Hochschule Karlsruhe
University of
Applied Sciences

Hochschule Karlsruhe - Technik und Wirtschaft
Fakultät für Informatik und Wirtschaftsinformatik

Documentation Project Work ML-Based Authentication for Mobile Applications

Project Team :

Nils Barzen

Professor at Karlsruhe University of Applied Sciences:

Prof. Dr. Oliver Waldhorst

Research Assistant at Karlsruhe University of Applied Sciences:

M. Sc. David Monschein

https://github.com/NilsBa/Behavior_Based_Authentication.git

Table of contents

Fundamentals **2**
- Introduction 2
- Goal of the Project 2
- Approach 2
- Project Implementation 3
- Which goals were achieved 4
- Which goals were not achieved 4

How to Use **5**

The Project **8**
- Description 9
- MainActivity 9
- TouchActivity 9
- MyForegroundService 9
- Rest: Read the Project Implementation 10

Fundamentals

Introduction

Constantly logging in your smartphone should be a thing of the past, yet we still are using one password to access our phone. We also log into our bank account with one or two factor authentication, yet people still are getting hacked through phishing or identity theft. But what if we had so many factors, that two correct ones are not enough?
It can not only be the security of the future, it can also simplify our lives by not having to log in everywhere. Only in places, such as banks or emails, where high security is expected.

Goal of the Project

The project should gather as much relevant data as possible so that the collected data can be further analysed. The analysis of the data in a machine learning model is not the subject of the project. Therefore, it is important not to filter out too much data, as the effectiveness of certain data is not known in advance. The data collected should of course be incorporated into the model in some form.

Approach

My first thought was to look at how other people had done similar projects. I found a few projects that had individual features that I should implement. Most of them were written in Java using Android Studio. So I started with that IDE and language as well. Even repositories using Kotlin were useful as well, because they were for my uscase in logic very similar. Only Java had much more boilerplate code. I started with watching YouTube tutorials, to write my first Android App. After I became a bit familiar with my environment, I started setting up projects that recorded individual sensors.

In case of problems, I searched the web for different names of the problem and usually found a solution on Stack Overflow, the Android Studio page itself or YouTube. However, there are still problems for which I have not yet found a solution (see Touch Input) even after a long search. I then tried to persist data and send it afterwards to a test Site. With some success, I then tried to implement everything in one project. This took a while though.

The problem with this approach is that I have not yet fully understood the basis and there are still plenty of areas that can be optimised.
The advantage is that putting everything into one project leads to a new project where only important features have been taken over.

Project Implementation

For the sensor types there were different possible solutions available. Some were plaintely better than others, some weren't available anymore. One good example is the collectivisation of the touch events. They are now only implemented to work in the app itself or on other apps, which do it themselves. A good solution was to have a transparent view above the screen that can detect touch events. When you exit the app, the view still returns a touch event, but the values are all zero. This has been changed in higher Android APIs for security reasons.

To run a Foreground service and collect data after closing the application, it must be started and stopped somewhere. This happens in the MainActivity. In the Foreground service data gets collected, saved and sent. It is the common main function of this project.
The MainActivity can also create a NewActivity, to test out the touch inputs in other activities.
Also not every sensor is being registered, only those named in the SensorLists.

To see all available sensors:
https://developer.android.com/guide/topics/sensors/sensors_overview
https://developer.android.com/guide/topics/sensors/sensors_motion
https://developer.android.com/guide/topics/sensors/sensors_position
https://developer.android.com/guide/topics/sensors/sensors_environment

Sending a sensor value every time it is updated is very costly for high frequency sensors. Therefore, not every small change is recorded and the programme only records new data every timerDelay. The value can be minimised or set to zero to get more accurate data.

To save a sensor value, it is stored in ThreeDPoints, which is simply something more than an ArrayList of ThreeDPoint. ThreeDPoint contains only the timestamp at which the data was collected and the value of three axes of the data.

Sending the collected data is realised over HTTP. The data is bundled in a JsonDataSet, which only contains a JSONObject and a constructor. The Foreground service calls the PostData function which simply adds a file path to the set link from the Foreground service and sends the data over the post method. This easy implementation uses the retrofit library.

To learn more about Retrofit: https://square.github.io/retrofit/

Which goals were achieved

1) The collecting of following data: sensors (rotation, light, …), touch inputs, location and Wlan
2) Creating a File and persisting latest data
3) Sending collected data over HTTP

Which goals were not achieved

1) As explained in the project implementation, touch inputs only work in the app itself.
2) Data is currently being sent as an uncompressed json, which isn't the most optimal way to send data. It could be improved by compressing the json with gzip.

