
Laborprotokoll

DezSys11 – Mobile Access to Web Services

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

Alexander Kölbl

Note:

Betreuer: Prof. Borko

Version 1.0

Begonnen am 15. April 2016

Beendet am 21. 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	Activities (Grafische Oberfläche).....	4
2.2	Activities (Controller)	6
2.3	Ausführen der Applikation mittels Emulator	9
2.4	Funktionstest	11
3	Zeitaufwand	15
4	Repository.....	15
5	Quellen.....	15

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 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

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

Die Anwendung wurde als Android App in der Entwicklungsumgebung Android Studio umgesetzt. Diese IDE bietet zusätzlich einen Emulator für Android Geräte an, um so den entwickelten Code zu testen. Als Anhaltspunkt wurde das Tutorial "Android Restful Webservice Tutorial – How to call RESTful webservice in Android" [1] verwendet. Benötigt wird eine App mit User Registrier- und Loginoberfläche sowie eine Home-Oberfläche bei erfolgreichem Login. Diese App soll auf den in DezSys09 erstellten Web Service zugreifen, um so E-Mail Adressen und Passwörter der erstellten Benutzer zu speichern und beim Login abzufragen.

2.1 Activities (Grafische Oberfläche)

So genannte Activities stellen in Android Studio die einzelnen Funktionalitäten der App in einer grafischen Oberfläche dar. Für sämtliche benötigten Activities der Aufgabe grafische Oberflächen entwickelt. Diese Layouts werden als XML-Dateien persistiert. Die erstellten Designs können jederzeit eingesehen werden.

Login Layout File & Designansicht:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="10dip" >

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dip"
            android:text="@string/login_title"
            android:textSize="25sp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/loginEmail"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:hint="Enter your Email ID"
            android:inputType="textEmailAddress" />

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dip"
```

```
        android:text="@string/pwd" />

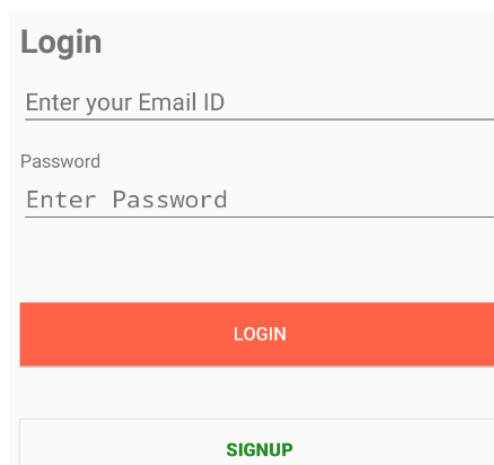
<EditText
    android:id="@+id/loginPassword"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="Enter Password"
    android:inputType="textPassword" />

<TextView
    android:id="@+id/login_error"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="10dip"
    android:textColor="#e30000"
    android:textStyle="bold" />

<Button
    android:id="@+id/btnLogin"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dip"
    android:background="#ff6347"
    android:onClick="loginUser"
    android:text="Login"
    android:textColor="#fff" />

<Button
    android:id="@+id/btnLinkToRegisterScreen"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dip"
    android:background="@null"
    android:onClick="navigatetoRegisterActivity"
    android:text="Signup"
    android:textColor="#228b22"
    android:textStyle="bold" />
</LinearLayout>

