JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll
Mobile Access to Web Services

Systemtechnik-Labor
5BHIT 2015/16, Gruppe B

Michael Weinberger 5BHIT

Version 1.0

Betreuer: Borko
Note:

Begonnen am 15.04.2016
Beendet am 21.04.2016

Inhaltsverzeichnis

Einführung

Diese Übung gibt einen Einblick in Entwicklungen von mobilen Applikationen.

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.

Voraussetzungen

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

Aufgabenstellung

Es ist eine mobile Anwendung zu implementieren, die sich an das Webservice aus der Übung DezSysLabor-09 "Web Services in Java" anbinden 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)

Ergebnisse

Im Zuge dieser Übung wurde das Tutorial von Android Guru verwendet. Der bereitgestellte Sourcecode musste noch angepasst werden, um mit DezSys09 kommunizieren zu können. [1, 4]

Anbindung einer mobilen Applikation an die Webservice-Schnittstelle

Diese Anwendung wurde als Android-App implementiert. Verwendet wurde das „Android Studio“ als IDE. Bei der Simulation muss statt Localhost die Adresse *10.0.2.2* angeben. DezSys 09 wurde mit POST und JSON realisiert, das Android Guru-Tutorial ist jedoch auf GET ausgelegt. In den bereitgestellten Klassen *LoginActivity* und *RegisterActivity* musste das RequestParams-Objekt umgeändert werden. Das JSON-File wird im Anschluss mit Daten befüllt. Auch die Verbindungsart des AsyncHttpClient muss auf POST und & JSON geändert werden.

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

```
//Fuellen des JSON-Files

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

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

In den xml-Dateien activity_login, activity_register und activity_success wird das gesamte Layout festgelegt. Beispielsweise:

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/prompt_email"
    android:inputType="textEmailAddress"
    android:maxLines="1"
    android:singleLine="true"/>

<Button
    android:id="@+id/submit_button"
    style="?android:textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:text="@string/action_sign_in"
    android:textStyle="bold"/>
```

Die Erstellung dieser XML-Dateien geschieht automatisch, es gibt einen bequemen WYSIWG-Editor.

Die onSuccess-Methode wird dementsprechend angepasst, je nach HTTP-Statuscode.
Registrieren:

```
@Override
public void onSuccess(int statusCode, Header[] headers, String
responseBody) {
    prgDialog.hide();
    // Wenn HTTP 201
    if (statusCode == 201) {
        Toast.makeText(getApplicationContext(), responseBody,
Toast.LENGTH_LONG).show();
        navigatetoLoginActivity(findViewById(android.R.id.content));
    }
}
```

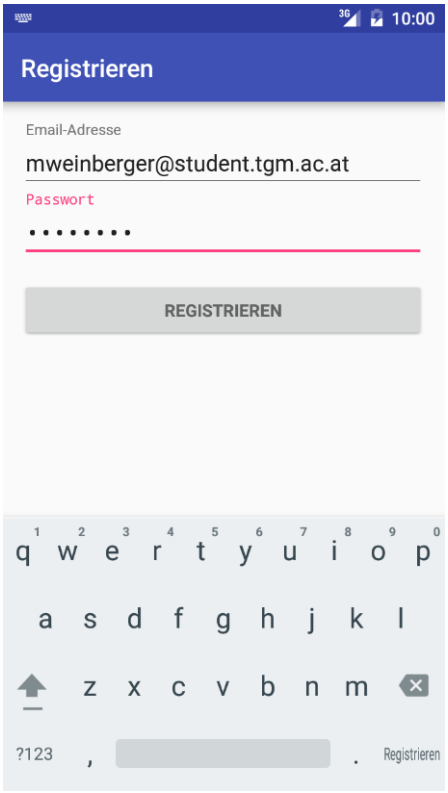
Login:

```
@Override
public void onSuccess(int statusCode, Header[] headers, String
responseBody) {
    prgDialog.hide();
    // Wenn HTTP 200
    if(statusCode == 200) {
        Toast.makeText(getApplicationContext(), responseBody,
Toast.LENGTH_LONG).show();
        navigatetoHomeActivity();
    }

}
```

Nach diesen Schritten hat es schlussendlich funktioniert.

