable();
```

The class WifiManager provides the primary API for managing all aspects of Wi-Fi connectivity. This is the API to use when performing Wi-Fi specific operations. I have used it in order to define the names of various Intent actions that are broadcast upon any sort of change in Wi-Fi state. In order to turn the wi-fi on and off I have used the following methodes from the class: [12]

```
wifiManager.setWifiEnabled(true);
wifiManager.setWifiEnabled(false);
```

In order to get the state that the wi-fi is currently in I have used some macros:

```
WifiManager.EXTRA_WIFI_STATE,
WifiManager.WIFI_STATE_UNKNOWN
```

[10]Thread https://developer.android.com/reference/java/lang/Thread

[11]BluetoothAdapter<https://developer.android.com/reference/android/ bluetooth/BluetoothAdapter>.

[12] WifiManager <https://developer.android.com/reference/android/net/wifi/WifiManager>.

Furthermore, for the Audio settings I have used the the AudioManger class. AudioManager provides access to volume and ringer mode control. In my application I have only used the ring control. In order to do that I have used the following methodes:[13]

```
audioManager.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
audioManager.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
```

The first one will turn on the sound, and the last on will put it on vibration.

Because I have implemented the BroadcastReceiver for them and I have register them in the onStart() method, and unregister them in the onStop() method. The user can change the state of the Bluetooth, wi-fi and sound, both from the application or phone, and the changes will be made automatically on both.

```
@Override
protected void onStart() {
    super.onStart();
    IntentFilter intentFilter = new IntentFilter(WifiManager.WIFI_STATE_CHANGED_ACTION);
    registerReceiver(wifiStateReceiver, intentFilter);

    IntentFilter filter1 = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
    registerReceiver(bluetoothStateReceiver, filter1);

    IntentFilter filter3= new IntentFilter(AudioManager.RINGER_MODE_CHANGED_ACTION);
    registerReceiver(audioStateReceiver, filter3);

}

@Override
protected void onStop() {
    super.onStop();
    unregisterReceiver(wifiStateReceiver);
    unregisterReceiver(bluetoothStateReceiver);
    unregisterReceiver(audioStateReceiver);
    enable = false;
}
```

In order to can use all those classes for Bluetooth, wi-fi and audio, some permissions should have been provided:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NOTIFICATION_POLICY" />
<uses-permission android:name="android.permission.VIBRATE" />
```

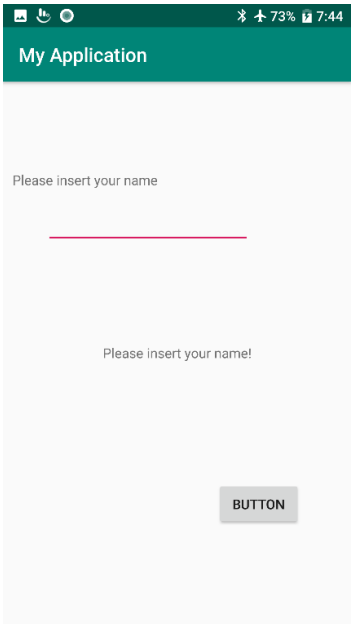
[13] AudioManager <https://developer.android.com/reference/android/media/AudioManager>.

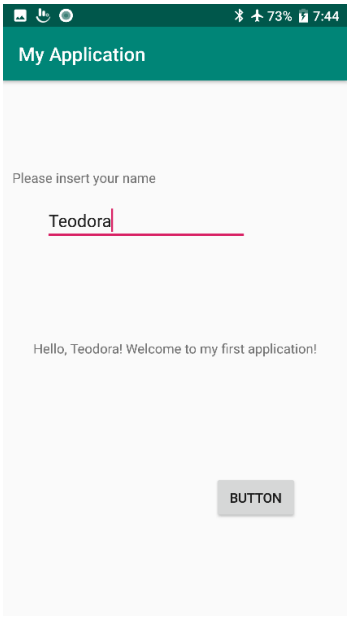
5.Testing

5.1 “Hello Word” application

1.When we first start the application it looks like this. You have a text field where you can insert your name.

