Fontys Hogeschool Techniek en Logistiek

Informatics

Software Factory

Work - to - Students

Handover document

January 12, 2018

Information

Authors	Julia Mödinger (2593467)
	Moritz Richter(2620731)
	Thilo Ritzerfeld (2462966)
	Merve Sahin (2621363)
	Niklas Schwerin (2645386)
Module	Software Factory
Supervising Docent	Ferd van Odenhoven
Institute	Fontys Hogeschool Techniek en Logistiek
Study program	Informatics
Study year	2017/2018
Place and Date	Venlo, January 12, 2018

Contents

Information I

List of Tables IV

List of Figures IV

1 Introduction 1

1.1 Purpose . . . 1

1.2 Project structure . . . 1

2 GitHub 2

2.1 GitHub Account Setup . . . 2

2.2 Checking out project . . . 2

3 Ionic 3

3.1 Installation of Ionic . . . 3

3.2 Test and deploy Ionic application . . . 4

3.3 Configuration of Ionic application . . . 4

3.3.1 Dependencies and plugins . . . 4

3.3.2 Connection components . . . 7

3.3.3 Interaction with GUI . . . 9

4 Firebase 10

4.1 Database Setup & Data import . . . 10

4.2 Push Notification Setup . . . 12

4.3 Cloud Functions . . . 13

4.3.1 Deploying Firebase Functions . . . 13

4.4 Share Firebase with your team . . . 15

5 Deployment 16

5.1 Android . . . 16

5.1.1 Emulator . . . 17

5.1.2 Run with AndroidStudio . . . 20

5.1.3 Run with command line 21
5.2 iOS 24

6 Prospect for the future **27**

7 References **28**
7.1 GitHub 28
7.2 Ionic 28
7.3 Firebase 29
7.4 Android 29
7.5 iOS 30

List of Tables

1 Dependencies and plugins 6

List of Figures

1 Setup GitHub account 2
2 Creating a Firebase Project 10
3 Import .json file 11
4 Firebase iOS setup 12
5 Sharing Firebase project 15
6 Adding user to Firebase project 16
7 Importing project to AndroidStudio 17
8 Open AVD Manager 18
9 Creating a Virtual Device 18
10 Downloading Android version 19
11 URL configuration 20
12 Importing Xcode project (1) 24
13 Importing Xcode project (2) 25
14 Device selection (1) 25
15 Device selection (2) 26

1 Introduction

1.1 Purpose

This handover document is intended as a guide for future developers or administrators of the "Work-to-students" application. It will provide the reader with the necessary knowledge for setting up/checking out the project, changing or maintaining the implementation of the application. In addition to this there is a prospect for the future, to give advices what has to be done.

1.2 Project structure

This handover document will give future developers or administrators a detailed explanation, how the project can be checked out from GitHub. It is also stated how to set up ionic and Firebase and how to work with it. In addition to that there is a description how to deploy the application on Android and iOS. At the end there is a prospect in the future, which will give hints on tasks that still have to be performed.

2 GitHub

GitHub is used as storage and access point (repository) of files for the project "Work-to-students". The complete application can be found there as well. It is a free service provided by GitHub, Inc. and can only be acquired with a registered GitHub Account.
GitHub can be accessed under the following URL `https://github.com/`.

2.1 GitHub Account Setup

1. Open the URL `https://github.com/` and create a personal GitHub account.

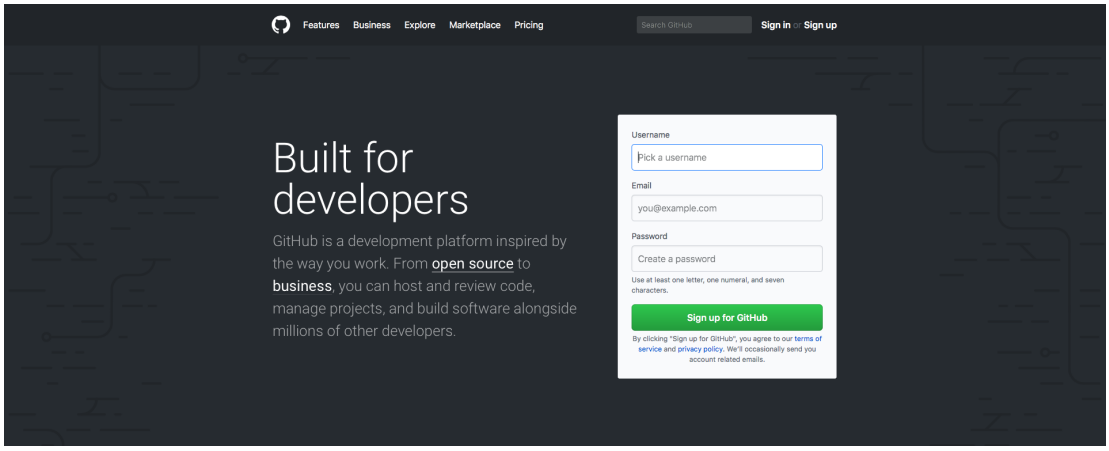


Figure 1: Setup GitHub account

2.2 Checking out project

After setting up the personal GitHub account you have to check out/clone the project "Work-to-students". For doing that, there are several tools that can be used, which almost work the same way like the procedure with the command line, explained in the following listing.

1. Open the terminal on your computer
2. Navigate via terminal to the folder where you want to clone the repository to
3. Type "git clone", and then paste the url `https://github.com/Merve40/wts`. Press Enter and your local clone will be created. You are ready to start.

```
1 $ git clone https://github.com/Merve40/wts
```

3 Ionic

Our Software factory group uses the Ionic framework to create our application. Ionic is an open-source framework used for hybrid mobile application development. The framework uses Web technologies like CSS, HTML5, and SCSS.
The software development kit can be used for mobile applications on Android, iOS or Windows Phone. The installation of Ionic works with the execution of commands in the terminal.

3.1 Installation of Ionic

For Windows you need to install "Git for Windows" to execute the commands in the Git Bash (Skip this step for Mac OS and Linux).

