WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EFMD EQUIS ACCREDITED

JoggingCoach

Fitness-Tracker for Android and Tizen

Harald Bernhard

1054776

August 22, 2016

Examiner: ao. Univ. Prof. Mag. Dr. Rony G. Flatscher

WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

Bachelor Thesis

English title of the Bachelor Thesis	JoggingCoach Fitness-Tracker for Android and Tizen
German title of the Bachelor Thesis	JoggingCoach Fitness-Tracker für Android und Tizen
Author **last name, first name(s)**	Bernhard Harald
Student ID number	1054776
Degree program	BaWiso-12
Examiner **degree, first name(s), last name**	ao. Univ. Prof. Mag. Dr. Rony G. Flatscher

I hereby declare that

1. I have written this Bachelor thesis independently and without the aid of unfair or unauthorized resources. Whenever content was taken directly or indirectly from other sources, this has been indicated and the source referenced.

2. this Bachelor thesis has neither previously been presented for assessment, nor has it been published.

3. this Bachelor thesis is identical with the assessed thesis and the thesis which has been submitted in electronic form.

4. (only applicable if the thesis was written by more than one author): this Bachelor thesis was written together with first name(s), last name(s). The individual contributions of each writer as well as the co-written passages have been indicated.

Date 22. August 2016

Signature

Abstract

JoggingCoach is an Android application to track and consequently improve the training performance of its users. Additionally, an interface to connect the app with a Samsung gear enables a more comfortable information retrieval from the wearable device.

This bachelor work is based on a seminar paper that was written for the course “IS Projektseminar”, which describes how a fitness tracking application can be created for Android devices. Moreover, it shows how access to location sensors is established and information about time, covered distance and speed are calculated.

Building up on this seminar work, this application needs to be expanded. Thus, a survey to get an insight about the actual market situation is carried out in order to show actual customer needs for fitness based tracking applications.

This paper is the graduation work of my study at the Vienna University Of Economics And Business and includes a profound review of several topic areas.

Content

Figures

1 Introduction

As part of the seminar project, a fitness tracker for Samsung Galaxy Note 3 and Samsung Galaxy Gear was established. It was shown how Android devices can be used to generate data about the actual position of an user in order to calculate the running speed and covered distance. Furthermore, a stopwatch to measure the time duration of a workout was implemented. This information has been transferred to a Samsung wearable device with the help of the Samsung Accessory Protocol. [Seminar16, P. 1 ff]

Set up on this seminar work, further functionalities were added to establish a competitive product that is available for download on Google Play Store. JoggingCoach is characterized by storing all sensible data directly on the device without sending it via a network. [Seminar16, P. 1ff]

Figure 1.1 - Android Phone And Samsung Gear

Source [Android/Samsung]

In order to get information on customer needs concerning fitness tracking applications, an empirical survey was conducted. People were invited to fill out an online questionnaire. The results of this evaluation have to be embedded into this project properly.

In this bachelor thesis, an examination of how applications for Android and Tizen devices can be created is offered. Therefore, fundamental concepts are described in detail so that readers who have no experience of developing apps for this two operating systems are able to understand all executions.

2 Evaluation of Market Research

2.1 Overview

To find out the benefits customers have from fitness tracking applications, an empirical study was made. This poll deals with the question “What functionality influences humans to buy tracking based fitness devices?” Over 100 people were examined, mostly young active people. The questions should give an insight into which implementations JoggingCoach should offer. The survey was executed online by umfrageonline.com. This website provides all necessary tools to create, distribute and evaluate investigation. [Umfrage_Online]

2.2 Result

Overall, as figure 2.1 shows, 70 percent of all 107 participants are male.

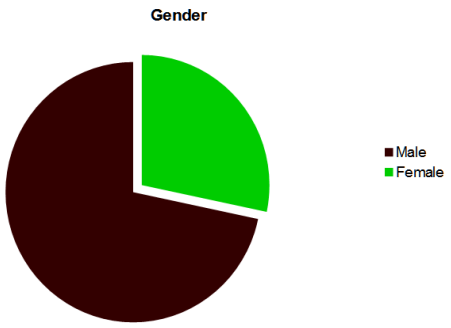

Figure 2.1 - Gender

The majority of the polled people indicates to do any sport multiple times a week.

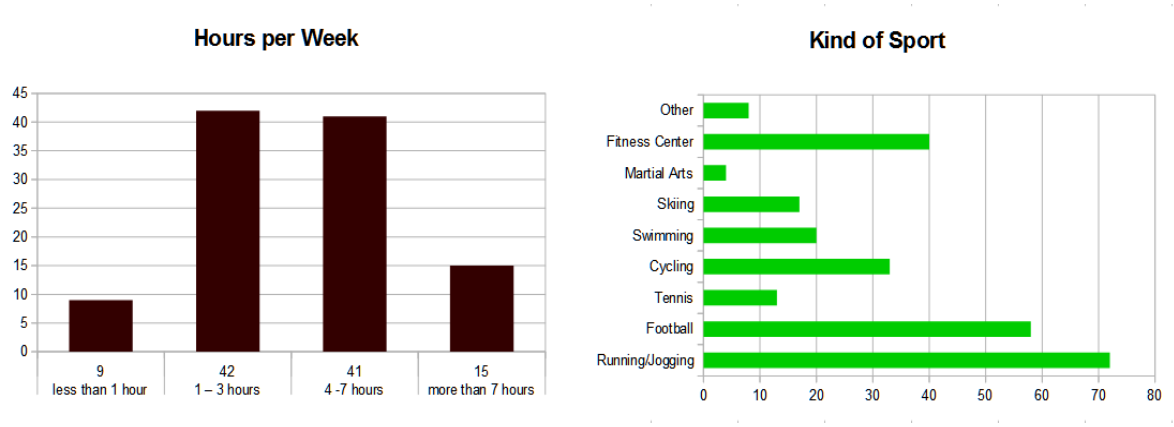

Figure 2.2 - Sporting Activity

The fundamental issue of this survey is, if a market for tracking based fitness devices exists. This consideration should also clarify the question if it makes sense to implement a fitness application for Android and Tizen. The result can be seen in figure 2.3.

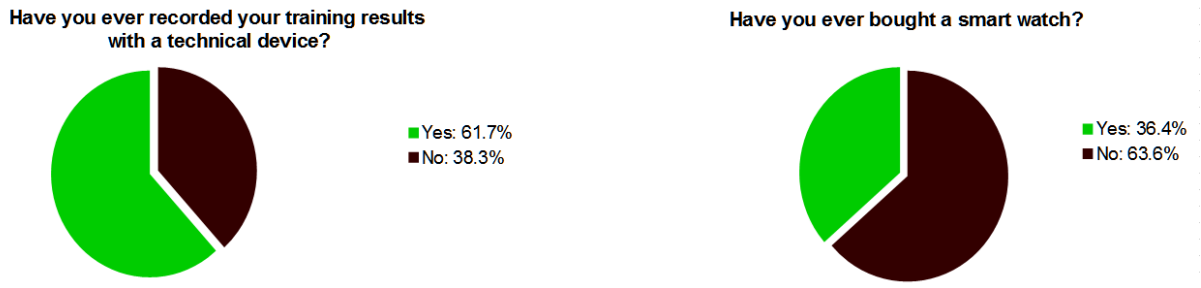

Figure 2.3 - Usage Of Fitness Tracker

During the training, it is possible to measure a large amount of values about the actual performance. JoggingCoach should only offer information about the most important features users wants to have. The participants have to evaluate several information according to their importance, indicated in figure 2.4.

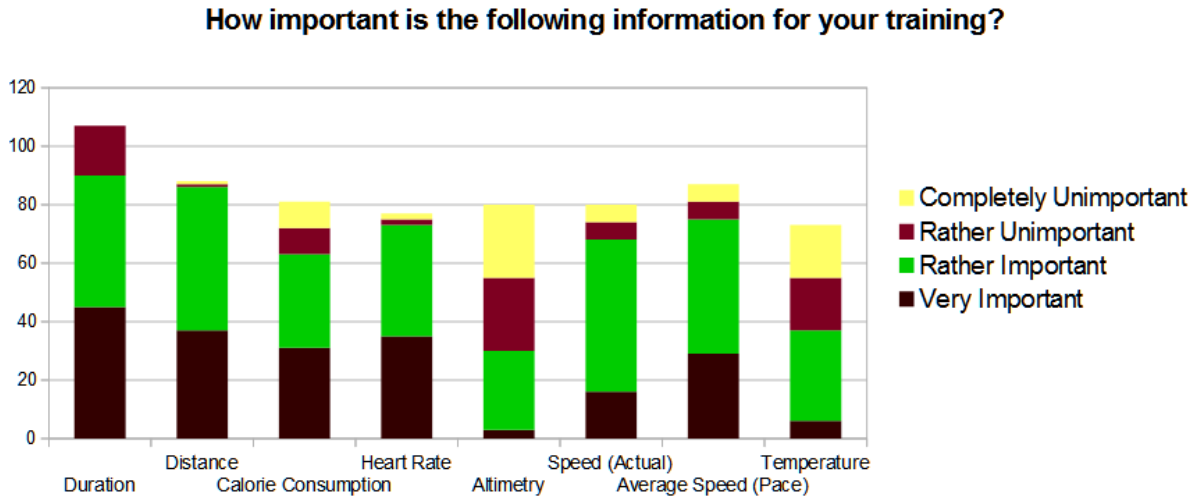

Figure 2.4 - Importance Of Training Information

Also the different functionalities the app should have can be identified with the questionnaire. The interviewed people have the choice to evaluate the functionalities concerning their importance. The main information for the implementation on the Android side is gained by questions concerning features during the workout on the one hand, and by questions dealing with the results after the workout on the other hand.

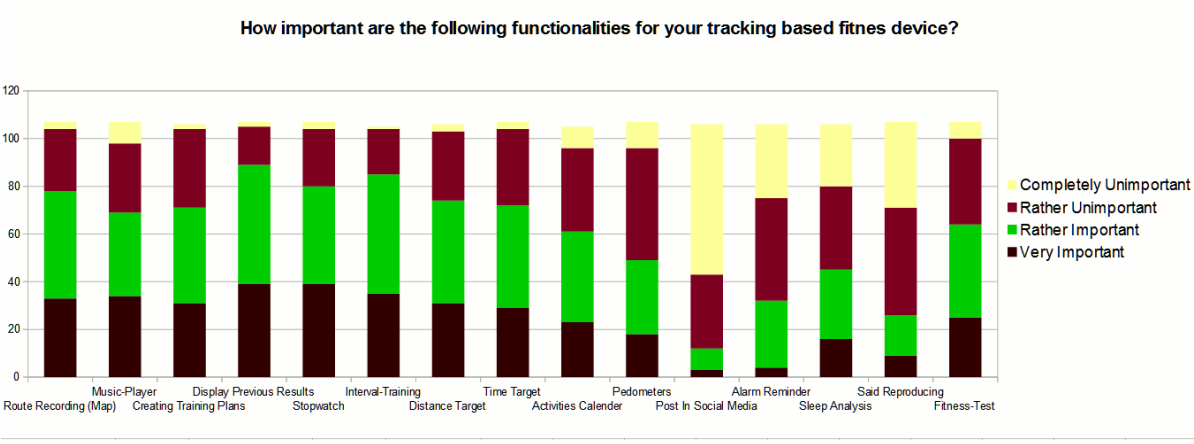

Figure 2.5 - Importance Of Functionality

For developing on the wearable device, it is also important to get feedback from the customers influence factors. Some characteristics have to be accepted because they are set from the Samsung gear itself but still give an exclusion on shopping behaviour of customers. Other features give an important exposure and influence the implementations of the application.

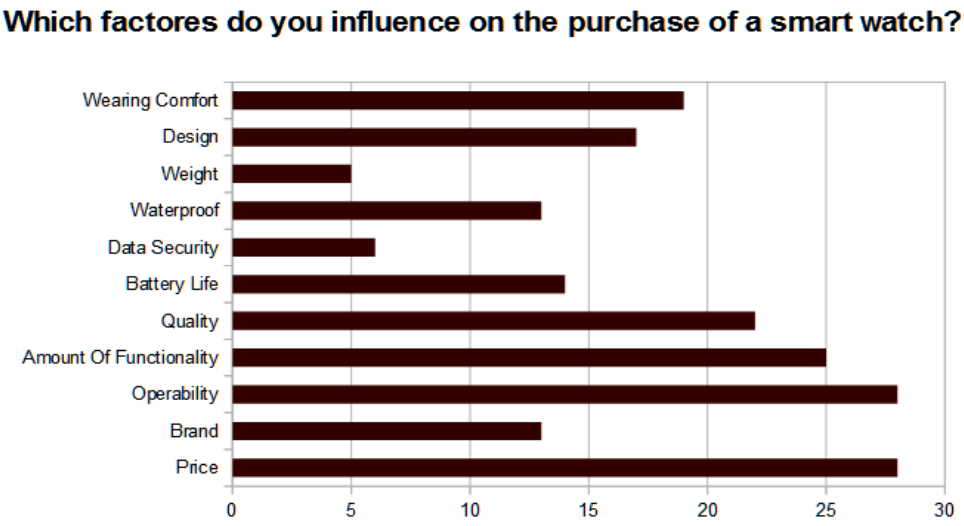

Figure 2.6 - Purchasing Factors

2.3 Interpretation

The most important question is, if there is a market for tracking based fitness apps at all and if it makes sense to create a fitness tracker for Android and Tizen. Figure 2.3 shows that a desire for such products still exists. Over two-thirds of the participants answered that they have already used a fitness app and over one third already own a smart watch. Consequently, it is legitimate to invest time to implement JoggingCoach.

But which characteristics should the application provide? The evaluation of the questions shown in figure 2.4 and 2.5 are interpreted by calculating the arithmetic medium of the given answers. [Wiki_Arithmetic]

As the value table has different weightings (very important, rather important, rather unimportant, completely unimportant), they have to be considered in the calculations. The formula of the weighted arithmetic medium is defined as followed: [Wiki_Arithmetic]

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i}.$$

Figure 2.7 - Formula Arithmetic Medium

Source [Wiki_Arithmetic]

For example, applied on the arithmetic medium of the stopwatch this means:

$$\text{Arithmetic Medium} = \frac{1}{107*(39*1+41*2+24*3+3*4)}$$

The result is rounded 1.92. [Wiki_Arithmetic]

The calculations give the following conclusion:

People want to know about:

1. Duration (1,74)
2. Distance (1,86)
3. Heart rate (1.98)
4. Calorie consumption (2.20)
5. Actual and average speed (2.26/2.08)

The actual altitude and temperature have no signification for the sportsmen.

The functionalities that the JoggingCoach should according to the demands of the customers are the following:

1. Display previous results (1.82)
2. Interval-Training (1.87)
3. Stopwatch (1.92)

3 Android

3.1 About Android

In 2005, Google bought a start-up establishment in California called Android Inc. This company was concerned with developing an operating system for mobile devices. The first Android SDK was promulgated in 2007 and was very bad compared with the competitive product by Apple. In the meantime, Android has been continuously improved and is market leader for smart phones. Nowadays, Android is not only an operating system, it also includes additional functions like an email client, a database or a web browser. [Android12, P. 21 ff]

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%

Figure 3.1 - Android Version

Source: [Android_Versions]

The actual version of Android is 6.0 and has a market share in contrast to other versions of 2.3% (March 2016). [Android_Statistik]

3.2 Architecture

Fundamental of the Android platform is the operating system which is based on Linux. Set up on the Linux kernel, Android provides several libraries and a virtual machine, called Dalvik, to compile source code. The virtual machine allows to transform implemented codes into runnable *apk* files. These programs are able to be executed on every device that has installed Dalvik. [Android12, P. 25 ff]

Android obtains additional pre-installed applications, such as a browser or the Google Play Store. [Android12, P. 25 ff]

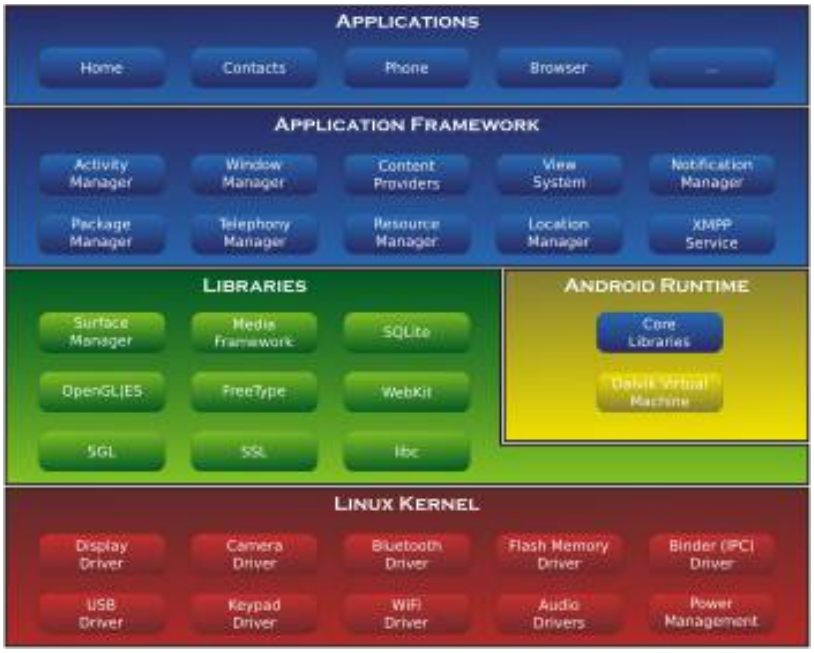

Figure 3.2 - Android Architecture

Source [Wiki_Android_Architecture]

3.3 Runtime

The source code of Android applications is written in Java which has to be compiled from a Java compiler firstly. The result, the Java byte code, is readable by the Dex compiler. This generated *dex* files are executable by the Dalvik virtual machine. One of the biggest advantages of Dalvik, which got its name from a village in Island, is the minimal storage consumption. [Android12, P. 27 ff]

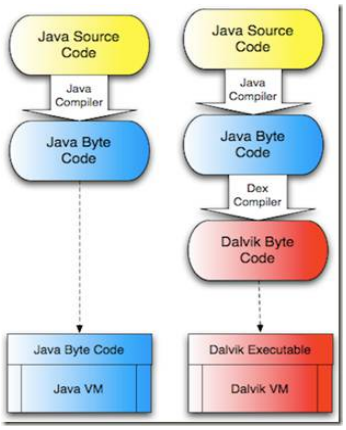

Figure 3.3 - Android Runtime

Source [Android_Runtime]

4 Tizen

4.1 About Tizen

In 2012, the Tizen operating system was created by the Linux foundation. It is used for smart phones, television and other domestic appliances. Today, the open system is built by a community of developers that consists of device manufacturers, application developers and independent software vendors. [Tizen_About]

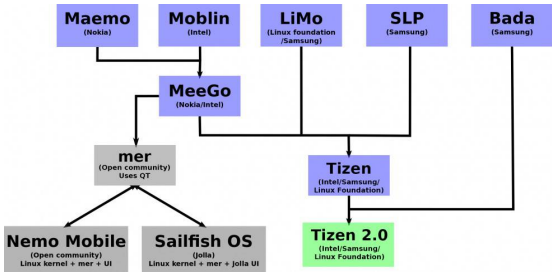

Figure 4.1 - Tizen Version
Source [Android_Tizen]

4.2 Web Application

It is reasonable to develop Tizen applications with two different methods. Firstly, it is possible to create a web application that needs a host device to run the code on an Android virtual machine. Such an application is written in HTML5, CSS and JavaScript. The second option is to build a native Tizen project. This approach is written in C or C++ and is installed and directly executable on the wearable device. The advantage of the native method is the ability to implement closer on the hardware in order to improve the performance. [Android_Tizen]

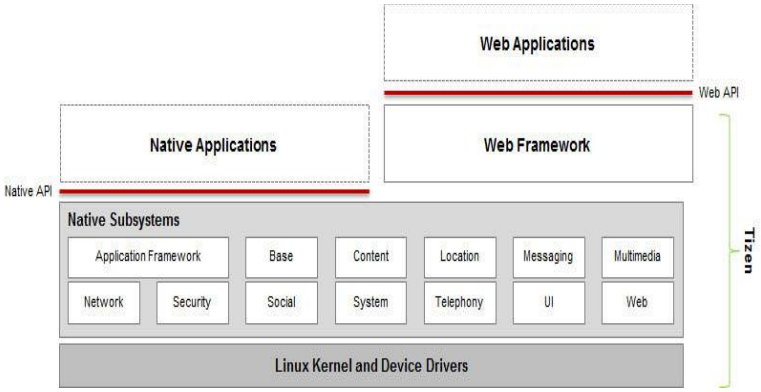

Figure 4.2 - Tizen Application
Source [Tizen_Application]

It is only necessary to transmit data from the Android to the Tizen wearable device. That is the reason why JoggingCoach is implemented as web application.

