JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

# Laborprotokoll

# Dezentrale Systeme - Mobile Access to Web Services


Systemtechnik Labor
5BHITT 2015/16, Gruppe B

Thomas Taschner


Version 1.0
Begonnen am 15. April 2016
Beendet am 22. April 2016

Note:
Betreuer: Michael Borko

# Inhaltsverzeichnis

# 1 Einführung

Diese Übung gibt einen Einblick in Entwicklungen von mobilen Applikationen.

## 1.1 Ziele

Das Ziel dieser Übung ist eine Anbindung einer mobilen Applikation an ein Webservices.

Die Anbindung soll mit Hilfe eines RESTful Webservice (Gruppe1) umgesetzt werden.

## 1.2 Voraussetzungen

- Grundlagen Java und XML
- Grundlegendes Verständnis über Entwicklungs- und Simulationsumgebungen
- Verständnis von RESTful Webservices

## 1.3 Aufgabenstellung

Es ist eine mobile Anwendung zu implementieren, die sich an das Webservice aus der Übung DezSysLabor-09 "Web Services in Javaänbinden soll. Dabei müssen die entwickelten Schnittstellen entsprechend angesprochen werden.

Es ist freigestellt, welche mobile Implementierungsumgebung dafür gewählt wird. Empfohlen wird aber eine Implementierung auf Android.

Bewertung: 16 Punkte

- Anbindung einer mobilen Applikation an die Webservice-Schnittstelle (6 Punkte)

  Registrierung von Benutzern (3 Punkte)
- Login und Anzeige einer Willkommensnachricht (3 Punkte)
- Simulation bzw. Deployment auf mobilem Gerät (2 Punkte)
- Protokoll (2 Punkte)

# 2 Ergebnisse

Zur Umsetzung dieser Übung wurde das Tutorial von Android Guru befolgt. Der Sourcecode wurde importiert, wobei noch einige Fehler behoben werden mussten. Die App kann nun mit der REST Schnittstelle aus DezSys09 kommunizieren. [3]

## 2.1 Erstellen der grafischen Oberfläche

Hierbei wurden primär die Dateien *home.xml*, *login.xml* und *register.xml* in *<projectroot>/app/src/main/res/layout* bearbeitet. So sieht die Datei *home.xml* aus. Zur einfacheren Generierung gibt es aber auch einen inkludierten YSIWYG-Editor.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.prgguru.example.HomeActivity"
    tools:ignore="MergeRootFrame" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="fill_vertical"
        android:layout_marginTop="20pt"
        android:gravity="center_horizontal"
        android:text="Welcome User"
        android:textSize="25dip" />

</FrameLayout>
```

Listing 1: home.xml

## 2.2 Anbindung einer mobilen Applikation an die Webservice-Schnittstelle

Das erste Problem trat beim Zugreifen auf localhost im Emulator auf. Nach einer kurzen Recherche im Internet konnte ein Post auf Stackoverflow weiterhelfen. [4] Im Simulator muss statt localhost die IP-Adresse 10.0.2.2 eingegeben werden.

Ein weiteres Problem war die Realisierung von DezSys09 mittels POST und JSON, während das Tutorial auf GET ausgelegt ist. In den Klassen LoginActivity und RegisterActivity muss noch das RequestParams Objekt auf ein JSONObject geändert werden.

```
ALT:
// Instantiate Http Request Param Object
RequestParams params = new RequestParams();

NEU:
JSONObject params = new JSONObject();
```

Listing 2: Ändern des RequestParams Objekts auf ein JSONObject

Nun muss das JSON-File noch mit angegebenen Daten befüllt werden.

```
StringEntity request = null;
try {
    request = new StringEntity(params.toString());
    request.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
}
```

Listing 3: Ändern des RequestParams Objekts auf ein JSONObject

Die Requestart wurde von GET und RequestParams auf POST und JSONObject geändert, die Variante der Verbindung muss noch von AsyncHttpClient geändert werden.