1. Visit https://git-scm.com/download/win and follow the instructions
2. Visit https://nodejs.org/en/, download for your system software and follow the instructions
3. Install Cordova

```
$ sudo npm install -g cordova
```

4. Install Ionic

```
$ sudo npm install -g ionic
```

   If you want to create a new project then execute the following commands

```
$ ionic start todo blank --type ionic1
```

   And enable iOS and Android platforms:

```
$ ionic cordova platform add ios
$ ionic cordova platform add android
```

3.2 Test and deploy Ionic application

There are four different approaches to test your Ionic application. You can test it in a desktop browser, in the simulator for iOS and Android (and Windows Phone), in a mobile browser on your phone, or as a native app on the phone. In the following, the two ways we used to test our applications are shown.
The best way is to test the application in browser for testing without device specific functionalities. For this approach you have to navigate to the root folder of the project in your terminal and run the following command:

```
1 $ ionic serve
```

Listing 1: Test your application

If your mobile phone is in the same network as the computer, you can connect to the IP address of the desktop computer to test the application on your mobile phone. Just load the address on the mobile phone (i.e. 192.168.1.123:8000).

3.3 Configuration of Ionic application

After you learned how to install Ionic and create a project, this chapter shows how to configure an Ionic application and how it is done for us.

3.3.1 Dependencies and plugins

In this chapter we describe the dependencies and plugins that we use in our project except from the plugins and dependencies we already installed in the installation and setup of Ionic.
We installed the Firebase dependency and the native plugin to use Firebase as our database. It implements our document oriented database model in which we store and query user data such as making the data available between users.
The promise-polyfill was installed because we had an error in specific browers when building and deploying. Therfore we installed this package to support native promises.
The node.js was mandatory for the installation of Ionic. It is a JavaScript run time environment for executing our TypeScript code.
The Crypto.js is a JavaScript library of crypto standards to crypt the passwords and other personal data of in example the login.

The TranslationService is to translate values in our application. We implemented the languages English and German and other languages can easily be added with the TranslationService plugin.
The Moment library is a library to use the data format. We use it in example in our profile as the time stamp or the birth of the users.
The Cloud Messaging/ Push is a plugin which provides push notifications for Cordova applications with Google Firebase. We use various push notifications to notify the user about new messages and new contact requests.
The Globalization plugin obtains information and performs operations specific to the user's language. It is used for reading the users device language and showing the app in the same language or English, in case the device language is not used in the application.
In the following table an overview over the plugins and dependencies is shown.

Plugin/ Dependency	Description	Installation
Firebase dependency	Store and query user data, and makes it available between users	$ sudo npm install firebase –save
Firebase native plugin	Store and query user data, and makes it available between users	$ ionic cordova plugin add cordova-plugin-firebase $ npm install –save @ionic-native/firebase
Promise-polyfill	For browser that does not support native promises	$ sudo npm install promise-polyfill –save-exact
Node.js	JavaScript run-time environment for executing JavaScript code	npm install @types/node –save-dev
CryptoJs	JavaScript library of crypto standards	npm install crypto-js
TranslationService	Translate values in the app	npm install @ngx-translate/core @ngx-translate/http-loader –save
Moment library	Library for date format	npm install –save moment
Cloud Messaging / Push	Push notification plugin for Cordova applications with Google Firebase	cordova plugin add cordova-plugin-fcm
Globalization	Obtains information and performs operations specific to the user's language	npm install –save @ionic-native/globalization

Table 1: Dependencies and plugins

3.3.2 Connection components

Because every Ionic application is a web page you need an index.html file which is the page that loads first when you start your app. The design of the HTML file is done in a CSS file. The logic of the page is stored in our case in a TypeScript file.
Because the files work together you need to connect them in different ways.
In the TypeScript file where you store the logic of the application is a reference to the HTML and CSS file as seen in the coding example below. The selector is the page-login which references to the CSS file and the template URL references to our HTML file which is called "login.html".

```
/**
 * Page for Log-In
 */
@Component({
  selector: 'page-login',
  templateUrl: 'login.html'
})
```

Listing 2: Extract typescript file

This coding example shows an extract of the CSS file of our login page with the identifier "page-login".

```
page-login {

    div.top{
        margin-top: 25px;
        text-align: center;
    }
}
```

Listing 3: Extract css file

Furthermore, if you want to add pages to your mobile applications you need to configure it in the app module and app components. The app components can be seen as the starting point for the Ionic application.
In the app components the pages are defined. They are stored in an array. A root page is showed when the application is launched. In our case it is the login page which is shown.

```
16 import { LoginPage } from '../pages/login/login';
17 export class MyApp {
18   @ViewChild(Nav) nav: Nav;
19
20   rootPage: any = LoginPage;
21
22   pages: Array<{ title: string, component: any }>;
23   }
24
25  translate.get(['NEWSFEEDPAGE']).subscribe(translations => {
26         { title: translations.NEWSFEEDPAGE, component: NewsfeedPage }
27       ];
28     });
```

Listing 4: Extract App components

The app module tells Angular what is included in your application and how to compile and launch it. It is an Angular module which is described with NgModule in the code.
In the declarations section the components are included so Angular can recognize them.
In the entryComponents section we define all page components since these are all loaded through the Navigation Controller in our application.

```
29 @NgModule({
30   declarations: [
31     MyApp,
32     LoginPage,
33     StudentProfilePage,
34     Profile_EditPage,
35     ListSearchPage,
36     MapPage,
37   ],
38   bootstrap: [IonicApp],
39   entryComponents: [
40     MyApp,
41     LoginPage,
42     StudentProfilePage,
43     Profile_EditPage,
44   ],
45 })
```

Listing 5: Extract App module

3.3.3 Interaction with GUI

The interaction with the Graphical User Interface (GUI) is handled with the so-called ngmodel in Angular which binds a variable of the TypeScript file to a specific field in the GUI.
The following extract shows the HTML file which binds the variable students with help of a modification of the ngmodel, the ngFor which is used to bind arrays to the GUI.

```
46 <ion-item *ngFor="let item of students;">
47         <ion-item ion-item (click)="test">
48           <h2>{{item.name}}</h2>
49           <p>{{item.description}}</p>
50           <p>{{item.message}}</p>
51         </ion-item>
52       </ion-item>
```

Listing 6: Extract Contact Request html

4 Firebase

Firebase is used as backend for saving data & files and for pushing notifications. It is a free service provided by Google and can only be acquired with a registered Google-Account or a Gmail-Account.
Firebase can be accessed under the following URL https://console.firebase.google.com/.