5 Requirements

5.1 Java SE

The basis for building Android and Tizen application is the Java Software Development Tool (Java SDK). It includes components to develop Java source code and also the Java Runtime Environment to execute the Android and Tizen Software Development Kid. The Java SDK is available in several occurrences and can be downloaded from *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. [Android12, P. 29 ff]

5.2 Android SDK

The Android Software Development Kit enables the generation of applications for smart phones. It contains the necessary software and additional tools, documentation and samples for easier development. An update manager ensures to install and delete the desired packages. The Android SDK is obtainable to download from https://developer.android.com/studio/intro/index.html. [Android12, P. 30 ff]

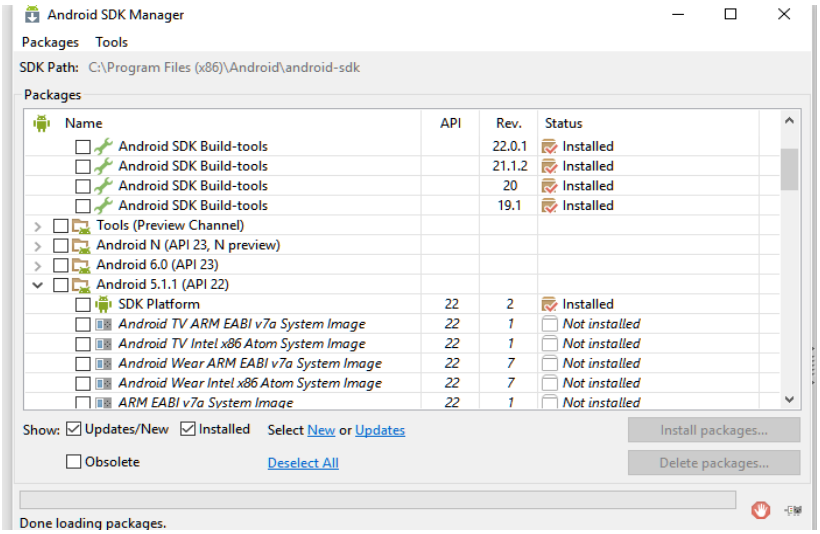

Figure 5.1 - Android SDK Manager

5.2.1 Android Studio

Inside the Android SDK, the Android Studio, an integrated development environment based on IntelliJ IDEA, is contained. This environment includes helpful instruments to develop Android applications. Android Studio is not only a syntax highlighted code editor, it also supports debugging, testing, profiling tools and many more. Furthermore, emulators to test applications directly on the system are provided. [Android_Studio]

Android uses the build automation system Gradle to describe and build the project structure. This system is helpful to customize, configure or reuse codes. Moreover, it allows to create multiple versions of one application if a free and paid version has to be offered, for example. [Android_Studio]

5.3 Tizen SDK

For developing on the wearable side, the Tizen Software Development Kit (Tizen SDK) is required. On https://developer.tizen.org/development/tools/download/installing-sdk, a GUI Installer is available. Also the Tizen Update Manager is supplied to install or reinstall additional software. [Tizen_SDK]

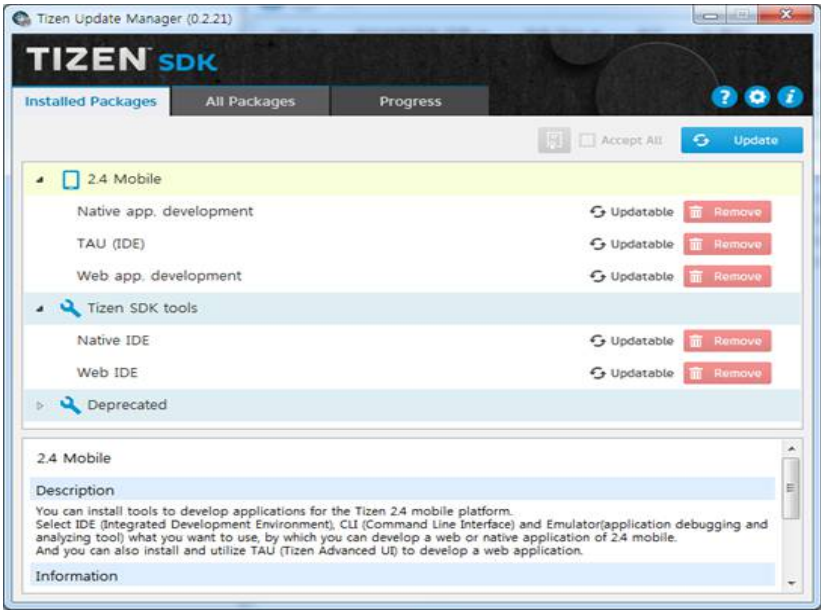

Figure 5.2 - Tizen SDK

The version of the installed SDK must be coincide with that one on the executable device. If an elder version is needed, an install instruction in more detail can be found in the seminar paper. [Seminar16, P. 7 ff]

Additionally to the software components of the Tizen SDK, an integrated development environment is installed. This Eclipse based IDE, helps to develop native and web applications, to debug and to test the project. [Tizen_SDK]

5.4 Samsung SDK

The Samsung Accessory Software Development Kit is needed to transmit data between the Samsung gear and the Android device. [Samsung_SDK]

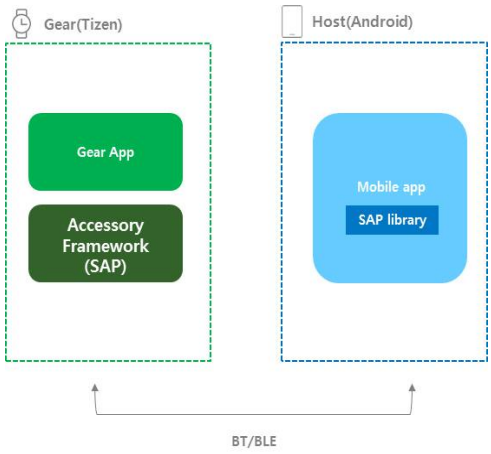

Figure 5.3 - Gear Applicatin

Source [Tizen_Application]

It contains all necessary libraries to establish a connection, exchange data or files and it closes them if it is not used any more. Furthermore, samples and guidelines show how the Samsung Accessory Protocol is working in practice. The Samsung SDK is located on http://developer.samsung.com/gear/develop/sdk. [Samsung_SDK]

6 Developing for Android

6.1 Overview

JoggingCoach is a fitness tracking application for Android devices. It consists of three different functionalities: a training mode to record time, distance, calorie consumption, actual and average speed plus an interval mode to define workout and recreation phases and a training diary to get an insight about the latest training results.

It is also possible to connect the Android device with a Samsung gear to get the information about the workout on the display of the Tizen wearable device. In order to perform this transmission, the Samsung Accessory Protocol is used. [Accessory_Guide1]

The application is characterised by storing all produced data on the device itself. Consequently, users do not have to fear that any private information can be intercept via the internet.

After starting JoggingCoach for the first time, it is necessary to create a new user account. Therefore, the name, age, size and weight has to be fill up. This personal traits are important to determine the calorie consumption and to save results in a database. If a new user is created, the menu selective of the app is ready to start to choose one of the three functionalities.

1. The training mode offers a free practice and the opportunity to save the information mentioned above. The user has also the possibility to connect a Samsung device to get the information on that screen. After the workout is done, the user gets the choice to either save or discard the result.
2. The second alternative is the interval training mode. It is able to train in certain time phases: time to burden and recreation. These periods are repeated as long as the user wants to.
3. To retrace the trend of the training results, the diary gives an overview of improvements or deterioration of the performance. For a meaningful analysis, a list provides the data about time, distance, calorie consumption and average speed.

6.2 Create a New Android Project

In order to build a new Android application, do the following:

1. Open Android Studio
2. Click on File > New > New Project

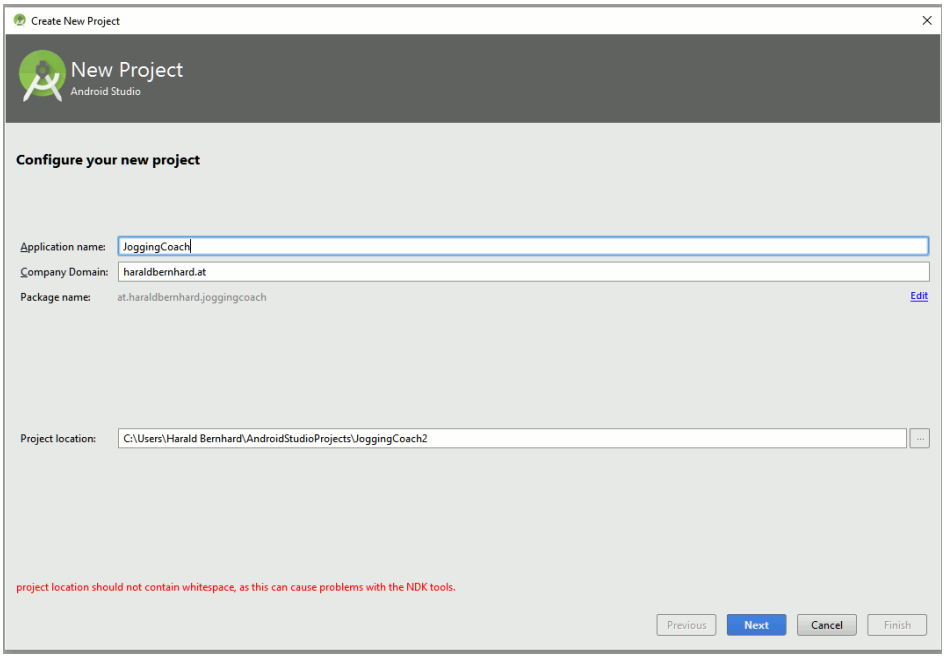

Figure 6.1 - Create New Android Project

Android Studio helps to assign a project name, the minimum SDK and a start activity. It is also possible to choose the device, for example TV, phone or tablet, where the application should be installed later. After finishing this procedure, the IDE creates a project structure with all necessary modules and files. [Android_Project]

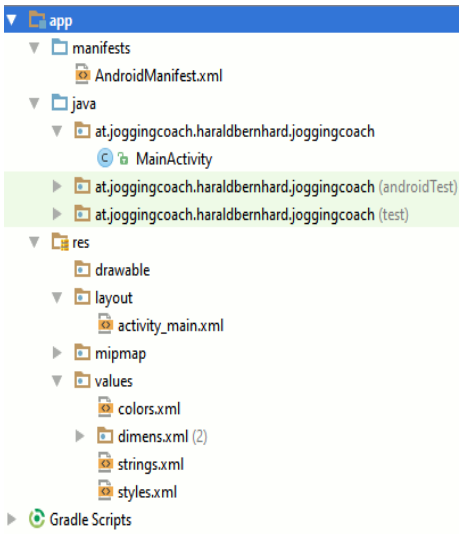

Figure 6.2 - Project Structure

The AndroidManifest.xml file contains information about the Android app itself. Here, permissions, receivers or the required SDK can be expanded or changed. The function of permissions and services is described in the following chapters. The source code for the activities are suited in the *java* folder. There, it is also possible to create new classes for the program logic. The design of the application is filed in the folder *res* in several XML files. [Android_Studio]

6.3 Activities

Android applications are structured in a logical and a graphical component. These blocks are designated as activities and mainly include a user interface and the associated code. [Android12, P. 87 ff]

An app insists one launch activity. This activity is opened when the program is started. If another activity is opened, the previous one is deposit on a stack. This concept allows to get the latest activity by pressing the back button on the device. [Android12, P. 87 ff]

6.4 Graphical User Interface

As already mentioned, every activity has its own graphical user interface. The appearance is described with XML code and inflated at run time. [Android12, P. 52 ff]

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="at.joggingcoach.haraldbernhard.joggingcoach.CreateAvatarActivity">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/editTextWeight"
        android:layout_marginTop="48dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/createUser"
        android:id="@+id/buttonCreateAvatar"
        android:onClick="createNewAvatar"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="45dp" />
</RelativeLayout>
```

Figure 6.3 - Activity Layout

To work with operating elements like buttons, text views and labels in the program logic, they have to be referenced. Therefore, an identification name has to be allocated. Android stores this information in the *layout* folder. To initialise the components, the onCreate() method is most suitable. The call of the method findViewById() expects the identification to reference the element. [Android12, P. 51 ff]

```
public class TrainingModeActivity extends Activity {

    //Labels
    TextView tvLabelTime, tvLabelDistance, tvLabelCalorie, tvLabelSpeed, tvLabelGearConnection;
    //TextViews for display
    TextView tvTime, tvDistance, tvCalorie, tvSpeedAverage, tvSpeedImmidiate;
    //Buttons
    Button startTraining, stopTraining, pauseTraining;

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

        //Labels
        tvLabelTime = (TextView)findViewById(R.id.textViewLabelTime);
        tvLabelDistance = (TextView)findViewById(R.id.textViewLabelDistance);
        tvLabelCalorie = (TextView)findViewById(R.id.textViewLabelCalorie);
        tvLabelSpeed = (TextView)findViewById(R.id.textViewLabelSpeed);
        tvLabelGearConnection = (TextView)findViewById(R.id.textViewGearConnection);
        //TextViews for display
        tvTime = (TextView) findViewById(R.id.textViewTime);
        tvDistance = (TextView)findViewById(R.id.textViewDistance);
        tvCalorie = (TextView) findViewById(R.id.textViewCalorie);
        tvSpeedImmidiate = (TextView) findViewById(R.id.textViewSpeedI);
        tvSpeedAverage = (TextView) findViewById(R.id.textViewSpeedA);
```

Figure 6.4 - Method onCreate()

6.4.1 Material Design

Smart phones are different to other computers in several ways. They have smaller displays than laptops or personal computers, the interaction is done by touching the display instead of using a mouse and many products have various sizes. These characteristics make it more complicated to create a good Graphical User Interface. [Android12, P. 47 ff]

The introduction of Android version 5 includes several innovations. One of them is the material design. “Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices“. It provides several design styles to define colours, layouts, animations and much more. [Android_Material]

Several predefined themes for material design exist. In order to use one of them, it has to be declared in the manifest file. [Android_Appbar]

```
<application
    android:theme="@style/AppTheme">
```

Figure 6.5 - Theme Manifest

The actual implementation is defined in the AppTheme.xml file. This file is located in the *style* folder. It includes the instruction of defining the basic look of the app. [Android_Appbar]

```
<resources>
    <!-- inherit from the material theme -->
    <style name="AppTheme" parent="Theme.AppCompat.NoActionBar">
        <!-- Main theme colors -->
        <!--   your app branding color for the app bar -->
        <item name="android:colorPrimary">#7CB342</item>
        <!--   darker variant for the status bar and contextual app bars -->
        <item name="android:colorPrimaryDark">#123456</item>
        <!--   theme UI controls like checkboxes and text fields -->
        <item name="android:colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

Figure 6.6 - Theme Appbar

Every view of JoggingCoach has the same construction. At the top, there is a toolbar located. It includes a button to navigate to the previous View and the name of the application.

Figure 6.7 - Toolbar

The required variables and methods to create a toolbar are implemented in the class AppCompatActivity. In order to get access to that components, the activity has to inherit from this class. [Android_Appbar]

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#7CB342"
    android:id="@+id/my_toolbar"
    >
</android.support.v7.widget.Toolbar>
```

Figure 6.8 - Toolbar

In the onCreate() method of the activity, the toolbar object is instantiated. The method setSupportActionBar() ensures that the Toolbar is used instead of the predefined ActionBar. With setDisplayHomeAsUpEnabled(), the button to navigate to the previous view is shown. [Android_Appbar]

```
//Toolbar
Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);
setSupportActionBar(toolbar);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

Figure 6.9 - Initialise Toolbar

6.5 Execute JoggingCoach

After JoggingCoach is installed and started on the Android device, the launch activity appears. Which activity the launch activity has to be is declared as setting in the manifest file. [Android_Activity]

Figure 6.10 - Launch Activity 1

After starting the application a welcome screen appears that can have two different looks. Moving from that point that the app is opened at the first time and no user account exist, the screen on figure 6.10 arises.

The user gets the possibility to create a new user by clicking the link “CREATE NEW USER”. In Android it is very easy to implement listeners to react to the interaction. In order to navigate through activities, only the onClick attribute in the XML file has to be declared and a method with exactly the same name has to be written. If a user clicks on the button, the associated method is called. [Android_Button]

```
<TextView
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="@string/createUser"
    android:id="@+id/textViewCreateNewUser"
    android:onClick="startCreateUserActivity"
    android:layout_marginTop="50dp"
    android:textAlignment="center"
    />
```

Figure 6.11 - OnClick Listener

In this method, an object of type Intent has to be initialised. The constructor needs two parameters. One for the actual activity, and the second one for the activity that should be started. The call of the method startActivity() triggers the event. [Android_OnClick]

```
public void startCreateUserActivity(View view){
    Intent intent = new Intent(this, CreateUserActivity.class);
    startActivity(intent);
}
```

Figure 6.12 - Change Activity

After clicking the link, the activity to create a new user appears. The entered data about the name, age, size and weight are stored in a database on the device. Here, it is important to react on wrong information. Checking the input ensures that the username consists of characters and the personal data can only be numbers in human dimension.

Figure 6.13 - Create New User

Android provides the interface TextWatcher to listen if something has been edited to the text fields. If this interface wants to be used, the class has to import the three methods beforeTextChanged(), onTextChanged() and afterTextChanged() and the elements have to add the text watcher. [Android_TextWatcher]

```
etUsername.addTextChangedListener(this);
etAge.addTextChangedListener(this);
etSize.addTextChangedListener(this);
etWeight.addTextChangedListener(this);
```

Figure 6.14 - Text Watcher

The input validation in JoggingCoach is implemented in the afterTextChanged() method. Therefore, the data of the text views are stored into variables and checked according to their rightness. If everything is correct, the button gets activated to create a new user. [Android_TextWatcher]

```
    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
    }

    @Override
    public void afterTextChanged(Editable editable) {
        username = etUsername.getText().toString();
        age = Integer.parseInt(etAge.getText().toString()+0)/10;
        size = Integer.parseInt(etSize.getText().toString()+0)/10;
        weight = Integer.parseInt(etWeight.getText().toString()+0)/10;
        if(!username.isEmpty() && age>0 && age<=100 && size>0 &size<200 && weight>0 && weight<200 ){
            createAvatar.setEnabled(true);
        }
        else{
            createAvatar.setEnabled(false);
        }
    }
}
```

Figure 6.15 - Input Validation

6.5.1 SQLite

The personal data about the user and the training results are often needed in the application. The storage of this information makes it easier to structure and retrieve them. All necessary functionalities provide Android in the database management system SQLite. The access and execution to use this system are located in the class android.database.sqlite. [Android12, P. 269 ff]

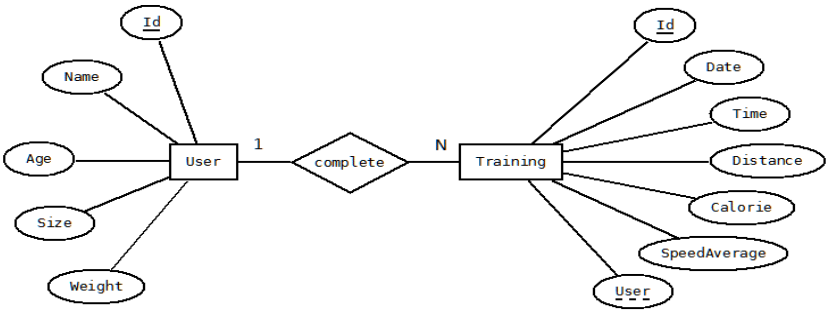

Figure 6.16 - ER Diagram

The organization of the SQL database is described in a schema. "A schema is a formal declaration of how the database is organized". It reflects the SQL statements and allows to change the database without alternating the whole implementation. SQLite is characterised by having no client/server architecture, such as many other database systems. [Android_SQLite]

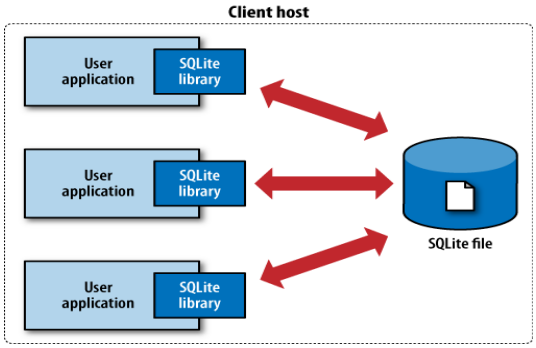

Figure 6.17 - SQLite Management System

The actual program logic of the database is implemented in DbHelper.java and acts as a helper class for the database connection. This class has to inherit from SQLiteOpenHelper and ensures that the methods onCreate() and onUpgrade() get overwritten. While onCreate() constructs a new table if no one already exits, onUpgrade() modifies the database if the version number has changed. [Android_SQLite]

Methods to insert, update or delete tables from the database are also implemented in this helper class. The method getWritableDatabase() returns an object of SQLiteDatabase. This object provides the methods insert(), update() and delete() to execute the SQL statements. [Android_SQLite]

For JoggingCoach, it is necessary to insert the entries of the applicants in the table *user* and *training*. Thus, a create table statement has to be sent. This statement is stored into a String object. The methods onCreate() help to construct a new table if the application is started the first time and the method execSQL() that expects the String to generate a new table. [Android_SQLite]

```
private static final String TABLE_TRAINING_CREATE =
        "CREATE TABLE "
                + TABLE_NAME_TRAINING + " (" + _ID
                + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + DATE + " TEXT, "
                + TIME + " TEXT, "
                + DISTANCE + " TEXT, "
                + CALORIE + " TEXT, "
                + SPEEDA + " TEXT);";

public void onCreate(SQLiteDatabase db) {
    db.execSQL(TABLE_USER_CREATE);
    db.execSQL(TABLE_TRAINING_CREATE);
}
```

Figure 6.18 - Create Table

It has to be considered, that all required tables have to be generated at once. [Android_SQLite]

The insert of the values in the table is executed by the eponymous method. First of all, the connection to the database has to be established. Then, all records of the form have to be added to a map whereby the keys are representing the columns of the table. The method insert() submits the assignment to store the data. It is important to capture faulty statements with a try/catch block. The execution of this method will be done by clicking on the create new user button. [Android_SQLite]

```
public void insert(User avatar){
    long rowId = -1;
    try{
        // Open Connection
        SQLiteDatabase db = getWritableDatabase();
        //Save Values
        ContentValues values = new ContentValues();
        values.put(USERNAME, avatar.getUsername());
        values.put(AGE, avatar.getAge());
        values.put(SIZE, avatar.getSize());
        values.put(WEIGHT, avatar.getWeight());
        //Insert into Table
        rowId = db.insert(TABLE_NAME_USER, null, values);
    }

    catch (SQLiteException e){
        Log.e(TAG, "insert() ", e);
    }
    finally {
        Log.d(TAG, "insert(): rowId= " + rowId);
    }
}
```

Figure 6.19 - Insert User

Additionally, the application is informed that a user exists now and the username and weight are stored in shared preferences. [Android_SharedPreferences]

6.5.2 Shared Preferences

Primitive data types can be stored easily with key-value pairs on the Android device. These shared preferences save information beyond sessions and enable a faster retrieval compared to the database. The information about the username and the weight are needed during the runtime of the application and therefore, they are very useful for this concept. [Android_SharedPreferences]

```
SharedPreferences.Editor editor = sharedPref.edit();
editor.putString(PREFERENCE_KEY_USERNAME, etUsername.getText().toString());
editor.putInt(PREFERENCE_KEY_WEIGHT, Integer.parseInt(etWeight.getText().toString()));
editor.apply();

LaunchActivity.userExists = true;
```

Figure 6.20 - Shared Preferences

And Editor variable of type SharedPreferences has to be established and the two information has to be handed over as parameters. The method apply() ensures that this information is stored in the background. [Android_SharedPreferences]

6.6 Ready to Start JoggingCoach

After creating a new user, the application navigates back to the LaunchActivity with a changed look.

Figure 6.22 - Launch Activity 3

Figure 6.21 - Launch Activity 2

The method getSharedPreferences() returns an SharedPreferences object and expects two parameters. While the first designates the file name, the second reveals the visibility of the data. With the method getString() the information is fetched. [Android_SharedPreferences]

```
//Shared Preferences
public static SharedPreferences sharedPref;
public static final String PREFERENCE_FILE_NAME ="UserPreference";
public static final String PREFERENCE_KEY_USERNAME="username";

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

    sharedPref = getSharedPreferences(PREFERENCE_FILE_NAME, Context.MODE_PRIVATE);
    username = sharedPref.getString(PREFERENCE_KEY_USERNAME, "");
```

Figure 6.23 - Shared Preferences Get Username

The text view in figure 6.22 needs the date of the last training. In order to get it, the database has to provide the data. At first, a connection to the database has to be established. All results that a SQL query contains, are stored in a Cursor object. This object can be received by calling the method query(). The query() method can be handed over up to seven parameters that represent the conditions of the SQL statement. The last parameter ensures that the return values of the database are sorted descending and thus, it makes sure that the latest date is retrieved firstly. Now, the Cursor object has to be put to the beginning. This is achived by calling the method moveToFirst(). If the Cursor is not empty, the result is stored into a variable. With the methods of the Cursor, getString() or getLong(), the values are extracted from the database. [Android_SQLite]

```
public String getLastTrainingDate(){

    String result;
    //Open Connection
    SQLiteDatabase db = getReadableDatabase();
    //Build SQL-Statement
    String[] column = {DATE};
    Cursor cursor = db.query    (TABLE_NAME_TRAINING,   // The table to query
                                 column,                // The columns to return
                                 null,                  // The columns for the WHERE clause
                                 null,                  // The values for the WHERE clause
                                 null,                  // don't group the rows
                                 null,                  // don't filter by row groups
                                 DATE+" DESC"           // The sort order
                                 );
    cursor.moveToFirst();
    if(cursor != null && cursor.getCount()>0) {
        result =  cursor.getString(cursor.getColumnIndexOrThrow(DATE));
    }
    else {
        result = "";
    }
    return result;
}
```

Figure 6.24 - Get Last Training Date

Now, the user has the possibility to start several training modes by clicking the start button.

6.7 Menu Activity

The menu activity acts as the guide of the application and allows the user to navigate to the different app modules. On the top, an invitation to choose a training mode is displayed. Below, there are three buttons for the different functionalities.

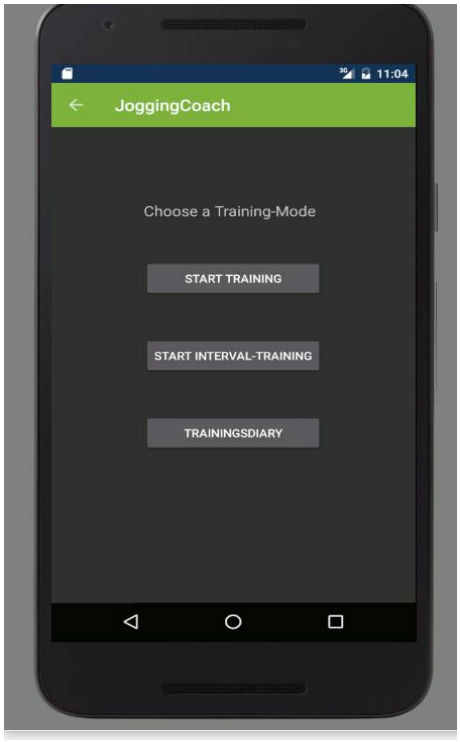

Figure 6.25 - Menu Activity

While the training mode enables the user to start the running session and get information about the performance, he/she gets the possibility to set workout and recovery phases in the interval training mode. The diary can be used to get the latest results of the trainings.

