JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

DEZSYS11

Mobile Access to Web Services

Systemtechnik Labor
5BHITT 2015/16, Gruppe X

Andreas Ernhofer

Version 1.0

Note:

Betreuer: M. Borko

Begonnen am 19. Februar 2016

Beendet am 22. Februar 2016

Inhaltsverzeichnis

Abbildungsverzeichnis

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

Das erstellte Gesamtprojekt hat folgenden Aufbau. In diesem Protokoll wird jedoch nur der Teil der Android Applikation erklärt. Um Informationen über das JEE WebApp zu erhalten muss das Protokoll der Übung DEZSYS09 zur Hand genommen werden.

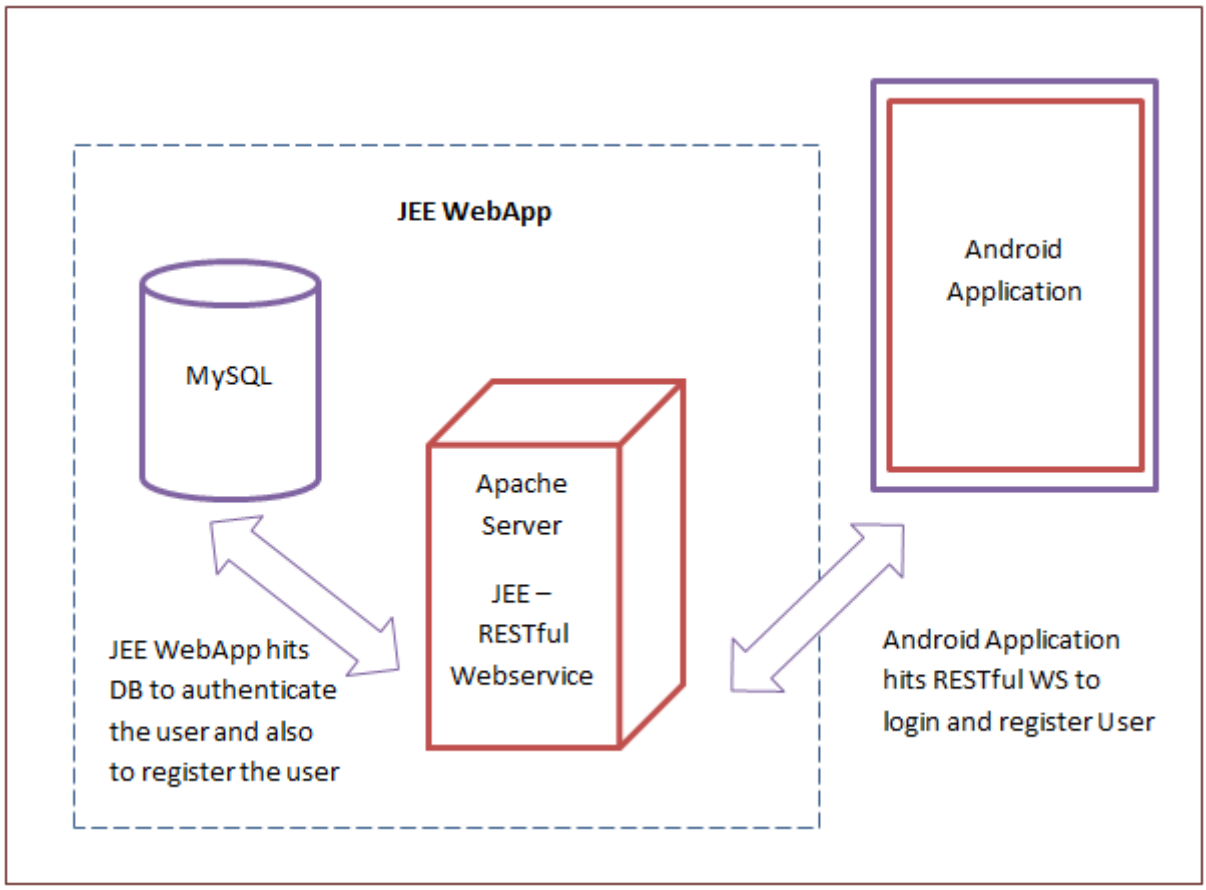


Abbildung 1: Funktionsweiße des Projektes [ANDR2]

