

---

# **Laborprotokoll**

## **Mobile Access to Web Services**

---

**Systemtechnik Labor  
5BHITT 2015/16, Gruppe 1**

**Stefan Erceg**

**Version 2.0**

**Note:**

**Betreuer: Prof. Borko**

**Begonnen am 19. Februar 2016**

**Beendet am 22. April 2016**

## Inhaltsverzeichnis

1	Einführung .....	3
1.1	Ziele .....	3
1.2	Voraussetzungen .....	3
1.3	Aufgabenstellung .....	3
1.4	Bewertung .....	3
2	Ergebnisse .....	4
2.1	Android-Applikation erstellen .....	4
2.1.1	XML-Layouts erstellen .....	4
2.1.2	Activity-Klassen erstellen .....	4
2.2	Starten der Applikation .....	6
2.2.1	WAR-File von DezSys09 auf OpenShift deployen .....	6
2.2.2	Gerät im AVD-Manager hinzufügen .....	7
2.2.3	Testdurchlauf .....	10
2.2.4	Github-Repository .....	12
3	Zeitaufwand .....	12
4	Quellen .....	12

# 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 Webservice.

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 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.

## 1.4 Bewertung

- 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

Voraussetzung für diese Übung ist, dass die RESTful Webservice-Übung (DezSys09) bereits implementiert wurde. Die gesamte Übung wurde mit der IDE „Android Studio“ (Downloadlink: [1]) durchgeführt.

### 2.1 Android-Applikation erstellen

Um eine Android-Applikation umzusetzen, welches ein RESTful Webservice aufruft, wurde ein Tutorial [2] als Hilfestellung verwendet.

#### 2.1.1 XML-Layouts erstellen

Unter dem Ordner „/res/layout“ werden die Layouts zu den jeweiligen Seiten in XML-Form abgelegt. Bei dieser Übung existieren 3 Layouts:

- „register.xml“: Seite zum Registrieren - hierbei existieren 3 Eingabefelder (Name, E-Mail-Adresse und Passwort des Users)
- „login.xml“: Seite zum Einloggen – hierbei existieren 2 Eingabefelder (E-Mail-Adresse und Passwort des Users)
- „home.xml“: Seite, welche nach erfolgreichem Login angezeigt wird – Nachricht „Welcome User“ wird angezeigt

#### 2.1.2 Activity-Klassen erstellen

Zu jedem Layout existiert jeweils eine Activity-Klasse, welche von der Klasse „Activity“ erbt. Mit der Anweisung

```
setContentView(R.layout.<xmlfile>);
```

wird in der jeweiligen Klasse angegeben, welches Layout angezeigt werden soll.

Daraufhin holt man sich die Texte von den Eingabefeldern. Folgendermaßen setzt man dies beispielsweise beim Login-Layout für das Eingabefeld der E-Mail-Adresse um:

```
EditText emailET = (EditText)findViewById(R.id.loginEmail);  
String email = emailET.getText().toString();
```

Die „Utility“-Klasse existiert zusätzlich zu den 3 Activity-Klassen dazu, um zu überprüfen, ob eine valide E-Mail-Adresse angegeben wurde bzw. kein Eingabefeld leer ist. In der Klasse existiert unter anderem folgende Methode, welche überprüft, ob ein bestimmter String null ist und darauf true oder false zurückliefert:

```
public static boolean isNotNull(String txt){
    return txt!=null && txt.trim().length()>0 ? true: false;
}
```

Um das RESTful Webservice in die Übung einzubauen, wird zu Beginn ein Objekt vom Typen „AsyncHttpClient“ erstellt:

```
AsyncHttpClient client = new AsyncHttpClient();
```

Das Objekt versucht daraufhin eine bestimmte URL mit den ganzen angegebenen Parametern aufzurufen:

```
client.get("http://localhost:8080/dezsys09/login/dologin",params,
new AsyncHttpResponseHandler() { ... });
```

Die Parameter holt man sich davor folgendermaßen:

```
RequestParams params = new RequestParams();
params.put("username", email);
```