6.8 Training Mode Activity

One of the main parts of JoggingCoach is the training mode activity. It effects a user to start a new workout and record time, distance, calorie consumption and average speed. The code of this functionality is implemented in the class TrainingModeActivity. A button to start the training session is located at the bottom of the display. If the button is pressed, the training starts and the labels get updated. One button which stops the activity is situated next to the start button.

Figure 6.26 - Training Mode Activity

6.8.1 Location

For calculations of the completed distance, calories consumption, actual and average speed, the position of the user is necessary. The class android.location.LocationManager provides the required methods to get access to the device sensor. A call of the method getSystemService(LOCATION_SERVICE) returns an object of this class. [Android12, P. 237 ff]

The determination of the location takes on a provider. Android devices mostly support more than one location provider. It is possible to generate data about the actual position with the help of the Global Positioning System (GPS) that enables a more accurate determination. A second way is to get the data about the location from radio stations. Even though this method is rather inaccurate, it consumes far less battery charge than the GPS method. The implementation of JoggingCoach is based on GPS. [Android12, P. 237 ff]

As of Android 6.0, users no longer need to accept permission before installing the app but during run time. Because of a better protection of the users privacy. Before an activity is allowed to use the location service, it needs to be checked if a user has granted access. [Android_Permission]

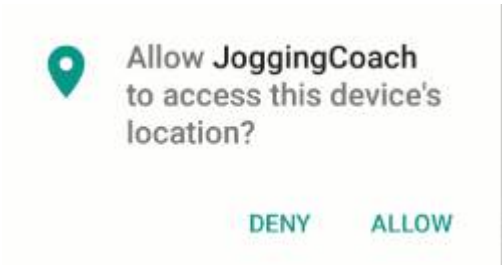

Figure 6.27 - Permission Dialog Box

This permissions are declared in the Android manifest file of the application. [Android_Permission]

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Figure 6.28 - Location Permission

The location sensor needs one of the so-called dangerous permissions, whereas the user has to give an explicit approval every time. Firstly, it should be checked if the permission is granted in the manifest file. This is executed by calling the method checkSelfPermission() that returns an object of type ContextCompat. Then, a dialog box appears which allows the user to inform the application about his/her decision. [Android_Permission]

```
public void gpsPermission() {
    // Check if the GPS permission is already available
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED)
        Toast.makeText(this, "Initialize Location Manager", Toast.LENGTH_LONG).show();
        LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        LocationListener ll = new AndroidLocationListener();
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 50, ll);
    }
    // Permission is not available
    else {
        // Should we show an explanation?
        if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.ACCESS_FINE_LOCATION)) {
            // Show an expanation to the user *asynchronously*
        } else {
            // No explanation needed, we can request the permission.
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, LOCATION_REQUEST);
        }
    }
}
```

Figure 6.29: - Check GPS Permission

After the communication of the decision, the method onRequestPermissionResult() is invoked. If a user refuses access, this method navigates him or her back to the menu activity.

```
@Override
public void onRequestPermissionsResult(int requestCode,  String permissions[], int[] grantResults) {
    switch (requestCode) {
        case LOCATION_REQUEST: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // permission was granted, yay! Do the
                // contacts-related task you need to do.
            } else {
                // permission denied, boo! Disable the
                // functionality that depends on this permission.
                Intent intent = new Intent (this, MenuActivity.class);
                startActivity(intent);
            }
            return;
        }
        // other 'case' lines to check for other
        // permissions this app might request
    }
}
```

Figure 6.30: - Handle Permission Response

If the user allows the permission, the training mode is ready to start. After the start button is clicked, the calculations of time, distance, calorie consumption and speed and the update of the user interface begin. This update is executed by a location listener every ten meters a user moves. Additionally, the stop button gets clickable, while the start button gets unclickable.

6.8.2 Stopwatch

In order to be able to stop time, another class which represents a stopwatch is created. It contains three Integer variables for hour, minute and second. Three interleaved loops representing time counts until they do not reach the value 60. If this happens, it is set to zero and the next higher value is incremented. [Seminar, P. 11 ff]

```
public void startStopwatch(){
    while(true){
        while (min<60){
            while (sec<60){
                try{
                    Thread.sleep(1000);
                    sec +=1;
                    setHour(hour);
                    setMin(min);
                    setSec(sec);
                    Log.i("Time", hour + " : " + min + " : " + sec);
                }
                catch(InterruptedException e){
                    e.printStackTrace();
                }
            }
            min +=1;
            sec = 0;
        }
        hour +=1;
        min = 0;
        sec = 0;
    }
}
```

Figure 6.31 - Stopwatch

To prevent the responsiveness of the Training Mode Activity from being interrupted, the calculations are made in the background. The Android SDK provides the class AsyncTask to implement this functionality. In order to be able to do executions parallel to the main thread, the desired class has to inherit from android.os.AsyncTask. In JoggingCoach, this is implemented as inner class of TrainingModeActivity. [Android_AsyncTask]

```
//AsyncTask for Stopwatch
class StopwatchTask extends AsyncTask<Void, String, Void>{

    @Override
    protected void onPreExecute() { super.onPreExecute(); }

    @Override
    protected Void doInBackground(Void... params) {
        stopwatch = new Stopwatch();
        stopwatch.startStopwatch();
        return null;
    }

    @Override
    protected void onPostExecute(Void aVoid) {}

    @Override
    protected void onProgressUpdate(String... values) {}
}
```

Figure 6.33 - Async Task Stopwatch

```
//AsyncTask For Update User Interface
class UpdateUITask extends AsyncTask<Void, Void, Void>{
    @Override
    protected void onPreExecute() {}

    @Override
    protected Void doInBackground(Void... voids) {

        while (isActive) {
            try {
                time = stopwatch.getTime();
                Thread.sleep(1000);
                publishProgress();
            }
            catch(InterruptedException e){
                e.printStackTrace();
            }
        }
        return null;
    }

    @Override
    protected void onProgressUpdate(Void... values) {
        tvTime.setText(time);
        tvDistance.setText(distance);
        tvCalorie.setText(calorie);
        tvSpeedAverage.setText((speedA));
        tvSpeedImmidiate.setText(speedI);
    }

    @Override
    protected void onPostExecute(Void aVoid) {}
}
```

Figure 6.32 - Async Task Update UI

As AsyncTask is defined as abstract, the methods onPreExecute(), doInBackground(), onProgressUpdate() and onPostExecute() should be overwritten. While background calculations can be delegated in onPreExecute(), doInBackground() and onPostExecute(), onProgressUpdate is responsible for updating the user interface. [Android_AsyncTask]

As the termination of the stopwatch is accomplished when the loop is finished, two separate tasks are necessary. One for the stopwatch and one to update the data and the graphical user interface. [Android_AsyncTask]

6.8.3 Calorie Calculation

The determination of the burnt calories during the workout is calculated by a rule of thumb which describes Williams in “Ernährung, Fitness und Sport”. According to this rule, the kilograms of the user are multiplied with the covered kilometre. Researches came to the result that this easy method is close to complex computational operations. [Calorie_Calculation]

6.8.4 Close and Save Training Mode

The stop button ensures that the training mode can be completed. In order to avoid the unintentional data storage, a dialog box pops up which asks the user if he/she really wishes to either save, or delete the result.

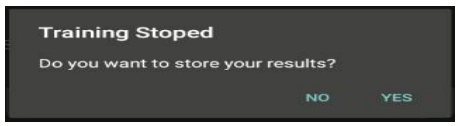

Figure 6.34 - Alert Dialog

Android offers the possibility for showing pre-built dialogs on the screen, which catches the users attention. The class AlertDialog is most suitable for this confirmation. It consists of a title, a message, one button to save and one to exit without saving the result. [Android_Dialog]

The behaviour of the AlertDialog is implemented in the method createAlertDialog(). At first, an object of type AlertDialog.Builder needs to be initialised. The methods setTitle() and setMessage(), are responsible for displaying information. Two buttons, one for acceptation and one for negation, are handled by the methods setPositiveButton() and setNegativeButton(). The listeners are provided as callback methods and are handed over as parameters. While the exit without saving has only the consequence to navigate to the menu activity, the other decision must establish a database connection and store the information. Finally, the methods create() and show() should be called so that the dialog appears. [Android_Dialog]

```
public void createAlertDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(R.string.title_ResultDiary);
    builder.setMessage(R.string.message_ResultDiary);
    builder.setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            // User clicked YES button
            //Date
            Calendar calendar = new GregorianCalendar();
            String date = calendar.get(Calendar.DAY_OF_MONTH) + ". " + calendar.get(Calendar.MONTH) + ". " + calendar.get(Calendar.YEAR);
            time = tvTime.getText()+"";
            distance = tvDistance.getText()+"";
            calorie = tvCalorie.getText()+"";
            speedA = tvSpeedAverage.getText()+"";
            //Insert into Database
            dbHelper = new DbHelper(TrainingModeActivity.this);
            dbHelper.insertTraining(date, time, distance, calorie, speedA);
            finish();
            System.exit(0);
            Intent intent = new Intent(TrainingModeActivity.this, MenuActivity.class);
            startActivity(intent);
            Toast.makeText(TrainingModeActivity.this, R.string.save_result, Toast.LENGTH_LONG).show();
        }
    });
    builder.setNegativeButton(R.string.no, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            // User clicked NO button
            finish();
            System.exit(0);
            Intent intent = new Intent(TrainingModeActivity.this, MenuActivity.class);
            startActivity(intent);
        }
    });
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

Figure 6.35 - Create Alert Dialog

```
//TRAINING
//NAME and Attribute of Table "training"
public static final String TABLE_NAME_TRAINING = "training";
public static final String DATE = "date";
public static final String TIME = "time";
public static final String DISTANCE = "distance";
public static final String CALORIE = "calorie";
public static final String SPEEDA = "speeda";

//SQL-Statement Create Table Training
private static final String TABLE_TRAINING_CREATE =
        "CREATE TABLE "
                + TABLE_NAME_TRAINING + " (" + _ID
                + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + DATE + " TEXT, "
                + TIME + " TEXT, "
                + DISTANCE + " TEXT, "
                + CALORIE + " TEXT, "
                + SPEEDA + " TEXT);";
```

Figure 6.36 - Create Table Training

In order to save the result to the table *training*, the date is needed. This information is supplied by a Calendar object of type GregorianCalendar. A String variable concatenates the day, month and year. [Android_Calendar]

The same process as the insertion new user is triggered. At first, a database connection is established. [Android_SQLite]

The invoke of the method insertTraining() ensures that the information about the actual date, time, distance, calorie consumption and average speed is persistently stored in the SQLite database. [Android_SQLite]

```
public void insertTraining(String date, String time, String distance, String calorie, String speedA){
    long rowId = -1;
    try {
        //Open Connection
        SQLiteDatabase db = getWritableDatabase();
        //Save Values
        ContentValues values = new ContentValues();
        values.put(DATE, date);
        values.put(TIME, time);
        values.put(DISTANCE, distance);
        values.put(CALORIE, calorie);
        values.put(SPEEDA, speedA);
        //Insert Into Table
        rowId = db.insert(TABLE_NAME_TRAINING, null, values);
    } catch (SQLiteException e) {
        Log.e(TAG, "insert() ", e);
    } finally {
        Log.d(TAG, "insert(): rowId= " + rowId);
    }
}
```

Figure 6.37 - Insert New Training

6.8.5 Samsung Accessory Protocol

It is possible to connect a Samsung wearable with the Android device and to get the information from the training mode on the screen of the wearable device. This allows the user to read the information directly from the screen of the Samsung accessory.

Therefore, a connection has to be established where data can be transmitted. For this task, Samsung provides the Samsung Accessory Protocol (SAP) which is located in the Samsung SDK. The SAP is not only suitable for a wristwatch, but also for TV, speakers, headphone and many more. This section shows how the source code on the Android device is implemented. [Accessory_Guide1]

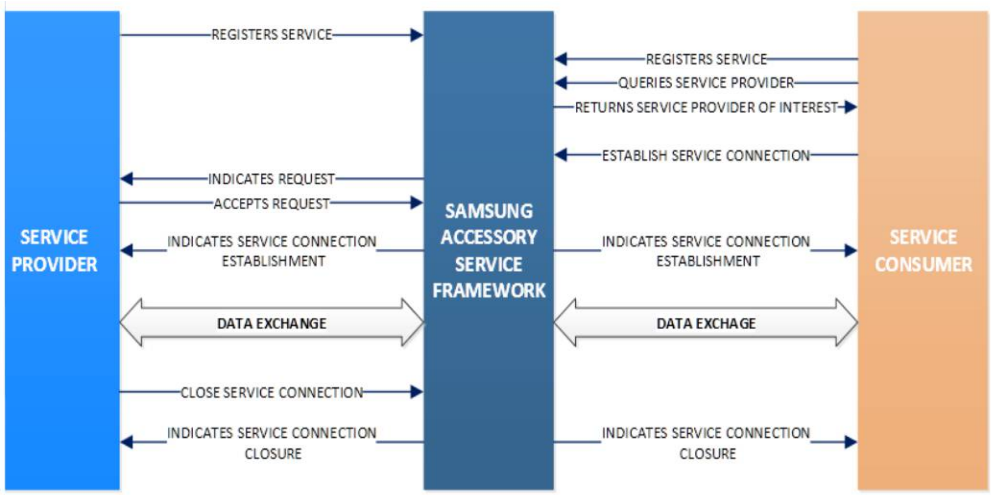

Figure 6.38 - Samsung Accessory Protocol Framework

The architecture of this network protocol consists of two Accessory Peer Agents which represent the Android and the Tizen device. While the Android device acts as the Service Provider, the Tizen device functions as the Service Consumer. In order to be able to transfer data, everyone needs to create a socket. A socket represents the endpoint of the communication channel for data transmission. [Accessory_Guide1]

The permissions to allow the usage of the accessory framework and to send data via Bluetooth must be entered into the manifest file.[Accessory_Guide1]

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="com.samsung.accessory.permission.ACCESSORY_FRAMEWORK"/>
<uses-permission android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY"/>
<uses-permission android:name="com.samsung.WATCH_APP_TYPE.Companion"/>
<uses-permission android:name="com.samsung.wmanager.ENABLE_NOTIFICATION"/>
```

Figure 6.39 - Accessory Manifest

```
<meta-data
    android:name="AccessoryServicesLocation"
    android:value="/res/xml/sapservices.xml"/>
```

Figure 6.40 - Accessory Location Manifest

To operate as Peer Agent, this service must be declared in the manifest file. What services are, as well as they differ from activities can be read in the seminar paper. [Seminar16, P. 15 ff]

```
<service android:name="at.haraldbernhard.joggingcoach.SAPServiceProvider"/>
<receiver android:name="com.samsung.android.sdk.accessory.RegisterUponInstallReceiver">
    <intent-filter>
        <action android:name="com.samsung.accessory.action.REGISTER_AGENT"/>
    </intent-filter>
</receiver>

<receiver android:name="com.samsung.android.sdk.accessory.ServiceConnectionIndicationBroadcastReceiver">
    <intent-filter>
        <action android:name="com.samsung.accessory.action.SERVICE_CONNECTION_REQUESTED"/>
    </intent-filter>
</receiver>
```

Figure 6.41 - Service Manifest

The sent and received messages between the devices are handled in the background and need no own graphical user interface. Android supports the possibility to do this job with broadcast receivers. Broadcast Receivers are interfaces between software components and react on notifications concerning the whole system. [Android12, P. 106]

The description of the service in sapservice.xml is deposit in *res/xml*. This file includes the settings of the Android device and is also clarified in the manifest file within a meta-data tag.

```
<resources>
    <application name="DevelopingForAndroid">
        <serviceProfile
            role="provider"
            name="sapsample"
            id="/system/sapsample"
            serviceImpl="at.haraldbernhard.developingforandroid.SAPServiceProvider"
            version="1.0"
            serviceLimit="any"
            serviceTimeout="10">
            <supportedTransports>
                <transport type="TRANSPORT_BT"/>
            </supportedTransports>
            <serviceChannel
                id="321"
                dataRate="low"
                priority="high"
                reliability="enable"/>
        </serviceProfile>
    </application>
</resources>
```

Figure 6.42 - SAP Service XML

The implementation of the Samsung Accessory Protocol is created in the class SAPProvider.java that extends from SAAgent. To use the necessary classes from the Samsung SDK, it needs to be imported into the Android Studio Project firstly. [Accessory_Guide1]

To import an external library:

1. Press Ctrl+Shift+Alt+S to open the Project Structure dialogue
2. Select Modules > Dependencies
3. Click on the plus sign
4. Import the desired libraries via drag and drop into the *libs* folder [Import_Library]

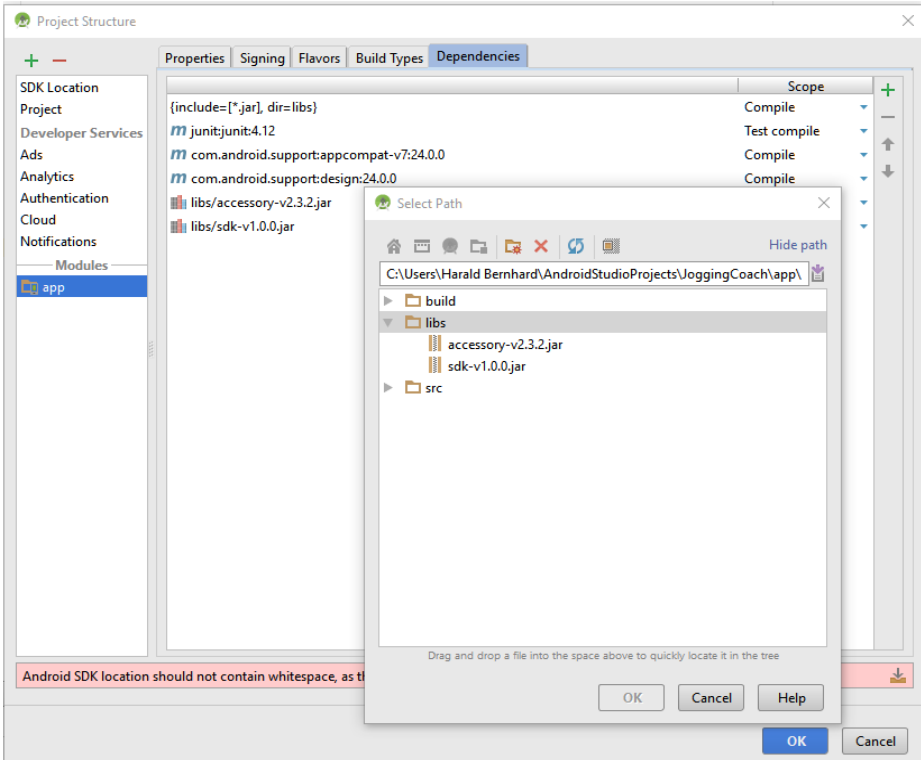

Figure 6.43 - Import Library

After the service is registered, the onCreate() method is called to initialize a SA object. This object represents the Accessory Service Provider. Then, the application listens if a service consumer, in this case the Tizen wearable, wants to establish a connection. [Accessory_Guide1]

```
@Override
public void onCreate() {
    super.onCreate();

    SA myAccessory = new SA();
    try{myAccessory.initialize(this);
        Log.d("SAP Provider", "On Create Try Block");
    }
    catch (SsdkUnsupportedException e){
        Log.d("SAP Provider",  "On create Try Block Error Unsupported Sdk");
    }
    catch(Exception e1){
        Log.e(TAG,  "Cannot initialize Accessory package.");
        e1.printStackTrace();
        stopSelf();
    }
}
```

Figure 6.44 - SAP Method onCreate()

If both devices have the same settings in the service profile, a match can be detected and the establishment process starts. In the body of the method onFindPeerAgentResponse(), influence can be taken on the different scenarios. [Accessory_Guide1]

```
@Override
protected void onFindPeerAgentResponse(SAPeerAgent peerAgent, int result) {
    switch(result){
        case PEER_AGENT_FOUND:
            //Peer Agent is found
            requestServiceConnection(peerAgent);
            Log.e(TAG,  "PeerAgent is found");
            break;
        case FINDPEER_DEVICE_NOT_CONNECTED:
            //Peer Agent are not found, no accessory device connected
            Log.e(TAG,  "PeerAgent not found");
            break;
        case FINDPEER_SERVICE_NOT_FOUND:
            //No matching service on connected accessory
            Log.e(TAG,  "No matching");
            break;
        default:
            Log.e(TAG,  "Default");
    }
}
```

Figure 6.45 - SAP Method onFindPeerAgentResponse()

If a remote Accessory Peer agent is found and the connection request is accepted, the program continues to create a socket. The application is notified on this circumstance and starts the onServiceConnectionResponse() callback method. A new object of type SAPServiceProviderConnection that extends from SASocket is instantiated and stored into a HashMap. [Accessory_Guide1]

```
protected void onServiceConnectionResponse(SAPeerAgent peerAgent, SASocket thisConnection, int result) {
    switch(result){
        case CONNECTION_SUCCESS:
            if(thisConnection != null){
                SAPServiceProviderConnection myConnection = (SAPServiceProviderConnection) thisConnection;
                if(connectionMap == null){
                    connectionMap = new HashMap<Integer, SAPServiceProviderConnection>();
                }
                myConnection.connectionID = (int) (System.currentTimeMillis() & 255);
                Log.d(TAG,  "onServiceConnection connectionID = " + myConnection.connectionID);
                connectionMap.put(myConnection.connectionID, myConnection);
                Toast.makeText(getBaseContext(),  "Connection Established",  Toast.LENGTH_LONG).show();
            }
            else{
                Log.e(TAG,  "SASocket object is null");
            }
            break;
        case CONNECTION_FAILURE_NETWORK:
            Log.e(TAG,  "connection already exists");
            break;
        default:
            Log.e(TAG,  "default");
    }
```

Figure 6.46 - SAP Method onServiceConnectionResponse

In JoggingCoach, the class SAPServiceProviderConnection is implemented as inner class in SAPServiceProvider. For possible failures, the methods onError() and onServiceConnectionLost() are implemented to intercept these mistakes. The method onReceive() ensures that incoming data from the service consumer is processed and the data about the training is sent back. [Accessory_Guide1]

```
@Override
public void onReceive(int channelID, byte[] data) {

    final String information = TrainingModeActivity.getInformationAsJSON();
    final SAPServiceProviderConnection uHandler = connectionMap.get(Integer.parseInt(String.valueOf(connectionID)));
    if(uHandler == null){
        Log.e(TAG,  "Error, can not get SAPServiceProviderConnection handler");
    }

    new Thread(new Runnable(){
        public void run(){
            try{
                uHandler.send(SAP_SERVICE_CHANNEL_ID,  information.getBytes());
            }
            catch(IOException e){
                e.printStackTrace();
            }
        }
    }).start();
}
```

Figure 6.47 - SAP Method onReceive()

This information about the actual time, distance, calorie consumption and average speed is prepared into a JavaScript Object Notation (JSON) format before they are sent to the Samsung gear. This makes it easier to process it. How JSON files look like and how they are used can be read in the seminar paper. [Seminar16, P. 22 ff]

```
public static String getInformationAsJSON(){

    return "{\"time\":\"" + time + " \", \"distance\":\"" + distance + "\", \"speed\": \"" + speedA + "\"}";
}
```

Figure 6.48 - Parse Data To JSON

6.9 Interval Mode

The interval mode is convenient to set workout and recreation phases for the training. It is also possible to select the quantity of how often these phases should be started. To choose the characteristics of the interval training, a pop up window appears.