4.1 Database Setup & Data import

1. Create a Firebase Project, name it "WorkToStudents" and choose the right country

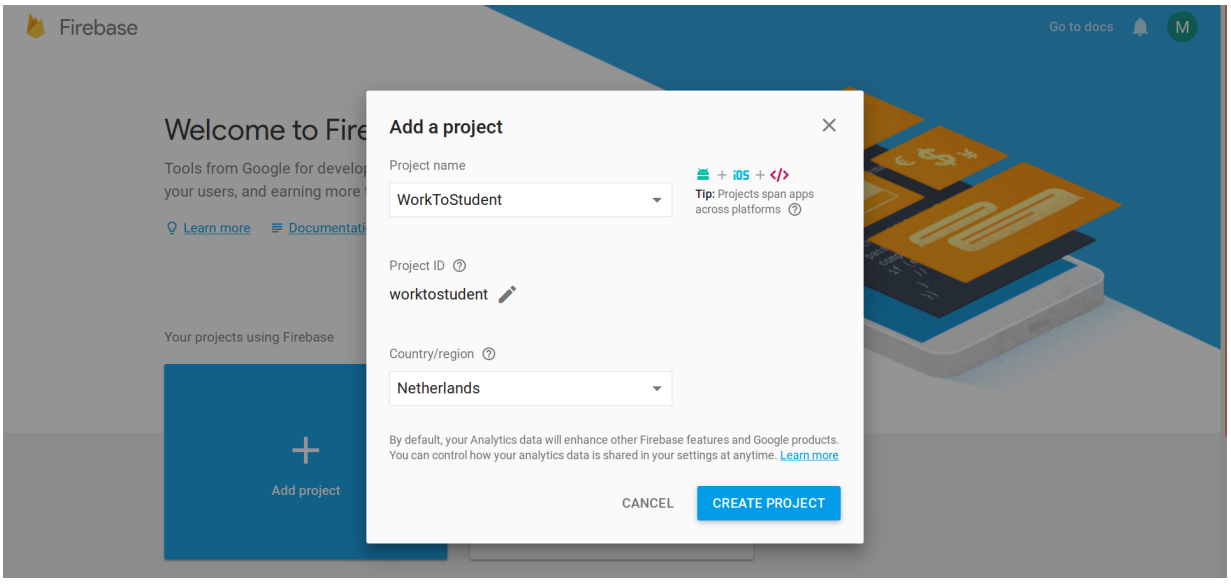


Figure 2: Creating a Firebase Project

After creating the project, you will be forwarded to the main page, which shows a menu containing Firebase Services. We are currently using three services: **Database**, **Cloud Functions** and **Notifications**.

2. Open database and click on the dots on the right corner to import *data.json*

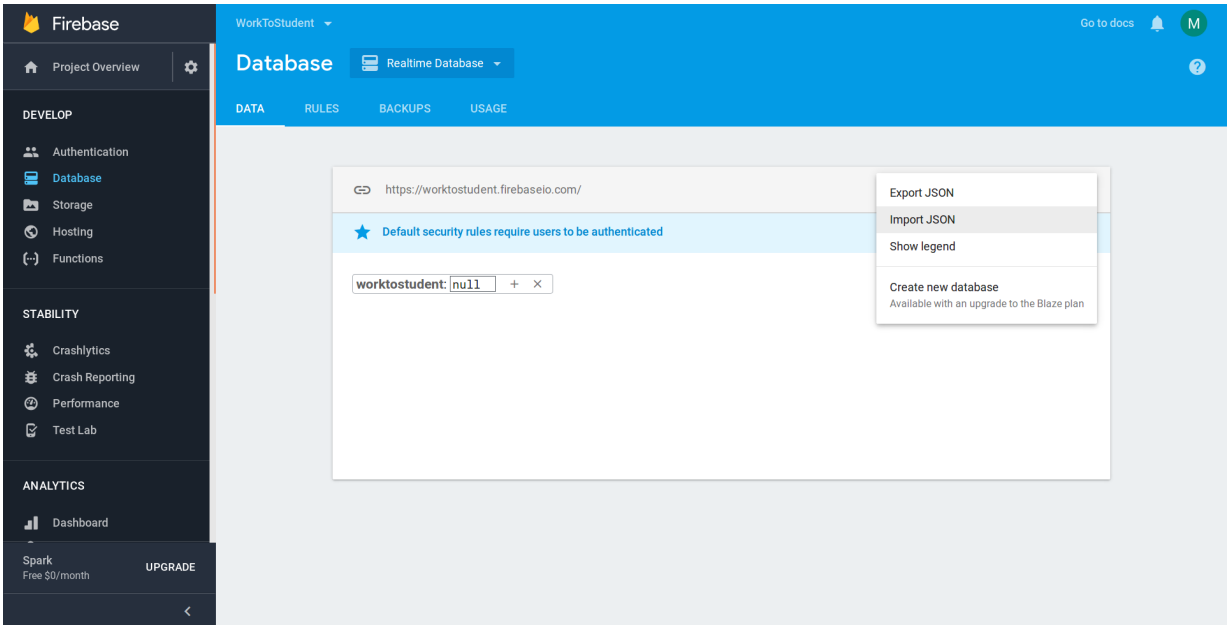


Figure 3: Import .json file

3. Open the **rules** tab at the top and replace its content with *rules.json* & click the **publish** button once you are finished.

4.2 Push Notification Setup

We will make use of the integrated Firebase Cloud Messaging (FCM) for sending Push Notifications. Ionic already has a plugin for FCM, which expects a configuration file, in order to communicate with the correct Server-Endpoint. The following instructions show how to import the configuration for iOS and Android.

1. Navigate back to the overview by clicking *Project Overview* (left panel) and click on the iOS icon to add firebase for iOS.

1.1 iOS Bundle ID should be: **com.wts**
Click on **Regiser App** and download the file.