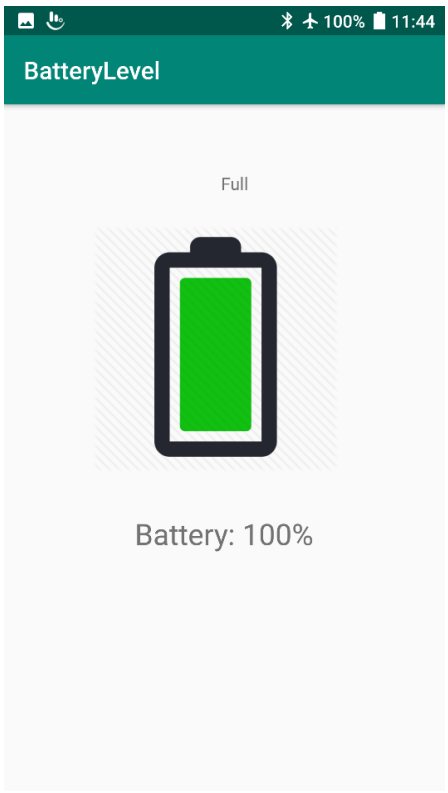
2.If you do not insert any name and you have pressed the button, and you have pressed the button, a message will appear that will tell you to insert your name.





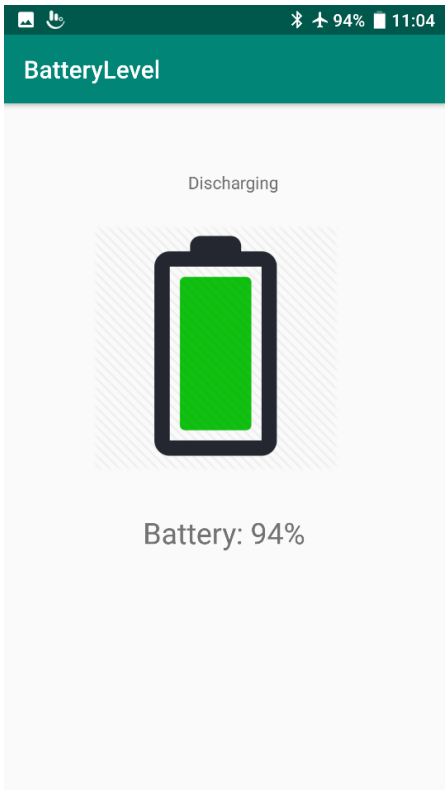
3. Then, if you insert a name, and press the button a message welcoming you will appear on the screen.

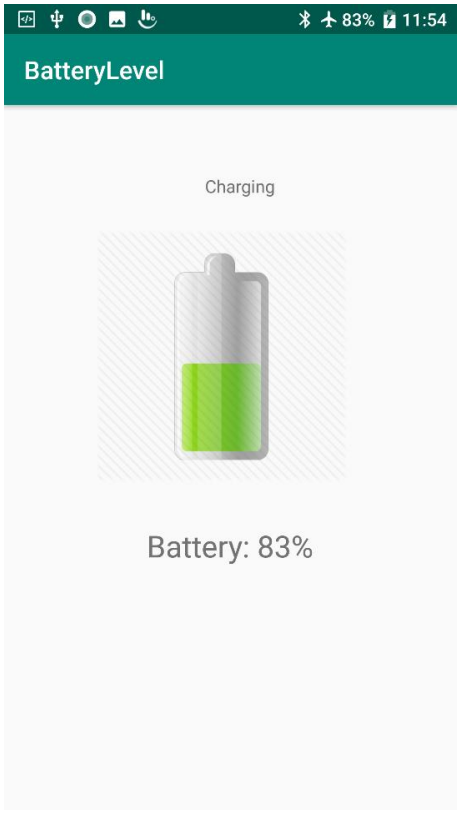
5.2 “Battery life” application



1.Dependeing on the level of battery there will be an image shown and a text. The first example is the one on the left where the battery level is 100% and is full. We can check by looking on the upper right conner where will se that the batery level is indeed 100%. The image battery is full.