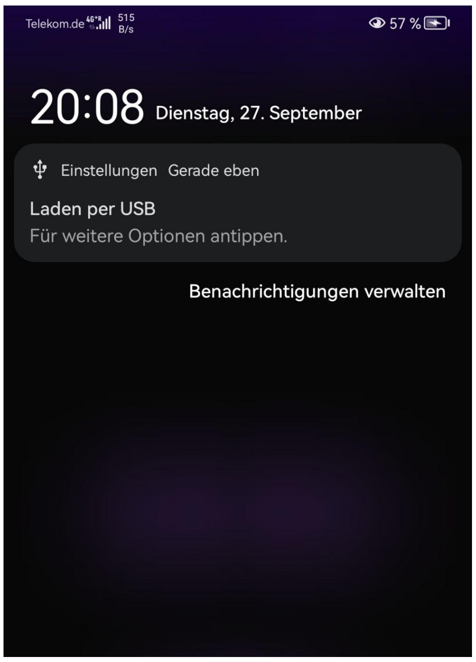
How to Use

You only need Android Studio to open the package.
The simplest way to test the program is to connect your mobile phone with your computer via cable. Also the phone needs to have developer mode activated.
For myself I needed to go to: Settings > About Phone > Software Info > Build Number and tap it seven times.
Then you need to go to click on your connection with the computer and change it to transfer files. A Pop-up should appear asking to enable debugging. After pressing OK you can return to the computer.
If your phone appears on the IDE you can start the program.

To lookup the internal file browser navigate to: View > Tool Windows > Device File Explorer
In the File Explorer you can find the File under: data > data > com.example.behavior_based_authentication > files > Behavior_Based_Authentication

After starting the APP and clicking on START, a lot of features need to be allowed. Especially click on always allow when asked about the location. If you don't do that, the location may not work in the background.

The START button is for starting the foreground service and the STOP button for closing it again. You can start the service, close the app and the service will still run. To close the service again, you have to open the app and click on STOP.
The NEW ACTIVITY exists to test touch inputs outside of the MainActivity scope.

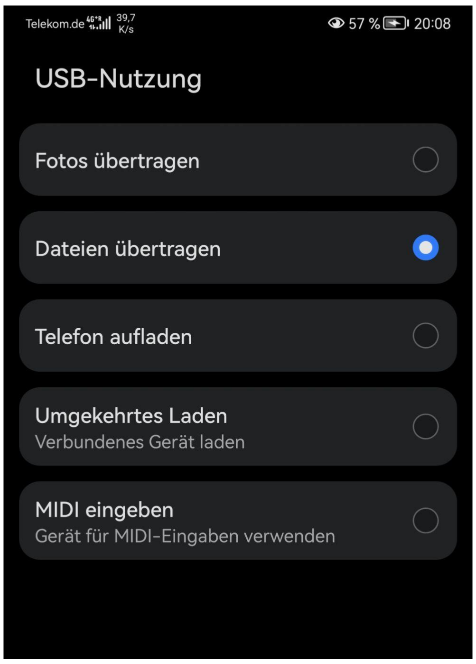

Telekom.de 515 B/s 57 %

20:08 Dienstag, 27. September

Einstellungen Gerade eben

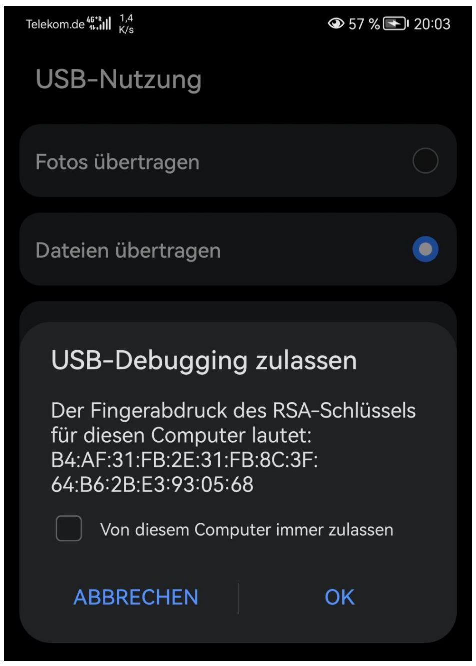
Laden per USB
Für weitere Optionen antippen.