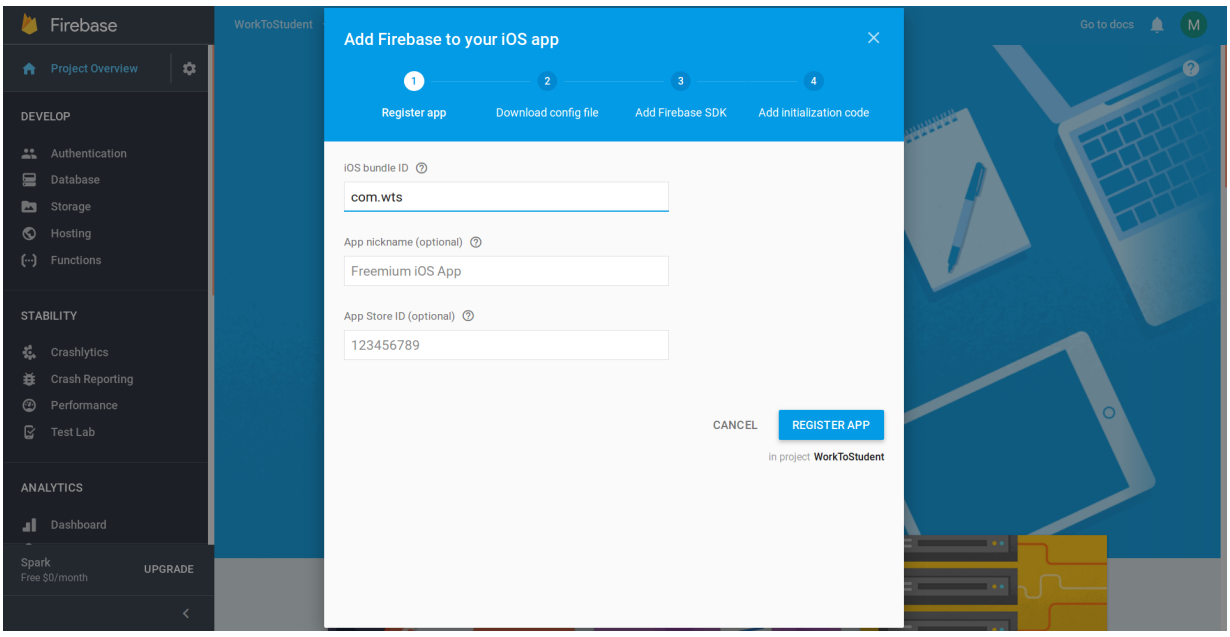


Figure 4: Firebase iOS setup

1.2 Replace the existing **GoogleService-Info.plist** file in the wts folder with the new one.

2. Navigate to the **Project Overview** again, click on **add another app** at the top-right position and choose Android.

2.1 Set **Android Package Name** to **com.wts**, click **Register App** and download the **google-services.json** file

2.2 Replace the existing **google-services.json** file in the wts folder with the new one

4.3 Cloud Functions

Cloud functions provide a way for setting up server-side triggers. We are making use HTTP and database triggers. Further information can be found here:
https://firebase.google.com/docs/functions/.
This feature lets you deploy code, that will be triggered each time a certain request was made.

The table below shows a list of existing triggers:

Function Name	**Event**	**Endpoint**
contains	http	*base-url/contains
sortBy	http	*base-url/sortBy
notfiyContactAccepted	write	/Kontaktanfragen/{pushId}
notifyContactRequested	create	/Kontaktanfragen/{pushId}
pushNotification	write	/Nachricht/{pushId}

*base-url = https://us-central1-worktostudents.cloudfunctions.net

Whereas HTTP-Triggers can only be invoked over a url, database triggers, such as **create** and **write**, are invoked when creating or updating entries on certain endpoints.

4.3.1 Deploying Firebase Functions

In order to deploy functions we need to install Firebase SDK and its dependencies with npm:

```
1 $ npm install -g firebase-tools
```

To install the latest version of Firebase functions run these commands:

```
1 $ npm install firebase-functions@latest --save
2 $ npm install -g firebase-tools
```

Create a new folder and name it after the trigger you want to implement. Navigate on the command line to your directory and login to firebase:

```
1 $ firebase login
```

The command below will generate a **functions** folder, which contains *package.json* and *index.js*.

```
1 $ firebase init functions
```

Put your implementation into *index.js* and your dependencies into *package.json* (For implementation see references).

To deploy your function after you have implemented it, run the command below. Replace "yourFunctionName" with the name of the new function:

```
1 $ firebase deploy --only functions:yourFunctionName
```

4.4 Share Firebase with your team

You can share your database and other Firebase-Services with your team members. Click on the settings icon and open "Users and permissions".

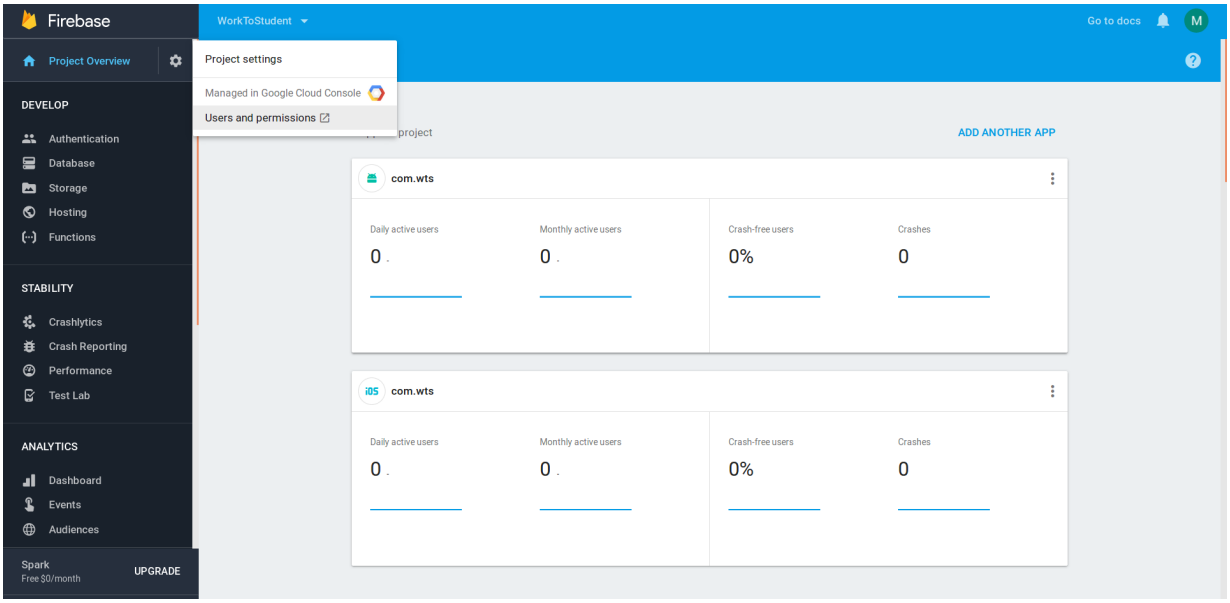


Figure 5: Sharing Firebase project

Click on "Add" at the top of the page and enter the email addresses of your team mates. You can enter multiple roles, but project owner should suffice.