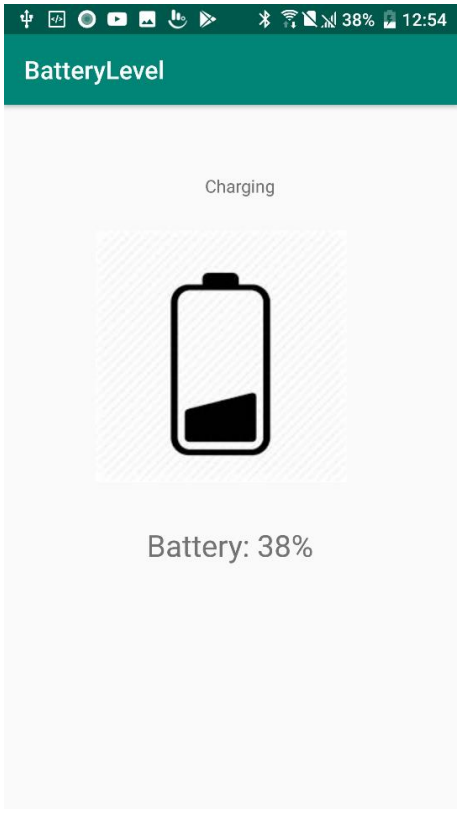
2.On the right we have a case that show us that the battery is 94% and is discharging. This is due to the fact that the phone is not connected to power. The image battery is full.



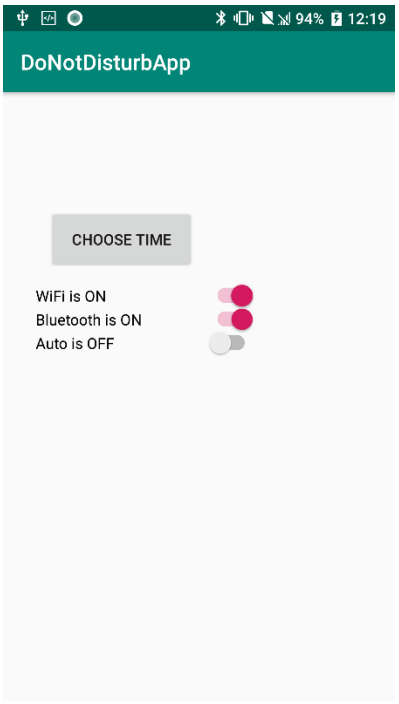


3.On the right we can test a case in which the battery is 83% and is charging because it is connected to the power. The image battery is half.

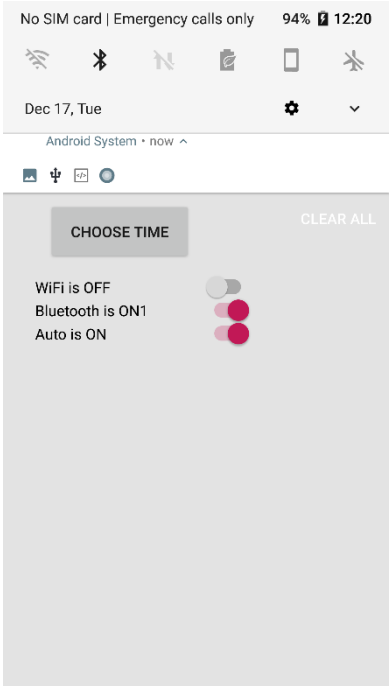
4.On the right the battery is 38% and is also charging. We can see that the image is changing depending on the level of the battery. The image battery is almost empty.