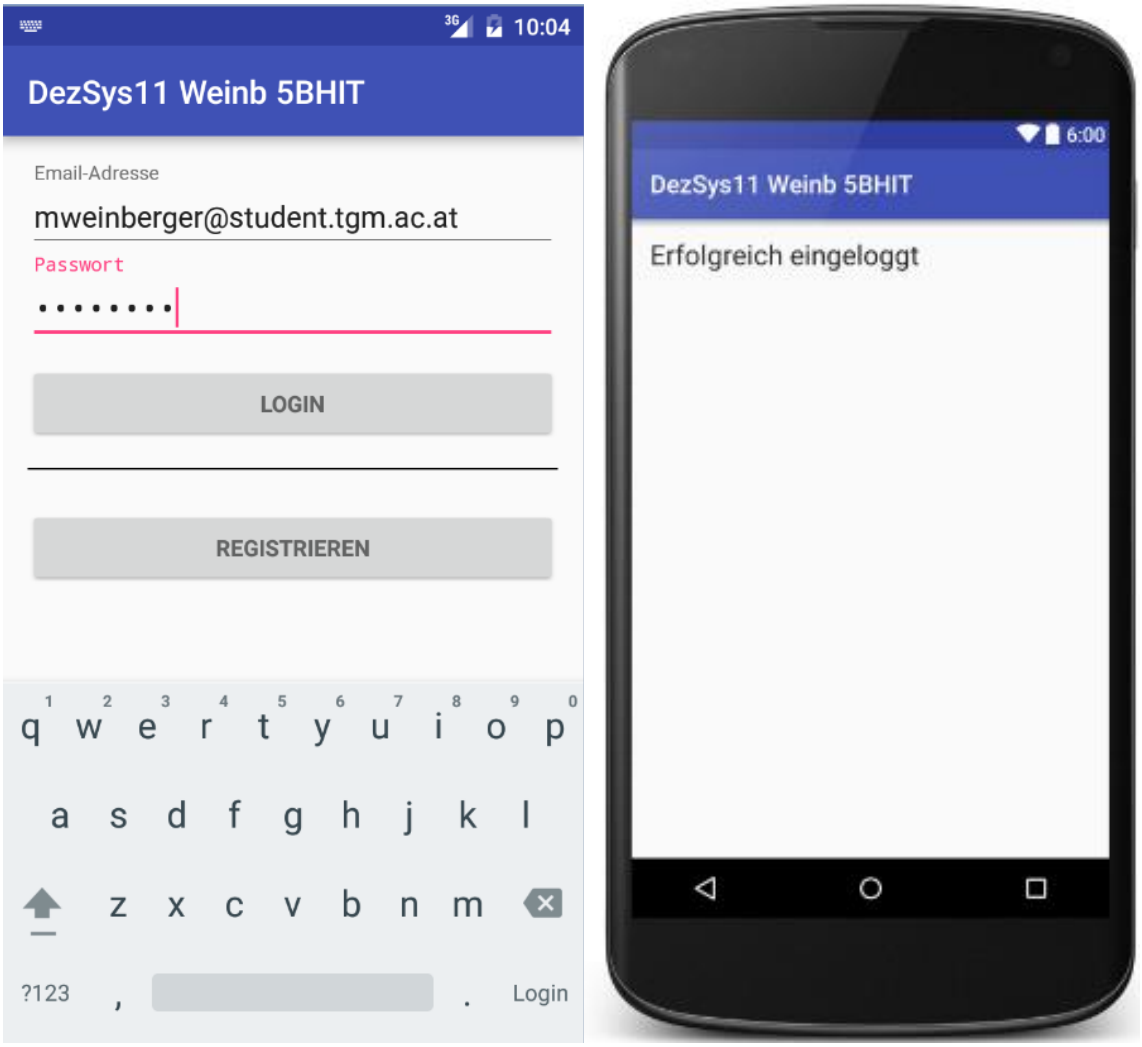
Registrierung von Benutzern



Der HTTP-Response wird als kurze Statusmeldung angezeigt. {„status“:200, „message“: “Registrierung erfolgreich“}