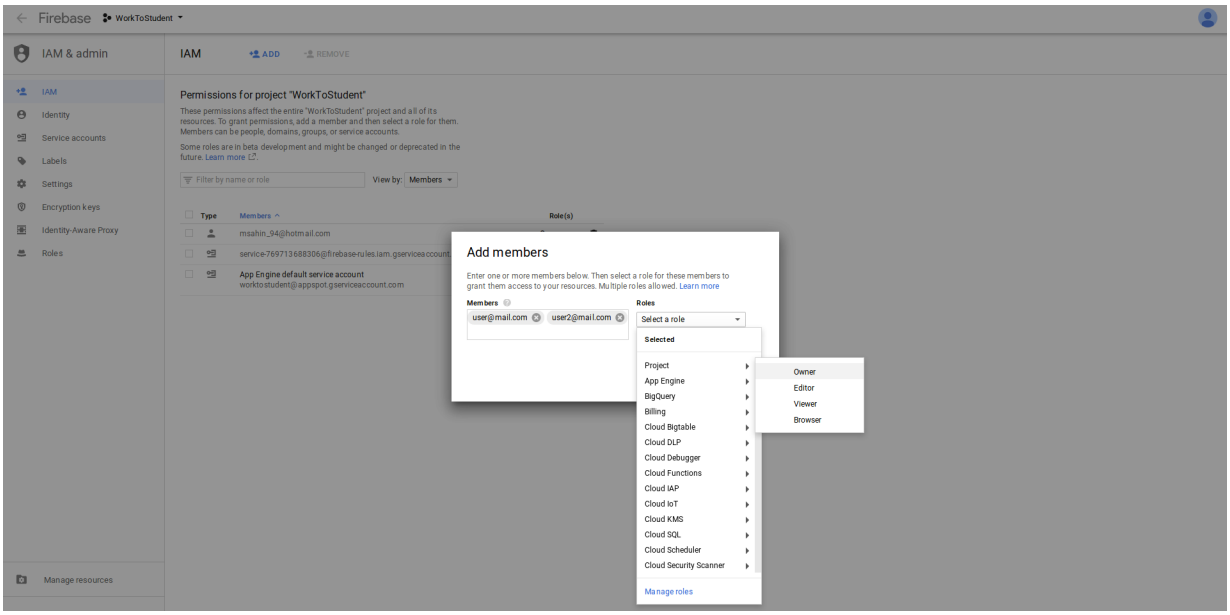


Figure 6: Adding user to Firebase project

5 Deployment

The following subsections will provide further information on how to deploy your application to an emulator/simulator or a mobile device.

5.1 Android

There are several ways to test your app on an Android device, we will make use of two different methods:

1. Open your Android-project from AndroidStudio
2. Test your app with help of the command line

Before you get started make sure you have the following software installed:

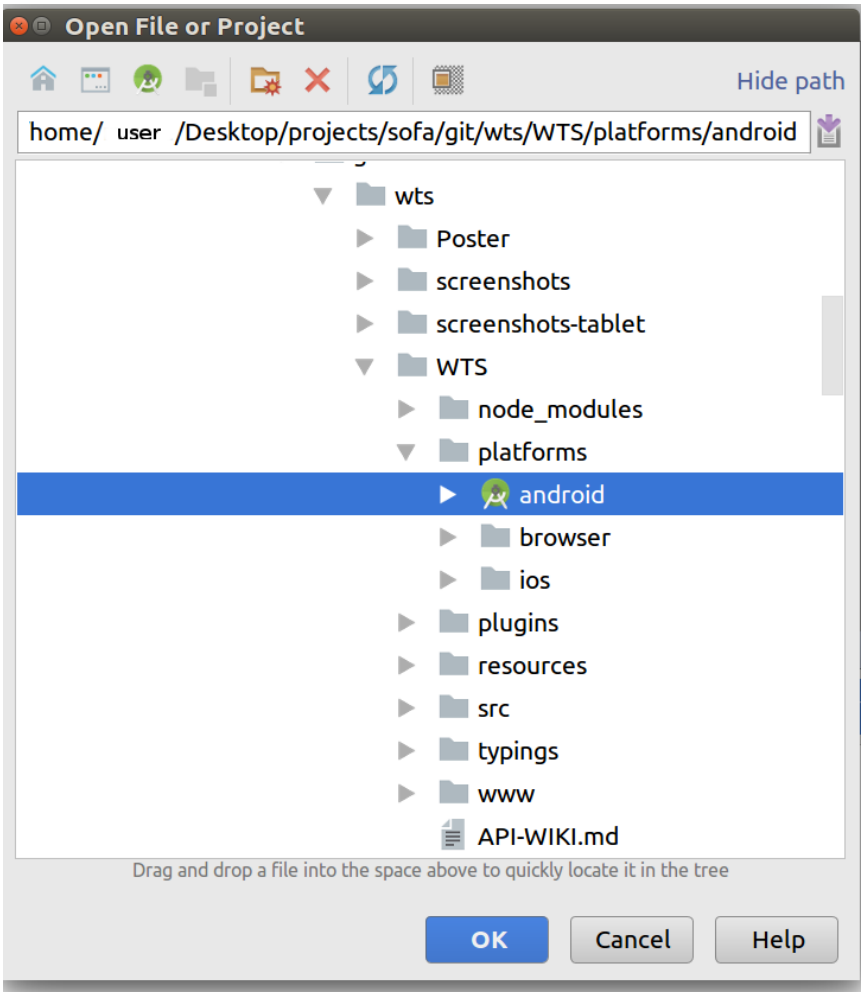
- **AndroidStudio**: https://developer.android.com/studio/index.html
- the emulator, if you did not do it in the first step
- **gradle**: https://gradle.org/releases/

The easiest way to test your application is to open **AndroidStudio** and navigate to your android-project which can be found in the folder *WTS/platforms/android*.