Falls die Seite erfolgreich aufgerufen werden konnte, wird ein JSON-Objekt erstellt, falls nicht wird eine entsprechende Fehlermeldung geworfen:

```
public void onSuccess(String response) {
    JSONObject obj = new JSONObject(response);
}
public void onFailure(int statusCode, Throwable error, String content) {
    Toast.makeText(getApplicationContext(), "Unexpected Error
    occurred!", Toast.LENGTH_LONG).show();
}
```

## 2.2 Starten der Applikation

### 2.2.1 WAR-File von DezSys09 auf OpenShift deployen

Damit das WAR-File, welches bei DezSys09 erstellt wird und auf einem Tomcat-Server läuft, auf OpenShift deployt werden kann, musste man sich zu Beginn auf der offiziellen OpenShift-Seite [3] mit einem Account registrieren. Daraufhin musste man das Tomcat Maven-Plugin, bei welchem man unter anderem die URL, den Usernamen und das Passwort angibt, zum pom.xml – File von DezSys09 hinzufügen:

```
<plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
    <url>https://useraccount-athc.rhcloud.com/manager/text</url>
    <server>TomcatServer</server>
    <path>/dezs09</path>
    <username>deploy</username>
    <password>DPfT0i2016</password>
  </configuration>
</plugin>
```

In den Java-Klassen „RegisterActivity“ und „LoginActivity“ wird auf folgende URLs zugegriffen:

URL zum Login:

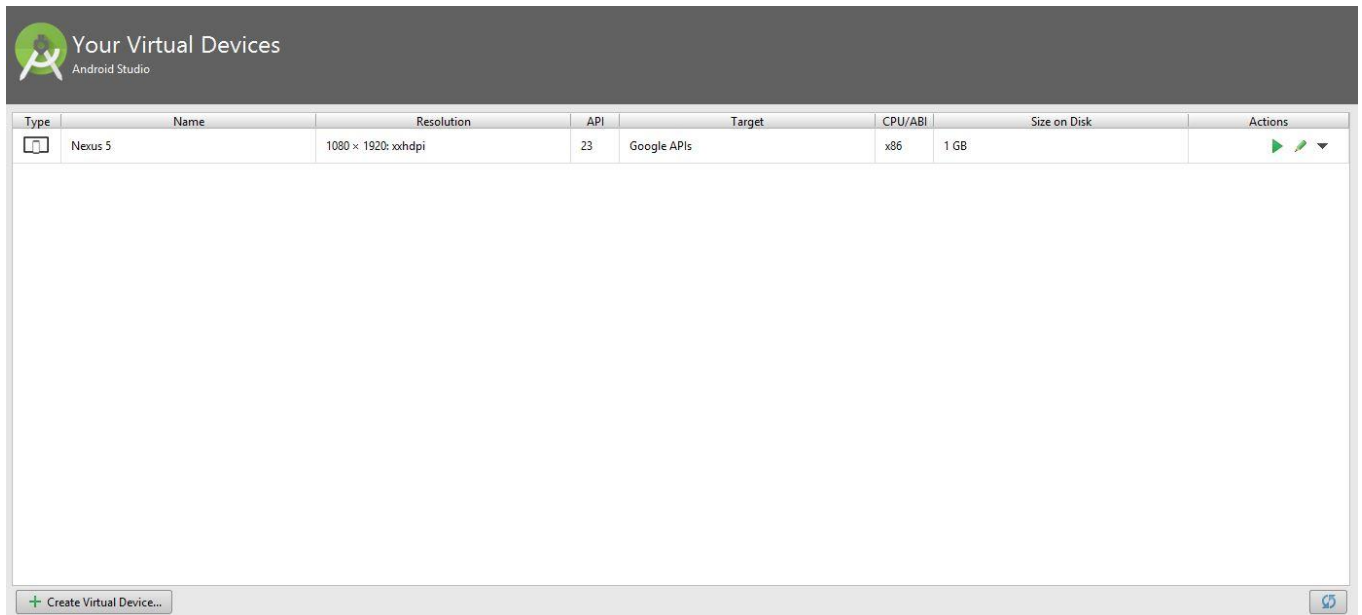
<https://useraccount-athc.rhcloud.com/dezs09/login/dologin>

