tgm
Die Schule der Technik

JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

Systemtechnik Labor

5xHIT 2017/18, Gruppe A

Protokolle in LaTeX

Laborprotokoll

Kevin Waldock

3. April 2018

Bewertung:		Version:	0.1
Betreuer:	Michael Borko	Begonnen:	31.1.18
		Beendet:	1.2.18

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 umgesetzt werden.

1.2 Voraussetzungen

- Grundlagen einer höheren Programmiersprache und XML
- Grundlegendes Verständnis über Entwicklungs- und Simulationsumgebungen
- Verständnis von RESTful Webservices
- Download der entsprechenden Entwicklungsumgebung

1.3 Aufgabenstellung

Es ist eine mobile Anwendung zu implementieren, die sich an das Webservice aus der Übung GK9.3 "Cloud-Datenmanagementänbinden 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

- Gruppengrösse: 1 Person
- Anforderungen "**überwiegend erfüllt**"
  - Dokumentation und Beschreibung der angewendeten Schnittstelle
  - Anbindung einer mobilen Applikation an die Webservice-Schnittstelle
  - Registrierung von Benutzern
  - Login und Anzeige einer Willkommensnachricht
- Anforderungen "**zur Gänze erfüllt**"
  - Simulation bzw. Deployment auf mobilem Gerät
  - Überprüfung der funktionalen Anforderungen mittels Regressionstests

2 Umsetzung

Diese Umsetzung basiert auf GK 9.3 und dem Tutorial von programmerguru.com [1]. Für die Entwicklung wurde Android Studio auf Windows verwendet.

2.1 REST-Schnittstelle

Die Client-REST-Schnittstelle besteht aus folgenden Klassen:

- `RESTConfig`
- `RESTExecutor`
- `RESTFailedCallback`
- `RESTSuccessCallback`

2.1.1 RESTConfig

Diese Klasse beschreibt die IP und den Hostname des REST-Service und dessen Endpunkte für Login und Registrierung. Es handelt sich dabei um konstante statische Felder die vom Entwickler selbst festgelegt werden.

2.1.2 RESTFailedCallback und RESTSuccessCallback

Diese Interfaces dienen als Callback-Interfaces für die RESTExecutor-Klasse. Da jeweils nur eine Interface-Methode deklariert wurde, kann dies mit Java Lambda-Expression verwendet werden.

2.1.3 RESTExecutor

Die Klasse `RESTExecutor` bietet statische Methoden für die Ausführung von REST-Befehlen. Intern wird die Klasse `AsyncHttpClient` verwendet, damit die GUI weiterhin responsiv ist. Daher müssen diese Methoden auch in der Android-Main-Loop ausgeführt werden. Für GUI-Applikationen ist dies kein Problem, da diese sowieso in der Android-Main-Loop ausgeführt werden. Unit-Tests hingegen führen Tests nicht in der Android-Main-Loop aus. Daher muss bei Unit-Tests explizit eine Task hinzugefügt werden.

RESTExecutor.ExecuteLoginRequest Diese Methode führt den REST-Request für ein Login aus. Der Endpunkt `<host>:port/login-request` wird angesprochen die die Parameter `username` und `password` werden übergeben. Zurückgegeben wird ein JSON-Objekt mit einem Statuscode. Je nach Statuscode wird der Callback für einen erfolgreichen Request oder der Fehler-Callback-Funktion bei einem Fehler aufgerufen.

```
public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {
    try {
        String response = new String(responseBody, "UTF-8");
        // JSON Object
        JSONObject obj = new JSONObject(response);

        int statusResponseCode = obj.getInt("code");
        // When the JSON response has status boolean value assigned with true
        if(statusResponseCode == 0) {
            successCallback.onSuccess();
        } else if(statusResponseCode == 1) {
            failedCallback.onFailed("User does not exist!");
        } else if(statusResponseCode == 2) {
            failedCallback.onFailed("Invalid password, try again!");
        } else { // Else display error message
        failedCallback.onFailed("Error: Unknown error code");
        }
    } catch (UnsupportedEncodingException e) {
        failedCallback.onFailed("Error Occured [Server's JSON response is not properly
        ↪ UTF8 encoded]!");
    } catch (JSONException e) {
        failedCallback.onFailed("Error Occured [Server's JSON response might be
        ↪ invalid]!");
    }
}

@Override
public void onFailure(int statusCode, Header[] headers, byte[] responseBody, Throwable
↪ error) {
    // When Http response code is '404'
    if(statusCode == 404){
        failedCallback.onFailed("Requested resource not found");
    } else if(statusCode == 500){
    failedCallback.onFailed("Something went wrong at server end");
    } else {
        failedCallback.onFailed("Unexpected Error occcured! [Most common Error: Device
        ↪ might not be     connected to Internet or remote server is not up and
        ↪ running]");
    }
}
```

Auflistung 1: Handler für den REST-Login-Request

RESTExecutor.ExecuteRegisterRequest

2.2 GUI

Die GUI-Klassen wurde vom Programmerguru-Tutorial [1] übernommen und nur leicht verändert [1]. So nimmt die Login-Activity keine E-Mail-Adresse sondern den Benutzername entgegen. Die Methode `invokeWS` wurde in `RESTExecutor` extrahiert, sodass diese mit Unit-Tests getestet werden kann.

2.3 Deployment

Bevor das Deployment gestartet werden kann muss der Hostname des REST-Services in der Klasse `RESTConfig.URL_MAIN` gesetzt werden. Dies inkludiert die IP/URL inkl. Port
(Beispiel: `URL_MAIN = "http://192.168.0.50:3000"`).

Als nächstes muss der REST-Service selbst gestartet werden. Dies ist im Protokoll von GK 9.3 beschrieben.

Als nächstes muss ein Emulator gestartet werden. Dies kann mit `emulator @name-of-your-emulator` erzielt werden (wo "name-of-your-emulator" ist der Name des Emulators). Will man wissen welche virtuelle Geräte vorhanden sind kann man dies mit `emulator -list-avds` auflisten. [2]

Die Emulator-Binary ist in `${ANDROID_SDK}/tools/emulator` zu finden.

Wurde der Emulator gestartet und das Android-Betriebssystem vollständig initialisiert, so kann nun das Projekt gestartet werden. Folgende Aktionen sind möglich:

- `gradlew build` - Baut das Projekt
- `gradlew installDebug` - Erstellt eine Debug-APK-Datei, welches auf einem virtuellen Android-Gerät installiert wird. Dabei muss ein virtuelles Android-Gerät bereits laufen.
- `gradlew connectedDebugAndroidTest` - Erstellt eine Debug-APK-Datei und lässt die Tests auf einem virtuellen Android-Gerät ausgeführt wird. Dabei muss eine virtuelles Android-Gerät bereits laufen.

Nach dem Starten des Emulators und dem Ausführen des Befehls `gradlew installDebug` sollte die App im Launcher des virtuellen Android-Gerät vorhanden und ausführbar sein.

Literaturverzeichnis

[1] *Android Restful Webservice Tutorial - How to call RESTful webservice in Android - Part 3 | Android Tutorial Blog*. 03.04.2018. programmerguru. URL: http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-call-restful-webservice-in-android-part-3/.

[2] *How do I launch the Android emulator from the command line? - Stack Overflow*. 03.04.2018. Stackoverflow. URL: https://stackoverflow.com/questions/4974568/how-do-i-launch-the-android-emulator-from-the-command-line.

Auflistungsverzeichnis