Benachrichtigungen verwalten


Telekom.de 39,7 K/s 57 % 20:08

USB-Nutzung

Fotos übertragen ○
Dateien übertragen ●
Telefon aufladen ○
Umgekehrtes Laden ○
Verbundenes Gerät laden
MIDI eingeben ○
Gerät für MIDI-Eingaben verwenden


Telekom.de 1,4 K/s 57 % 20:03

USB-Nutzung

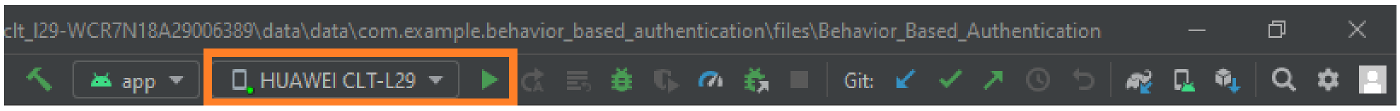
Fotos übertragen ○
Dateien übertragen ●

USB-Debugging zulassen

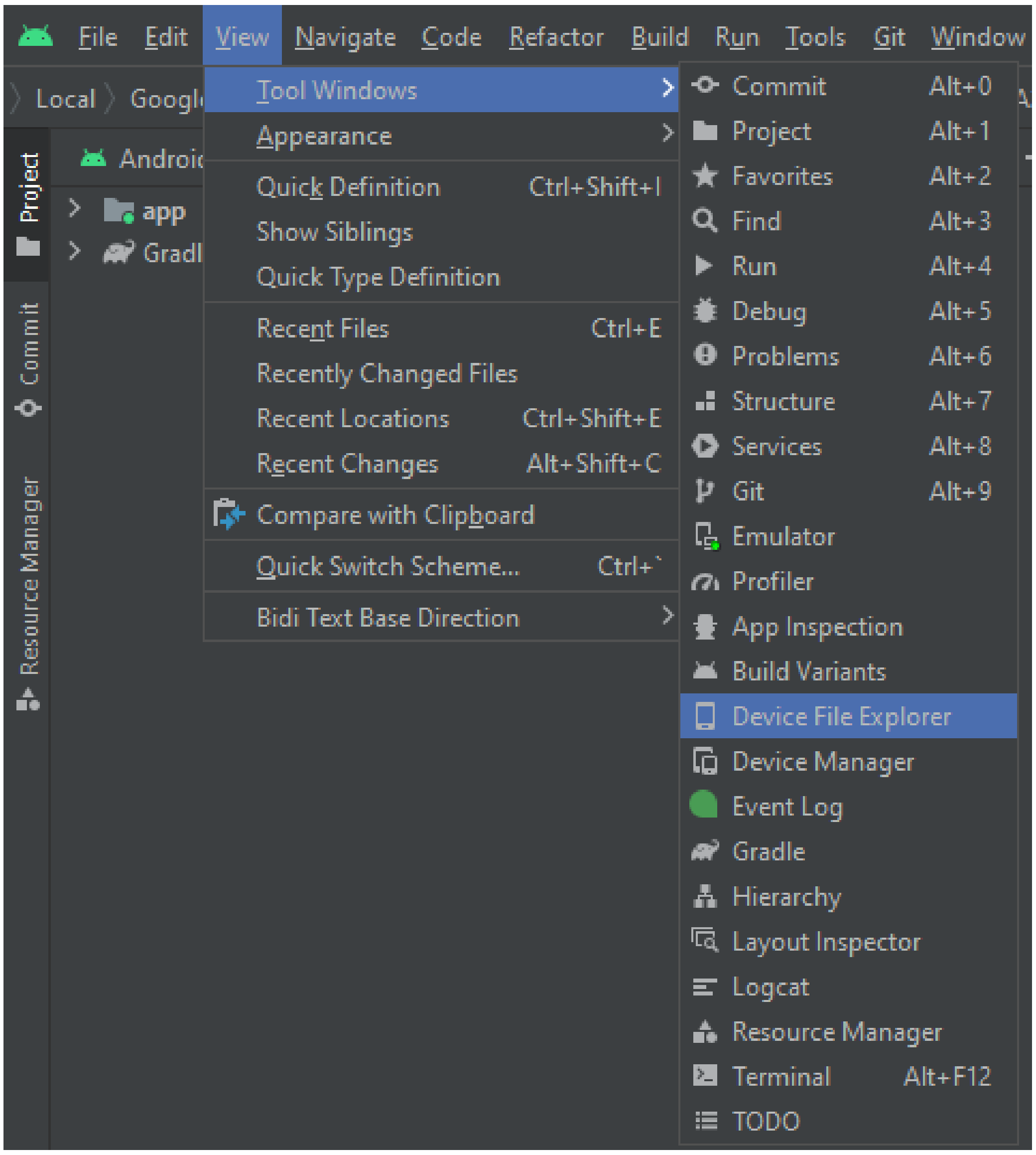
Der Fingerabdruck des RSA-Schlüssels für diesen Computer lautet:
B4:AF:31:FB:2E:31:FB:8C:3F:
64:B6:2B:E3:93:05:68