5.1.1 Emulator

If you want to test on an emulator, the first thing you have to do is set up a device in your **AVD Manager** and install the packages for that device:

1. Import the project to **AndroidStudio**.



2. Open **AVD Manager**.

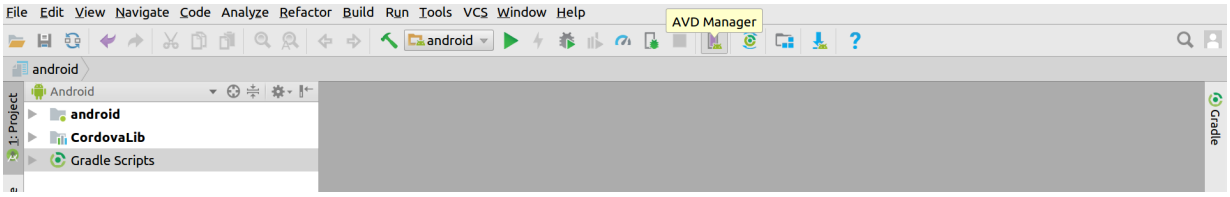


Figure 7: Importing project to AndroidStudio

3. Create a **Virtual Device**.

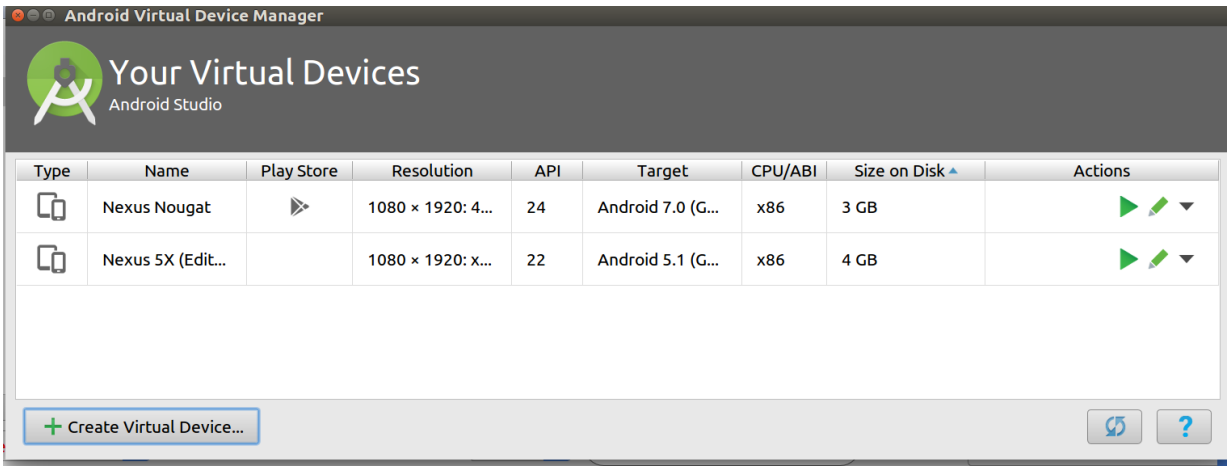


Figure 8: Open AVD Manager

4. Choose a device & click **next**.

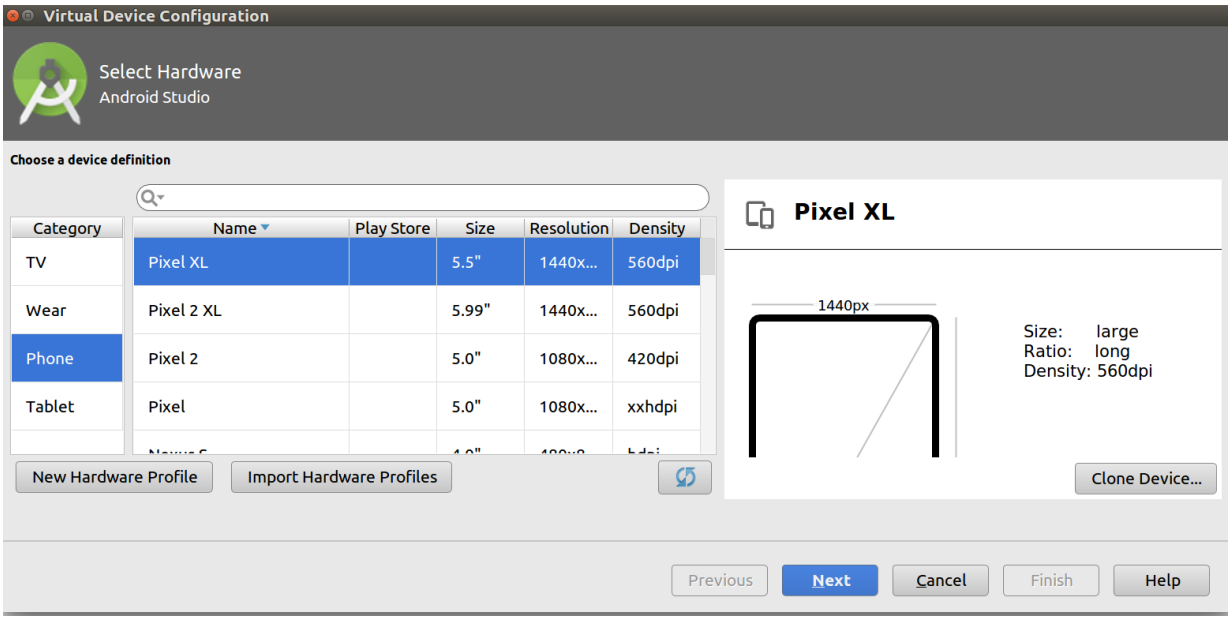


Figure 9: Creating a Virtual Device

5. Download the desired Android version (Recommend: testing with several versions, start with the latest one) & click **next**.