```
ALT:
AsyncHttpClient client = new AsyncHttpClient();
client.get("http://192.168.2.2:9999/useraccount/login/dologin",params ,new AsyncHttpResponseHandler()

NEU:
AsyncHttpClient client = new AsyncHttpClient(); client.post(this.getApplicationContext(), "http
    ://10.0.2.2:8080/register", request, "application/json", new TextHttpResponseHandler() {
```

Listing 4: Ändern des AsyncHttpClients

Nun muss die onSuccess-Methode geändert werden.

```
@Override
public void onSuccess(String response) {
// Hide Progress Dialog
prgDialog.hide();
try {
// JSON Object
JSONObject obj = new JSONObject(response);
// When the JSON response has status boolean value assigned with true
if(obj.getBoolean("status")){
Toast.makeText(getApplicationContext(), "You are successfully logged in!", Toast.LENGTH_LONG).show();
// Navigate to Home screen
navigatetoHomeActivity();
}
// Else display error message
else{
errorMsg.setText(obj.getString("error_msg"));
Toast.makeText(getApplicationContext(), obj.getString("error_msg"), Toast.LENGTH_LONG).show();
}
} catch (JSONException e) {
// TODO Auto-generated catch block
Toast.makeText(getApplicationContext(), "Error Occured [Server's JSON response might be invalid]!",
    Toast.LENGTH_LONG).show();
e.printStackTrace();
}
}
```

Listing 5: onSuccess Methode

Die Registrierung wurde entsprechend angepasst.

```
@Override
public void onSuccess(int statusCode, Header[] headers, String responseBody) {
    prgDialog.hide();
    // When Http response code is '201'
    if (statusCode == 201) {
        Toast.makeText(getApplicationContext(), responseBody, Toast.LENGTH_LONG).show();
        // Navigate to Home screen      navigatetoLoginActivity(findViewById(android.R.id.content));
            }
}
```

Listing 6: Neue Registrierung

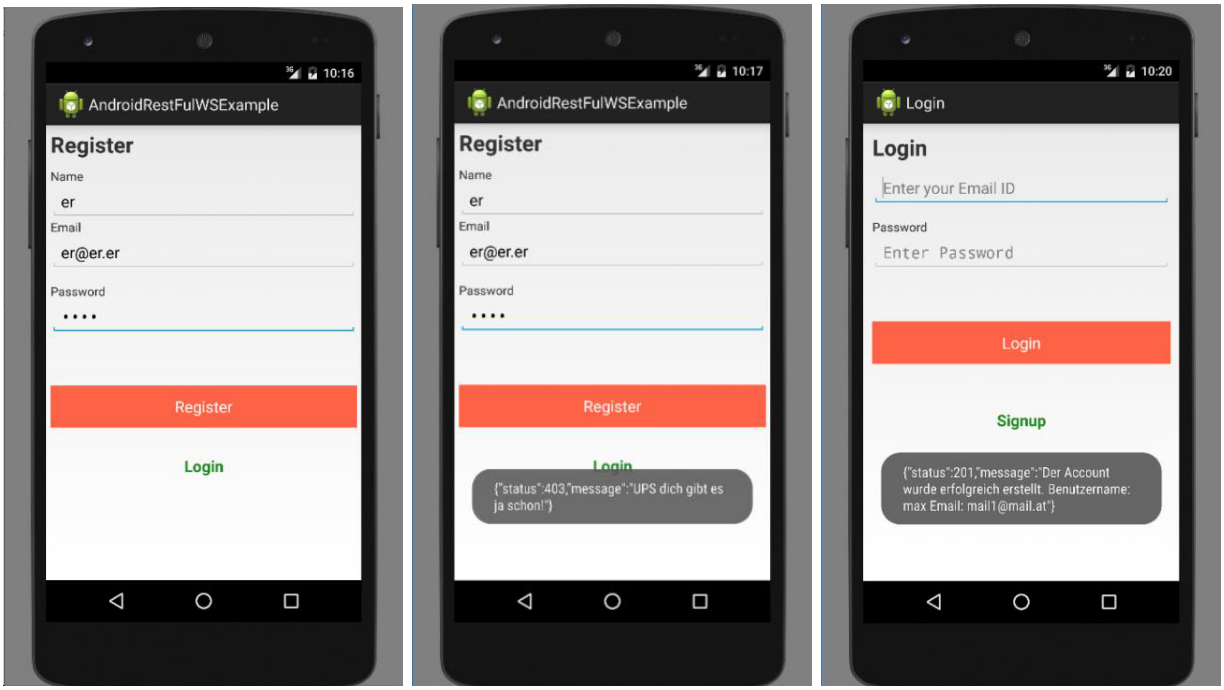
Der Login wurde entsprechend angepasst.

