
Laborprotokoll

11 – Mobile Access to Web Services

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

Mathias Ritter

Note:

Betreuer: Prof. Borko

Version 1.0

Begonnen am 15. April 2016

Beendet am 18. April 2016

Inhaltsverzeichnis

1Einführung.....	3
1.1Ziele.....	3
1.2Voraussetzungen.....	3
1.3Aufgabenstellung.....	3
2Ergebnisse.....	4
2.1Activities.....	4
UserForm.....	4
LoginActivity & RegisterActivity.....	5
SuccessActivity.....	6
2.2Task.....	6
2.3Aufruf der REST-API.....	7
2.4Simulation der App.....	8
LoginActivity.....	8
SuccessActivity.....	11
RegisterActivity.....	12
3Literaturverzeichnis.....	13

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.

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 mobile Anwendung wurde als Android App implementiert. Zur Entwicklung und zur Simulation auf einem mobilen Gerät wurde die IDE „Android Studio“ verwendet, welche kostenfrei genutzt werden kann.

Die App wurde in fünf Packages unterteilt. Im Package „activity“ wird das User-Interface konfiguriert, daher stellt dieses die View da. Ein spezielles UI-Element wurde in „layout“ definiert. Um den Aufruf der Rest-API durchzuführen, ist ein Background-Task erforderlich, welcher im Package „task“ implementiert wurde. Außerdem wurden weitere erforderliche Klassen zum Aufruf in „rest“ realisiert. Zwei Model-Klassen wurden in „model“ definiert.

2.1 Activities

Activities stellen in Android die einzelnen Screens der GUI dar.

In Android Studio wurde zu Beginn eine neue Login- bzw. Registrierungs-Activity generiert. Dadurch wurde eine entsprechende Klasse und im Ressourcenordner unter „layout“ ein Layout-XML-File angelegt. Diese Activity stellt ein Formular dar, welches die Eingabe von E-Mail-Adresse und Passwort ermöglicht. Mittels eines Buttons kann das Formular abgeschickt und somit ein Task ausgelöst werden. Während der Abarbeitung des Tasks wird ein Ladebildschirm angezeigt.

UserForm

Da sich die Registrierung und das Login sehr ähneln, wurde eine abstrakte Klasse „UserForm“ definiert. In dieser wird die Konfiguration und das Abschicken des Formulars (inklusive Validierung) sowie das Auslösen des Hintergrundtasks bereits übernommen, während in abstrakten Methoden beispielsweise der explizite Aufruf der RESTful API stattfindet.

In der Methode „setupForm()“ wird das Formular konfiguriert. Dies bedeutet, dass ActionListener und Referenzen zu den UI-Elementen definiert werden. ActionListener wurden für den Submit-Button und für das Passwort-Feld definiert, welche beide ein Abschicken des Formulars auslösen. Die UI-Elemente werden durch die ID, welche im Layout-XML-File definiert ist, referenziert. Die Konfiguration des Submit-Buttons erfolgte beispielsweise folgendermaßen:

```
Button mSubmitButton = (Button) findViewById(R.id.submit_button);
mSubmitButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        submitForm();
    }
});
```

In der Methode „submitForm()“ werden die Formularinhalte validiert und ein neuer Background-Task ausgelöst. Für die Validierung der E-Mail-Adresse wird eine in der API bereits vorhandene Methode verwendet, das Passwort wird auf eine Mindestlänge von 5 Zeichen überprüft. Falls alle Eingaben valide sind, wird ein Ladebildschirm angezeigt und der Background-Task gestartet, welcher die Rest-API aufruft (siehe Kapitel „2.2 Task“).

```
showProgress(true);
new UserTask(this).execute((Void) null);
```

LoginActivity & RegisterActivity

Die „LoginActivity“ stellt den Login-Screen dar, während „RegisterActivity“ den Registrierungs-Screen darstellt. Beide Klassen erben von „UserForm“.