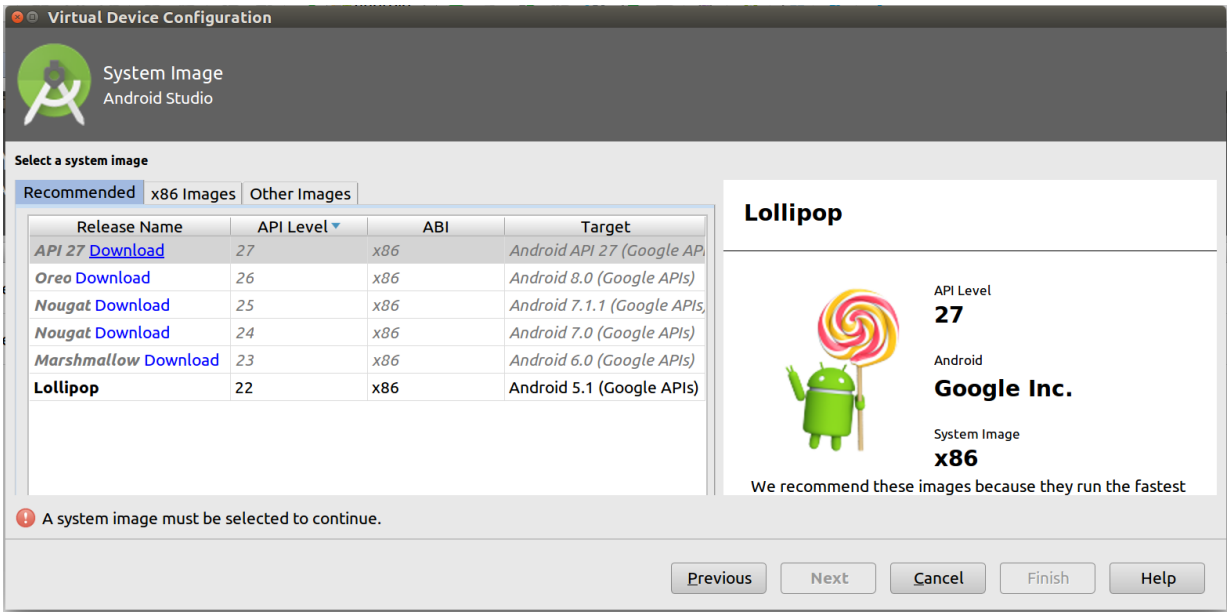


Figure 10: Downloading Android version

6. Give the device a name or leave the default one and click **finish**.

5.1.2 Run with AndroidStudio

After you have setup the emulator you can simply execute the project. Cordova creates a **webview** inside the native application. When testing with the emulator, the webview will try to access a URL, which is established by this command:

```
$ ionic serve
```

This command needs to be executed (inside the ionic project) beforehand. Ionic will expose a port (usually 8100, 8101,...) on the network, which can be accessed over the IP of the laptop running ionic, for example: **http://145.93.85.53:8100**.
The configuration of the URLs can be found in **config.xml** under **allow-navigation**:

config.xml

```
        <splash height="2732" src="resources/ios/splash/Default@2x~universal~anyany.png" width="2732" />
    </platform>
    <allow-navigation href="http://192.168.2.116:8101" />
    <allow-navigation href="http://145.93.85.25:8101" />
    <allow-navigation href="http://145.93.117.255:8100" />
    <allow-navigation href="http://192.168.2.116:8100" />
    <allow-navigation href="http://145.93.80.207:8100" />
    <allow-navigation href="http://192.168.2.116:8100" />
    <allow-navigation href="http://145.93.80.207:8100" />
    <allow-navigation href="http://145.93.84.155:8101" />
    <allow-navigation href="http://145.93.85.53:8101" />
    <allow-navigation href="http://145.93.85.53:8100" />
    <allow-navigation href="http://192.168.2.116:8102" />
    <engine name="ios" spec="~4.5.4" />
    <plugin name="cordova-plugin-firebase" spec="^0.1.24" />
    <plugin name="ionic-plugin-keyboard" spec="~2.2.1" />
    <plugin name="cordova-plugin-whitelist" spec="1.3.1" />
    <plugin name="cordova-plugin-statusbar" spec="git+https://github.com/apache/cordova-plugin-statusbar.git" />
    <plugin name="cordova-plugin-ionic-webview" spec="^1.1.11" />
    <plugin name="cordova-plugin-device" spec="^1.1.7" />
    <plugin name="cordova-plugin-background-mode" spec="^0.7.2" />
    <plugin name="cordova-plugin-fcm" spec="^2.1.2" />
</widget>
```

Figure 11: URL configuration

Sometimes Ionic has issues with generating the **config.xml**, which results in the application not being able to connect to the correct URL. Another disadvantage is, that the project in AndroidStudio is outdated during development. This forces the user to reopen the project and rerun the application. The next alternative is a stable method and offers more flexibility.

5.1.3 Run with command line

Ionic offers great features for testing during development. One of them being the **livereload**, which automatically updates the app on the phone during development.

To make use of this method we need to set some environment variables to be able to execute from the command line. You either need to install **gradle** or find the path of the gradle installation of **AndroidStudio**.

Windows Configuration:

- ANDROID_HOME : The path to where the Android SDK is installed
- PATH : Two entries, one where Tools are installed for the Android SDK, and another for platform tools
- GRADLE_HOME : The path to where gradle is installed

Open the command line and set the following variables:

```
1 set ANDROID_HOME=C:\<installation location>\sdk
2 set PATH=%PATH%;%ANDROID_HOME%\tools;%ANDROID_HOME%\platform-tools
3 set GRADLE_HOME=C:\<installation location>
```

Linux Configuration:

We need to set the environment variables at the bottom of the *.bashrc* file. This file is executed everytime the user logs in to the terminal. The Android-Path is usually the same on Linux machines, however you should make sure that the paths are correct. The gradle path and version might differ on your machine, make sure it's the correct one.

```
1 $ nano ~/.bashrc
```

Put these lines at the end of the file:

```
1 ANDROID_HOME="/home/user/Android/Sdk"
2 PATH="$PATH:$ANDROID_HOME/tools:/opt/gradle/gradle-4.3.1/bin"
3 export ANDROID_HOME
```

Apply changes afterwards:

```
1 $  source ~/.bashrc
```

OSX Configuration:

This step is similar to the linux configuration. All you need to do is create *.bash_profile*:

```
$ touch ~/.bash_profile
$ open -a TextEdit ~/.bash_profile
```

Now you can copy the variables:

```
# Create a JAVA_HOME variable, determined dynamically
export JAVA_HOME=$(/usr/libexec/java_home)
# Add that to the global PATH variable
export PATH=${JAVA_HOME}/bin:$PATH
# Set Android_HOME
export ANDROID_HOME=~/Library/Android/sdk/
# Add the Android SDK to the ANDROID_HOME variable
export PATH=$ANDROID_HOME/platform-tools:$PATH
export PATH=$ANDROID_HOME/tools:$PATH
#Set GRADLE_HOME
export GRADLE_HOME=/Library/gradle/gradle-3.2
export PATH=$PATH:$GRADLE_HOME/bin
```

Apply changes afterwards:

```
$  source ~/.bash_profile
```

Execution:

There are two kinds of deployments: one is for testing, which can be used with **livereload** and the other is for release.