This view consists NumberPickers to adjust the ranges of the intervals. Additionally, one button to start and another one to cancel the interval training mode are located on the dialog box. [Android_Dialog]

Figure 6.49 - Interval Dialog

For NumberPickers, two implementations are necessary to work properly. Firstly, a style must be declared for the theme attribute. Secondly, it is essential to determine the minimum and maximum value. This can be done with the methods setMinValue() and setMaxValue(). Otherwise, an error may occur. [Android_NumberPicker]

```
<NumberPicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/npWorkoutMin"
    android:theme="@android:style/Theme.Dialog"/>
```

Figure 6.50 - Theme Style Number Picker

If prefabricated dialog boxes such as the AlertDialog are unsatisfactory, it is also possible to use an Activity for this task. The only thing that should be noted is to declare it in the manifest file. An entry ensures that the Activity is treated like a dialog.
[Android_Dialog]

```
<activity android:name="IntervalDialogActivity"
    android:theme="@android:style/Theme.Holo.Dialog"/>
```

Figure 6.51 - Interval Dialog Manifest

The look of the Activity is made in the associated XML file as usual. The implementation of the dialog box is done in the class IntervalDialogActivity that includes only the references and the listeners of the elements. The program logic of the interval training mode is written in IntervalModeActivity that appears when the start button of the dialog box is clicked. [Android_Dialog]

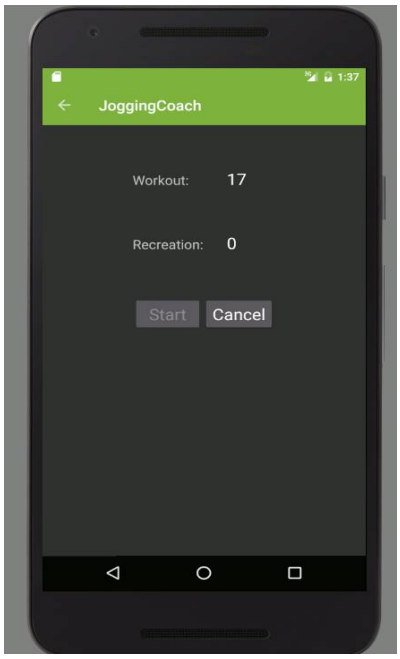

Figure 6.52 - Interval Mode Activity

The calculations of the interval mode is, like the training mode, done in the background with an AsyncTask. More specifically, three separate tasks are required. One of them uses the stopwatch to make the time calculations, while the other ones are used for updating the workout and recreation text views. These tasks are alternating in a loop which is passed through until the count of the intervals is reached. [Android_AsyncTask]

```
public void startTraining(){
    workSec = IntervalDialogActivity.getSecWorkout();
    recSec = IntervalDialogActivity.getSecRecreation();
    intervals = IntervalDialogActivity.getIntervals();

    for(int i = 0; i<intervals; i++){

        //Workout Phase
            stopwatchTask = (StopwatchTask) new StopwatchTask().executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
            updateWorkoutUITask = (UpdateWorkoutUITask) new UpdateWorkoutUITask().executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
            try {
                    Thread.sleep(IntervalDialogActivity.getSecWorkout()*1000);
                    stopwatchTask.cancel(true);
                    updateWorkoutUITask.cancel(true);
                }
                catch (InterruptedException e){
                    e.printStackTrace();
                }

        //Recreation Phase
        stopwatchTask = (StopwatchTask) new StopwatchTask().executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
        updateRecreationUITask = (UpdateRecreationUITask) new UpdateRecreationUITask().executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
        try {
            Thread.sleep(IntervalDialogActivity.getSecRecreation() * 1000);
            stopwatchTask.cancel(true);
            updateRecreationUITask.cancel(true);
        }
        catch(InterruptedException e){
            e.printStackTrace();
        }
    }
}
```

Figure 6.53 - Start Interval Training

6.10 Training Diary

The third possibility in the menu is to look for the latest results. A click on the diary button opens a new screen and offers information about the performance trend. The aim of this activity is to put the result information of the table training into a list view. Android facilitates many programming work with adapters. One implementation is the cursor adapter that provides the activity with necessary data from the database.

Figure 6.54 - Training Diary Activity

[Android_CursorAdapter]

The class TrainingDiaryAdapter that extends from CursorAdapter needs to be created and should implement two methods. The method newView() is called when the application is invoked at the first time and thus, it is responsible for creating the list.

[Android12, P. 281 ff]

```
@Override
public View newView(Context context, Cursor cursor, ViewGroup viewGroup){
    return inflator.inflate(R.layout.training_result, null);
}
```

Figure 6.55 - New View

The second method bindView() accepts the Cursor object as parameter and fills the list view with the data. [Android12, P. 281 ff]

```
@Override
public void bindView(View view, Context context, Cursor cursor) {
    TextView tvDate = (TextView) view.findViewById(R.id.textViewDateAdapter);
    tvDate.setText("Date: " + cursor.getString(ciDate));
    TextView tvTime = (TextView) view.findViewById(R.id.textViewTimeAdapter);
    tvTime.setText("Time: " + cursor.getString(ciTime));
    TextView tvDistance = (TextView) view.findViewById(R.id.textViewDistanceAdapter);
    tvDistance.setText("Distance: " + cursor.getString(ciDistance));
    TextView tvCalorie = (TextView) view.findViewById(R.id.textViewCalorieAdapter);
    tvCalorie.setText("Calorie: " + cursor.getString(ciCalorie));
    TextView tvSpeedA = (TextView) view.findViewById(R.id.textViewSpeedAAdapter);
    tvSpeedA.setText("Speed: " + cursor.getString(ciSpeedA));
}
```

Figure 6.56 - Bind View

In TrainingDiaryActivity, there is only the invocation of the TrainingDiaryAdapter constructor that needs the Cursor object as parameter. [Android12, P. 281 ff]

To get the cursor with the data from the table, a SQL query statement has to be sent. This information is sorted descendingly in order to get the latest result displayed on the top. All other tasks are taken over from the Adapter. [Android_SQLite]

```
public Cursor getAllTraining(){
    SQLiteDatabase db = getReadableDatabase();
    String[] column = {_ID, DATE, TIME, DISTANCE, CALORIE, SPEEDA};
    Cursor cursor = db.query(TABLE_NAME_TRAINING, column, null, null, null, null, DATE+" DESC");
    return cursor;
}
```

Figure 6.57 - Get All Training Results

7 Devoloping for Tizen

7.1 Create a New Tizen Web Application

The Tizen implementation of JoggingCoach is created as web application. This means that this app needs a host device to install and run the program. Therefore, the Gear Manager must be installed on the Android. The Gear Manager is available on Google Play Store. [Seminar16, P. 25 ff]

The Tizen Web application is created with HTML, CSS and JavaScript. While the look of the Tizen consumer application is written in HTML and CSS, the program logic is implemented in JavaScript. [Tizen_WebApp]

To create a new Tizen Web Application:

1. Open the Tizen IDE
2. Click File > New > Tizen Web Project
3. Select a template, enter a project name and click on finish [Tizen_WebApp]

] n-

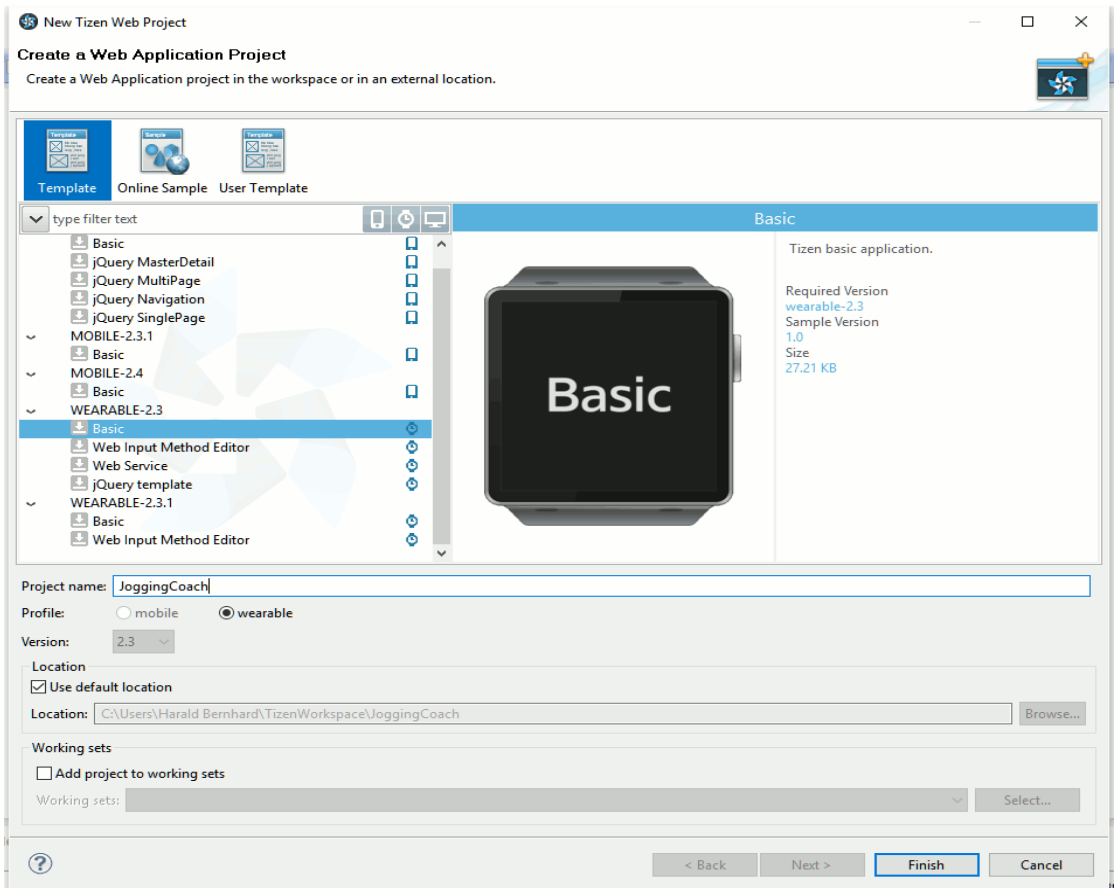

Figure 7.1 - Create Tizen Web Application

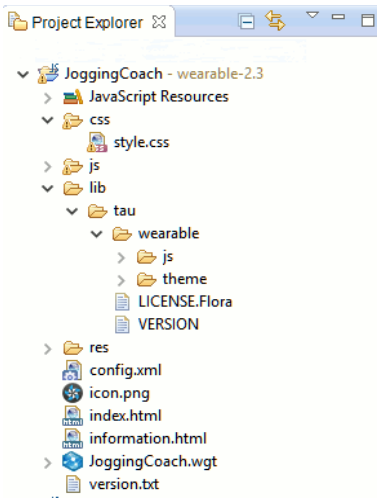

Figure 7.2 - Tizen Project Structure

7.2 Overview

The Tizen wearable device is used to make it easier to get the information procurement during the workout. Consequently, a user interface and the Samsung Accessory Protocol for data exchange needs to be implemented.

7.3 Graphical User Interface

7.3.1 Tizen Advanced User Interface (TAU)

The displays of a Samsung gear differs from mobile devices in the size of their screen. However, all displayed information can be easily read by a user. TAU is a user interface framework that allows a more professional implementation of design and progam logic. [Tizen_UI]

The user interface of the Tizen device consists of two views. The welcome screen contains the greeting and a button to connect it with the Android device, while the main screen includes the text fields about time, distance, calorie consumption and average speed. At the bottom, there is a button to update the data.

Figure 7.3 - Launch Page UI

Figure 7.4 - Main Page UI

Tizen has established a *tau* folder that includes the necessary CSS and JavaScript files. These files must be imported firstly to the HTML file in order to be able to use them. The path of the framework is added in the head element. Furthermore, the scaling of the device should be adjusted. [Tizen_TAU]

```
<meta name="viewport" content="width=device-width, user-scalable=no"/>
<link rel="stylesheet" href="./lib/tau/wearable/theme/default/tau.css"/>
```

Figure 7.5 - Import TAU CSS

The pre-implemented Tau JavaScript functions are imported to the body section. [Tizen_TAU]

```
<script type="text/javascript" src="./lib/tau/wearable/js/tau.js"></script>
```

Figure 7.6 - Import TAU JavaScript

The screen of the user interface is denoted as “pages” in TAU. The structure of a page consists of a header, content and footer area. [Tizen_TAU]

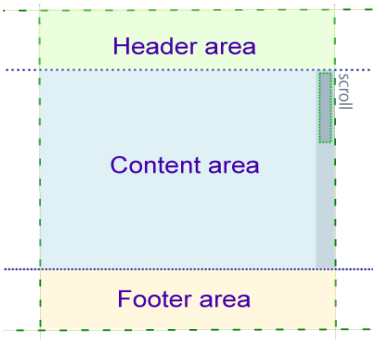

Figure 7.7 - Page Structure

Source: [Tizen_Page]

If the user wants to use the predefined design structure, the desired element needs to be marked up with the ui-page attribute. [Tizen_Page]

```
<div class="ui-page" id="main">
```

Figure 7.8 - Define Class Page

The connect button in the footer area of the launch page starts the connection establishment and navigates the user to the main screen.

```
<!-- LAUNCH PAGE -->
<div class="ui-page ui-page-active" id="launch">
    <!-- Content -->
    <div class="ui-content">
        <div id="welcomeInfo">
            <h3>Welcome to JoggingCoach </h3>
        </div>
    </div>

    <div class="ui-footer">
    <button id="buttonConnect" class="ui-btn">Connect</button>
            <script>
            var btCon = document.getElementById("buttonConnect");
            btCon.addEventListener("click", function(){
                tau.changePage("information.html");
                connect();
                });
            </script>
    </div>
</div>
```

Figure 7.9 - Launch Page

To indicate the launch page, the class ui-page-active has to be added. The behaviour of the button is implemented into JavaScript tags. After clicking it, the TAU function changePage() navigates the user to the information.html site. Additionally, the connect() function to establish a data transfer channel is executed. [Tizen_Page]

The second page, which informs the user about his/her actual performance, is divided into the content and footer area. While the content area includes a table with the information about the workout, the footer has a button to update the data.

```
<body>
    <!-- MAIN PAGE -->
    <div class="ui-page " id="main">
        <!-- CONTENT -->
        <div class="ui-content">
            <table class="table">
                <tr>
                    <th class="info-tv">Time</th>
                    <th class="info-tv">Distance</th>
                </tr>
                <tr>
                    <td class="info">0:0:0</td>
                    <td class="info">0.0</td>
                </tr>
            </table>
            <table class="table">
                <tr>
                    <th class="info-tv">Calorie</th>
                    <th class="info-tv"> Speed</th>
                </tr>
                <tr>
                    <td class="info">0</td>
                <td class="info">0.0</td>
                </tr>
            </table>
        </div>

        <!-- FOOTER -->
        <div class="ui-footer">
            <button id="buttonFetch" class="info-btn ui-btn " onClick="fetch();">Fetch</button>
        </div>
    </div>
    <script type="text/javascript" src="./lib/tau/wearable/js/tau.js"></script>
</body>
```

Figure 7.10 - Main Page

7.4 Program Logic

7.4.1 Samsung Accessory Protocol

Generally, two tasks have to be implemented on the Tizen side. The data should be fetched from the Android device and displayed on the screen. In order to do this, a connection needs to be established to retrieve the required information. The data transmission is executed with the help of the Samsung Accessory Protocol. [Accessory_Guide2]

To use this framework, some definitions should be announced in the configuration file. Firstly, the location of the service description must be added. [Accessory_Guide2]

```
<tizen:metadata key="AccessoryServicesLocation" value="res/xml/sapservices.xml"/>
```

Figure 7.11 - SAP Service Description

Secondly, the required privilege has to be declared.

```
<tizen:privilege name="http://developer.samsung.com/privilege/accessoryprotocol"/>
```

Figure 7.12 - SAP Privilege

Furthermore, the service consumer is described in sapservice.xml which is located in the *res* folder. Apart from other features, it defines the name, role, transport type and identification. [Accessory_Guide2]

```
<resources>
    <application name="JoggingCoach">
        <serviceProfile
            id="/system/sapsample"
            name="sapsample"
            role="consumer"
            version="2.0"
            serviceLimit="ANY"
            serviceTimeout="10">
            <supportedTransports>
                <transport type="TRANSPORT_BT"/>
            </supportedTransports>
            <serviceChannel
                id="321"
                dataRate="low"
                priority="low"
                reliability="enable"/>
        </serviceProfile>
    </application>
</resources>
```

Figure 7.13 - SAP Service Description

The connection establishment is triggered by clicking the connect button. That leads to the invocation of the associated function connect(). Firstly, the user should check if a socked to a remote device already exists. If that is not the case, it must be checked if a service profile is specified in the configuration file by calling the function requestSAAgent(). Two callback functions onsuccess() and onerror() as parameters need to be handed over. [Accessory_Guide2]

```
function connect() {
    if (SASocket) {
        alert('Already connected!');
        return false;
    }
    try {
        webapis.sa.requestSAAgent(onsuccess, onerror);
    } catch(err) {
        createHTML("exception [" + err.name + "] msg[" + err.message + "]");
    }
}
```

Figure 7.14 - Function connect()

While onerror() is responsible for intercepting possible failures, onsuccess() starts to find the remote peer agent. The characterisation of the specified service is stored as an object and checks if a match with the remote peer agent occurs. The information about the remote system provides the function findPeerAgents(). If the settings of the two devices fit, a new socket is created and the data exchange can be started. [Accessory_Guide2]

```
var peerAgentFindCallback = {
    onpeeragentfound : function(peerAgent) {
        try {
            if (peerAgent.appName == ProviderAppName) {
                console.log("peerAgentFindCallback :: onpeeragentfound " + peerAgent.appname + " || " + ProviderAppName);
                SAAgent.setServiceConnectionListener(agentCallback);
                SAAgent.requestServiceConnection(peerAgent);
            } else {
                console.log("peerAgentFindCallback::onpeeragentfound else");
                alert("Not expected app!! : " + peerAgent.appName);
            }
        } catch(err) {
            console.log("peerAgentFindCallback::onpeeragentfound exception {" + err.name + "] msg [" + err.message + "]");
        }
    },
    onerror : onerror
};

function onsuccess(agents) {
    try {
        if (agents.length > 0) {
            SAAgent = agents[0];
            SAAgent.setPeerAgentFindListener(peerAgentFindCallback);
            SAAgent.findPeerAgents();
            console.log("onsuccess " + SAAgent.name);
        } else {
            createHTML("Not found SAAgent!!");
            console.log("onsuccess else");
        }
    } catch(err) {
        console.log("onsuccess exception [" + err.name + "] msg[" + err.message + "]");
    }
}
```

Figure 7.15 - Find Peer Agent

The retrieve of actual data is handled by pressing the fetch button on the smart gear. The setDataReceiveListener() function of the socket object notes if an updated message was received and it takes care of the information updating on the display. [Accessory_Guide2]

```
function fetch(){
    try{
        SASocket.setDataReceiveListener(onreceive);
        SASocket.sendData(CHANNELID,"");
    }
    catch(err){
        console.log("exception [" + err.name + "] msg [ " + err.message + "]");
    }
}
```

Figure 7.16 - Fetch Data

The received data about time, distance, calorie consumption and average speed is parsed into a JSON object to allow a more easier handling. This split information is now assigned to the associated text views. [Accessory_Guide2]

```
function createHTML(log_string){
    var log= document.getElementById('logResult');
    log.innerHTML = log_string;
}

function createInfoHTML(info){
    var obj = JSON.parse(info);
    document.getElementById('time').innerHTML=obj.time;
    document.getElementById('distance').innerHTML=obj.distance;
    document.getElementById('speed').innerHTML=obj.speed;
    document.getElementByID('calorie').innerHTML=obj.calorie;
}
```

Figure 7.17 - Create Information

8 Testing and Installing

8.1 AVD Manager

The Android SDK includes software to test applications on emulators. An emulator imitates a physical Android device. With the Android Virtual Device Manager (AVD) it is possible to reconstruct devices and set all desired characterisations a real device has. The AVD Manager starts by pressing the icon on the menu bar of Android Studio. [Android_Virtual_Device] 

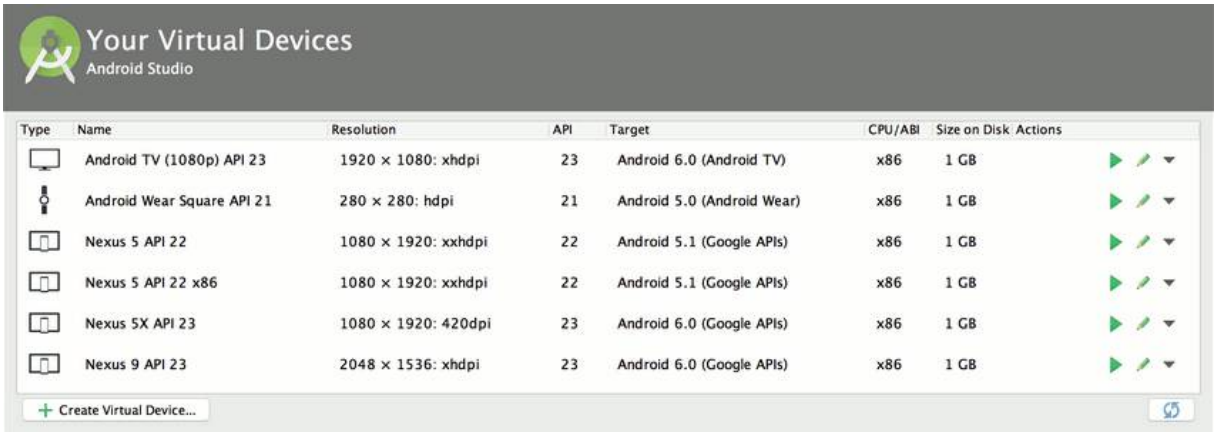

Figure 8.1 - Android Virtual Devices

Source: [Android_Virtual_Device]

At the left bottom there is a button to add a new Virtual Device and the AVD Manager which helps to set the desired configurations. After that, the application on the virtual device starts by clicking Run > Run. This process can be very time consuming. [Android_Virtual_Device]

8.2 Android Debug Bridge

The Android Debug Bridge (ADB) allows engineers to run and test applications on an emulator or real device via command line. This program provides a client to send commands, a deamon to execute commands and a server for interaction between deamon and client. [Android_Debug_Bridge]

In order to use the ADB, the developer options must be unlocked on the Android device firstly.

1. Click on *Settings* > *About phone*
2. Tap *Build number* seven times

3. Open the locked *Developer options*
4. Check *USB-Debugging*
5. Connect the device via USB to the computer
6. Agree the RSA conditions

The device is now connected to the computer and commands can be executed via a shell. [Android_Debug_Bridge]

8.2.1 Testing SQLite

Sometimes it is better to test applications directly on the computer instead of installing and running the app every time on the device. For SQLite 3 it is also possible to send commands from a remote shell to an emulator. [Android_Test_SQLite]

1. Run the application on an emulator
2. Open a terminal and enter *adb shell* (if no environment variable is set, navigate to the platform-tools folder of the Android SDK)
3. Enter *sqlite3*
4. Navigate to the path where the database is stored on the emulator

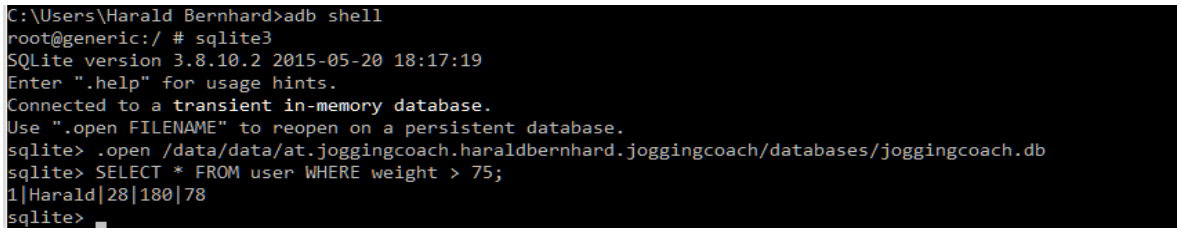

Figure 8.2 - SQLite Command Line

Now SQL-statements can be entered to create, read, update or delete and sent to the emulator where they get executed. Other interesting commands are .help to give possible instructions and .exit to leave the ADB shell. [Android_Test_SQLite]

8.2.2 Tizen Certification

Tizen applications have been certificated before they are executable. An author and device certification must be created to protect the device. This safety measure serves for defective and unintentional apps.