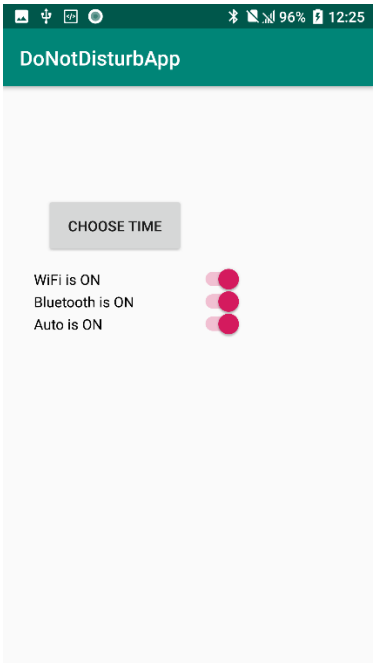
5.3 “Do Not Disturb” application



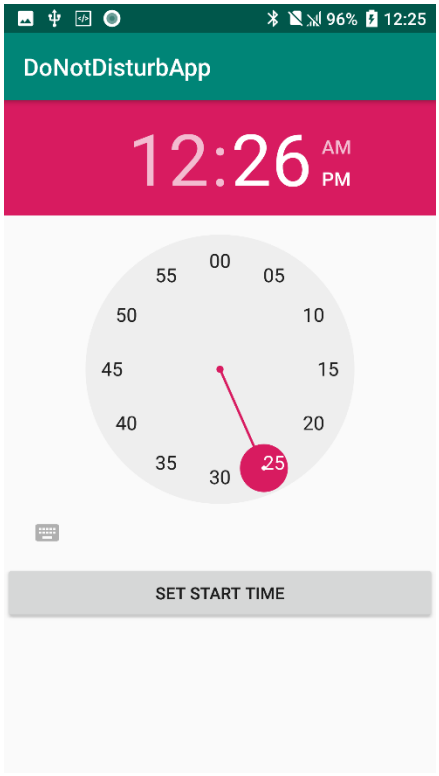
1.When you open the application it will look like this. This is depending on the current status of the wi-fi, Bluetooth and audio.If we look on the upper bar we will se that the Bluetooth is on and the sound is on vibration.



2.On the image on the left we will se that if we change the settings from the phone, the swiches will also change. The wi-fi is off, the Bluetooth is on and the sound is also on.

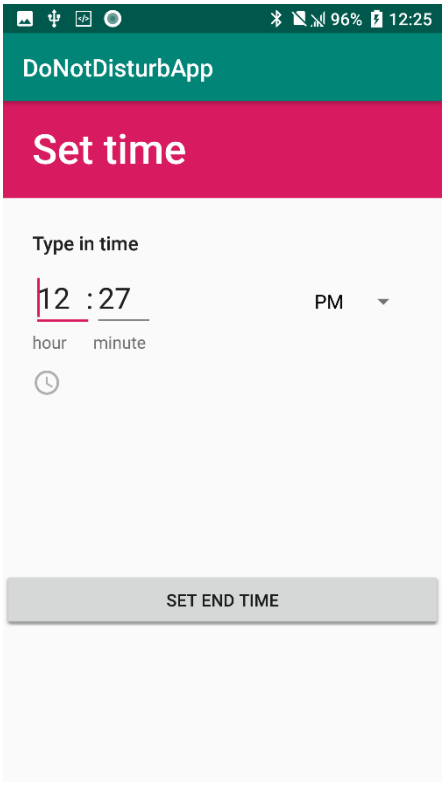


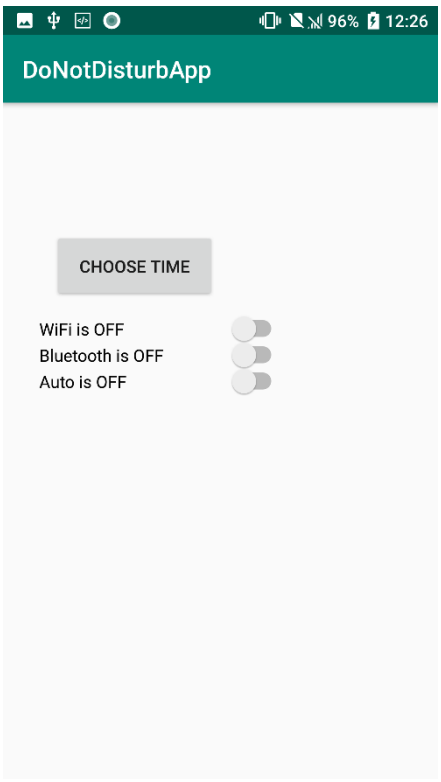
3.After that we can change an interval of time in which all three will be off. To illustrate this first I selected all of them as being on. Right now the time is 12:25. When I click on the Choose time button a new window will appear helping me selecting a time interval.



4.Then I have selected a time interval as follows: start time 12:26 and end time 12:27. There are two ways of selecting the time, one in which you have a clock and you select from it. Another one when you introduce the time by hand. I have illustrated both possibilities, first one when selecting the start time and the other one when selecting the end time.