URL zum Registrieren:

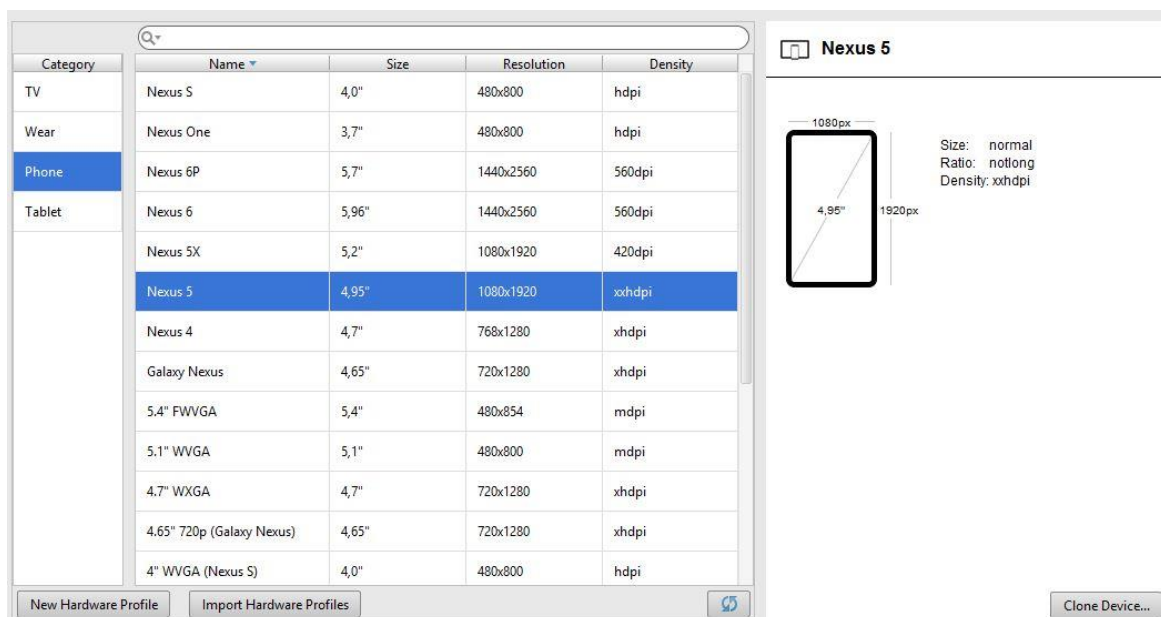
<https://useraccount-athc.rhcloud.com/dezs09/register/doregister>

## 2.2.2 Gerät im AVD-Manager hinzufügen

Damit die Android-Applikation auf einem Emulator getestet werden kann, muss ein Gerät im Android Virtual Device Manager hinzugefügt werden. Diesen öffnet man in Android Studio indem man auf „Tools → Android → AVD Manager“ klickt. Nachdem man daraufhin auf „Create Virtual Device“ gedrückt hat, kann man eine Hardware (Handys, Tablets, etc.) auswählen.



Ich habe das Handy „Nexus 5“ angeklickt und als System-Image Android 6.0 definiert:




Release Name	API Level	ABI	Target
Marshmallow	23	x86	Android 6.0 (with Google APIs)
KitKat	19	armeabi-v7a	Android 4.4
KitKat	19	armeabi-v7a	Android 4.4
KitKat	19	x86	Android 4.4

☐ Show downloadable system images

Refreshing...

**Marshmallow**



API Level  
**23**

Android  
**6.0**

Google Inc.