```
@Override
public void onSuccess(int statusCode, Header[] headers, String responseBody) {
    prgDialog.hide();
    // When Http response code is '200'
    if(statusCode == 200) {
        Toast.makeText(getApplicationContext(), responseBody, Toast.LENGTH_LONG).show();
        // Navigate to Home screen
        navigatetoHomeActivity();
    }
}
```

Listing 7: Neue Registrierung

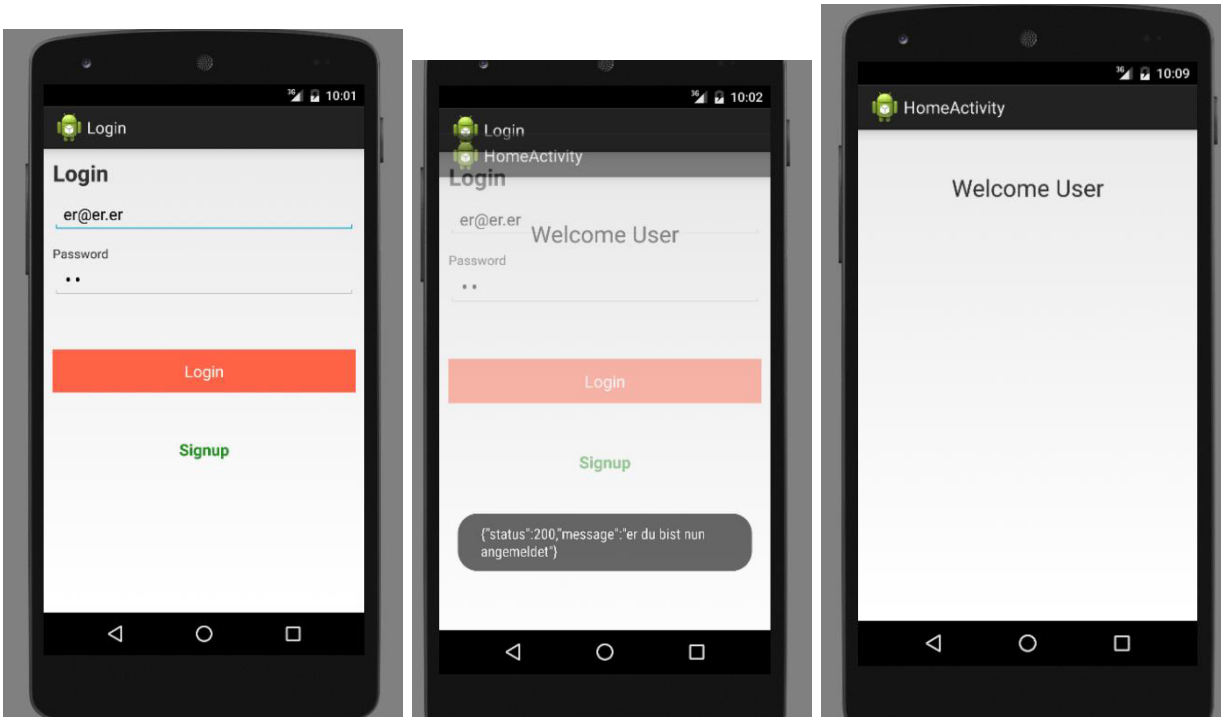
## 2.3 Registierung von Benutzern

Die konfigurierte Schnittstelle ermöglicht nun das Anlegen eines Benutzers.



## 2.4 Login und Anzeige einer Willkommensnachricht

Nachdem ein User in der Datenbank abgespeichert wurde, kann nun ein Login als dieser erfolgen.




## 2.5 Ausführen der App mittels Simulationsumgebung

Android Studio bringt neben den bereits genannten noch weitere nützliche integrierte Funktionen mit sich. Dazu gehört