2.1 Erstellen des Projektes

Das Projekt wurde mittels Android Studio [ANDS] erstellt. Dazu wurde folgendermaßen vorgegangen. Zunächst wurde ein neues Projekt angelegt:

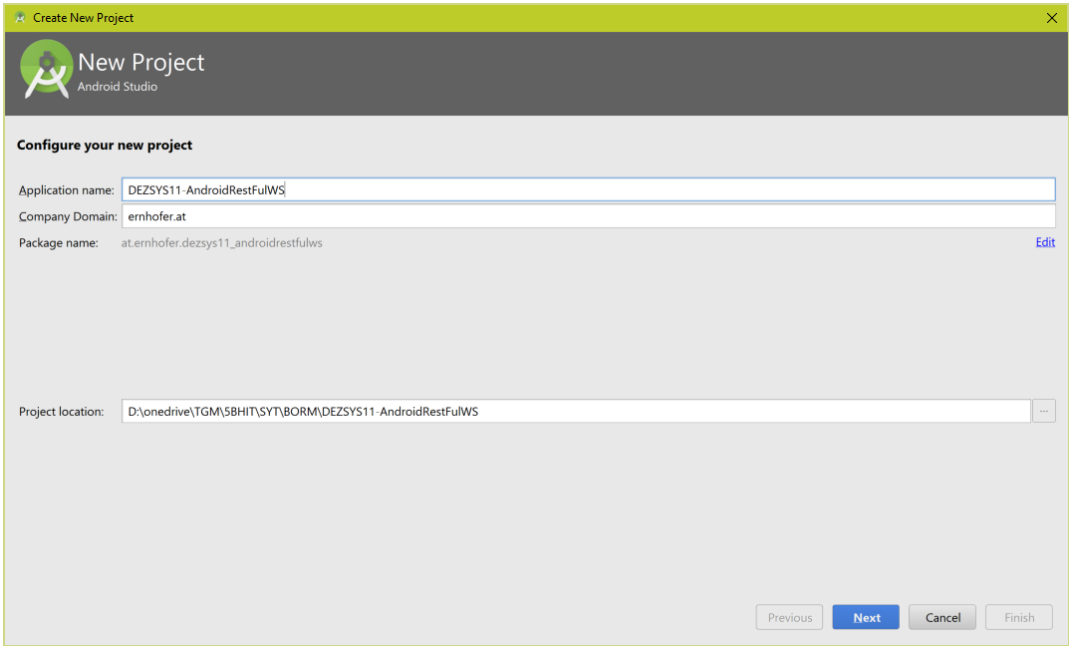


Abbildung 2: Anlegen eines neuen Projektes

Danach musste eine Auswahl dafür getroffen werden, auf welchem Gerät die App später laufen wird.