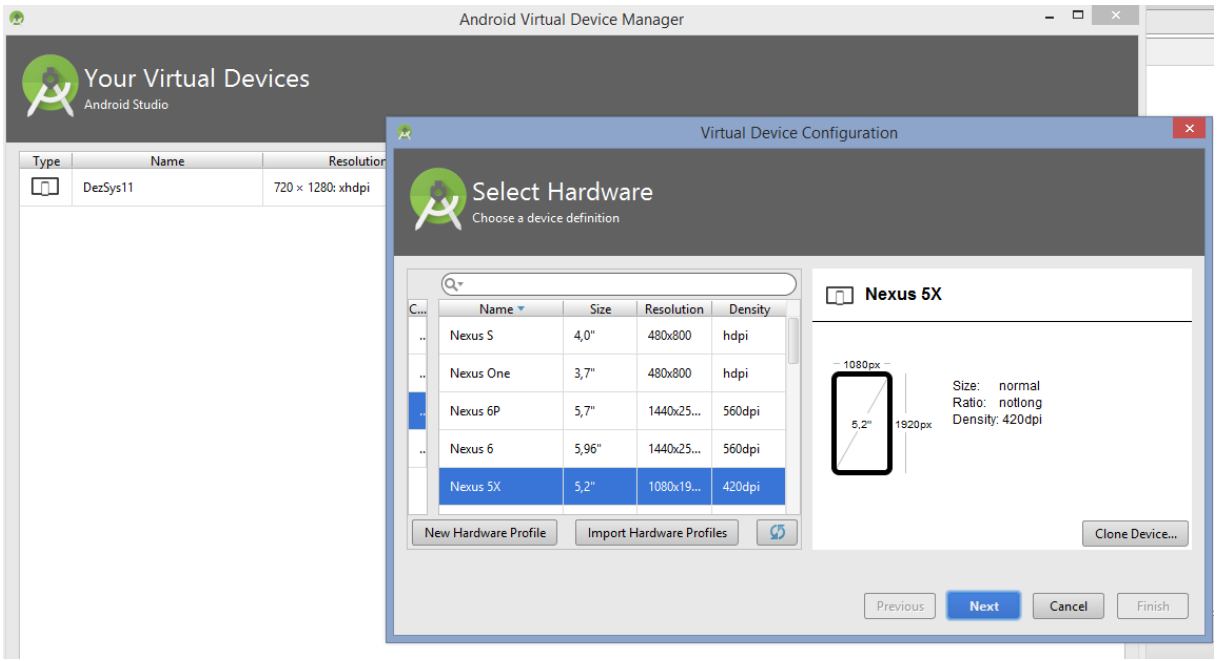
Login und Anzeige einer Willkommensnachricht

Nachdem der User in der Datenbank gespeichert wurde, lässt sich dieser auch anmelden. Die Willkommensmeldung wird nach der erfolgreichen Anmeldung angezeigt.