unter anderem die Simulation mittels Android Virtual Devices. Um die Applikation auszuführen muss lediglich der „Play-Button“ betätigt werden, sofern keine Run-Konfiguration ausgewählt ist. Sollte sich kein Fenster für den Emulator öffnen, muss die Run-Konfiguration „Android Application“ ausgewählt werden. Nach einem erfolgreichen Starten der Run-Konfiguration sollte die folgende Maske zu sehen sein.

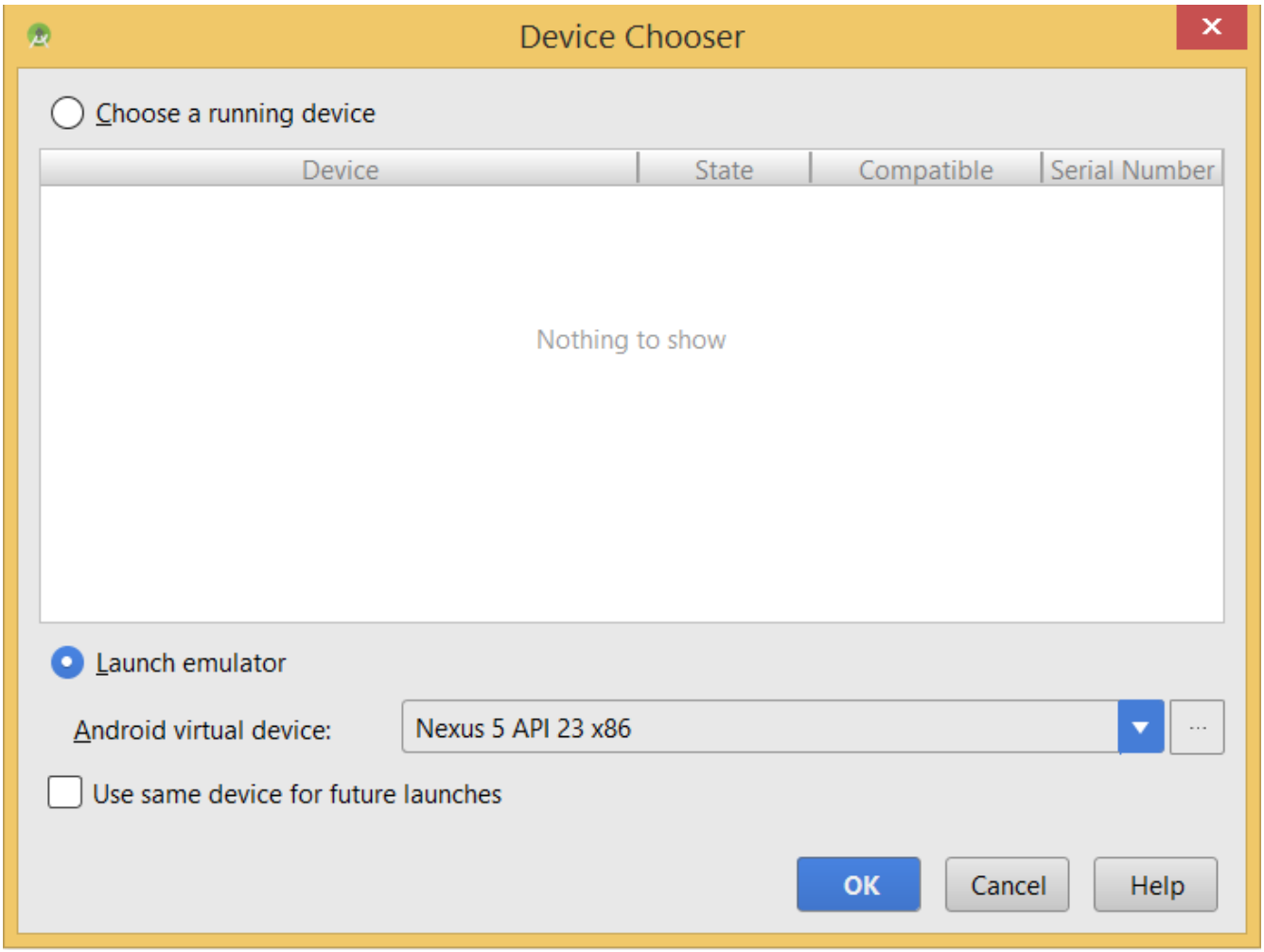



Abbildung 1: Device Chooser

Sollte beim Punkt „Android virtual device“ noch kein Gerät ausgewählt sein, muss dieses erst über den Button mit den drei Punkten („...“) hinzugefügt werden. Dort angelangt sollte man den Button „Create Virtual Device...“ auffinden. Dieser wiederum öffnet die folgende Maske.

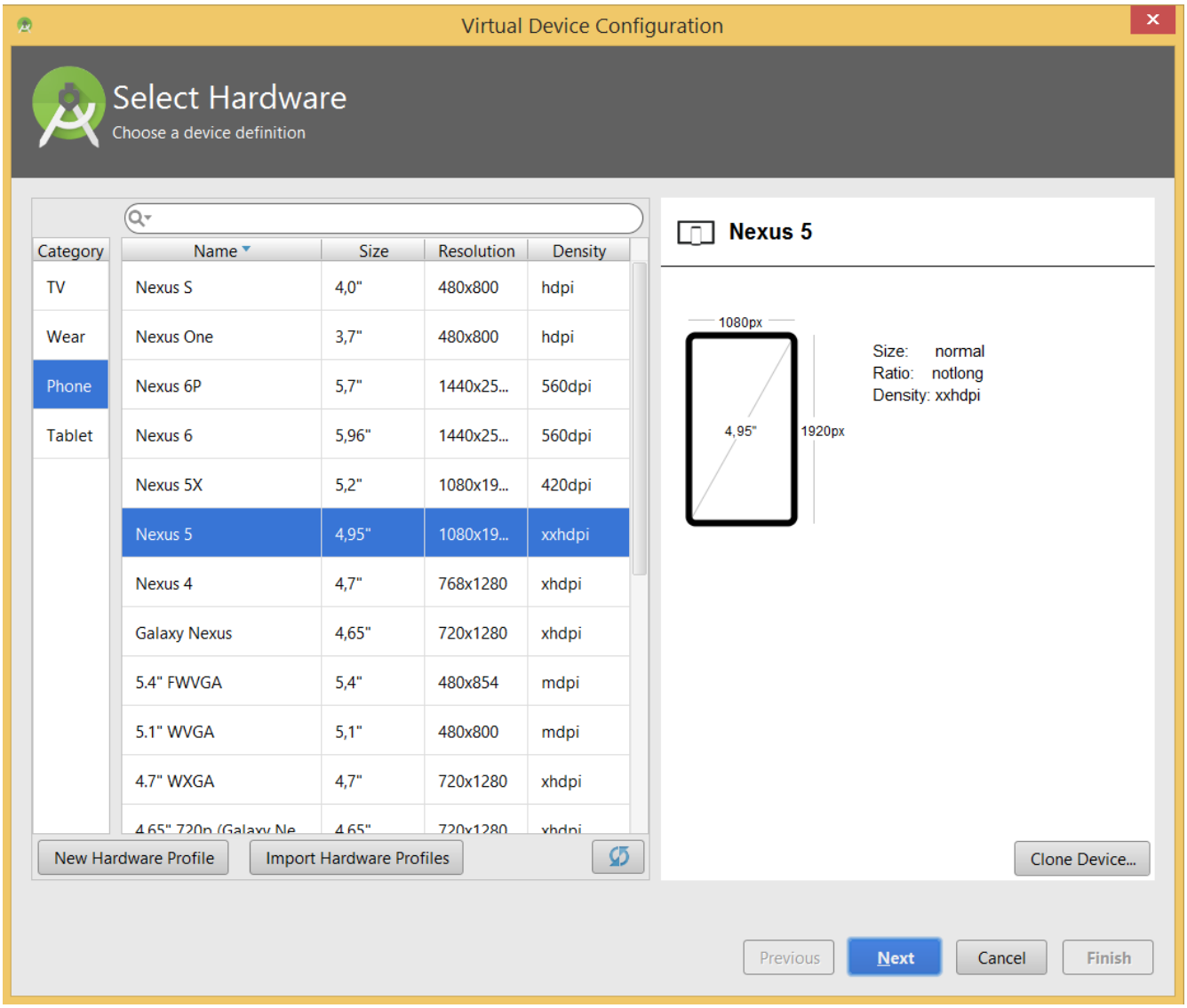



Abbildung 2: Neues Device erstellen