To create an author certificate in the Tizen IDE, the Certificate Extension is needed. This functionality can be installed with the Update Manager. [Tizen_Certificate]

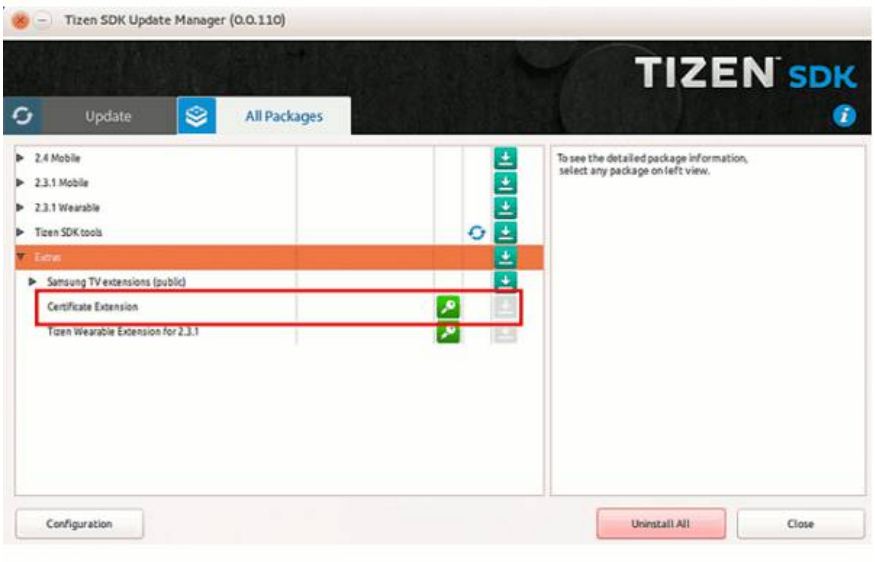

Figure 8.3 - Install Certificate Extension

A new button to request and register a certificate appears in the menu bar of the Tizen IDE. A Certificate Signing Request file (CSR) gets generated by clicking the orange marked button which is indicated in the picture above. Then, the form must be filled in to receive the file. This file needs to be signed on the Samsung web page and gets retrieved automatically. [Seminar16, P. 25 ff]

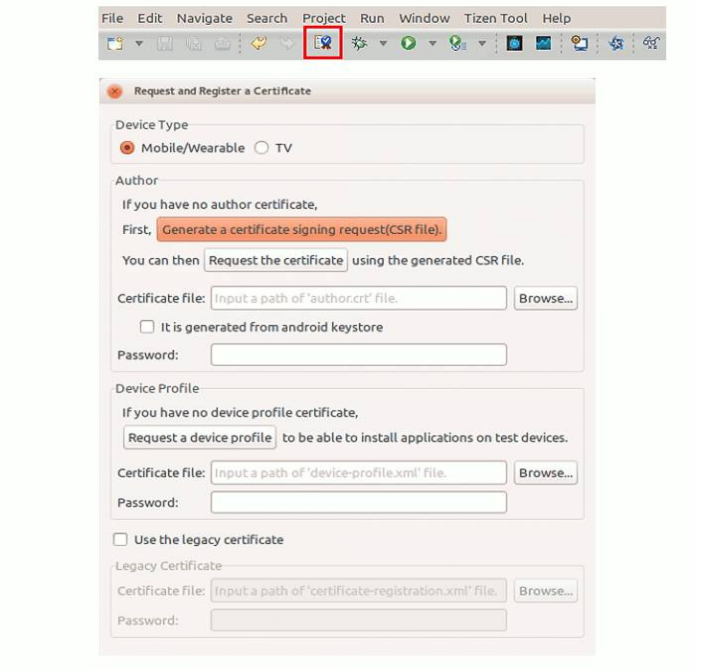

Figure 8.4 - Register Certificate

The *csr* file must be uploaded to require an email with the signed certificate file. This process needs to be done for a device certificate file too. A detailed description for getting the device certificate file can be found in the seminar paper. [Seminar16, P. 25 ff]

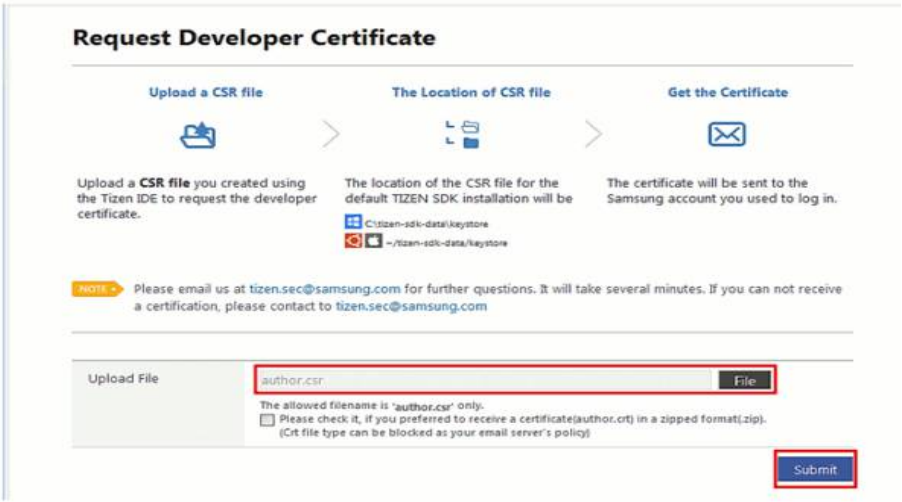

Figure 8.5 - Request Developer Certificate

Finally, the two files, one for the author and one for the device, must be added to the security profile. To open this mask, click on Window > Preferences > TizenSDK-Security Profiles. [Seminar16, P. 25ff]

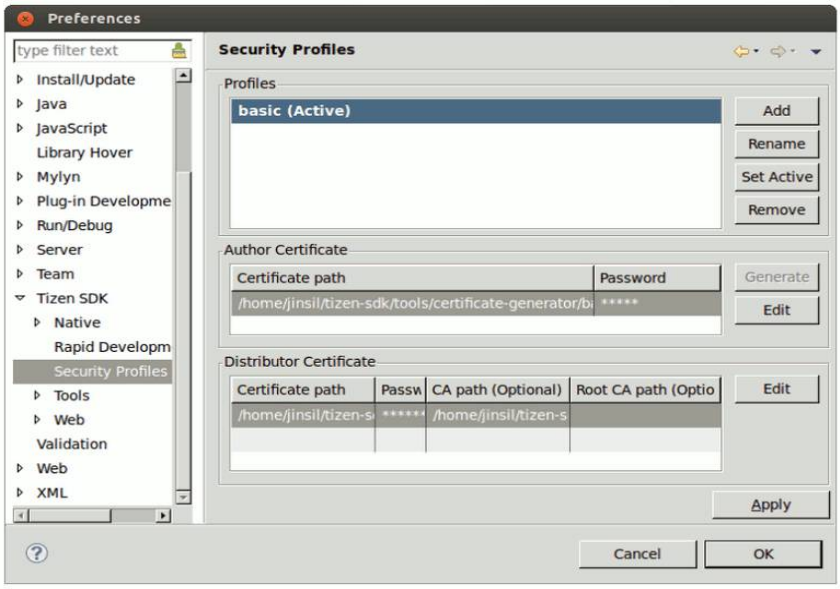

Figure 8.6 - Security Profiles

8.2.3 Test Tizen Applications

The Tizen SDK includes, like the Android SDK, an Emulator Manager to create virtual devices to test the application. The Emulator Manager can be started from the Tizen SDK folder or directly from the Tizen IDE. [Tizen_Emulator]

The user interface for the Emulator Manager allows to create and adapt emulators for different devices. If a virtual device is not used any more, it can also be deleted. [Tizen_Emulator]

Figure 8.7 - Tizen Emulator

After the emulator got booted up completely, it is possible to test the application. [Tizen_Emulator]

8.2.4 Testing Joggingcoach

One solution for testing Android and Tizen applications together without installing them every time is to connect one device via USB. This section shows the testing of the Android side on a smartphone and the Tizen application on an emulator. Located in the *tools* folder of the Samsung SDK, there are the two files SAAccessoryService_Emul.apk and SASystemProviders_Emul.apk which should be installed on the Android smart device. [Accessory_Guide1]

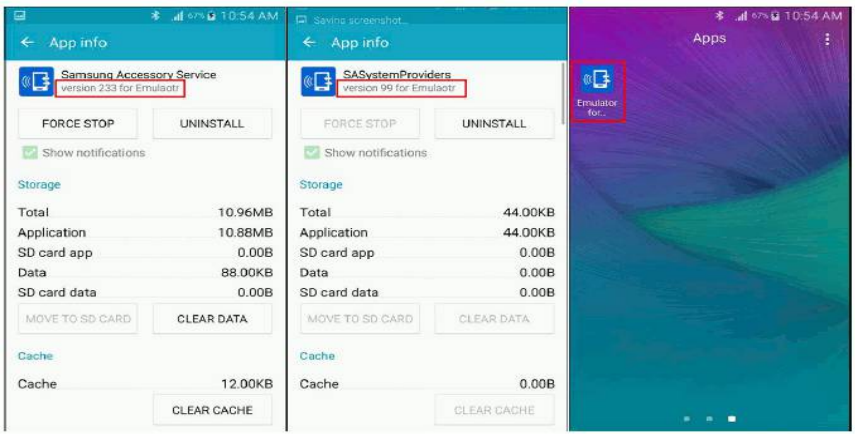

Figure 8.8 - App Info

Source: [Accessory_Guide1]

To use the Android Debug Bridge in order to test the application:

1. Start the *Emulator for Samsung Accessory* application
2. Connect the Android device and the computer via USB
3. Execute *adb -d forward tcp:8230 tcp:8230* in the command line of the terminal
4. Start the application in the Tizen IDE and wait for a complete boot up

The text of the emulator program on the Android device has changed from Disconnected to Connected. The construction is now ready for testing. [Accessory_Guide1]

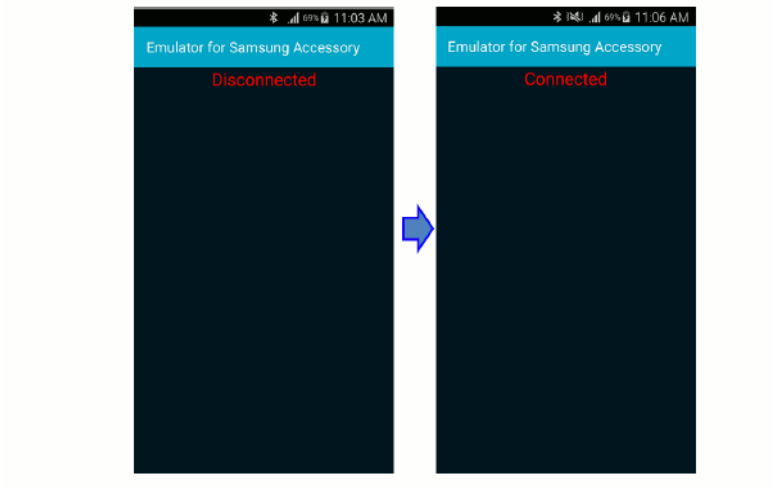

Figure 8.9 - Emulator For Samsung Accessory

Source: [Accessory_Guide1]

8.2.5 Installing Applications

To run Tizen web applications, the Samsung Gear Manager is required on the Android device. It helps to connect an Android host device with a Samsung wearable. [Seminar16, P. 19 ff]

As web application, the Tizen program must be inserted into the Android application only. The installation is done by the Samsung Gear Manager. To create an executing WGT file, open *Project* > *Build Project* in the Tizen IDE. This file has to be inserted into the *assets* folder of the Android Studio project. [Tizen_Installing]

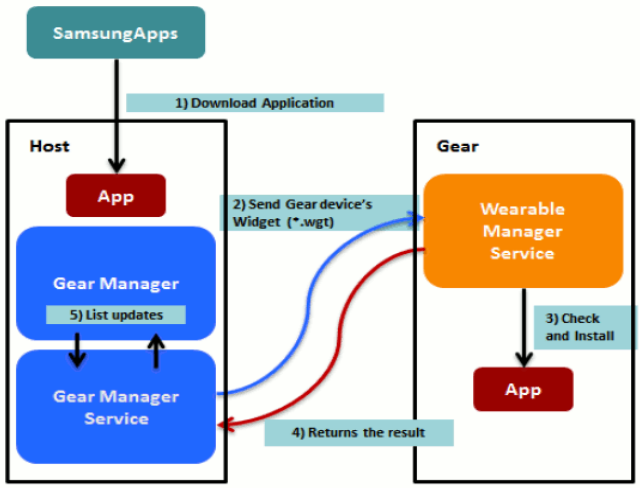

Figure 8.10 - Installing Web Application

The Android project with the *wgt* file, has also be packaged now. Before this can be done, it needs to be signed firstly. In order to do so, click on *Build* > *Generate Signed APK*. It is necessary to have a keystore for this procedure. [Android12, P. 79 ff]

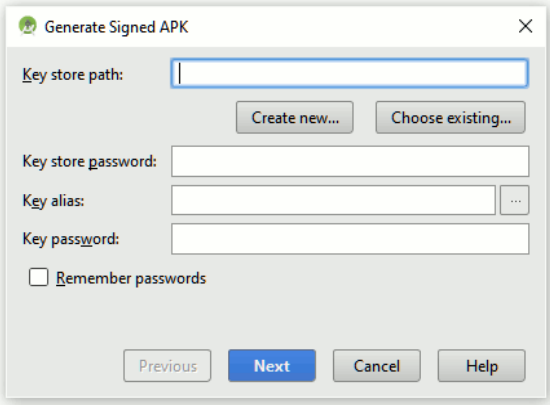

Figure 8.11 - Generate Signed APK

If no keystore exists, it can be generated by pressing *Create new*... The keystore is a digital certificate that consists of a public/private key pair and ensures the authentication of the app. The public key is attached to the app, whereas the private key keeps in possession of the producer. Thus, only the owner of the private key is able to distribute and update the application. [Android_SignApp]

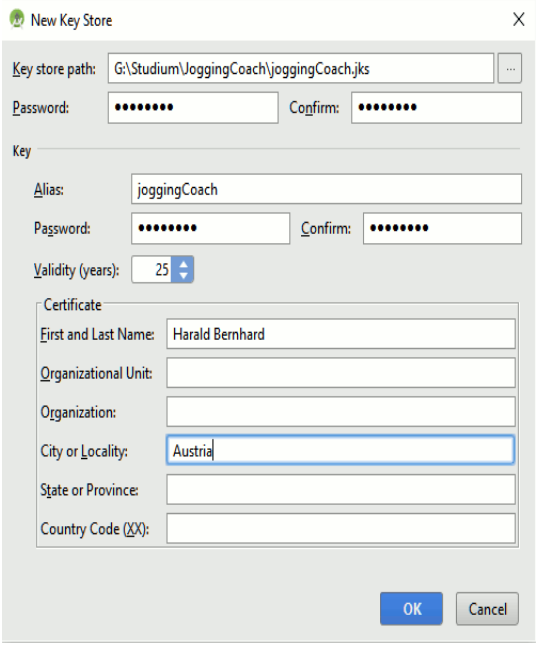

Figure 8.12 - Create New Key Store

The generated Android Application Package (APK) file includes all components and is ready for distribution on Google Play Store [Android12, P. 79 ff]

9 Deployment

9.1 Google Play Store

The finished application is now ready for uploading to the Google Play Store. The Play Store is a platform to make applications available for users and is pre-installed on most android devices. However, some arrangements must be made. [Android12, P. 79 ff]

The first thing a user see after downloading and installing JoggingCoach is the logo on the display to start the app. Thus, it is important to include a picture for the app. Android Studio stores a default logo by creating a new Android application. These sample pictures exist in different sizes and are located in *ic_launcher.png*, a subfolder of *res/mipmap*. [Android12, 79 ff]

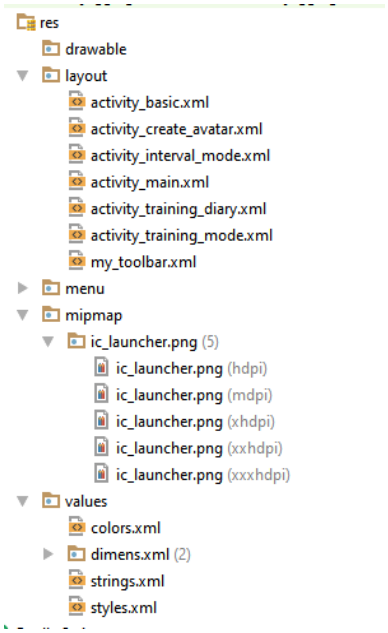

Figure 9.1 - Mipmap

In order to display an own logo, pictures in the sizes 36 x 36, 48 x 48, 72 x 72 and 96 x 96 are sufficient for the possible variations of smart phones. [Android12, 79 ff]

To upload an application, Google provides a web page to perform this process. The URL of this page is *https://play.google.com/apps/publish*. A Gmail account to log in is required. The Developer Console appears and allows the user to make all important settings for engineers. To use this service, the account must be registered as developer account. The costs will be at 25 $. [Google_Publish1]

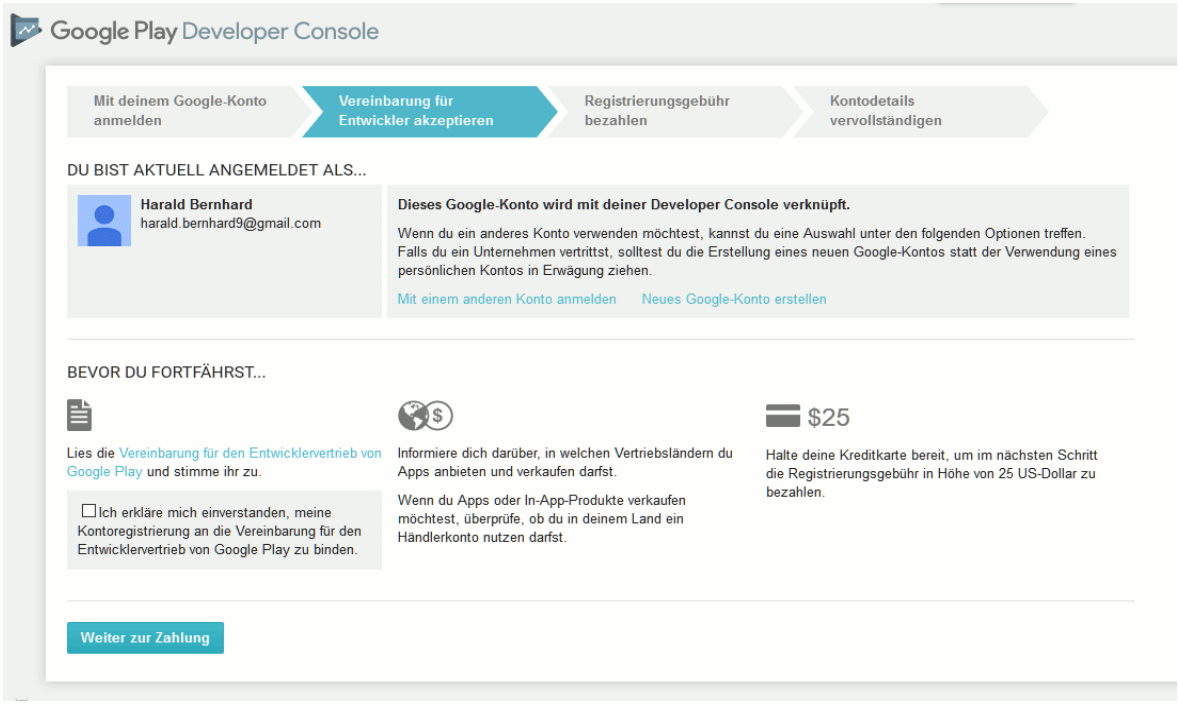

Figure 9.2 - Google Play Developer Console

After the registration is done, JoggingCoach is ready for the upload. [Google_Publish1]

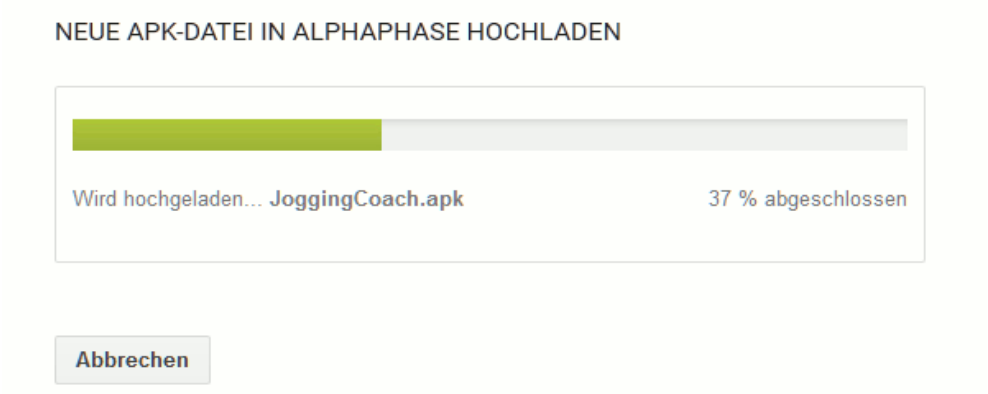

Figure 9.3 - Upload JoggingCoach

A helpful feature of the Developer Console is the testing environment. It is possible to mark the application as alpha or beta version before the distribution and to select a community for this issue. The testing phase enables the developer to get feedback on possible failures and therefore prevents a delivery of an erroneous application. [Google_Publish2]

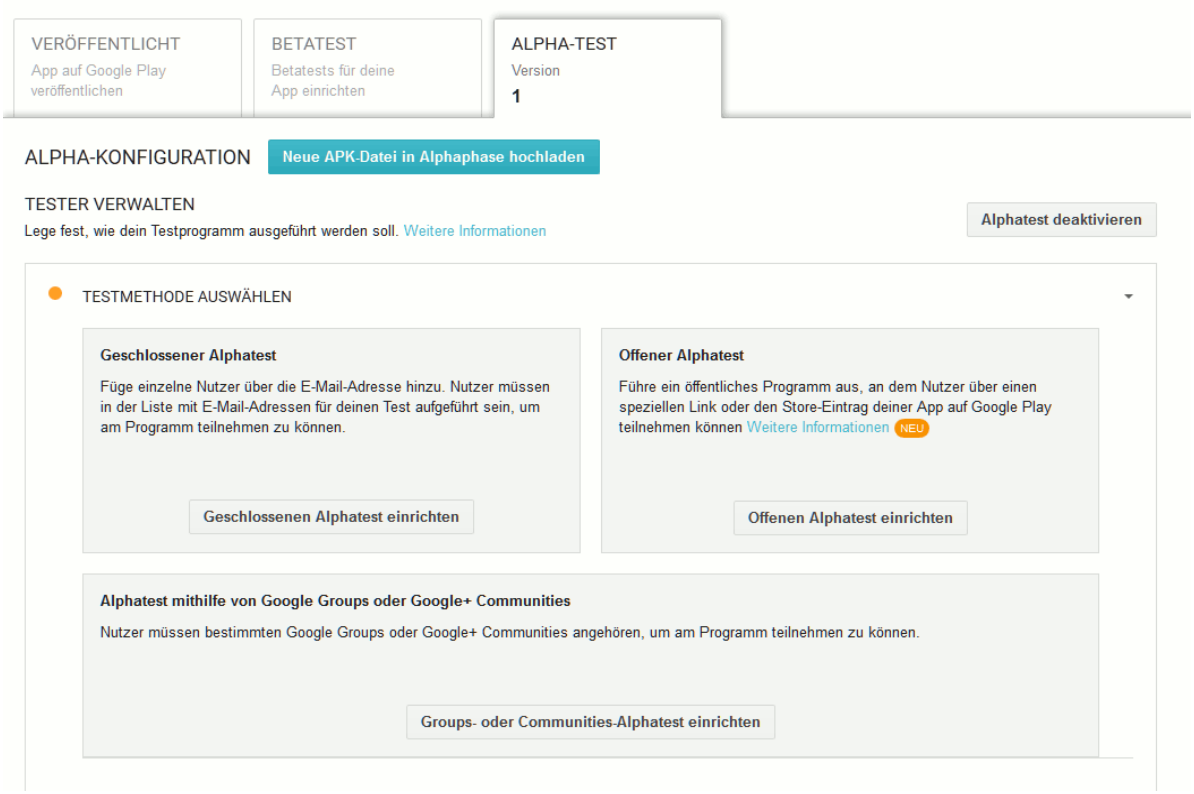

Figure 9.4 - Alpha Test

After JoggingCoach got properly, this configuration is deactivated and published for all Android users. [Google_Publish2]

10 Conclusion

The for creating a fitness tracking app emerged because of the IS Projekseminar. The requirements of this project were to create an application for Android and Tizen devices. As a more or less inexperienced programmer for this two operating systems, it was a big challenge that required a very high degree of induction.