</ScrollView>
```



Android Studio verfügt außerdem über vorgefertigte Activities, darunter auch Activities für Login und Registrierung. Zusätzlich zu dem Layout File wird bei den vorgefertigten Activities auch ein Controller File, in der die Logik der grafischen Oberfläche implementiert wird, erzeugt.

Benötigten Activities und deren Funktionalität:

RegisterActivity: Registrierungsfläche (E-Mail Adresse & Passwort),
Userregistrierung über Web Service & Weiterleitung auf LoginActivity

LoginActivity: Loginfläche (E-Mail Adresse & Passwort), Userdatenkontrolle
über Web Service & Weiterleitung auf HomeActivity bei erfolgreichem Login

HomeActivity: Homebildschirm der App nach erfolgreichem Login

2.2 Activities (Controller)

In den Controller Klassen wird mittels der Android-Async-http Bibliothek auf den Web Service zugegriffen. In der Referenzimplementierung ist ein beispielhafter Zugriff auf einen Web Service implementiert. Der angegebene Web Services der Referenzimplementierung muss auf den selbst erstellten Web Service umgeändert werden. Die eingegebenen Daten (E-Mail Adresse & Passwort) werden als JSON-Objekt gespeichert und mittels POST-Anfrage an den Web Service gesendet. Zugriff auf den selbst erstellten Web Service erfolgt über die IP Adresse `http://10.0.2.2:8080/login` bzw.

`http://10.0.2.2:8080/register`. Die IP-Adresse 10.0.2.2 entspricht der localhost-Adresse meines Rechners, auf dem der Web Service läuft. Wenn anstelle der Adresse 10.0.2.2 localhost angegeben wäre, würde auf dem Android Emulator nach dem Web Service gesucht werden.

Methode, welche die Anfragen an den Web Service versendet:

```
public void invokeWS(JSONObject params){
    // Show Progress Dialog
    progressDialog.show();
    // Make RESTful webservice call using AsyncHttpClient object
    AsyncHttpClient client = new AsyncHttpClient();
    StringEntity request = null;
    try {
        request = new StringEntity(params.toString());
        request.setContentType(new BasicHeader(HTTP.CONTENT_TYPE,
"application/json"));
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    client.post(this.getContext(), "http://10.0.2.2:8080/login",
request, "application/json", new TextHttpResponseHandler() {
```

```

    /**
     * Method, when the response has the http response code 200
     */
    @Override
    public void onSuccess(int statusCode, Header[] headers, String
responseBody) {
        prgDialog.hide();
        if(statusCode == 200) {
            Toast.makeText(getApplicationContext(), responseBody,
Toast.LENGTH_LONG).show();

            // Navigate to Home screen
            navigatetoHomeActivity(responseBody);
        }
    }

    /**
     * Method, when the response has not the http response code 200
     */
    @Override
    public void onFailure(int statusCode, Header[] headers, String
responseBody, Throwable error) {
        prgDialog.hide();
        // When the response has the http response code 403
        if(statusCode == 403){
            Toast.makeText(getApplicationContext(), responseBody,
Toast.LENGTH_LONG).show();
        }
        // When Http response code is other than 403
        else{
            Toast.makeText(getApplicationContext(), "Error occcured!",
Toast.LENGTH_LONG).show();
        }
    }
});
}

```

Login Methode:

```

public void loginUser(View view){
    // Get Email Edit View Value
    String email = emailET.getText().toString();
    // Get Password Edit View Value
    String password = pwdET.getText().toString();
    // Instantiate Http Request Param Object
    JSONObject params = new JSONObject();
    // When Email Edit View and Password Edit View have values other than Null
    if(Utility.isNotNull(email) && Utility.isNotNull(password)){
        // When Email entered is Valid
        if(Utility.validate(email)){
            try {
                // Put Http parameter username with value of Email Edit View
control
                params.put("email", email);
                // Put Http parameter password with value of Password Edit Value

```

```

control
    params.put("password", password);
    // Invoke RESTful Web Service with Http parameters
    invokeWS(params);
} catch (JSONException e) {
    e.printStackTrace();
}
}
// When Email is invalid
else{
    Toast.makeText(getApplicationContext(), "Please enter an valid
email", Toast.LENGTH_LONG).show();
}
}
// When any of the Edit View control left blank
else{
    Toast.makeText(getApplicationContext(), "Don't leave any fields blank",
Toast.LENGTH_LONG).show();
}
}
}

```

In der Login Methode werden die über die grafische Oberfläche eingegebenen Daten (E-Mail Adresse & Passwort) als JSON Objekt gespeichert und kontrolliert. Falls die Daten den Konventionen entsprechen (korrekte E-Mail Adresse, leer gelassene Felder) werden diese über die Methode „invokeWS“ an den Web Service versendet. Wenn der Login erfolgreich war, wird der Benutzer zu der Home Oberfläche weitergeleitet, ansonsten wird eine Fehlermeldung ausgegeben.

Weiterleitung auf eine andere Activity:

```

/**
 * Method which navigates from Login Activity to Home Activity
 */
public void navigatetoHomeActivity(String response){
    Intent homeIntent = new Intent(getApplicationContext(),HomeActivity.class);
    homeIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    startActivity(homeIntent);
}

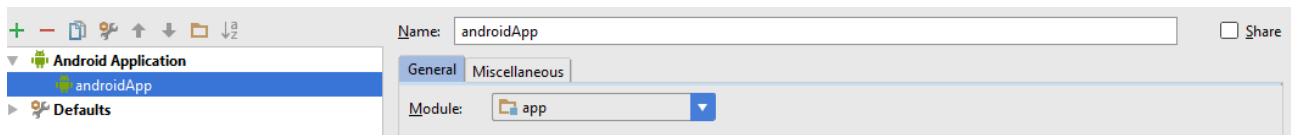
/**
 * Method gets triggered when Register button is clicked
 * @param view
 */
public void navigatetoRegisterActivity(View view){
    Intent loginIntent = new
Intent(getApplicationContext(),RegisterActivity.class);
    loginIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    startActivity(loginIntent);
}

```

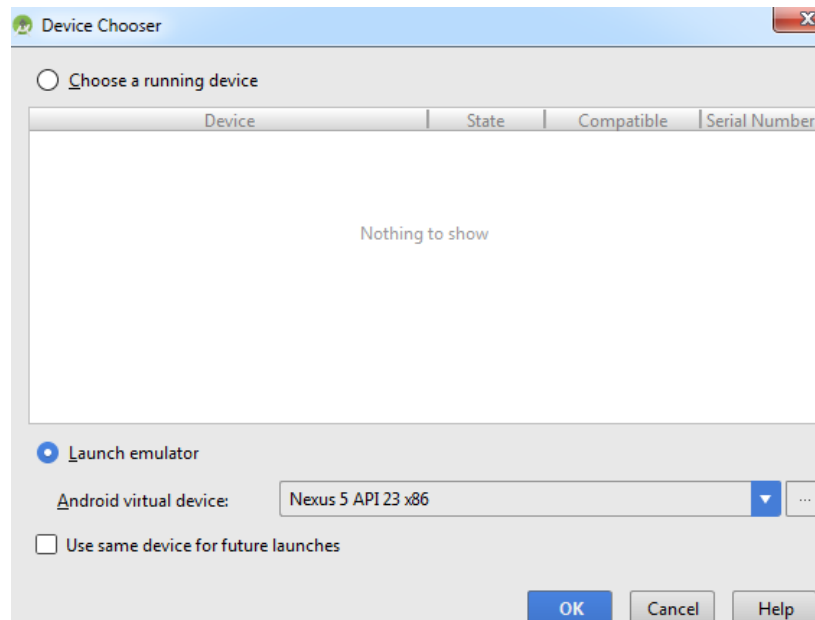