System Image  
**x86**

Questions on API level?  
See the [API level distribution chart](#)

Einige Einstellungen, wie zum Beispiel, ob das Gerät im Hoch- oder Querformat angezeigt bzw. wie viel RAM dem virtuellen Gerät zugewiesen werden soll, konnten zusätzlich angegeben werden:

Startup size and orientation

Scale: Auto

Orientation: Portrait Landscape

Camera

Front: None

Back: None

Network

Speed: Full

Latency: None

Emulated Performance

☒ Use Host GPU

☐ Store a snapshot for faster startup

You can either use Host GPU or Snapshots

Memory and Storage

RAM: 1536 MB

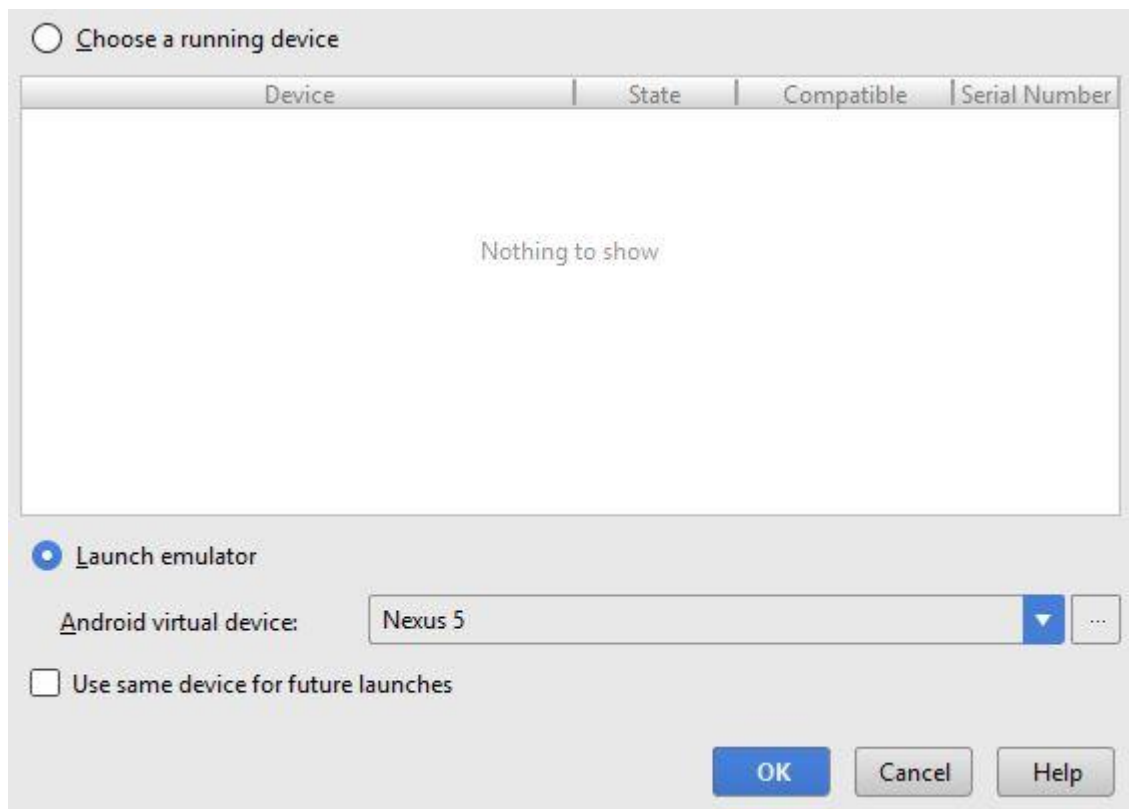
VM heap: 64 MB

Internal Storage: 200 MB

SD card: Studio-managed 100 MB

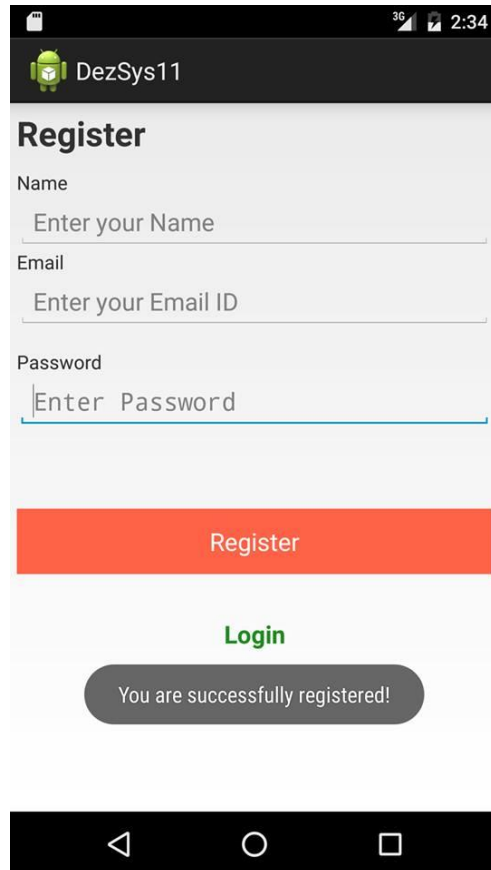
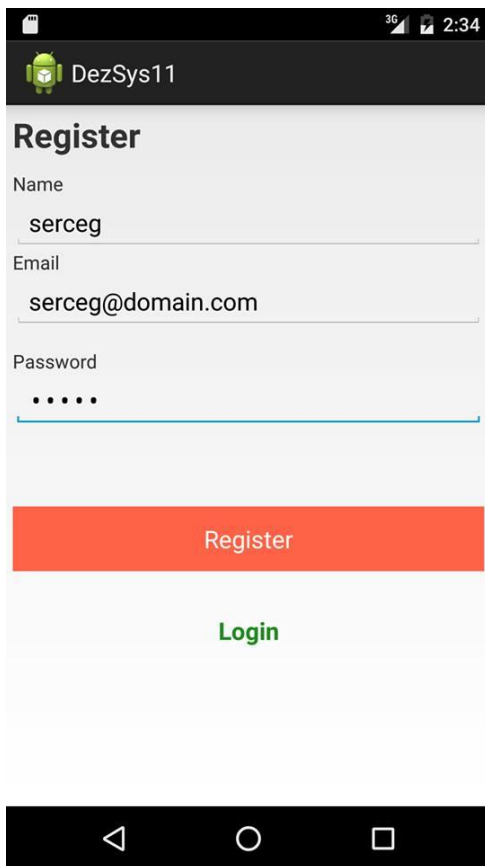


Indem die Applikation per Play-Button gestartet und als Emulator Nexus 5 ausgewählt wurde, wird die Android-Applikation auf dem Gerät bereits geöffnet und man kann sich registrieren bzw. einloggen:

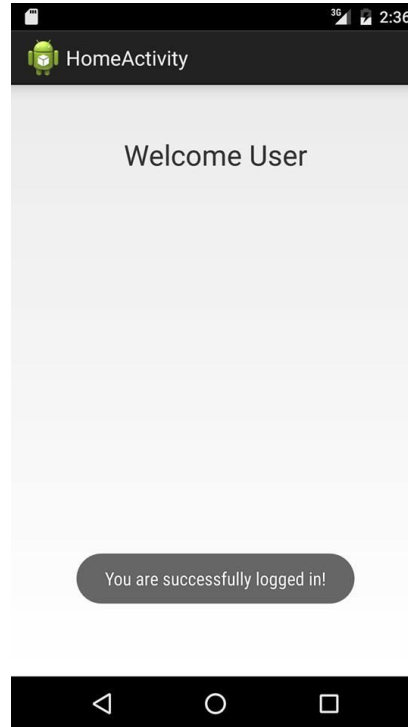
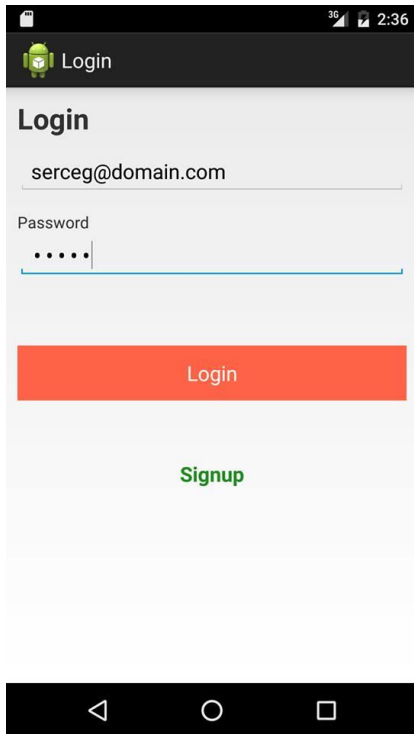