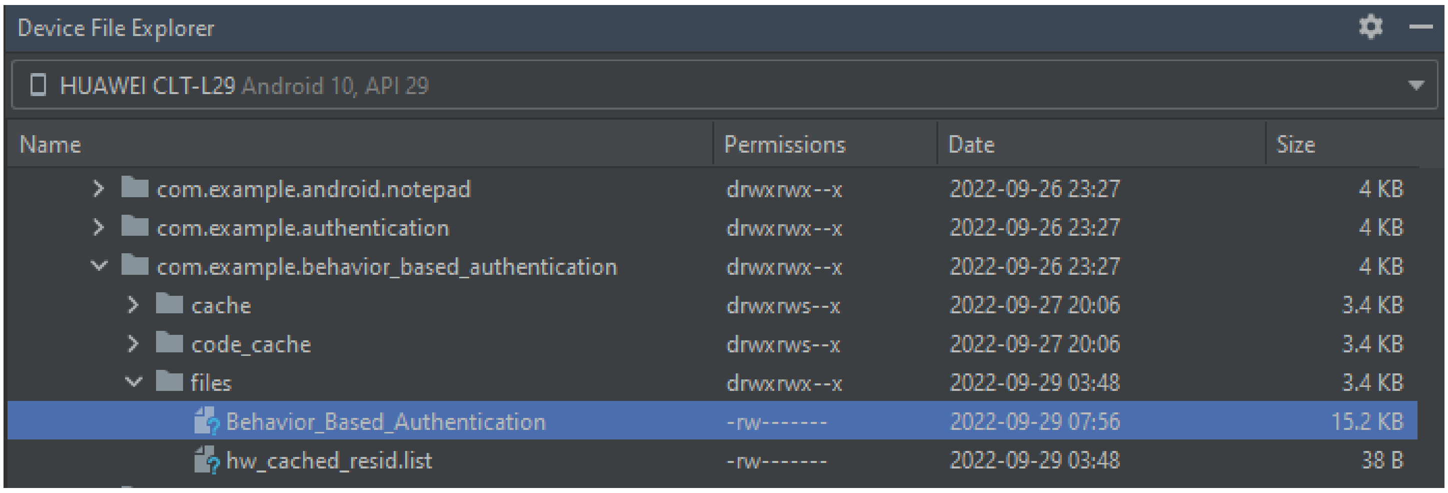
☐ Von diesem Computer immer zulassen

ABBRECHEN | OK


clt_l29-WCR7N18A29006389\data\data\com.example.behavior_based_authentication\files\Behavior_Based_Authentication