Nun kann das entsprechende Gerät aus der Liste ausgewählt werden. Durch einen Klick auf „Next“ kann das zu installierende Image ausgewählt werden und durch einen erneuten Klick auf „Next“ gelangt man zur Settings-Seite. Nun sollte dort eine wichtige Einstellung überprüft werden. Diese ist in den erweiterten Einstellungen zu finden, welche über den Button „Show Advanced Settings“ angezeigt werden können. Konkret handelt es sich um die Checkbox „Enable Keyboard Input“, welche angehakt sein sollte. Dadurch ist es möglich jegliche Eingabe auf der Tastatur an das virtuelle Device weiterzuleiten.

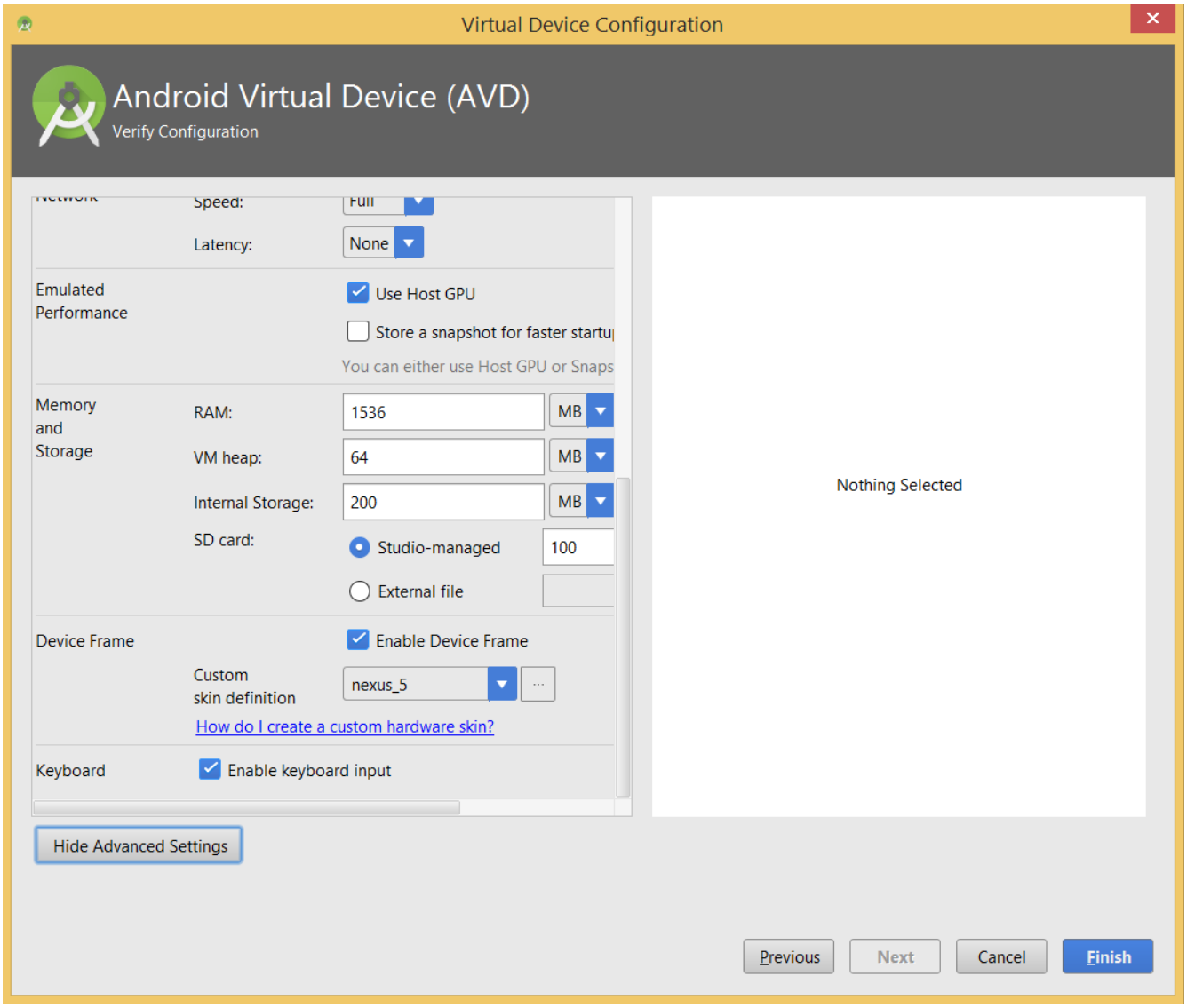



Abbildung 3: Keyboard Input aktivieren

Nun kann die Konfiguration über den Button „Finish“ abgeschlossen werden. Es sollte noch erwähnt werden, das Intel HAXM für das Ausführen des AVDs installiert sein muss. Dies wird allerdings meist bereits