2.3 Ausführen der Applikation mittels Emulator

Da ich selber kein Android Gerät besitze, wird die Applikation mittels des mitgelieferten Emulators von Android Studio ausgeführt. Dieser Emulator erzeugt Android Virtual Devices.

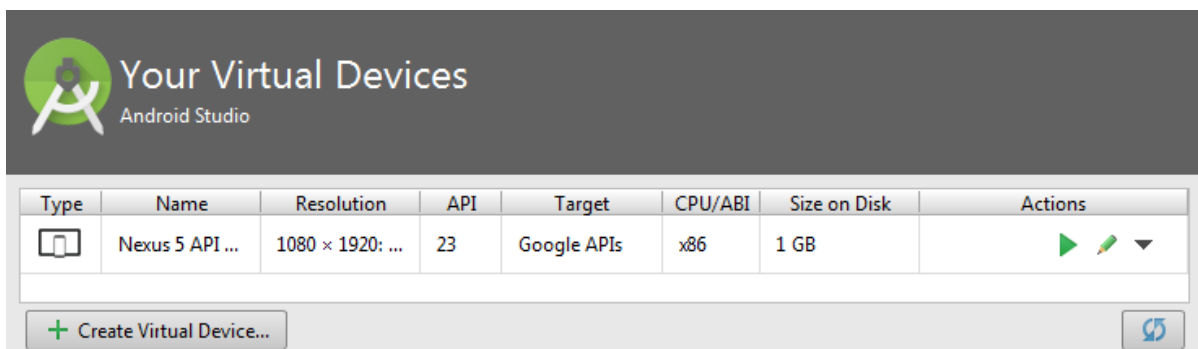
Über die Run Configuration kann man bei dem Punkt Android Application eine neue Konfiguration erstellen und muss bei dem Punkt „Module“ den Ordner „app“ auswählen.



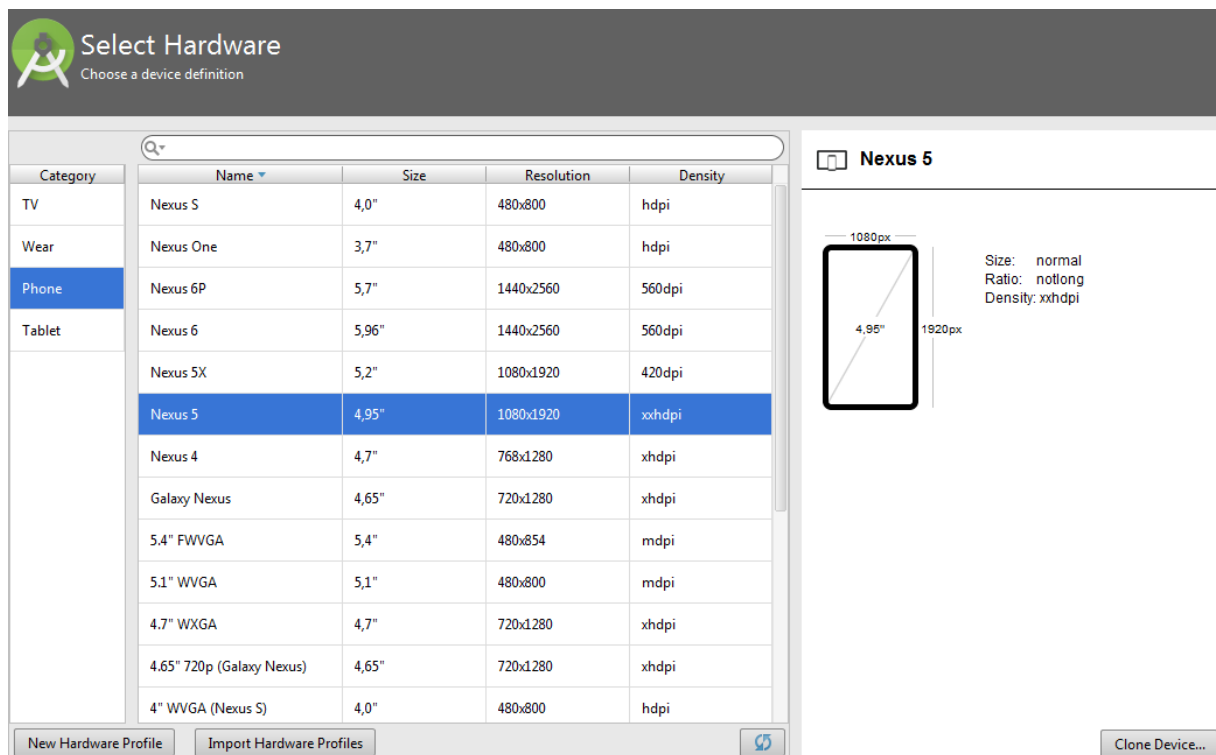
Danach kann die App gestartet werden. Folgendes Fenster wird geöffnet, bei dem man das virtuelle Android Gerät auswählt:



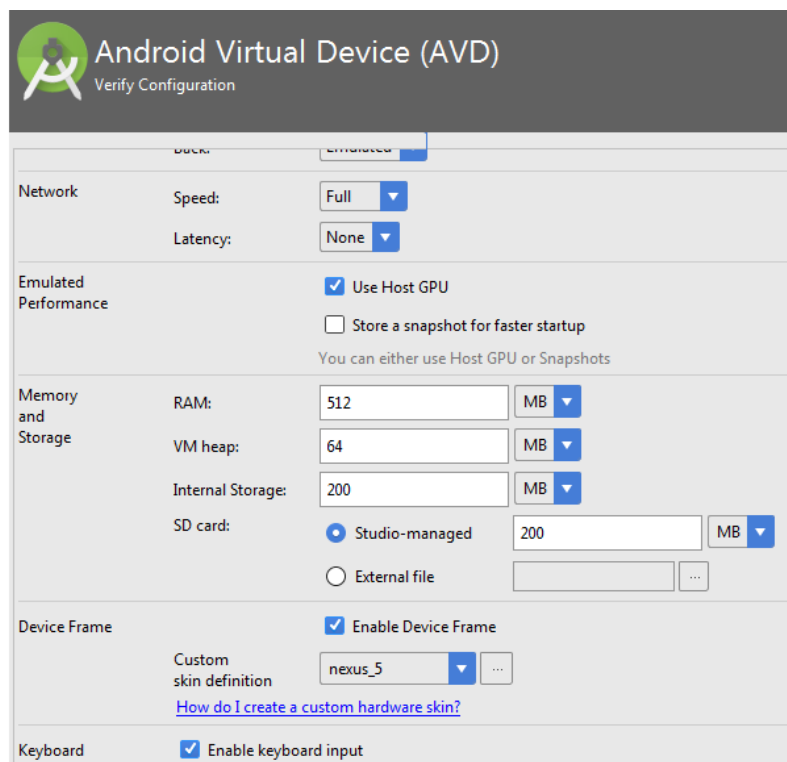
Über den Button „...“ kann ein gewünschtes Android Gerät ausgewählt und konfiguriert werden.