The result of this project was a rather modest application with a simple user interface to track a users workout. Nevertheless, at the end of the seminar, this first version of JoggingCoach was working completely.

At that point, the question was raised if there is a demand for such an application and therefore, if it makes sense to expand the seminar project to a bachelor thesis. Thus, a survey was needed to answer this issue. Moreover, this questionnaire was also used to find out about customer claims for fitness trackers. After the evaluation, it was clear that a desire for a fitness app exists and which functionalities are essential.

Building up on these results, JoggingCoach got three functionalities: firstly, a training mode to record time, distance, calorie consumption, actual and average speed. Concerning this function, it is possible to connect a Samsung device to get this information on the wearable instrument. The second implementation is an interval training mode to select both workout and calm phases. Thirdly, a training diary to get information about the latest results is also included. Furthermore, the user centric design is crucial in order to increase the usability.

The application is available on Google Play Store. The provision of information for students that are faced with the same exercises and not to earn profit is at the center of this project. Therefore, the download is free and the sensible data about the user are stored on the device and not handed over to third parties.

In conclusion, it needs to be stated that this bachelor thesis is written in a way that even programming beginners are able to understand basic Android and Tizen concepts. For better intelligibility, all explanations include instrumental pictures, tips and links to further documentations.

References

Accessory_Guide1: Samsung, Programming Guide: Accessory, http://developer.samsung.com/html/tech-doc/ProgrammingGuide_Accessory.pdf, Retrieved: 2016-08

Accessory_Guide2: Samsung, Samsung Gear Application, Getting started, http://imgdeveloper.samsung.-com/contents/cmm/SamsungGearApplication_GettingStarted_1.0.pdf, Retrieved: 2016-08

Android_Activity: Android, Starting an Activity, https://developer.android.com/training/basics/activity-li-fecycle/starting.html, Retrieved: 2016-08

Android_Appbar: Android, Setting Up the App Bar, https://developer.android.com/training/appbar/set-ting-up.html, Retrieved: 2016-08

Android_AsyncTask: Android, Creating a Background Service, https://developer.android.com/training/run-background-service/create-service.html, Retrieved: 2016-08

Android_Button: Android, Button, https://developer.android.com/reference/android/widget/Button.html, Retrieved: 2016-08

Android_Calendar: Android, GregorianCalendar, https://developer.android.com/reference/java/util/Gre-gorianCalendar.html, Retrieved: 2016-08

Android_CursorAdapter: Android, CursorAdapter, https://developer.android.com/reference/android/wid-get/CursorAdapter.html, Retrieved: 2016-08

Android_Debug_Bridge: Android, Android Debug Bridge, https://developer.android.com/studio/comman-d-line/adb.html, Retrieved: 2016-08

Android_Dialog: Android, Dialogs, https://developer.android.com/guide/topics/ui/dialogs.html Retrieved: 2016-08

Android_Material: Android, Material Design for Android, https://developer.android.com/design/material/index.html, Retrieved: 2016-08

Android_NumberPicker: Android, NumberPicker, https://developer.android.com/reference/android/wid-get/NumberPicker.html, Retrieved: 2016-08

Android_OnClick: Android, Button, https://developer.android.com/reference/android/widget/Button.html, Retrieved: 2016-08

Android_Permission: Android, Requesting Permissions at Run Time, https://developer.android.com/trai-ning/permissions/requesting.html, Retrieved: 2016-08

Android_Project: Android, Create a Project, https://developer.android.com/studio/projects/create-projec-t.html, Retrieved: 2016-08

Android_Runtime: Forays In Software Development, 2012, https://markfaction.wordpress.com/2012/07/15/stack-based-vs-register-based-virtual-machine-architec-ture-and-the-dalvik-vm/, Retrieved: 2016-08

Android_SignApp: Android, Sign Your App, https://developer.android.com/studio/publish/app-si-gning.html, Retrieved: 2016-08

Android_Statistik: Android-Statistik: Lollipop jetzt am meisten verbreitet, 2016, http://www.heise.de/newsticker/meldung/Android-Statistik-Lollipop-jetzt-am-meisten-verbrei-tet-3131083.html, Retrieved: 2016-08

Android_Studio: Android, Meet Android Studio, , https://developer.android.com/studio/intro/index.html, Retrieved: 2016-08

Android_Test_SQLite: Android, sqlite3, , https://developer.android.com/studio/command-line/sqlite3.html, Retrieved: 2016-08

Android_TextWatcher: Android, TextWatcher, https://developer.android.com/reference/android/text/TextWatcher.html, Retrieved: 2016-08

Android_Tizen: Ron Amadeo, The first Tizen smartphone isn't an "Android killer" - it's a bad Android clone, 2015, http://arstechnica.com/gadgets/2015/02/samsung-z1-review-the-first-tizen-smartphone-still-feels-like-plan-b/, Retrieved: 2016-08

Android_SharedPreferences: Android, Saving Key-Value Sets, https://developer.android.com/training/basics/data-storage/shared-preferences.html, Retrieved: 2016-08

Android_SQLite: Android, Saving Data in SQL Databases, https://developer.android.com/training/basics/data-storage/databases.html, Retrieved: 2016-08

Android_Versions: Android-OS: Aktuelle Statistik über die Verbreitung der Versionen, 2015, http://www.stereopoly.de/statistik-android-os/, Retrieved: 2016-08

Android_Virtual_Device: Android, Create and Manage Virtual Devices, https://developer.android.com/studio/run/managing-avds.html, Retrieved: 2016-08

Android/Samsung: Stefan, Samsung Gear S, Gear 2 & Gear 2 Neo auch mit anderen Android Smartphones, 2014, http://www.go2android.de/samsung-gear-s-gear-2-gear-2-neo-auch-mit-anderen-android-smartphones/, Retrieved: 2016-08

Android12, Thomas Künneth, Android 4, Apps entwickeln mit dem Android SDK, Galileo Computing, 2012

Calorie_Calculation: Christian Riedel, Wie hoch ist eigentlich der Kalorienverbrauch beim Laufen, , http://www.netzathleten.de/fitness/richtig-trainieren/item/2327-wie-hoch-ist-eigentlich-der-kalorienverbrauch-beim-laufen, Retrieved: 2016-08

Google_Publish1: Android, Get Started with Publishing, https://developer.android.com/distribute/google-play/start.html, Retrieved: 2016-08

Google_Publish2: Android, Get real user feedback with beta tests, https://developer.android.com/distribute/engage/beta.html, Retrieved: 2016-08

Import_Library: Configuring Module Dependencies and Libraries, https://www.jetbrains.com/help/idea/2016.1/configuring-module-dependencies-and-libraries.html, Retrieved: 2016-08

Samsung_SDK: Samsung, Tizen SDK, , http://developer.samsung.com/gear/develop/sdk#samsung-accessory-sdk, Retrieved: 2016-08

Seminar16: Harald Bernhard, Developing for Android - Fitness Tracker for Samsung Galaxy Note 3 and Samsung Gear, 2016

Tizen_About: Tizen, About, https://www.tizen.org/about, Retrieved: 2016-08

Tizen_Application: Samsung Developers, Getting Started, http://developer.samsung.com/gear/develop/getting-started. Retrieved: 2016-08

Tizen_Certificate: Tizen, Issuing a Tizen Certificate (From Certificate Extension ver 1.2), https://developer.tizen.org/ko/community/tip-tech/issuing-tizen-certificate-certificate-extension-ver-1.2?langredirect=1#CommercialDevices, Retrieved: 2016-08

Tizen_Emulator: Tizen, Emulator, https://developer.tizen.org/development/tools/common-tools/emulator, Retrieved: 2016-08

Tizen_Installing: Tizen, Web Application Development Process, , https://developer.tizen.org/ko/development/getting-started/web-application/application-development-process?langredirect=1, Retrieved: 2016-08

Tizen_Page: Tizen, Managing Pages: Creating and Routing a Page, , https://developer.tizen.org/ko/development/guides/web-application/user-interface/tizen-advanced-ui/managing-pages, Retrieved: 2016-08

Tizen_SDK: Tizen, Installing the Tizen SDK, https://developer.tizen.org/development/tools/download/installing-sdk, Retrieved: 2016-08

Tizen_TAU: Tizen, Hello World: Creating a Basic Hello World Page, https://developer.tizen.org/development/guides/web-application/user-interface/tizen-advanced-ui, Retrieved: 2016-08

Tizen_UI: Tizen, Tizen Advanced UI, https://developer.tizen.org/development/guides/web-application/user-interface/tizen-advanced-ui, Retrieved: 2016-08

Tizen_WebApp: Tizen, Creating Your First Tizen Wearable Web Application, https://developer.tizen.org/development/getting-started/web-application/creating-your-first-tizen-wearable-web-application, Retrieved: 2016-08

Umfrage_Online: enuvo GmbH, www.umfrageonline.com, Retrieved: 2016-08

Wiki_Android_Architecture: Wikipedia, Android (operating system), 2016, https://en.wikipedia.org/wiki/Android_(operating_system), Retrieved: 2016-08

Wiki_Arithmetic: Wikipedia, Arithmetisches Mittel, 2016, https://de.wikipedia.org/w/index.php?title=Spezial:Zitierhilfe&page=Arithmetisches_Mittel&id=155778305, Retrieved: 2016-08

Appendix

Welche Faktoren beeinflussen Menschen beim Erwerb Tracking basierter Fitnessgeräte

Alter:_____ Geschlecht:_____

Wie viele Stunden pro Woche treiben Sie Sport?

□ weniger als 1 Stunde □ 1-3 Stunden □ 4- 7 Stunden □ mehr als 7 Stunden

Welche Sportarten üben Sie aus?

□ Laufen/Joggen □ Fußball □ Tennis □ Schwimmen □ Radfahren
□ Fitness Center □ (Beach-) Volleyball □ Kampfsport □ Sonstiges:_____

Haben Sie Ihre Resultate schon einmal mithilfe technischer Geräte aufgezeichnet? (Runtastic, Smartwatch, ...)

□ Ja □ Nein

Welche Informationen sind Ihnen für die Analyse Ihrer Trainingsleistung wichtig?

□ Trainingsdauer □ Distanz □ Kalorienverbrauch
□ Herzfrequenz □ Höhenmessung □ Durchschnitsgeschwindigkeit
□ Sonstiges: _____

Welche Funktionalitäten sollte Ihrer Meinung nach ein Gerät zum Aktivitätsaufzeichnung bieten?

□ Routenaufzeichnung (Karte) □ Musik-Player □ Erstellung von Trainingsplänen
□ Trainingsrangliste □ Stoppuhr □ Intervalltraining (Belastung/Erholung)
□ Kilometer-Zielvorgabe □ Zeit-Zielvorgabe □ Aktivitätenkalender
□ Schrittzähler □ Social Media posten □ Erinnerungsalarm
□ Stonstiges: _____

Haben Sie schon einmal eine Fitnessuhr gekauft?

□ Ja □ Nein

Wenn Ja, welche Faktoren waren für Sie dabei ausschlaggebend?

□ Preis □ Marke □ Bedienbarkeit □ Menge an Funktionalitäten
□ Qualität □ Sonstiges: _____