mit der Installation von Android Studio mitinstalliert. Wird die „Android Application“ Run-Konfiguration nun erneut ausgeführt, sollte das erstellte Device in der Liste angeführt sein. Nun startet das AVD und nach einem erfolgreichen Boot-Vorgang wird die Applikation automatisch deployed und gestartet.

## 2.6 Builden der Anwendung

Das Builden der Applikation ist sehr einfach, da das von Android Studio erstellte Projekt ohnehin bereits Gradle verwendet. Um eine APK-Datei zu erzeugen kann eine neue „Gradle“ Run-Konfiguration mit Android Studio mit den folgenden Einstellungen erzeugt werden.

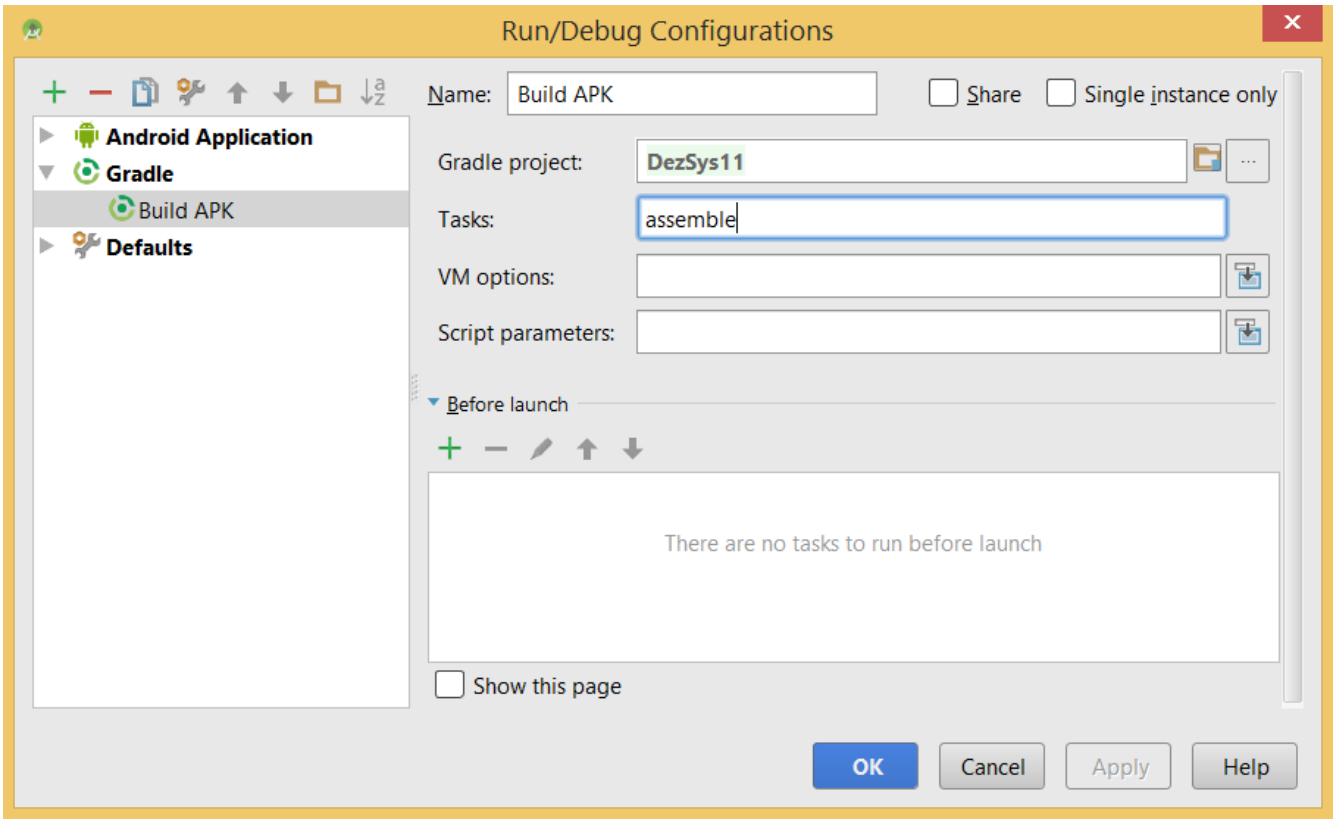



Abbildung 4: Run-Konfigurationen

Der Eintrag „Gradle project“ verweist dabei auf die Projektwurzel. (Project-Root) Die Erstellte Run-Konfiguration kann nun ausgeführt werden und erstellt das APK-File in den Ordner app/-build/outputs/apk.