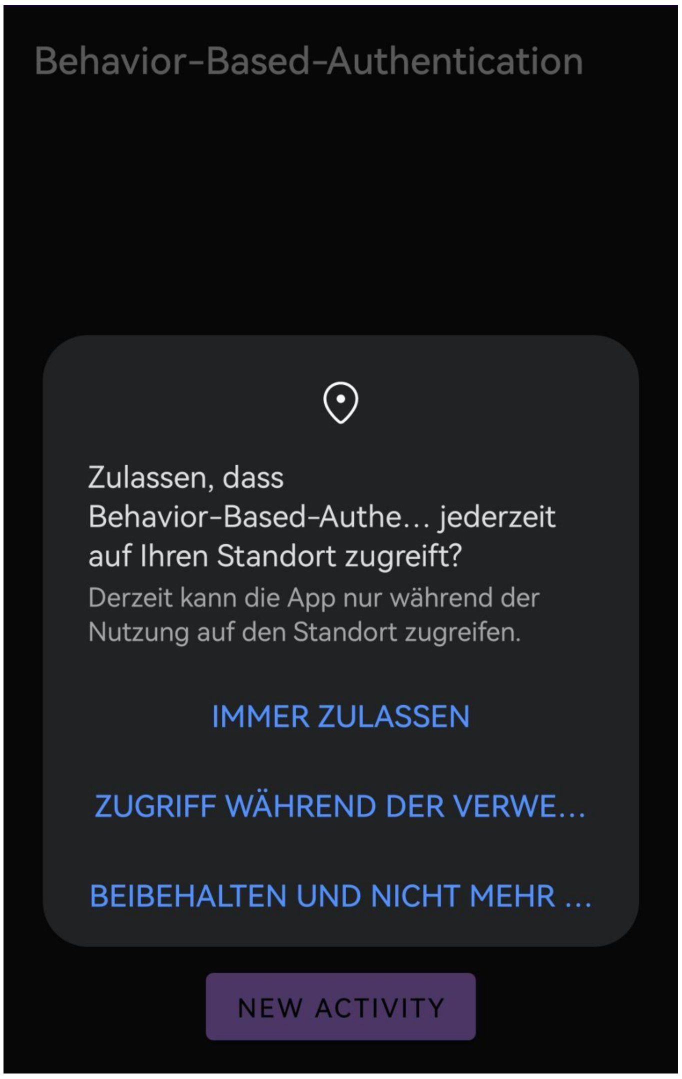
app | HUAWEI CLT-L29 | Git:




Device File Explorer

HUAWEI CLT-L29 Android 10, API 29

Name	Permissions	Date	Size
com.example.android.notepad	drwxrwx--x	2022-09-26 23:27	4 KB
com.example.authentication	drwxrwx--x	2022-09-26 23:27	4 KB
com.example.behavior_based_authentication	drwxrwx--x	2022-09-26 23:27	4 KB
cache	drwxrws--x	2022-09-27 20:06	3.4 KB
code_cache	drwxrws--x	2022-09-27 20:06	3.4 KB
files	drwxrwx--x	2022-09-29 03:48	3.4 KB
Behavior_Based_Authentication	-rw-------	2022-09-29 07:56	15.2 KB
hw_cached_resid.list	-rw-------	2022-09-29 03:48	38 B