Bei der Initialisierung einer Activity wird die „onCreate“-Methode aufgerufen. Darin rufen beide Activities unter anderem die „setUpForm“-Methode der Superklasse auf, um das Formular zu initialisieren. Bei der LoginActivity wird zusätzlich der Button zum Wechseln zur Registrierung initialisiert:

```
mRegisterButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
        startActivity(intent);
    }
});
```

In beiden Activities wird der Aufruf zu der RESTful Resource implementiert. Dabei wird zuerst die Model-Klasse „Response“ initialisiert, welche später die Response enthalten wird. Danach wird mittels „UserRequester“ der eigentliche Aufruf durchgeführt und zum Schluss die Response zurückgegeben. Der Aufruf erfolgt synchron. In der LoginActivity wird dies folgendermaßen durchgeführt:

```
public Response callRestAPI(User user){
    Response response = new Response();
    UserRequester caller = new UserRequester(
        getApplicationContext(), new ResponseHandler(response));
    caller.login(user);
    return response;
}
```

SuccessActivity

Diese Activity zeigt nach einem erfolgreichen Login eine Willkommensnachricht an. Die E-Mail-Adresse des Users wird diesem „Intent“ über die „putExtra“-Methode übergeben. Dies erfolgt in der LoginActivity:

```
Intent intent = new Intent(this, SuccessActivity.class);
intent.putExtra("email", super.getEmail());
startActivity(intent);
```

In der SuccessActivity wird dieser Wert mittels „getString(...)“ abgerufen.

2.2 Task

Der Task wurde in der Klasse „UserTask“ implementiert. Dieser führt den Rest-API-Call in einem eigenen Thread im Hintergrund aus. Dazu verfügt der Task über eine „doInBackground“-Methode. Diese ruft „callRestAPI“ (siehe oben) der jeweiligen Activity auf. Die Methode gibt als Boolean zurück, ob der Aufruf erfolgreich war oder nicht.

```
protected Boolean doInBackground(Void... params) {
    response = userForm.callRestAPI(user);
    return response.getStatus() == 200 || response.getStatus() == 201;
}
```

Wurde diese Methode abgearbeitet, wird „onPostExecute“ aufgerufen und das Resultat übergeben. Darin wird wieder die jeweilige „success“-Methode der Activity aufgerufen, falls der Rest-Call erfolgreich war. Sonst wird eine Fehlermeldung im Rahmen eines „Toast“ angezeigt:

```
protected void onPostExecute(final Boolean success) {  
    if (success) {  
        userForm.success();  
    } else {  
        userForm.showProgress(false);  
        Toast.makeText(userForm.getContext(), response.getMessage(),  
            Toast.LENGTH_LONG).show();  
    }  
}
```

2.3 Aufruf der REST-API

Zum Aufrufen der REST-API wird die Library „Android Asynchronous Http Client“ verwendet. [1] Diese wird in der Klasse „UserRequester“ verwendet. Außerdem wurde ein „ResponseHandler“ implementiert, welche die HTTP-Response weiter verarbeitet.

Im Konstruktor des „UserRequester“ wird ein neuer „SyncHttpClient“ initialisiert. Das Timeout wird auf 50 Sekunden gesetzt, da das Starten der Heroku-App (Übung „DezSys09“) einige Zeit in Anspruch nehmen kann. Es wurden ebenfalls Methoden für das Login und die Registrierung implementiert, welche ein User-Objekt entgegennehmen.