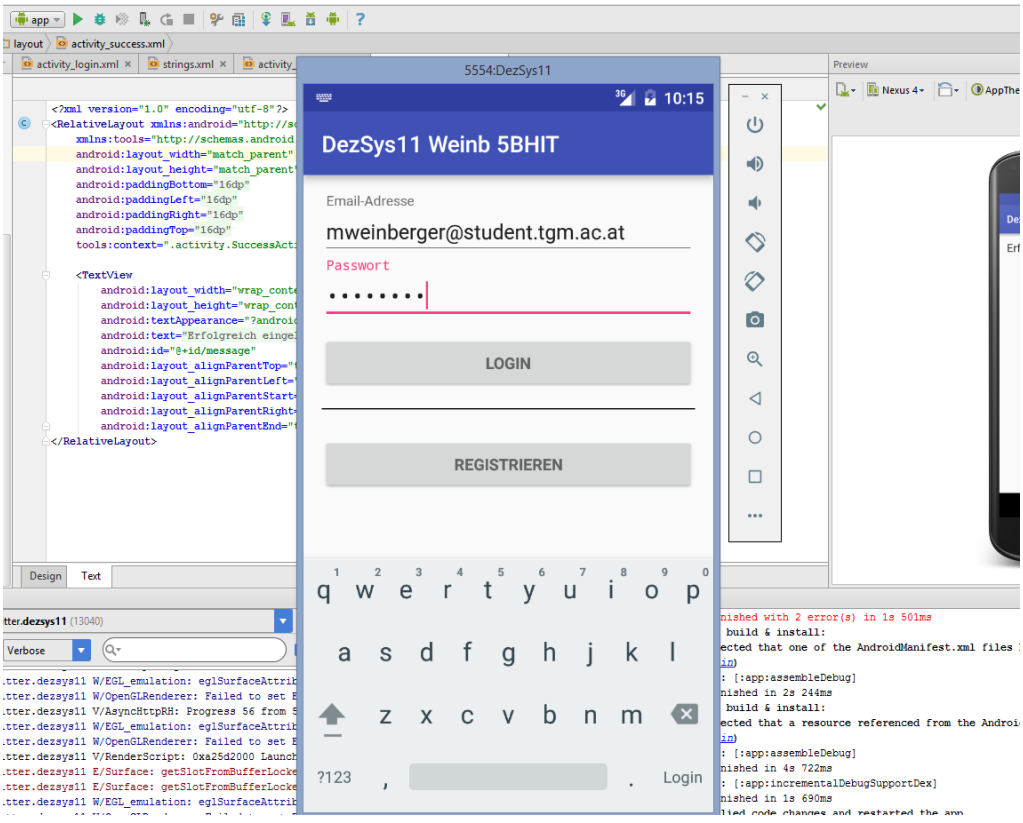


Simulation bzw. Deployment auf mobilem Gerät

Android Studio bietet brauchbare Funktionen wie etwa Android Virtual Devices, die im Laufe der Übung zum Einsatz kamen. Diese lassen sich durch einen leicht verständlichen Wizard erstellen. Bei einem Klick auf den Play-Button lässt sich die apk-Datei auf dieses virtuelle Betriebssystem deployen und sofort öffnen. Es benötigt lediglich etwas Zeit, das es zum Booten benötigt. Die häufigste Fehlermeldung war, dass das AVD zu wenig RAM hat, in der config.ini lässt sich dieses Problem sehr einfach umgehen.



Das von Android Studio erstellte Projekt verwendet ohnehin Gradle, im ‚Run/Debug Configuration' müssen nur Projekt und Task (‚assemble') ausgewählt werden. Jetzt ist das Deployment vollständig, und die App läuft in der virtuellen Umgebung.



Quellen

[1] "Android Restful Webservice Tutorial – How to call RESTful webservice in Android – Part 3"; Posted By Android Guru on May 27, 2014; online: http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-call-restful-webservice-in-android-part-3/
[2] "Referenzimplementierung von DezSys09"; Paul Kalauner; online: https://github.com/pkalauner-tgm/dezsys09-java-webservices
[3] „How to connect to my http://localhost web server from Android Emulator in Eclipse “; stackoverflow; online: http://stackoverflow.com/questions/5806220/how-to-connect-to-my-http-localhost-web-server-from-android-emulator-in-eclips
[4] “DezSys09”; mweinberger-tgm; https://github.com/mweinberger-tgm/DezSys09.git