Vielen Dank für Ihre Teilnahme!

```java
//Class LaunchActivity

package at.haraldbernhard.joggingcoach;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class LaunchActivity extends AppCompatActivity {
    TextView greeting, createUser, personalGreeting, information1, information2;
    Button start ;
    DbHelper dbHelper;
    String username, lastTraining;
    //Shared Preferences
    public static SharedPreferences sharedPref;
    public static final String PREFERENCE_FILE_NAME ="UserPreference";
    public static final String PREFERENCE_KEY_USERNAME="username";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_launch);
        sharedPref = getSharedPreferences(PREFERENCE_FILE_NAME, Context.MODE_PRIVATE);
        username = sharedPref.getString(PREFERENCE_KEY_USERNAME, "");
        greeting = (TextView) findViewById(R.id.textViewGreeting) ;
        createUser = (TextView) findViewById(R.id.textViewCreateNewUser);
        start = (Button) findViewById(R.id.buttonStartTraining);
        personalGreeting = (TextView) findViewById(R.id.textViewPersonalGreeting);
        information1 = (TextView) findViewById(R.id.textViewInformation1);
        information2 = (TextView) findViewById(R.id.textViewInformation2);
        dbHelper = new DbHelper(this);
        lastTraining = dbHelper.getLastTrainingDate();
        if(!username.isEmpty()) {
            if(lastTraining !="") {
                personalGreeting.setText("Hello " + username + " \n \n \n Welcome back, \n your last training was on \n" + lastTraining);
            }
            else{
                personalGreeting.setText("Hello " + username + " \n \n Welcome to JoggingCoach");
            }
            greeting.setVisibility(View.INVISIBLE);
            createUser.setVisibility(View.INVISIBLE);
            information1.setVisibility(View.INVISIBLE);
            information2.setVisibility(View.INVISIBLE);
        }
        else{
            start.setVisibility(View.INVISIBLE);
            personalGreeting.setVisibility(View.INVISIBLE);
        }
    }
    public void startCreateUserActivity(View view){
        Intent intent = new Intent(this, CreateUserActivity.class);
        startActivity(intent);
    }
    public void startBasicActivity(View view){
        Intent intent = new Intent(this, MenuActivity.class);
        startActivity(intent);
        finish();
    }
}
```

```java
//Class CreateUserActivity

package at.haraldbernhard.joggingcoach;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class CreateUserActivity extends AppCompatActivity implements TextWatcher{
    DbHelper dbHelper;
    EditText etUsername, etAge, etSize, etWeight;
    Button createAvatar;
    String username;
    int age, size, weight;
    //Shared Preferences
    public static SharedPreferences sharedPref;
    public static final String PREFERENCE_FILE_NAME ="UserPreference";
    public static final String PREFERENCE_KEY_USERNAME="username";
    public static final String PREFERENCE_KEY_WEIGHT="weight";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_user);
        setTitle("Create new User");
        //Toolbar
        Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        sharedPref = getSharedPreferences(PREFERENCE_FILE_NAME, Context.MODE_PRIVATE);
        createAvatar = (Button) findViewById(R.id.buttonCreateAvatar);
        createAvatar.setEnabled(false);
        etUsername = (EditText) findViewById(R.id.editTextUsername);
        etAge = (EditText) findViewById(R.id.editTextAge);
        etSize = (EditText) findViewById(R.id.editTextSize);
        etWeight = (EditText) findViewById(R.id.editTextWeight);
        etUsername.addTextChangedListener(this);
        etAge.addTextChangedListener(this);
        etSize.addTextChangedListener(this);
        etWeight.addTextChangedListener(this);
    }

    public void createNewAvatar(View view){
        User user = new User (username, age, size, weight);
        dbHelper = new DbHelper(this);
        dbHelper.insertUser(user);
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putString(PREFERENCE_KEY_USERNAME, etUsername.getText().toString());
        editor.putInt(PREFERENCE_KEY_WEIGHT, Integer.parseInt(etWeight.getText().toString()));
        editor.commit();
        Toast.makeText(this, "User '" + username + "' successfully created", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(this, LaunchActivity.class);
        startActivity(intent);
    }

    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
```

```
67      }
68
69      @Override
70      public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
71      }
72
73      @Override
74      public void afterTextChanged(Editable editable) {
75          username = etUsername.getText().toString();
76          age = Integer.parseInt(etAge.getText().toString()+0)/10;
77          size = Integer.parseInt(etSize.getText().toString()+0)/10;
78          weight = Integer.parseInt(etWeight.getText().toString()+0)/10;
79          if(!username.isEmpty() && age>0 && age<=100 && size>0 &size<250 && weight>0 && weight<200 ){
80              createAvatar.setEnabled(true);
81          }
82          else{
83              createAvatar.setEnabled(false);
84          }
85      }
86  }
87
```

```java
//Class MenuActivity

package at.haraldbernhard.joggingcoach;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MenuActivity extends AppCompatActivity {
    Button startNewTraining, diary, startNewIntervalTraining;
    TextView greeting;
    SharedPreferences sharedPref;
    DbHelper dbHelper;
    public static final String preferenceFileName ="UserPreference";
    public static final String preferenceKeyId="id";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);
        sharedPref = getSharedPreferences(preferenceFileName, Context.MODE_PRIVATE);
        dbHelper = new DbHelper(this);
        //Toolbar
        Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        greeting = (TextView) findViewById(R.id.textViewGreeting);
        int id = sharedPref.getInt(preferenceKeyId, 0);
        startNewTraining = (Button) findViewById(R.id.buttonStartNewTraining);
        startNewIntervalTraining = (Button)findViewById(R.id.buttonStartIntervalTraining);
        diary = (Button) findViewById(R.id.buttonDiary);
    }

    public void startTrainingSession(View view){
        Intent intent = new Intent(this, TrainingModeActivity.class);
        startActivity(intent);
    }

    public void startDiary(View view){
        Intent intent = new Intent(this, TrainingDiaryActivity.class);
        startActivity(intent);
    }

    public void startIntervalTraining(View view){
        Intent intent = new Intent(this, IntervalModeActivity.class);
        startActivity(intent);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
67          switch(item.getItemId()){
68              case R.id.action_exit:
69                  finish();
70                  System.exit(0);
71          }
72          return super.onOptionsItemSelected(item);
73      }
74  }
75
```

```java
//Class TrainingModeActivity

package at.haraldbernhard.joggingcoach;

import android.Manifest;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.util.Calendar;
import java.util.GregorianCalendar;

public class TrainingModeActivity extends AppCompatActivity {
    //Labels
    TextView tvLabelTime, tvLabelDistance, tvLabelCalorie, tvLabelSpeed;
    //TextViews for display
    TextView tvTime, tvDistance, tvCalorie, tvSpeedAverage, tvSpeedImmidiate;
    //Buttons
    Button startTraining, stopTraining, pauseTraining;
    //Strings for display information
    static String time = "0:0:0", distance = "0,0", calorie = "0,0", speedA = "0,0", speedI = "0,0 ";
    //is start button pressed?
    boolean isActive = false;
    Stopwatch stopwatch;
    //Database
    DbHelper dbHelper;
    //AsyncTask
    StopwatchTask stopwatchTask;
    UpdateUITask updateUITask;
    public static final int LOCATION_REQUEST = 1337;
    public static SharedPreferences sharedPref;
    public static final String PREFERENCE_FILE_NAME = "UserPreference";
    public static final String PREFERENCE_KEY_WEIGHT = "weight";
    private static int tmHour, tmMin, tmSec;
    private static double tmDistance, tmSpeedA, tmSpeedI, tmCalorie;
    private static int tmKilo = 75;  //TODO SHARED PREF

    public static double getTmSpeedI() {
        return tmSpeedI;
    }

    public static void setTmSpeedI(double tmSpeedI) {
        TrainingModeActivity.tmSpeedI = tmSpeedI;
    }

    public static double getTmSpeedA() {
        return tmSpeedA;
    }

    public static void setTmSpeedA(double tmSpeedA) {
```

```
67          TrainingModeActivity.tmSpeedA = tmSpeedA;
68      }
69
70      public static double getTmDistance() {
71          return tmDistance;
72      }
73
74      public static void setTmDistance(double tmDistance) {
75          TrainingModeActivity.tmDistance = tmDistance;
76      }
77
78      public static double getTmCalorie() {
79          return tmCalorie;
80      }
81
82      public static void setTmCalorie(double tmCalorie) {
83          TrainingModeActivity.tmCalorie = tmCalorie;
84      }
85
86      public static int getTmKilo() {
87          return tmKilo;
88      }
89
90      public static int getTmSec() {
91          return tmSec;
92      }
93
94      public static void setTmSec(int tmSec) {
95          TrainingModeActivity.tmSec = tmSec;
96      }
97
98      public static int getTmMin() {
99          return tmMin;
100     }
101
102     public static void setTmMin(int tmMin) {
103         TrainingModeActivity.tmMin = tmMin;
104     }
105
106     public static int getTmHour() {
107         return tmHour;
108     }
109
110     public static void setTmHour(int tmHour) {
111         TrainingModeActivity.tmHour = tmHour;
112     }
113
114     @Override
115     protected void onCreate(Bundle savedInstanceState) {
116         super.onCreate(savedInstanceState);
117         setContentView(R.layout.activity_training_mode);
118         //Toolbar
119         Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);
120         setSupportActionBar(toolbar);
121         getSupportActionBar().setDisplayHomeAsUpEnabled(true);
122         //Labels
123         tvLabelTime = (TextView) findViewById(R.id.textViewLabelTime);
124         tvLabelDistance = (TextView) findViewById(R.id.textViewLabelDistance);
125         tvLabelCalorie = (TextView) findViewById(R.id.textViewLabelCalorie);
126         tvLabelSpeed = (TextView) findViewById(R.id.textViewLabelSpeed);
127         //TextViews for display
128         tvTime = (TextView) findViewById(R.id.textViewTime);
129         tvDistance = (TextView) findViewById(R.id.textViewDistance);
130         tvCalorie = (TextView) findViewById(R.id.textViewCalorie);
131         tvSpeedImmidiate = (TextView) findViewById(R.id.textViewSpeedI);
132         tvSpeedAverage = (TextView) findViewById(R.id.textViewSpeedA);
```

```
133         //Starting values
134         tvTime.setText(time);
135         tvDistance.setText(distance);
136         tvCalorie.setText(calorie);
137         tvSpeedImmidiate.setText(speedI);
138         tvSpeedAverage.setText(speedA);
139         sharedPref = getSharedPreferences(PREFERENCE_FILE_NAME, Context.MODE_PRIVATE);
140         tmKilo = sharedPref.getInt(PREFERENCE_KEY_WEIGHT, 0);
141         Log.e("log", tmKilo+"");
142         //Buttons
143         startTraining = (Button) findViewById(R.id.buttonStart);
144         stopTraining = (Button) findViewById(R.id.buttonStop);
145         pauseTraining = (Button) findViewById(R.id.buttonPause);
146         pauseTraining.setVisibility(View.INVISIBLE);
147         stopTraining.setEnabled(false);
148         gpsPermission();
149     }
150
151     public void gpsPermission() {
152         // Check if the GPS permission is already available
153         if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED) {
154             Toast.makeText(this, "Initialize Location Manager", Toast.LENGTH_LONG).show();
155             LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
156             LocationListener ll = new AndroidLocationListener();
157             lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 50, ll);
158         }
159         // Permission is not available
160         else {
161             // Should we show an explanation?
162             if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.
    ACCESS_FINE_LOCATION)) {
163                 // Show an expanation to the user *asynchronously*
164             } else {
165                 // No explanation needed, we can request the permission.
166                 ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.
    ACCESS_FINE_LOCATION}, LOCATION_REQUEST);
167             }
168         }
169     }
170
171     @Override
172     public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
173         switch (requestCode) {
174             case LOCATION_REQUEST: {
175                 // If request is cancelled, the result arrays are empty.
176                 if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
177                     // permission was granted, yay!
178                 } else {
179                     // permission denied, boo! Disable the
180                     // functionality that depends on this permission.
181                     Intent intent = new Intent(this, MenuActivity.class);
182                     startActivity(intent);
183                 }
184                 return;
185             }
186             // other 'case' lines to check for other
187             // permissions this app might request
188         }
189     }
190
191     public void createAlertDialog() {
192         AlertDialog.Builder builder = new AlertDialog.Builder(this);
193         builder.setTitle(R.string.title_ResultDiary);
194         builder.setMessage(R.string.message_ResultDiary);
195         builder.setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
```

```
196             public void onClick(DialogInterface dialog, int id) {
197                 // User clicked YES button
198                 //Date
199                 Calendar calendar = new GregorianCalendar();
200                 String date = calendar.get(Calendar.DAY_OF_MONTH) + ". " + calendar.get(Calendar.MONTH)
    + ". " + calendar.get(Calendar.YEAR);
201                 time = tvTime.getText()+"";
202                 distance = tvDistance.getText()+"";
203                 calorie = tvCalorie.getText()+"";
204                 speedA = tvSpeedAverage.getText()+"";
205                 //Insert into Database
206                 dbHelper = new DbHelper(TrainingModeActivity.this);
207                 dbHelper.insertTraining(date, time, distance, calorie, speedA);
208                 finish();
209                 System.exit(0);
210                 Intent intent = new Intent(TrainingModeActivity.this, MenuActivity.class);
211                 startActivity(intent);
212                 Toast.makeText(TrainingModeActivity.this, R.string.save_result, Toast.LENGTH_LONG).show(
    );
213             }
214         });
215         builder.setNegativeButton(R.string.no, new DialogInterface.OnClickListener() {
216             public void onClick(DialogInterface dialog, int id) {
217                 // User clicked NO button
218                 finish();
219                 System.exit(0);
220                 Intent intent = new Intent(TrainingModeActivity.this, MenuActivity.class);
221                 startActivity(intent);
222             }
223         });
224         AlertDialog dialog = builder.create();
225         dialog.show();
226     }
227
228     //AsyncTask for Stopwatch
229     class StopwatchTask extends AsyncTask<Void, String, Void> {
230
231         @Override
232         protected void onPreExecute() {
233             super.onPreExecute();
234         }
235
236         @Override
237         protected Void doInBackground(Void... params) {
238             stopwatch = new Stopwatch();
239             stopwatch.startStopwatch();
240             return null;
241         }
242
243         @Override
244         protected void onPostExecute(Void aVoid) {
245         }
246
247         @Override
248         protected void onProgressUpdate(String... values) {
249         }
250     }
251
252     //AsyncTask For Update User Interface
253     class UpdateUITask extends AsyncTask<Void, Void, Void> {
254         @Override
255         protected void onPreExecute() {
256         }
257
258         @Override
259         protected Void doInBackground(Void... voids) {
```

```
260
261             while (isActive) {
262                 try {
263                     TrainingModeActivity.setTmHour(stopwatch.getHour());
264                     TrainingModeActivity.setTmMin(stopwatch.getMin());
265                     TrainingModeActivity.setTmSec(stopwatch.getSec());
266                     TrainingModeActivity.setTmSpeedA(TrainingModeActivity.getTmDistance()/(((double)
    TrainingModeActivity.getTmHour())+((double)TrainingModeActivity.getTmMin()/60)+((double)
    TrainingModeActivity.getTmSec()/3600))) ;
267                     TrainingModeActivity.setTmCalorie(TrainingModeActivity.getTmKilo()*
    TrainingModeActivity.getTmDistance());
268                     Thread.sleep(1000);
269                     publishProgress();
270                 } catch (InterruptedException e) {
271                     e.printStackTrace();
272                 }
273             }
274             return null;
275         }
276
277         @Override
278         protected void onProgressUpdate(Void... values) {
279             Log.e("LOG", tmHour + ":" + tmMin +":" + tmSec+ " | " + tmDistance + " | " + tmCalorie + " | "
     " + tmSpeedA + " | " + tmSpeedI + " ");
280             Log.e("Kilo", TrainingModeActivity.getTmKilo()+"");
281             tvTime.setText(TrainingModeActivity.getTmHour() + " : " + TrainingModeActivity.getTmMin() + "
     : " + TrainingModeActivity.getTmSec() );
282             tvDistance.setText(String.format("%.3f", TrainingModeActivity.getTmDistance()));
283             tvSpeedAverage.setText( String.format("%.2f", TrainingModeActivity.getTmSpeedA()));
284             tvSpeedImmidiate.setText(String.format("%.2f", TrainingModeActivity.getTmSpeedI()));
285             tvCalorie.setText(String.format("%.1f", TrainingModeActivity.getTmCalorie()));
286         }
287
288         @Override
289         protected void onPostExecute(Void aVoid) {
290         }
291     }
292
293     public void buttonStartTraining(View view) {
294         startTraining.setEnabled(false);
295         pauseTraining.setEnabled(true);
296         stopTraining.setEnabled(true);
297         isActive = true;
298         stopwatchTask = (StopwatchTask) new StopwatchTask().executeOnExecutor(AsyncTask.
    THREAD_POOL_EXECUTOR);
299         updateUITask = (UpdateUITask) new UpdateUITask().executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR
    );
300     }
301
302     public void buttonStopTraining(View view) {
303         createAlertDialog();
304     }
305
306     public void buttonPauseTraining(View view) {
307         onPause();
308     }
309
310     @Override
311     protected void onPause() {
312         super.onPause();
313     }
314
315     public static void setTime(String stopwatchTime) {
316         time = stopwatchTime;
317     }
318
```

```
319     public static void setDistance(String locationDistance) {
320         distance = locationDistance;
321     }
322
323     public static void setSpeedA(String locationSpeed) {
324         speedA = locationSpeed;
325     }
326
327     public static void setSpeedI(String locationSpeed) {
328         speedI = locationSpeed;
329     }
330
331     public static String getInformationAsJSON() {
332         return "{\"time\":\"" + time + " \", \"distance\":\"" + distance + "\", \"speed\": \"" + speedA
    + "\"}";
333     }
334
335     public class AndroidLocationListener implements LocationListener {
336         double speedI, distance;
337
338         @Override
339         public void onLocationChanged(Location location) {
340
341             if (location != null && isActive) {
342                 if (location.getLongitude() != 0 && location.getLatitude() != 0) {
343                     distance += 0.050;
344                 }
345                 speedI = location.getSpeed();
346                 TrainingModeActivity.setTmDistance(distance);
347                 TrainingModeActivity.setTmSpeedI(speedI);
348             }
349         }
350
351         @Override
352         public void onStatusChanged(String provider, int status, Bundle extras) {
353         }
354
355         @Override
356         public void onProviderEnabled(String provider) {
357         }
358
359         @Override
360         public void onProviderDisabled(String provider) {
361         }
362     }
363 }
364
```

```java
//Class IntervalModeActivity

package at.haraldbernhard.joggingcoach;

import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class IntervalModeActivity extends AppCompatActivity {
    Button btCancel, btStart;
    TextView workout, recreation, tvWorkout, tvRecreation;
    public static boolean showDialog;
    boolean workoutIsActive, recreationIsActive = true;
    Stopwatch stopwatch;
    StopwatchTask stopwatchTask;
    UpdateWorkoutUITask updateWorkoutUITask;
    UpdateRecreationUITask updateRecreationUITask;
    //Input from user
    private static int userInterval, userWork, userRec;
    //Control variable
    private static int imInterval, imWork, imRec;

    public static int getUserInterval() {
        return userInterval;
    }

    public static void setUserInterval(int userInterval) {
        IntervalModeActivity.userInterval = userInterval;
    }

    public static int getUserWork() {
        return userWork;
    }

    public static void setUserWork(int userWork) {
        IntervalModeActivity.userWork = userWork;
    }

    public static int getUserRec() {
        return userRec;
    }

    public static void setUserRec(int userRec) {
        IntervalModeActivity.userRec = userRec;
    }

    public static int getImInterval() {
        return imInterval;
    }

    public static void setImInterval(int imInterval) {
        IntervalModeActivity.imInterval = imInterval;
    }

    public static int getImWork() {
        return imWork;
    }

    public static void setImWork(int imWork) {
        IntervalModeActivity.imWork = imWork;
```

```
67      }
68
69      public static int getImRec() {
70          return imRec;
71      }
72
73      public static void setImRec(int imRec) {
74          IntervalModeActivity.imRec = imRec;
75      }
76
77      @Override
78      protected void onCreate(Bundle savedInstanceState) {
79          super.onCreate(savedInstanceState);
80          setContentView(R.layout.activity_interval_mode);
81          btCancel = (Button) findViewById(R.id.buttonIntervalCancel);
82          btStart = (Button) findViewById(R.id.buttonIntervalStart);
83          workout = (TextView) findViewById(R.id.textViewWorkout);
84          recreation = (TextView) findViewById(R.id.textViewRecreation);
85          tvWorkout = (TextView) findViewById(R.id.tvWorkout);
86          tvRecreation = (TextView) findViewById(R.id.tvRecreation);
87          btStart.setEnabled(false);
88          IntervalModeActivity.setImInterval(IntervalModeActivity.getUserInterval()*2);
89          if(showDialog){
90              btStart.setEnabled(true);
91          }
92          userWork = IntervalDialogActivity.getSecWorkout();
93          userRec = IntervalDialogActivity.getSecRecreation();
94          userInterval = IntervalDialogActivity.getIntervals();
95
96          //Toolbar
97          Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);
98          setSupportActionBar(toolbar);
99          getSupportActionBar().setDisplayHomeAsUpEnabled(true);
100
101         //showIntervalDialog();
102         if(!showDialog) {
103             showDialog = true;
104             Intent intent = new Intent(this, IntervalDialogActivity.class);
105             startActivity(intent);
106         }
107     }
108
109     //Button Cancel
110     public void cancelIntervalTraining (View view){
111         showDialog= false;
112         finish();
113         Intent intent = new Intent(this, MenuActivity.class);
114         startActivity(intent);
115     }
116
117     //Button Start
118     public void startIntervalTraining (View view){
119         startTraining();
120     }
121
122     public void startTraining(){
123         recreationIsActive = true;
124         startTimerWorkout();
125     }
126
127     public void startTimerWorkout(){
128         btStart.setEnabled(false);
129         if(IntervalModeActivity.getImInterval() ==0) {
130             stopwatchTask = (StopwatchTask) new StopwatchTask().executeOnExecutor(AsyncTask.
    THREAD_POOL_EXECUTOR);
131         }
```

```
132         updateWorkoutUITask = (UpdateWorkoutUITask) new UpdateWorkoutUITask().executeOnExecutor(AsyncTask
    .THREAD_POOL_EXECUTOR);
133
134     }
135     public void startTimerRecreation(){
136         updateRecreationUITask = (UpdateRecreationUITask) new UpdateRecreationUITask().executeOnExecutor(
    AsyncTask.THREAD_POOL_EXECUTOR);
137     }
138
139     //AsyncTask for Stopwatch
140     class StopwatchTask extends AsyncTask {
141
142         @Override
143         protected void onPreExecute() {
144             super.onPreExecute();
145         }
146
147         @Override
148         protected Object doInBackground(Object[] objects) {
149             stopwatch = new Stopwatch();
150             stopwatch.startStopwatch();
151             return null;
152         }
153
154         @Override
155         protected void onProgressUpdate(Object[] values) {
156             super.onProgressUpdate(values);
157         }
158
159         @Override
160         protected void onPostExecute(Object o) {
161             super.onPostExecute(o);
162         }
163     }
164
165     //AsyncTask For Update Worker User Interface
166     class UpdateWorkoutUITask extends AsyncTask {
167
168         @Override
169         protected void onPreExecute() {
170             super.onPreExecute();
171         }
172
173         @Override
174         protected Object doInBackground(Object[] objects) {
175                 if (IntervalModeActivity.getImInterval() < IntervalModeActivity.getUserInterval()  ) {
176                     while (IntervalModeActivity.getUserWork() <= IntervalModeActivity.getImWork()){
177                         try {
178                             Thread.sleep(1000);
179                             IntervalModeActivity.setImWork(stopwatch.getSec());
180                             publishProgress();
181                         } catch (InterruptedException e) {
182                             e.printStackTrace();
183                         }
184                     }
185                     Log.i("INTERVAL", IntervalModeActivity.getImInterval() + "");
186                     IntervalModeActivity.setImInterval(IntervalModeActivity.getImInterval() + 1);
187                     workoutIsActive = false;
188                     recreationIsActive = true;
189                     IntervalModeActivity.setImWork(0);
190                     stopwatch.setSec(0);
191                     publishProgress();
192                     startTimerRecreation();
193                 }
194             else{
195                     Log.i("FINISH", "Finish");
```

```
196                     finish();
197                 }
198             return null;
199         }
200
201         @Override
202         protected void onProgressUpdate(Object[] values) {
203             super.onProgressUpdate(values);
204             tvWorkout.setText(IntervalModeActivity.getImWork() + "");
205             Log.e("WORKOUT", IntervalModeActivity.getImWork() + "/" + IntervalModeActivity.getUserWork() ↵
    );
206         }
207
208         @Override
209         protected void onPostExecute(Object o) {
210             super.onPostExecute(o);
211         }
212     }
213
214     //AsyncTask For Update Recreation User Interface
215     class UpdateRecreationUITask extends AsyncTask<Void, Void, Void> {
216         @Override
217         protected void onPreExecute() {
218         }
219
220         @Override
221         protected Void doInBackground(Void... voids) {
222             if(IntervalModeActivity.getUserInterval() != IntervalModeActivity.getImInterval()) {
223                 do {
224                     try {
225                         Thread.sleep(1000);
226                         IntervalModeActivity.setImRec(stopwatch.getSec());
227                         publishProgress();
228                     } catch (InterruptedException e) {
229                         e.printStackTrace();
230                     }
231                 }
232                 while (IntervalModeActivity.getUserRec() != IntervalModeActivity.getImRec());
233             }
234                 Log.e("RECREATION", "Initialize");
235                 IntervalModeActivity.setImInterval(IntervalModeActivity.getImInterval() + 1);
236                 workoutIsActive = true;
237                 recreationIsActive = false;
238                 IntervalModeActivity.setImRec(0);
239                 stopwatch.setSec(0);
240                 publishProgress();
241                 startTimerWorkout();
242             return null;
243         }
244
245         @Override
246         protected void onProgressUpdate(Void... values) {
247             super.onProgressUpdate(values);
248             tvRecreation.setText(IntervalModeActivity.getImRec() + "");
249             Log.e("RECREATION", IntervalModeActivity.getImRec() + "/" + IntervalModeActivity.getUserRec()↵
    );
250         }
251
252         @Override
253         protected void onPostExecute(Void aVoid) {
254         }
255     }
256 }
257
258
```

```java
//Class IntervalDialogActivity

package at.haraldbernhard.joggingcoach;

import android.app.Activity;
import android.content.Intent;
import android.support.v4.app.DialogFragment;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.NumberPicker;
import org.w3c.dom.Text;

public class IntervalDialogActivity extends Activity{
    private static int  secWorkout, secRecreation, intervals;
    NumberPicker npWorkSec, npRecSec, npInterval;
    Button btStart, btCancel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_interval_dialog);
        npWorkSec = (NumberPicker) findViewById(R.id.nPWorkoutSec);
        npRecSec = (NumberPicker) findViewById(R.id.npRecreationSec);
        npInterval = (NumberPicker) findViewById(R.id.npIntervals);
        btStart = (Button) findViewById(R.id.buttonIntervalDialogStart);
        btCancel = (Button)findViewById(R.id.buttonIntervalDialogCancel);
        npWorkSec.setMinValue(0);
        npWorkSec.setMaxValue(300);
        npRecSec.setMinValue(0);
        npRecSec.setMaxValue(300);
        npInterval.setMinValue(0);
        npInterval.setMaxValue(10);
    }

    public void onStart(View view){
        //Store Input
        secWorkout = npWorkSec.getValue();
        secRecreation = npRecSec.getValue();
        intervals = npInterval.getValue();
        //Exit Dialog
        IntervalModeActivity.showDialog=true;
        Intent intent = new Intent (this, IntervalModeActivity.class);
        startActivity(intent);
    }

    public void onCancel(View view){
        Intent intent = new Intent(this, MenuActivity.class);
        startActivity(intent);
    }

    public static int getSecWorkout() {return secWorkout;}
    public static int getSecRecreation() {
        return secRecreation;
    }

    public static int getIntervals() {
        return intervals;
    }
}

```

```
1  //Class TrainingDiaryActivity
2
3  package at.haraldbernhard.joggingcoach;
4
5  import android.app.ListActivity;
6  import android.database.Cursor;
7  import android.os.AsyncTask;
8  import android.os.Bundle;
9  import android.support.v7.widget.Toolbar;
10
11 public class TrainingDiaryActivity extends ListActivity {
12     TrainingDiaryAdapter tdAdapter;
13     DbHelper dbHelper;
14     Cursor cursor;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         dbHelper = new DbHelper(this);
20         cursor = dbHelper.getAllTraining();
21         tdAdapter = new TrainingDiaryAdapter(this, cursor);
22         setListAdapter(tdAdapter);
23     }
24 }
25
```

```java
//Class TrainingDiaryAdapter

package at.haraldbernhard.joggingcoach;

import android.content.Context;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;

public class TrainingDiaryAdapter extends CursorAdapter{
    private LayoutInflater inflator;
    int ciDate, ciTime, ciDistance, ciCalorie, ciSpeedA;

    TrainingDiaryAdapter (Context context, Cursor cursor){
        super(context, cursor);
        inflator = LayoutInflater.from(context);
        ciDate = cursor.getColumnIndex(DbHelper.DATE);
        ciTime = cursor.getColumnIndex(DbHelper.TIME);
        ciDistance = cursor.getColumnIndex(DbHelper.DISTANCE);
        ciCalorie = cursor.getColumnIndex(DbHelper.CALORIE);
        ciSpeedA = cursor.getColumnIndex(DbHelper.SPEEDA);
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup viewGroup){
        return inflator.inflate(R.layout.training_result, null);
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        TextView tvDate = (TextView) view.findViewById(R.id.textViewDateAdapter);
        tvDate.setText("Date: " + cursor.getString(ciDate));
        TextView tvTime = (TextView) view.findViewById(R.id.textViewTimeAdapter);
        tvTime.setText("Time: " + cursor.getString(ciTime));
        TextView tvDistance = (TextView) view.findViewById(R.id.textViewDistanceAdapter);
        tvDistance.setText("Distance: " + cursor.getString(ciDistance));
        TextView tvCalorie = (TextView) view.findViewById(R.id.textViewCalorieAdapter);
        tvCalorie.setText("Calorie: " + cursor.getString(ciCalorie));
        TextView tvSpeedA = (TextView) view.findViewById(R.id.textViewSpeedAAdapter);
        tvSpeedA.setText("Speed: " + cursor.getString(ciSpeedA));
    }
}

```

```java
//Class DbHelper

package at.haraldbernhard.joggingcoach;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import java.util.ArrayList;
import java.util.List;

public class DbHelper extends SQLiteOpenHelper {
    private static final String TAG = DbHelper.class.getSimpleName();
    // Name and Version of Database
    private static final String DATABASE_NAME = "joggingcoach.db";
    private static final int DATABASE_VERSION = 1;

    //USER
    //Name and Attribute of Table "user"
    public static final String _ID = "_id";
    public static final String TABLE_NAME_USER = "user";
    public static final String USERNAME = "username";
    public static final String AGE = "age";
    public static final String SIZE = "size";
    public static final String WEIGHT = "weight";
    //Create Table
    private static final String TABLE_USER_CREATE =
            "CREATE TABLE "
                    + TABLE_NAME_USER + " (" + _ID
                    + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                    + USERNAME + " TEXT, "
                    + AGE + " INTEGER, "
                    + SIZE + " INTEGER, "
                    + WEIGHT + " INTEGER);";
    //Delete Table
    public static final String TABLE_USER_DROP = "DROP TABLE IF EXISTS " + TABLE_NAME_USER;

    //TRAINING
    //NAME and Attribute of Table "training"
    public static final String TABLE_NAME_TRAINING = "training";
    public static final String DATE = "date";
    public static final String TIME = "time";
    public static final String DISTANCE = "distance";
    public static final String CALORIE = "calorie";
    public static final String SPEEDA = "speeda";
    //SQL-Statement Create Table Training
    private static final String TABLE_TRAINING_CREATE =
            "CREATE TABLE "
                    + TABLE_NAME_TRAINING + " (" + _ID
                    + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                    + DATE + " TEXT, "
                    + TIME + " TEXT, "
                    + DISTANCE + " TEXT, "
                    + CALORIE + " TEXT, "
                    + SPEEDA + " TEXT);";

    public void onCreate(SQLiteDatabase db) {
        db.execSQL(TABLE_USER_CREATE);
        db.execSQL(TABLE_TRAINING_CREATE);
    }
    //Delete Table
    public static final String TABLE_TRAINING_DROP = "DROP TABLE IF EXISTS " + TABLE_NAME_TRAINING;