Weiters wurde eine „post“-Methode implementiert, welche den Host sowie das zu übertragende User-Objekt entgegennimmt. Dabei wird zuerst eine StringEntity generiert, welche den Inhalt des Users-Objekts (E-Mail-Adresse und Passwort) in JSON-Form enthält. Danach wird der POST-Request durchgeführt:

```
private void post(String host, User user) {  
    StringEntity entity = this.createEntity(user);  
    client.post(this.context, host, entity, "application/json", this.handler);  
}
```

Diesem wird ebenfalls der ResponseHandler übergeben, welcher zwei Methoden „onSuccess“ und „onFailure“ zur weiteren Verarbeitung des Response enthält. In beiden Methoden wird der Statuscode und die Message der Response in das Response-Objekt gesetzt:

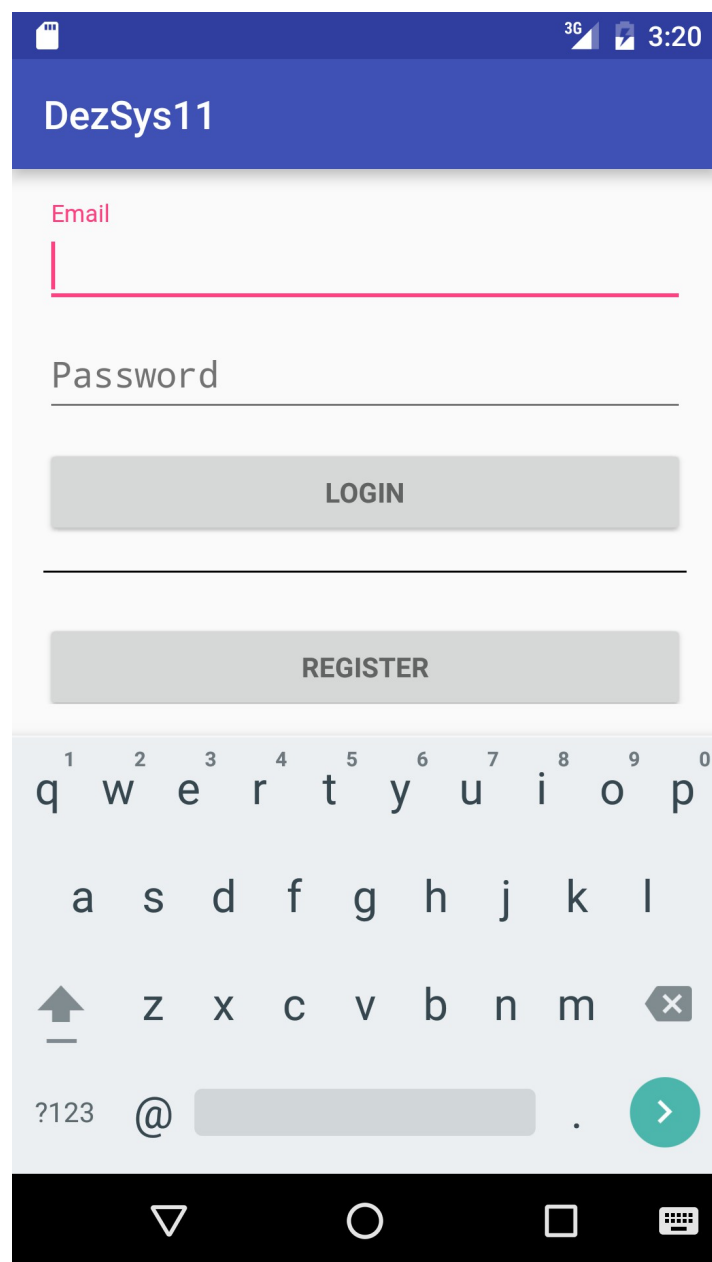
```
JSONObject jsonResponse = new JSONObject(responseString);  
if (jsonResponse.has("status") && jsonResponse.has("message")) {  
    response.setStatus(jsonResponse.getInt("status"));  
    response.setMessage(jsonResponse.getString("message"));  
}
```