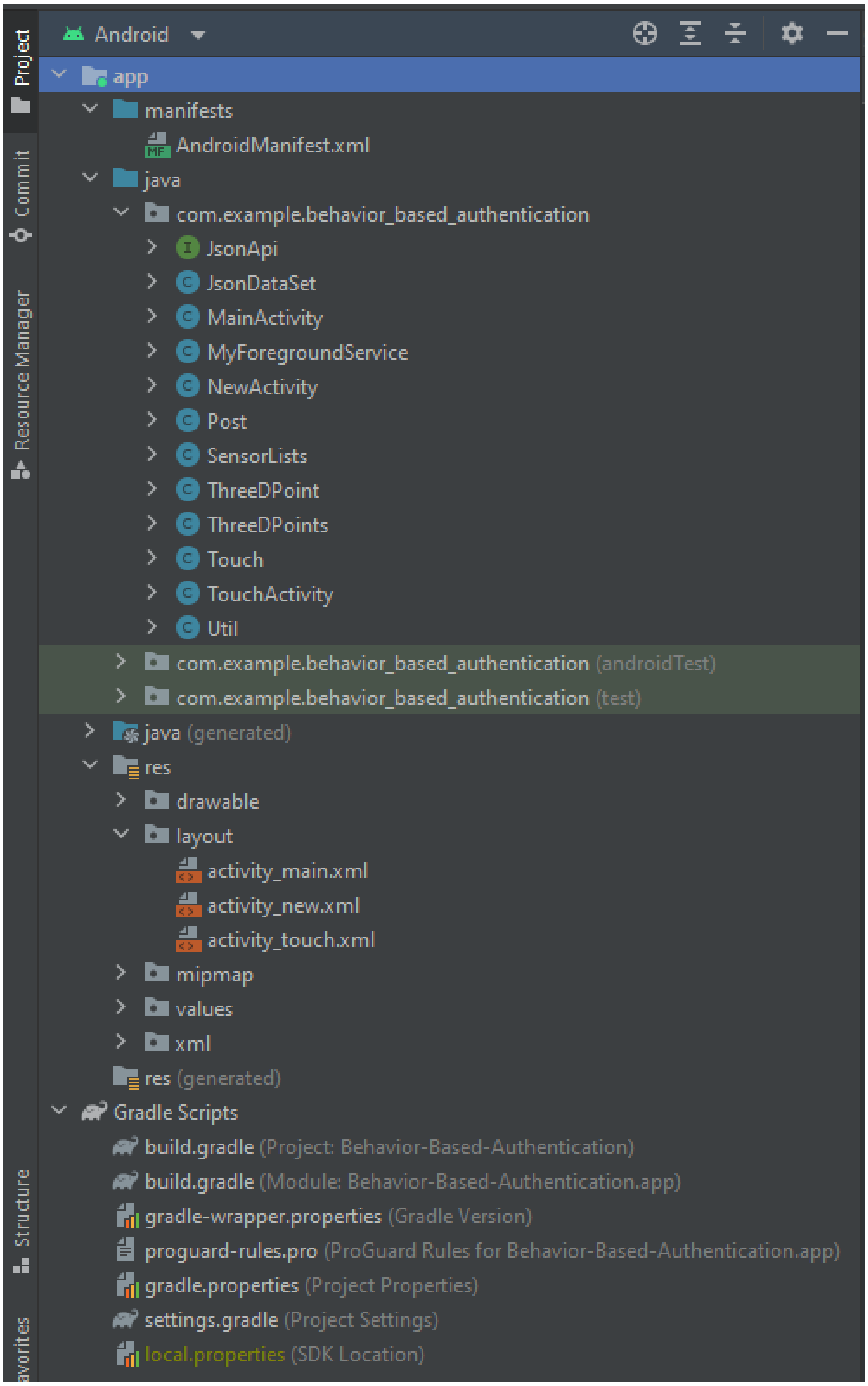


Behavior-Based-Authentication

Zulassen, dass Behavior-Based-Authe… jederzeit auf Ihren Standort zugreift?

Derzeit kann die App nur während der Nutzung auf den Standort zugreifen.

The Project



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.behavior_based_authentication
      JsonApi
      JsonDataSet
      MainActivity
      MyForegroundService
      NewActivity
      Post
      SensorLists
      ThreeDPoint
      ThreeDPoints
      Touch
      TouchActivity
      Util
    com.example.behavior_based_authentication (androidTest)
    com.example.behavior_based_authentication (test)
  java (generated)
  res
    drawable
    layout
      activity_main.xml
      activity_new.xml
      activity_touch.xml
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
  build.gradle (Project: Behavior-Based-Authentication)
  build.gradle (Module: Behavior-Based-Authentication.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for Behavior-Based-Authentication.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Description

The Project consists mainly of a manifest file, java classes, activities and the gradle scripts.
The manifest contains the permissions a user has to provide to use this application.
Java classes contain the executable code.
Activities contain the layout and components of the GUI.
Gradle is a Toolkit and it contains the dependencies of other libraries.

MainActivity

The program starts in the MainActivity. When the MainActivity gets created, the buttons are being initialised. If START is pressed, the permissions are getting verified and requested if missing. Then the foreground service is getting started.
The STOP button creates a new service to overwrite the running one.
The NEW ACTIVITY button only creates a new activity to test if touch inputs get collected outside of the main activity.

TouchActivity

All activity classes extend the TouchActivity. It is a basic class, which stores and logs touch inputs. These are sent through a broadcast to MyForegroundService.

MyForegroundService

The foreground service is the backbone of the project.
The sensorPoints and lastTimeStamps are hashmaps. They represent the data and the last time a sensor has collected data. The key of these maps represents the global ID of a sensor.
Touch inputs are obtained through the BroadCastReceiver and stored in the touchArrayList.

OnCreate starts the location updates, registers the receiver for touch inputs and starts a timer to save data locally and send data over REST API.

OnStartCommand compares the available sensors to the sensors in SensorLists. The matching ones are being registered.

OnSensorChanged gets called every time a sensor has a new value. If the new value is too new, it gets dropped. Same as the touch data, the sensor data gets stored into a list.
Every timerDelay a json is created and parsed. The wifiJSON, locationJSON and touchJSON are created in their respective functions and concatenated with the json of every sensor.
In order to parse the json containing all the data, it is converted into a gson to prettify it. The json then gets locally persisted and sent to a test URL.

To send data, a JSONObject is sent via JsonApi. A base URL is set up in the sendData function of MyForegroundService to which the data is sent. The filepath is added in the JsonApi. Both HttpInterceptor and OkHttpClient are present for logging purposes only.

The toasts and notifications are only a visual help and can be removed. This also applies for the log functions.

Rest: Read the Project Implementation