Durch drücken des Buttons „Create Virtual Device“ kann ein neues virtuelles Android Gerät erstellt werden.



Aus dieser Liste kann man sich das gewünschte Gerät aussuchen, von dem dann das Image heruntergeladen wird.

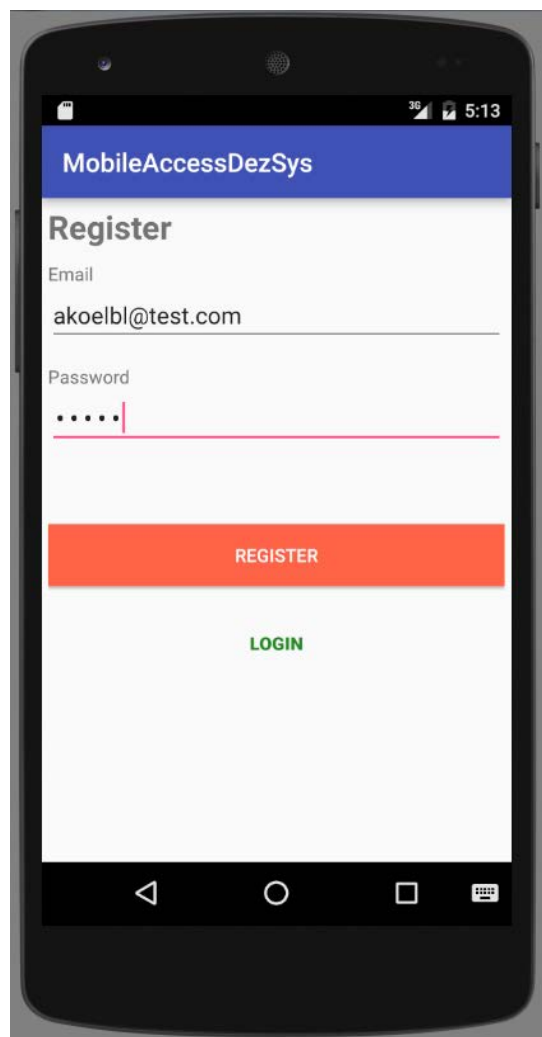


Über die „Advanced Settings“ können weitere Einstellungen wie z.B. RAM Größe konfiguriert werden. Bei den Advanced Settings sollte kontrolliert werden, ob die Option „Enable keyboard input“ ausgewählt ist, damit die App auch auf Tastatureingaben reagiert. Zusätzlich wird für das Ausführen des Android Virtual Devices Intel HAXM benötigt (wird meistens bei Installation von Android Studio mitinstalliert). Danach ist die Konfiguration abgeschlossen und die Applikation kann gestartet werden.