2.4 Simulation der App

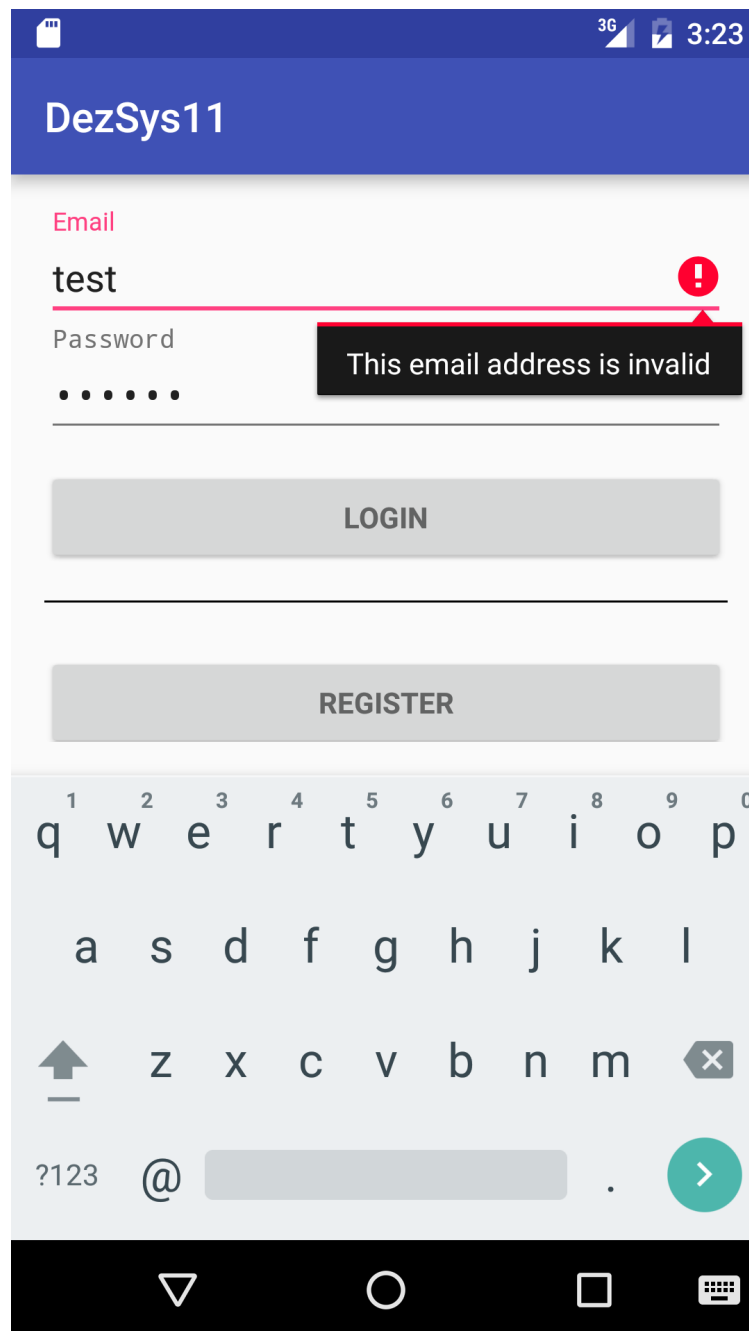
Die App konnte in AndroidStudio per Emulator ausgeführt werden. Dabei konnte die Funktionalität der App getestet werden.