Execute this in your command line:

```
$ adb devices
```

The output might look like this:

```
List of devices attached
adb server is out of date.  killing...
* daemon started successfully *
emulator-5554 device
00c760c78faf4821  device
```

As one can observe there is both an emulator and an actual device connected to the machine. For testing you can execute the following command:

```
$ ionic cordova run android -c -l --target 00c760c78faf4821
```

The *-c* stands for console-logs, *-l* for livereload and *–target* for the device id. With this approach you can simply disconnect your phone from your machine and the app will update the changes over the local network. This is only meant for testing and will not work outside of the network. You will also be able to see the outputs of *console.log* on the command line.

In order to deploy the app completely you can execute the following command:

```
$ ionic cordova run android --target 00c760c78faf4821
```

5.2 iOS

In order to deploy to an iOS device or emulator, you first need to download **Xcode**. Xcode might require you to register an Apple Developer ID. Sign up for the free version and login to Xcode with your new account. In order to deploy the on a device, you might need to upgrade your account to a provisioned one.
Further details on deployment can be found here https://ionicframework.com/docs/intro/deploying/.

First we need to build the Xcode-project with the following command in our ionic project:

```
1 $ ionic cordova build ios
```

Keep in mind that each time the code was changed the project must be reopened and re-executed.

1. Open Xcode and import the Xcode-project.

Figure 12: Importing Xcode project (1)

2. Navigate to the Xcode-project inside the Ionic project under *WTS/platforms/ios* .

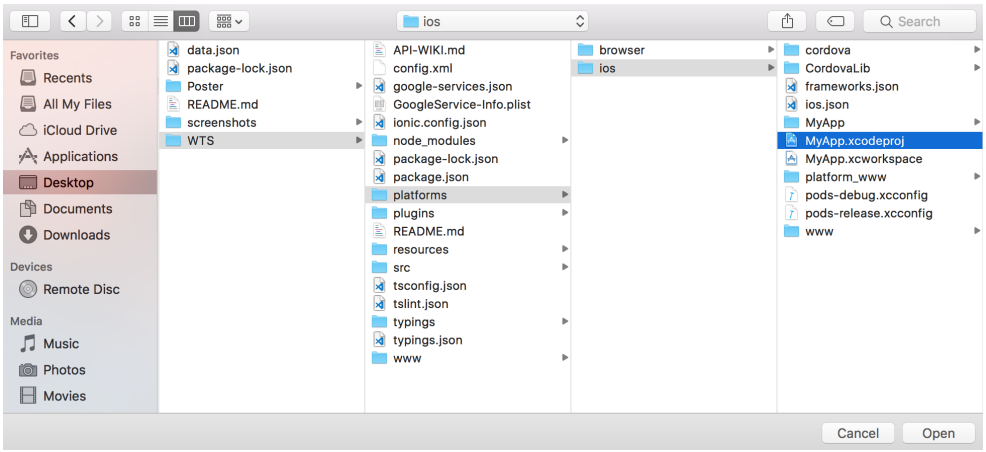


Figure 13: Importing Xcode project (2)

3. On the top bar you will see an iOS device.

Figure 14: Device selection (1)

4. If you click on the link, you will see a list of devices/simulators to choose from. Devices are listed at the top and simulators at the bottom.

Figure 15: Device selection (2)

5. Choose your device and click the play button to execute the app.

6 Prospect for the future

To have a fully working application of "Work-to-students", there are still some tasks to do:

1. Connecting the application to the WTS web application database
2. Display further information from that database
3. Push notifications for IOS
4. Editing pins on the map
5. Sortable message view design
6. List search autofill

After the training and the delivery of the project, the students are not longer responsible for the project. For getting further information, please contact the project owner:

Screenware GmbH & Co. KG
Thomas Müllers
tm@screenware.eu
Gewerbepark, Talstraße 27
41751 Viersen
Phone: +49 151 571 241 78
`https://www.screenware.eu/de/home/`

7 References

7.1 GitHub

GitHub Overview
https://github.com/

GitHub clone repository
https://help.github.com/articles/cloning-a-repository/

7.2 Ionic

Ionic Overview
https://ionicframework.com

Ionic Installation
https://ionicframework.com/docs/v1/guide/installation.html

Ionic Testing your app
https://ionicframework.com/docs/v1/guide/testing.html

Package Manager for JavaScript
https://www.npmjs.com

App Component and App Module
https://gonehybrid.com/exploring-app-module-and-app-component-in-an-ionic-2-app/

AngularJS ngModel
https://docs.angularjs.org/api/ng/directive/ngModel

7.3 Firebase

Firebase Overview
https://firebase.google.com/

Firebase Console
https://console.firebase.google.com

Firebase Database REST API
https://firebase.google.com/docs/reference/rest/database/

Firebase Push Notification with Ionic (Android example)
https://goo.gl/dDaXAC

Firebase Functions Guidelines
https://firebase.google.com/docs/functions/get-started

Firebase Functions Tutorials
https://goo.gl/Fn2Cqh

7.4 Android

AndroidStudio Installation
https://developer.android.com/studio/index.html

Gradle Installation
https://gradle.org/releases/

Configuration for Windows
https://ionicframework.com/docs/developer-resources/platform-setup/windows-setup.html

Configuration for Linux
https://hackernoon.com/how-to-run-ionic-application-in-android-using-ubuntu-78b500f7688f

Configuration for OSX
https://ionicframework.com/docs/developer-resources/platform-setup/mac-setup.html

7.5 iOS

Apple Developer ID
https://goo.gl/wnGmM4

Xcode Download
https://developer.apple.com/xcode/

Deployment
https://ionicframework.com/docs/intro/deploying/