## 2.7 Probleme

- Aufbauen einer Verbindung mit localhost, musste auf 10.0.2.2 geändert werden - Ändern von GET zu POST und JSON - Neue Methode für die Überprüfung eines erfolgreichen Logins - Asynchroner HTTP Client, da von GET zu POST und JSON gewechselt wurde

## 2.8 Lessons Learned

- Mit Hilfe des Tutorials wurde eine simple Android App erstellt - Verbindung zu einem Mobile Device von einem Web Service in Java über eine Schnittstelle - Vertiefung der Kenntnisse über die Kommunikation über Schnittstellen - Auffrischung des Umgangs mit JSON-Objekten

## 2.9 Link zum Repository

Die Abgabe wurde, wie üblich, auf GitHub zur Verfügung gestellt. [5]

# Literatur

[1] Android Guru. Android restful webservice tutorial – how to call restful webservice in android – part 3. online: `http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-call-restful-webservice-in-android-part-3/`. Zuletzt besucht: 22.04.2016.

[2] Paul Kalauner. Referenzimplementierung von dezsys09. online: `https://github.com/pkalauner-tgm/dezsys09-java-webservices`. Zuletzt besucht: 22.04.2016.

[3] Thomas Taschner. tgm-ttaschner/dezsys09. online: `https://github.com/tgm-ttaschner/DezSys09`. Zuletzt besucht: 22.04.2016.

[4] Stackoverflow. How to connect to my http://localhost web server from android emulator in eclipse. online: `http://stackoverflow.com/questions/5806220/how-to-connect-to-my-http-localhost-web-server-from-android-emulator-in-eclipse`. Zuletzt besucht: 22.04.2016.

[5] Thomas Taschner. tgm-ttaschner/dezsys11-mobile-access-to-web-services. online: `https://github.com/tgm-ttaschner/DezSys11-Mobile-Access-to-Web-Services`. Zuletzt besucht: 22.04.2016.

# Listings



# Abbildungsverzeichnis