LoginActivity

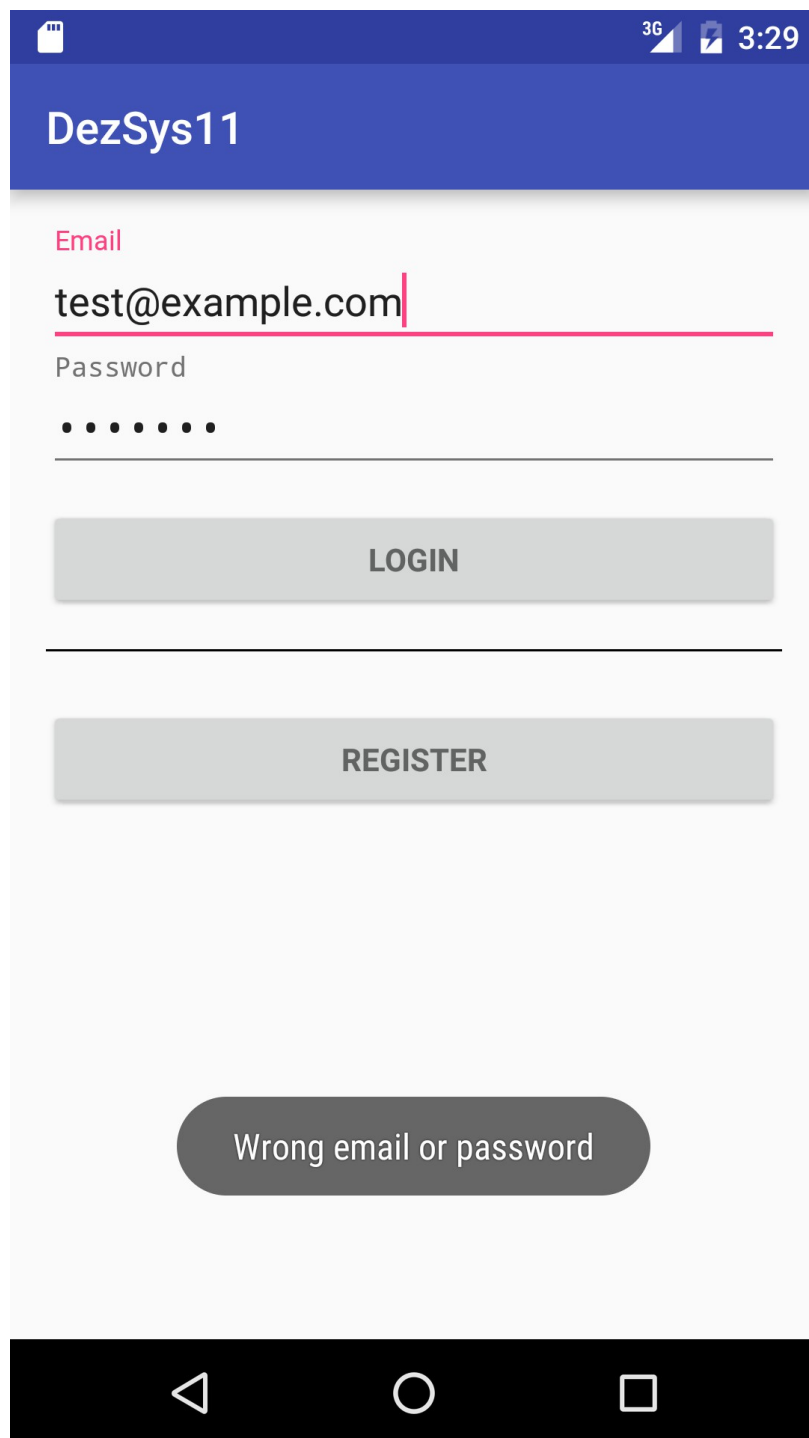
Der Benutzer kann E-Mail-Adresse und Passwort eingeben und sich einloggen. Alternativ kann der Benutzer auf „Register“ drücken, um das Registrierungsformular zu öffnen.



Da Eingaben validiert werden, wird eine falsche Eingabe beispielsweise folgendermaßen angezeigt:

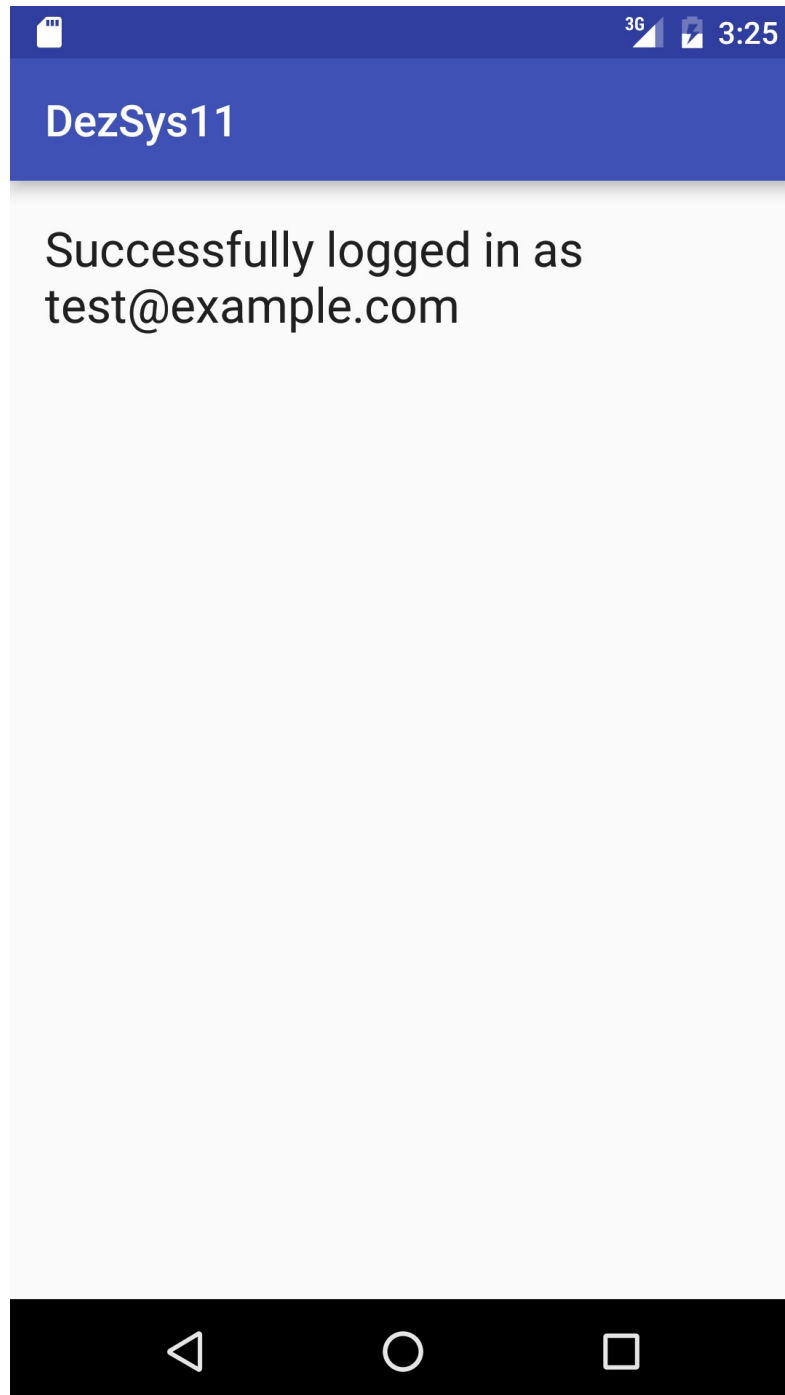


Meldungen (wie z.B. falsche E-Mail/Passwort) werden als Toast angezeigt. Ein Toast wird ebenfalls nach erfolgreicher Registrierung angezeigt:



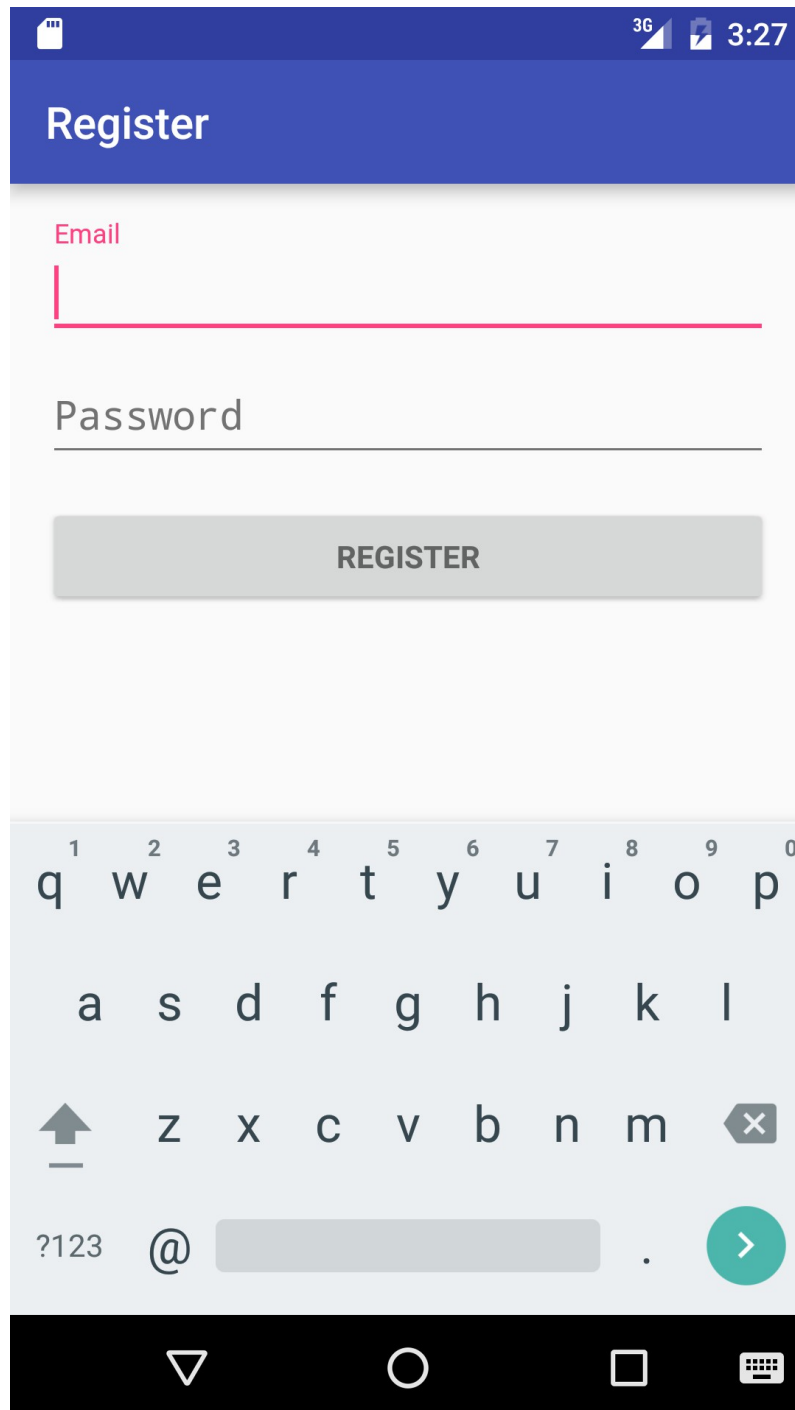
SuccessActivity

Nach einem erfolgreichen Login erscheint folgende Activity:



RegisterActivity

Falls der Benutzer zu Beginn (LoginActivity) den Button „Register“ wählt, öffnet sich eine neue Activity zur Registrierung:



The screenshot displays a mobile application interface for registration. At the top, a dark blue header bar contains the title "Register" in white text. Below the header, the interface features two input fields: "Email" and "Password". The "Email" field is highlighted with a red border, indicating it is the active field. Below these fields is a grey button labeled "REGISTER". At the bottom of the screen, a virtual keyboard is visible, showing the letters q, w, e, r, t, y, u, i, o, p on the first row, a, s, d, f, g, h, j, k, l on the second row, and z, x, c, v, b, n, m on the third row. The keyboard also includes a shift key, a spacebar, and a backspace key. The status bar at the very top shows a 3G signal, a battery icon, and the time 3:27.

3 Literaturverzeichnis

- [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/>
[zuletzt abgerufen am 18.04.2016]