## 2.2.3 Testdurchlauf

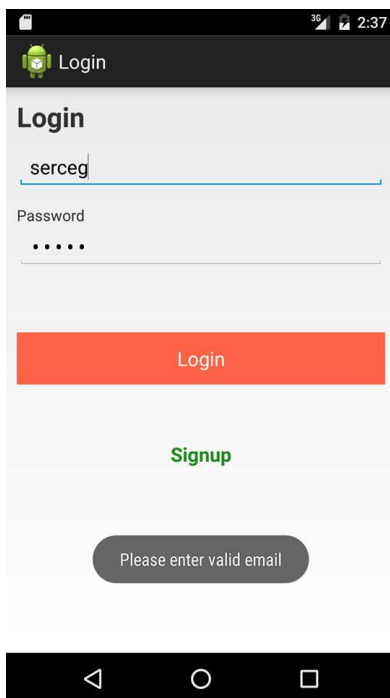
Nachdem die App im Emulator geöffnet wurde, wird zu Beginn die Registrier-Seite angezeigt. Dabei wird als Name „serceg“, als E-Mail-Adresse „serceg@domain.com“ und als Passwort „12345“ angegeben. Daraufhin wird eine Erfolgsmeldung angezeigt:



Daraufhin kann man sich mit dem gerade registrierten User einloggen. Nach einer erfolgreichen Anmeldung wird man auf die Willkommensseite weitergeleitet:



Bei falschen Eingaben (z.B. invalide E-Mail-Adresse) wird eine entsprechende Fehlermeldung geworfen:



## 2.2.4 Github-Repository

Mein Github-Repository, welches unter anderem den Source-Code und die notwendigen XML-Files für das Layout der Applikation enthält, ist unter folgender URL aufrufbar: [4].

## 3 Zeitaufwand

Arbeit	Datum	Zeit
AndroidStudio aufgesetzt und entsprechend vorbereitet	19.02.2016	2 h 30 min
Tutorial zum Laufen gebraucht	25.02.2016	1 h 30 min
WAR-File von DezSys09 auf OpenShift deployt	26.02.2016	1 h
Gesamt	26.02.2016	<b>5 h</b>

## 4 Quellen

- [1] Creative Commons Attribution 2.5 (2016).  
Download Android Studio [Online].  
<http://developer.android.com/sdk/index.html>  
[zuletzt abgerufen am 25.02.2016]
- [2] Android Guru (Mai 2014). Android Restful Webservice Tutorial – How to call RESTful webservice in Android – Part 3 [Online]. Available at:  
<http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-call-restful-webservice-in-android-part-3/>  
[zuletzt abgerufen am 25.02.2016]
- [3] redhat (2016). OpenShift Online by RedHat [Online].  
Available at: <https://www.openshift.com/>  
[zuletzt abgerufen am 26.02.2016]
- [4] Stefan Erceg (Februar 2016). Github-Repository  
„DezSys11\_MobileAccessToWebServices“ [Online]. Available at:  
[https://github.com/serceg-tgm/DezSys11\\_MobileAccessToWebServices](https://github.com/serceg-tgm/DezSys11_MobileAccessToWebServices)  
[zuletzt abgerufen am 26.02.2016]