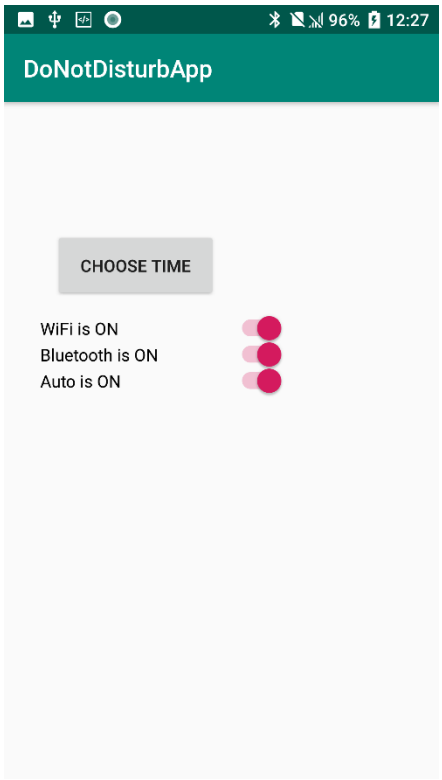
When I have clicked on the “set end time” button, the time interval will be set.





5. On the left you can see that the time is 12:26(upper right corner), which means that we are on the interval and all the wi-fi, Bluetooth and audio are off.

6.On the right you can see that the time is 12:27, that means the interval has ended, and they are all on.



6. Conclusions

When I started this project, I did not know anything about Android and how it works. Now that I have finished it, I learned how to create an application from beginning to end. This project has had as purpose to understand the Android operating system and to create three applications, the first one being a simple on and then adding complexity to next two ones.

When it comes to a structural point of view, an Android application has three main levels: the UI which is the level that the users sees and interact, the logic level where we obtain the information that the user askes and the data level where we save the information that are necessary for the logic level of the application. I have learned to use all three parts and to interconnect them.

In the future the applications could be improved by adding more elements in the UI, for them to look better. When it comes to the first application there could have been an image that pops up welcoming the user, and maybe a log in system.

For the second application, there could be more information of the phone, such as the health of the battery, the capacity, the voltage and the technology. Moreover, we can add a future that will make the application pop up a notification, or start an alarm when the phone battery is under a percentage, he/she has picked.

For the third one, we could have a button that will end the interval even though the interval has not passed. There could also be a button that will explicitly say that the interval is selected. In the future we can also implement an history of the interval, such that if a user wants to use a older interval he/she can. Furthermore, we can implement a daily routine, such that the user can select the interval and the days he/she wants the Do Not Disturb mode to be on.

In conclusion, I consider that the Android platform provides many utilities necessary for the creation of an application. It has a lot of classes that are already implemented, and this helps the developer get to the lower lever of the Android operating system. There are a lot of components that helps the developer create an entertaining user interface which helps the user have a good experience with the android applications. The difference between applications will be consists of the combination and usage of those components.

7. References

[1] Android Developers: < https://developer.android.com/about>.

[2] The history of Android OS: its name, origin and more https://www.androidauthority.com/history-android-os-name-789433/

[3] Android (Operating System),Wikipedia<https://en.wikipedia.org/wiki/Android_(operating_system)>.

[4] Android hardware components <https://subscription.packtpub.com/book/networking_and_servers/9781789131017/1/ch01lvl1sec15/android-hardware-components>

[5]Introduction to Activities <https://developer.android.com/guide/components/activities/intro-activities>

[6] Understand the Activity Lifecycle <https://developer.android.com/guide/components/activities/activity-lifecycle>

[7] Application Fundamentals<https://developer.android.com/guide/components/fundamentals>

[8] Build a simple user interface <https://developer.android.com/training/basics/firstapp/building-ui>.

[9] Monitor the Battery Level and Charging State <https://developer.android.com/training/monitoring-device-state/battery-monitoring>

[10]Thread https://developer.android.com/reference/java/lang/Thread

[11] BluetoothAdapter<https://developer.android.com/reference/android/bluetooth/BluetoothAdapter>.

[12] WifiManager <https://developer.android.com/reference/android/net/wifi/WifiManager>.

[13] AudioManager <https://developer.android.com/reference/android/media/AudioManager>.