2.4 Funktionstest

Mittels des Emulators wurden die Funktionalitäten der App getestet und ob die Verbindung mit dem Web Service erfolgreich war.

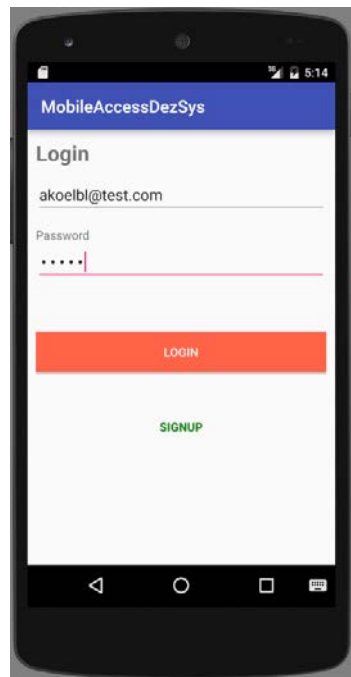
User mit der E-Mail Adresse „akoelbl@test.com“ und dem Passwort „12345“ erstellen:



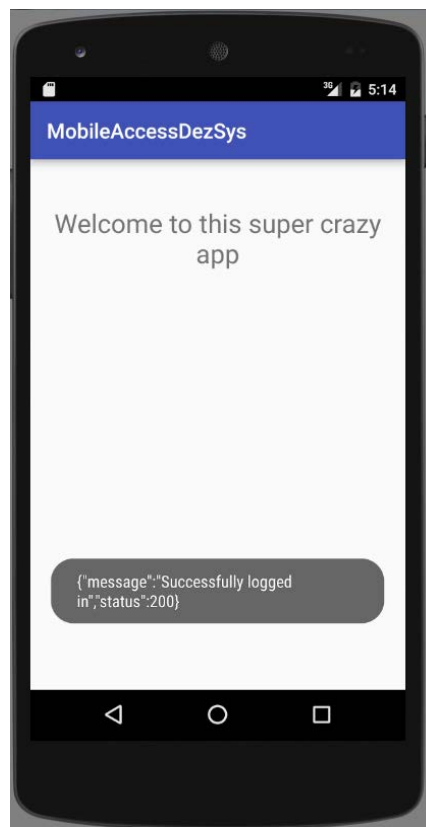
Bei erfolgreicher Erzeugung wird der User auf die Login Oberfläche

weitergeleitet.

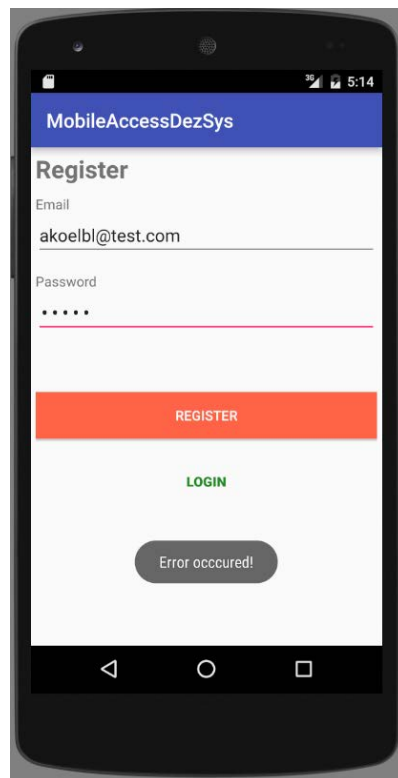
Login mit der E-Mail Adresse „akoelbl@test.com“ und dem Passwort „12345“:



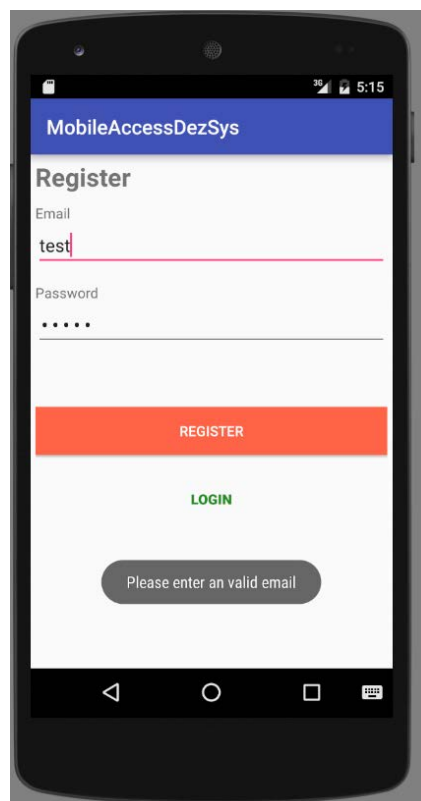
Bei erfolgreichem Login wird der Benutzer auf die Home Oberfläche weitergeleitet:



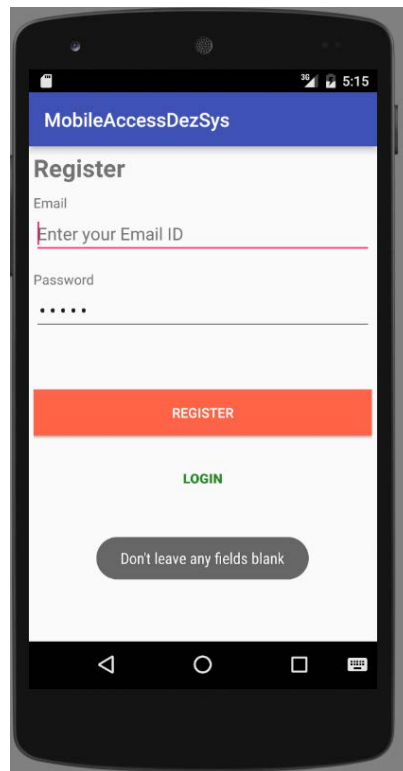
User mit der E-Mail Adresse „akoelbl@test.com“ und dem Passwort „12345“ erstellen (E-Mail Adresse bereits vergeben):



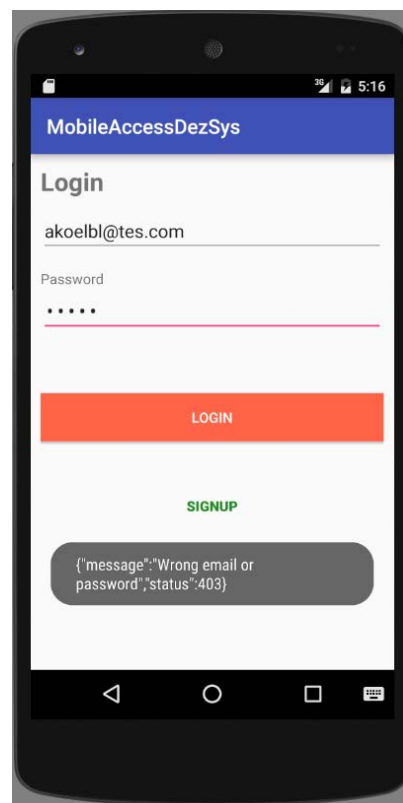
User mit nicht valider E-Mail Adresse erstellen:



Eingabefeld freilassen:



Login mit der E-Mail Adresse „akoelbl@tes.com“ und dem Passwort „12345“ (falsche E-Mail Adresse):



3 Zeitaufwand

Der Zeitaufwand für diese Aufgabe (Programmieren & Dokumentation) betrug 8 Stunden.

4 Repository

Das Git-Repository zu dieser Aufgabe ist unter folgender URL verfügbar:

<https://github.com/akoelbl-tgm/DezSys11-MobileAccess>

5 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>