```

```
67      public DbHelper (Context context){
68          super(context, DATABASE_NAME, null, DATABASE_VERSION);
69      }
70
71      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
72          // This database is only a cache for online data, so its upgrade policy is
73          // to simply to discard the data and start over
74          db.execSQL(TABLE_USER_DROP);
75          db.execSQL(TABLE_TRAINING_DROP);
76          onCreate(db);
77      }
78      public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
79          onUpgrade(db, oldVersion, newVersion);
80      }
81      //CRUD for USER
82      public int getUserId(String name){
83          //Open Connection
84          SQLiteDatabase db = getReadableDatabase();
85          String[] column = {_ID};
86          Cursor cursor = db.query(TABLE_NAME_USER, column, USERNAME+" = '"+name+"'", null, null, null,
    null);
87          cursor.moveToFirst();
88          if(cursor !=null && cursor.getCount()>0){
89              return cursor.getInt(cursor.getColumnIndexOrThrow(_ID));
90          }
91          else {
92              return 0;
93          }
94      }
95
96      public int getUserWeight(String weight){
97          //Open Connection
98          SQLiteDatabase db = getReadableDatabase();
99          String[] column = {WEIGHT};
100         Cursor cursor = db.query(TABLE_NAME_USER, column, WEIGHT+" = '"+weight+"'", null, null, null,
    null);
101         cursor.moveToFirst();
102         if(cursor.getColumnIndexOrThrow(WEIGHT)>0){
103             return cursor.getColumnIndexOrThrow(_ID);
104         }
105         else {
106             return 0;
107         }
108     }
109
110     public void insertUser(User user){
111         long rowId = -1;
112         try{
113             // Open Connection
114             SQLiteDatabase db = getWritableDatabase();
115             //Save Values
116             ContentValues values = new ContentValues();
117             values.put(USERNAME, user.getUsername());
118             values.put(AGE, user.getAge());
119             values.put(SIZE, user.getSize());
120             values.put(WEIGHT, user.getWeight());
121             //Insert into Table
122             rowId = db.insert(TABLE_NAME_USER, null, values);
123         }
124         catch (SQLiteException e){
125             Log.e(TAG, "insert() ", e);
126         }
127         finally {
128             Log.d(TAG, "insert(): rowId= " + rowId);
129         }
130     }
```

```
131
132     //CRUD for TRAINING
133     public void insertTraining(String date, String time, String distance, String calorie, String speedA){
134         long rowId = -1;
135         try {
136             //Open Connection
137             SQLiteDatabase db = getWritableDatabase();
138             //Save Values
139             ContentValues values = new ContentValues();
140             values.put(DATE, date);
141             values.put(TIME, time);
142             values.put(DISTANCE, distance);
143             values.put(CALORIE, calorie);
144             values.put(SPEEDA, speedA);
145             //Insert Into Table
146             rowId = db.insert(TABLE_NAME_TRAINING, null, values);
147         } catch (SQLiteException e) {
148             Log.e(TAG, "insert() ", e);
149         } finally {
150             Log.d(TAG, "insert(): rowId= " + rowId);
151         }
152     }
153
154     public String getLastTrainingDate(){
155         String result;
156         //Open Connection
157         SQLiteDatabase db = getReadableDatabase();
158         //Build SQL-Statement
159         String[] column = {DATE};
160         Cursor cursor = db.query    (TABLE_NAME_TRAINING,   // The table to query
161                 column,                 // The columns to return
162                 null,                   // The columns for the WHERE clause
163                 null,                   // The values for the WHERE clause
164                 null,                   // don't group the rows
165                 null,                   // don't filter by row groups
166                 DATE+" DESC"            // The sort order
167         );
168         cursor.moveToFirst();
169         if(cursor != null && cursor.getCount()>0) {
170             result =  cursor.getString(cursor.getColumnIndexOrThrow(DATE));
171         }
172         else {
173             result = "";
174         }
175         return result;
176     }
177
178     public Cursor getAllTraining(){
179         SQLiteDatabase db = getReadableDatabase();
180         String[] column = {_ID, DATE, TIME, DISTANCE, CALORIE, SPEEDA};
181         Cursor cursor = db.query(TABLE_NAME_TRAINING, column, null, null, null, null, DATE+" DESC");
182         return cursor;
183     }
184 }
185
```

```java
//Class SAPServiceProvider

package at.haraldbernhard.joggingcoach;

import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.util.Log;
import android.widget.Toast;
import com.samsung.android.sdk.SsdkUnsupportedException;
import com.samsung.android.sdk.accessory.SA;
import com.samsung.android.sdk.accessory.SAAgent;
import com.samsung.android.sdk.accessory.SAPeerAgent;
import com.samsung.android.sdk.accessory.SASocket;
import java.io.IOException;
import java.util.HashMap;

public class SAPServiceProvider extends SAAgent{
    public final static String TAG = "SAPServiceProvider";
    public final static int SAP_SERVICE_CHANNEL_ID = 321;
    private final IBinder mIBinder = new LocalBinder();
    HashMap<Integer, SAPServiceProviderConnection> connectionMap = null;

    public SAPServiceProvider() {
        super(TAG, SAPServiceProviderConnection.class);
    }

    @Override
    protected void onFindPeerAgentResponse(SAPeerAgent peerAgent, int result) {
        switch(result){
            case PEER_AGENT_FOUND:
                //Peer Agent is found
                requestServiceConnection(peerAgent);
                Log.e(TAG,  "PeerAgent is found");
                break;
            case FINDPEER_DEVICE_NOT_CONNECTED:
                //Peer Agent are not found, no accessory device connected
                Log.e(TAG,  "PeerAgent not found");
                break;
            case FINDPEER_SERVICE_NOT_FOUND:
                //No matching service on connected accessory
                Log.e(TAG,  "No matching");
                break;
            default:
                Log.e(TAG,  "Default");
        }
    }

    @Override
    protected void onServiceConnectionResponse(SAPeerAgent peerAgent, SASocket thisConnection, int result
) {
        switch(result){
            case CONNECTION_SUCCESS:
                if(thisConnection != null){
                    SAPServiceProviderConnection myConnection = (SAPServiceProviderConnection)
thisConnection;
                    if(connectionMap == null){
                        connectionMap = new HashMap<Integer, SAPServiceProviderConnection>();
                    }
                    myConnection.connectionID = (int) (System.currentTimeMillis() & 255);
                    Log.d(TAG,  "onServiceConnection connectionID = " + myConnection.connectionID);
                    connectionMap.put(myConnection.connectionID, myConnection);
                    Toast.makeText(getBaseContext(),  "Connection Established",  Toast.LENGTH_LONG).show(
);
                }
                else{
```

```
64                      Log.e(TAG,  "SASocket object is null");
65                  }
66                  break;
67              case CONNECTION_FAILURE_NETWORK:
68                  Log.e(TAG,  "connection already exists");
69                  break;
70              default:
71                  Log.e(TAG,  "default");
72          }
73      }
74
75      @Override
76      public IBinder onBind(Intent intent) {
77          return mIBinder;
78      }
79
80      @Override
81      public void onCreate() {
82          super.onCreate();
83
84          SA myAccessory = new SA();
85          try{myAccessory.initialize(this);
86              Log.d("SAP Provider", "On Create Try Block");
87          }
88          catch (SsdkUnsupportedException e){
89              Log.d("SAP Provider",  "On create Try Block Error Unsupported Sdk");
90          }
91          catch(Exception e1){
92              Log.e(TAG,  "Cannot initialize Accessory package.");
93              e1.printStackTrace();
94              stopSelf();
95          }
96      }
97
98      @Override
99      protected void onServiceConnectionRequested(SAPeerAgent peerAgent) {
100         acceptServiceConnectionRequest(peerAgent);
101     }
102
103     public class LocalBinder extends Binder {
104
105         public SAPServiceProvider getService(){
106             return SAPServiceProvider.this;
107         }
108     }
109
110
111     public class SAPServiceProviderConnection extends SASocket{
112         private int connectionID;
113         protected SAPServiceProviderConnection() {
114             super(SAPServiceProviderConnection.class.getName());
115         }
116
117         @Override
118         public void onError(int channelID, String errorString, int error) {
119             Log.e(TAG,  "ERROR:" + errorString + " : " + error);
120         }
121
122         @Override
123         public void onReceive(int channelID, byte[] data) {
124
125             final String information = TrainingModeActivity.getInformationAsJSON();
126             final SAPServiceProviderConnection uHandler = connectionMap.get(Integer.parseInt(String.
    valueOf(connectionID)));
127             if(uHandler == null){
128                 Log.e(TAG,  "Error, can not get SAPServiceProviderConnection handler");
```

```
129             }
130             new Thread(new Runnable(){
131                 public void run(){
132                     try{
133                         uHandler.send(SAP_SERVICE_CHANNEL_ID,  information.getBytes());
134                     }
135                     catch(IOException e){
136                         e.printStackTrace();
137                     }
138                 }
139             }).start();
140         }
141 
142         @Override
143         protected void onServiceConnectionLost(int arg0) {
144             if(connectionMap !=null){
145                 connectionMap.remove(connectionID);
146             }
147         }
148     }
149 }
150 
```

```java
//Class Stopwatch

package at.haraldbernhard.joggingcoach;

import android.util.Log;

public class Stopwatch {
    private int sec, min, hour;
    private String time;

    public Stopwatch(){
    }

    public int getSec() {
        return sec;
    }

    public void setSec(int sec) {
        this.sec = sec;
    }

    public int getMin() {
        return min;
    }

    public void setMin(int min) {
        this.min = min;
    }

    public int getHour() {
        return hour;
    }

    public void setHour(int hour) {
        this.hour = hour;
    }

    public void startStopwatch(){
        while(true){
            while (min<60){
                while (sec<60){
                    try{
                        Thread.sleep(1000);
                        sec +=1;
                        setHour(hour);
                        setMin(min);
                        setSec(sec);
                        Log.i("Time", hour + " : " + min + " : " + sec);
                    }
                    catch(InterruptedException e){
                        e.printStackTrace();
                    }
                }
                min +=1;
                sec = 0;
            }
            hour +=1;
            min = 0;
            sec = 0;
        }
    }
}

```

```java
//Class User

package at.haraldbernhard.joggingcoach;

public class User {
    String username;
    int age, size, weight;

    public User (String username, int age, int size, int weight){
        this.username = username;
        this.size = size;
        this.age = age;
        this.weight = weight;
    }

    public String getUsername() {
        return username;
    }

    public int getAge() {
        return age;
    }

    public int getSize() {
        return size;
    }

    public int getWeight() {
        return weight;
    }
}

```

```
//Android Manifest

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="at.haraldbernhard.joggingcoach">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="com.samsung.accessory.permission.ACCESSORY_FRAMEWORK"/>
    <uses-permission android:name="com.samsung.android.providers.context.permission.
WRITE_USE_APP_FEATURE_SURVEY"/>
    <uses-permission android:name="com.samsung.WATCH_APP_TYPE.Companion"/>
    <uses-permission android:name="com.samsung.wmanager.ENABLE_NOTIFICATION"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <service android:name="at.haraldbernhard.joggingcoach.SAPServiceProvider"/>
        <receiver android:name="com.samsung.android.sdk.accessory.RegisterUponInstallReceiver">
            <intent-filter>
                <action android:name="com.samsung.accessory.action.REGISTER_AGENT"/>
            </intent-filter>
        </receiver>
        <receiver android:name="com.samsung.android.sdk.accessory.
ServiceConnectionIndicationBroadcastReceiver">
            <intent-filter>
                <action android:name="com.samsung.accessory.action.SERVICE_CONNECTION_REQUESTED"/>
            </intent-filter>
        </receiver>
        <meta-data
            android:name="AccessoryServicesLocation"
            android:value="/res/xml/sapservices.xml"/>
        <activity android:name=".LaunchActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MenuActivity" />
        <activity android:name=".CreateUserActivity" />
        <activity android:name=".TrainingModeActivity" />
        <activity android:name=".IntervalModeActivity" />
        <activity android:name=".TrainingDiaryActivity" />
        <activity android:name=".IntervalDialogActivity"
            android:theme="@android:style/Theme.Dialog"/>
    </application>
</manifest>
```

```
1  //Layout LaunchActivity
2
3  <?xml version="1.0" encoding="utf-8"?>
4  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context="at.haraldbernhard.joggingcoach.LaunchActivity">
9
10     <include layout="@layout/my_toolbar"
11         android:id="@+id/my_toolbar"/>
12
13     <TableLayout
14         android:layout_below="@id/my_toolbar"
15         android:layout_height="wrap_content"
16         android:layout_width="wrap_content"
17         android:layout_centerHorizontal="true"
18         >
19         <TableRow>
20             <TextView
21                 android:text="@string/greetings"
22                 android:textColor="#FFFFFF"
23                 android:textSize="24dp"
24                 android:layout_marginTop="60dp"
25                 android:layout_marginBottom="30dp"
26                 android:id="@+id/textViewGreeting"
27                 android:textAlignment="center"/>
28         </TableRow>
29         <TableRow>
30             <TextView
31                 android:textAppearance="?android:attr/textAppearanceLarge"
32                 android:id="@+id/textViewPersonalGreeting"
33                 android:text="Hello Harald"
34                 android:textAlignment="center"
35                 android:layout_marginTop="30dp"/>
36         </TableRow>
37         <TableRow>
38             <TextView
39                 android:layout_width="40dp"
40                 android:layout_height="match_parent"
41                 android:textAppearance="?android:attr/textAppearanceMedium"
42                 android:text="Thank you for installing JoggingCoach. \n It will help you to improve your ↵
   training performance."
43                 android:id="@+id/textViewInformation1"
44                 android:layout_marginTop="30dp"
45                 android:textAlignment="center"/>
46         </TableRow>
47         <TableRow>
48             <Button
49                 android:layout_width="wrap_content"
50                 android:text="@string/start"
51                 android:id="@+id/buttonStartTraining"
52                 android:onClick="startBasicActivity"
53                 android:layout_marginTop="30dp"/>
54         </TableRow>
55         <TableRow>
56             <TextView
57                 android:textAppearance="?android:attr/textAppearanceLarge"
58                 android:text="@string/createUser"
59                 android:id="@+id/textViewCreateNewUser"
60                 android:onClick="startCreateUserActivity"
61                 android:layout_marginTop="50dp"
62                 android:textAlignment="center"
63                 />
64         </TableRow>
65         <TableRow>
```

```
66              <TextView
67                  android:textAppearance="?android:attr/textAppearanceSmall"
68                  android:text="All data are stored locally on the device."
69                  android:id="@+id/textViewInformation2"
70                  android:layout_marginTop="40dp"
71                  android:textAlignment="center"
72                  />
73          </TableRow>
74      </TableLayout>
75  </RelativeLayout>
```

```
1  //Layout CreateUserActivity
2
3  <?xml version="1.0" encoding="utf-8"?>
4  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context="at.haraldbernhard.joggingcoach.CreateUserActivity">
9
10     <include layout="@layout/my_toolbar"
11         android:id="@+id/my_toolbar"/>
12
13     <TableLayout
14         android:layout_below="@id/my_toolbar"
15         android:layout_width="wrap_content"
16         android:layout_height="match_parent"
17         android:layout_centerHorizontal="true"
18         android:layout_marginTop="30dp">
19         <TableRow
20             android:layout_marginTop="20dp">
21             <EditText
22                 android:layout_width="wrap_content"
23                 android:layout_height="wrap_content"
24                 android:inputType="textPersonName"
25                 android:hint="@string/username"
26                 android:ems="10"
27                 android:id="@+id/editTextUsername"
28                 android:layout_alignParentTop="true"
29                 android:layout_centerHorizontal="true" />
30         </TableRow>
31         <TableRow
32             android:layout_marginTop="30dp">
33             <EditText
34                 android:layout_width="wrap_content"
35                 android:layout_height="wrap_content"
36                 android:inputType="number"
37                 android:digits="0123456789"
38                 android:ems="10"
39                 android:id="@+id/editTextAge"
40                 android:hint="Age (0-100)" />
41         </TableRow>
42         <TableRow
43             android:layout_marginTop="30dp">
44             <EditText
45                 android:layout_width="wrap_content"
46                 android:layout_height="wrap_content"
47                 android:inputType="number"
48                 android:digits="0123456789"
49                 android:ems="10"
50                 android:id="@+id/editTextSize"
51                 android:hint="Size (0-250 cm)" />
52         </TableRow>
53         <TableRow
54             android:layout_marginTop="30dp">
55             <EditText
56                 android:layout_width="wrap_content"
57                 android:layout_height="wrap_content"
58                 android:inputType="number"
59                 android:digits="0123456789"
60                 android:ems="10"
61                 android:id="@+id/editTextWeight"
62                 android:hint="Weight (0-200 kg)" />
63         </TableRow>
64         <TableRow
65             android:layout_marginTop="30dp">
66             <Button
```

```
67                  android:layout_width="wrap_content"
68                  android:layout_height="wrap_content"
69                  android:text="@string/createUser"
70                  android:id="@+id/buttonCreateAvatar"
71                  android:onClick="createNewAvatar"
72                  android:layout_alignParentBottom="true"
73                  />
74          </TableRow>
75      </TableLayout>
76  </RelativeLayout>
77
```

```
//Layout MenuActivity

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="at.haraldbernhard.joggingcoach.MenuActivity">

    <include layout="@layout/my_toolbar"
        android:id="@+id/my_toolbar"/>

    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp">
        <TableRow
            android:layout_marginTop="100dp">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/chooseTrainingMode"
                android:id="@+id/textViewGreeting"
                android:layout_centerHorizontal="true"
                android:textAppearance="?android:attr/textAppearanceMedium"/>
        </TableRow>
        <TableRow
            android:layout_marginTop="50dp">
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/startTraining"
                android:id="@+id/buttonStartNewTraining"
                android:onClick="startTrainingSession"
                android:layout_centerHorizontal="true" />
        </TableRow>
        <TableRow
            android:layout_marginTop="50dp" >
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/startIntervalTraining"
                android:id="@+id/buttonStartIntervalTraining"
                android:onClick="startIntervalTraining"
                android:layout_centerHorizontal="true"
                />
        </TableRow>
        <TableRow
            android:layout_marginTop="50dp">
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/showDiary"
                android:id="@+id/buttonDiary"
                android:onClick="startDiary"
                android:layout_centerHorizontal="true"
                />
        </TableRow>
    </TableLayout>
</RelativeLayout>
```

```
//Layout TrainingModeActivity

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="at.haraldbernhard.joggingcoach.TrainingModeActivity">

    <include layout="@layout/my_toolbar"
        android:id="@+id/my_toolbar"/>

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/my_toolbar"
        android:stretchColumns="2">
        <TableRow>
            <TextView
                android:text="Time"
                android:id="@+id/textViewLabelTime"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:layout_column="0"
                android:layout_margin="20dp"/>
            <TextView
                android:id="@+id/textViewTime"
                android:text="0:0:0"
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:layout_column="1"
                android:layout_margin="20dp"/>
        </TableRow>
        <TableRow>
            <TextView
                android:text="Distance"
                android:id="@+id/textViewLabelDistance"
                android:layout_column="0"
                android:layout_margin="20dp"
                android:textAppearance="?android:attr/textAppearanceMedium"/>
            <TextView
                android:id="@+id/textViewDistance"
                android:text="0:0"
                android:layout_column="1"
                android:layout_margin="20dp"
                android:textAppearance="?android:attr/textAppearanceLarge"/>
        </TableRow>
        <TableRow>
            <TextView
                android:id="@+id/textViewLabelCalorie"
                android:text="Calorie"
                android:layout_column="0"
                android:layout_margin="20dp"
                android:textAppearance="?android:attr/textAppearanceMedium"/>

            <TextView
                android:id="@+id/textViewCalorie"
                android:text="0"
                android:layout_column="1"
                android:layout_margin="20dp"
                android:textAppearance="?android:attr/textAppearanceLarge"/>
        </TableRow>
        <TableRow>
            <TextView
                android:id="@+id/textViewLabelSpeed"
                android:text="Speed"
                android:layout_column="0"
                android:layout_margin="20dp"
```

```
67                  android:textAppearance="?android:attr/textAppearanceMedium"/>
68              <TextView
69                  android:id="@+id/textViewSpeedA"
70                  android:text="0"
71                  android:layout_column="1"
72                  android:layout_margin="20dp"
73                  android:textAppearance="?android:attr/textAppearanceLarge"/>
74              <TextView
75                  android:id="@+id/textViewSpeedI"
76                  android:text="0"
77                  android:layout_column="2"
78                  android:layout_margin="20dp"
79                  android:textAppearance="?android:attr/textAppearanceLarge"/>
80          </TableRow>
81          <TableRow>
82              <Button
83                  android:text="@string/start"
84                  android:id="@+id/buttonStart"
85                  android:onClick="buttonStartTraining"
86                  android:textAppearance="?android:attr/textAppearanceLarge"
87                  android:layout_margin="20dp"/>
88              <Button
89                  android:text="@string/pause"
90                  android:id="@+id/buttonPause"
91                  android:onClick="buttonPauseTraining"
92                  android:textAppearance="?android:attr/textAppearanceLarge"
93                  android:layout_margin="20dp"/>
94              <Button
95                  android:text="@string/stop"
96                  android:id="@+id/buttonStop"
97                  android:onClick="buttonStopTraining"
98                  android:textAppearance="?android:attr/textAppearanceLarge"
99                  android:layout_margin="20dp"/>
100         </TableRow>
101     </TableLayout>
102 </RelativeLayout>
103
```

```
//Layout IntervalModeActivity

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="at.haraldbernhard.joggingcoach.IntervalModeActivity">

    <include layout="@layout/my_toolbar"
        android:id="@+id/my_toolbar"
        />

    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/my_toolbar"
        android:layout_centerHorizontal="true">
        <TableRow
            android:layout_marginTop="60dp"
            android:layout_gravity="center_horizontal">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Workout:"
                android:id="@+id/textViewWorkout"
                />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:layout_marginLeft="30dp"
                android:text="0"
                android:id="@+id/tvWorkout"/>
        </TableRow>
        <TableRow
            android:layout_marginTop="60dp">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Recreation:"
                android:id="@+id/textViewRecreation" />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:layout_marginLeft="30dp"
                android:text="0"
                android:id="@+id/tvRecreation"/>
        </TableRow>
        <TableRow
            android:layout_marginTop="60dp"
            android:layout_width="wrap_content">
            <Button
                android:layout_width="wrap_content"
                android:text="@string/start"
                android:id="@+id/buttonIntervalStart"
                android:onClick="startIntervalTraining"
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:layout_gravity="center"
                />
            <Button
                android:layout_width="wrap_content"
                android:text="@string/cancel"
```

```
67                  android:id="@+id/buttonIntervalCancel"
68                  android:onClick="cancelIntervalTraining"
69                  android:textAppearance="?android:attr/textAppearanceLarge"
70                  android:layout_gravity="center"/>
71          </TableRow>
72      </TableLayout>
73  </RelativeLayout>
74
```

```
//Layout IntervalDialog

<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/interval_dialog_layout">

    <TableRow
        android:layout_marginTop="20dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Workout"
            android:id="@+id/textViewWorkout"
            />
        <NumberPicker
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/npWorkoutMin"
            android:theme="@android:style/Theme.Dialog"/>
        <NumberPicker
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/nPWorkoutSec"
            android:theme="@android:style/Theme.Dialog"/>
    </TableRow>
    <TableRow
        android:layout_marginTop="20dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Recreation"
            android:id="@+id/textViewRecreation" />
        <NumberPicker
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/npRecreationMin"
            android:theme="@android:style/Theme.Dialog"/>
        <NumberPicker
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/npRecreationSec"
            android:theme="@android:style/Theme.Dialog"/>
    </TableRow>
    <TableRow
        android:layout_marginTop="20dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Number of intervals"
            android:id="@+id/textViewIntervals" />
        <NumberPicker
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/npIntervals"
            android:theme="@android:style/Theme.Dialog"/>
    </TableRow>
</TableLayout>
```

```
//Layout IntervalDialogActivity

<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context="at.haraldbernhard.joggingcoach.IntervalDialogActivity">

    <TableRow
        android:layout_marginTop="10dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Workout"
            android:id="@+id/textViewWorkout"
            />
        <NumberPicker
            android:layout_marginLeft="30dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/nPWorkoutSec"
            android:theme="@android:style/Theme.Dialog"/>
    </TableRow>
    <TableRow
        android:layout_marginTop="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Recreation"
            android:id="@+id/textViewRecreation" />
        <NumberPicker
            android:layout_marginLeft="30dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/npRecreationSec"
            android:theme="@android:style/Theme.Dialog"/>
    </TableRow>
    <TableRow
        android:layout_marginTop="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Number of intervals"
            android:id="@+id/textViewIntervals" />
        <NumberPicker
            android:layout_marginLeft="30dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/npIntervals"
            android:theme="@android:style/Theme.Dialog"
            />
    </TableRow>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/start"
        android:id="@+id/buttonIntervalDialogStart"
        android:onClick="onStart"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/cancel"
```

```
67          android:id="@+id/buttonIntervalDialogCancel"
68          android:onClick="onCancel"/>
69  </TableLayout>
70
```

```
//Layout TrainingDiaryActivity

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <include layout="@layout/my_toolbar"
        android:id="@+id/my_toolbar"/>

    <TextView
        android:id="@+id/trainingDiary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/diary"
        android:textColor="#FFFFFF"
        android:layout_centerHorizontal="true"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_marginTop="20dp"
        />
</RelativeLayout>

```

```
//Layout TrainingResultAdapter

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="match_parent">
    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="80dp"
        android:layout_centerHorizontal="true">
        <TableRow>
            <TextView
                android:layout_marginTop="20dp"
                android:id="@+id/textViewDateAdapter"
                android:text="Date"
                android:paddingBottom="3dp"
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"/>
        </TableRow>
        <TableRow>
            <TextView
                android:id="@+id/textViewTimeAdapter"
                android:layout_toRightOf="@id/textViewDateAdapter"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Time"
                android:paddingRight="40dp"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                />
            <TextView
                android:id="@+id/textViewDistanceAdapter"
                android:layout_below="@id/textViewTimeAdapter"
                android:layout_alignLeft="@id/textViewTimeAdapter"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Distance"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                />
        </TableRow>
        <TableRow>
            <TextView
                android:id="@+id/textViewCalorieAdapter"
                android:layout_toRightOf="@id/textViewTimeAdapter"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Calorie"
                android:paddingRight="40dp"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                />
            <TextView
                android:id="@+id/textViewSpeedAAdapter"
                android:text="Speed"
                android:layout_below="@id/textViewCalorieAdapter"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                />
        </TableRow>
    </TableLayout>
</RelativeLayout>
```

```
//Toolbar

<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#7CB342"
    android:id="@+id/my_toolbar"
    >
</android.support.v7.widget.Toolbar>
```

```
//Menu

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_exit"
        android:title="@string/exit"
        ></item>
</menu>
```