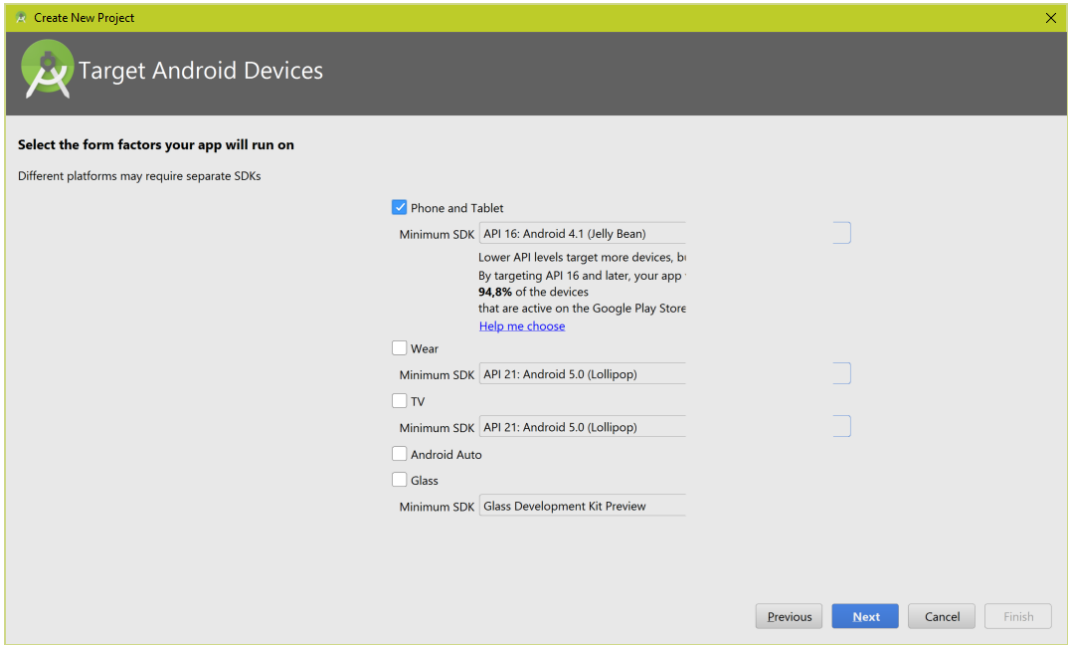


Abbildung 3: Wählen des Gerätes des neuen Projektes

Um die App von Anfang an lauffähig zu machen und um ein Beispielhaftes Ergebniss zu erhalten wurde ein Empty Activity als Activity gewählt. Dies ist für den weiteren Verlauf des Projektes jedoch nicht relevant.

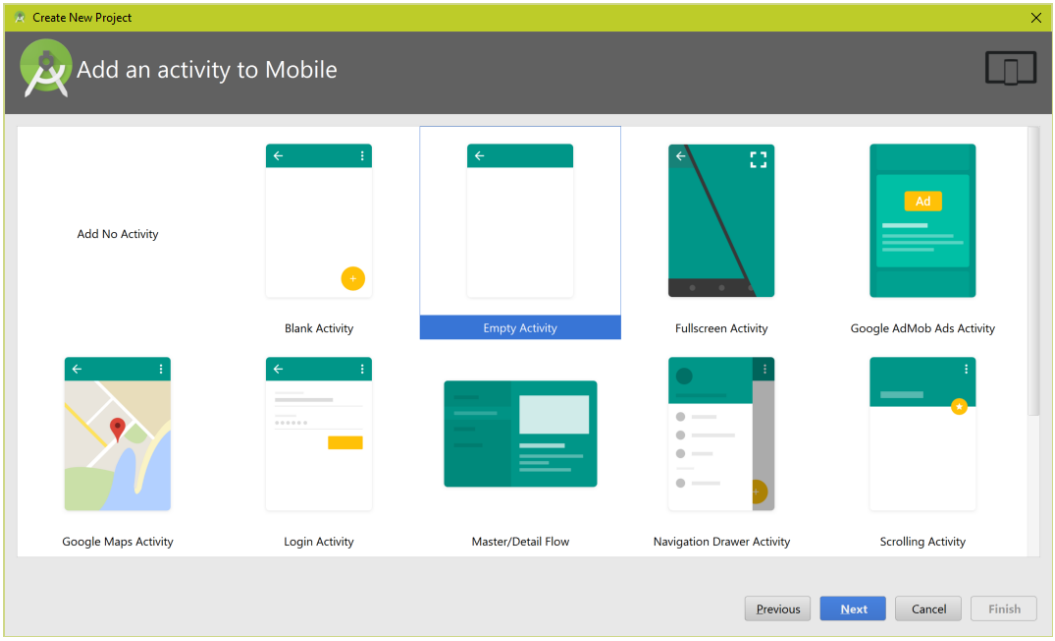


Abbildung 4: Einfügen eines Beispielactivities ins neue Projekt

Das Anlegen der einzelnen Dateien entspricht zum Großteil der Anleitung [ANDR3].

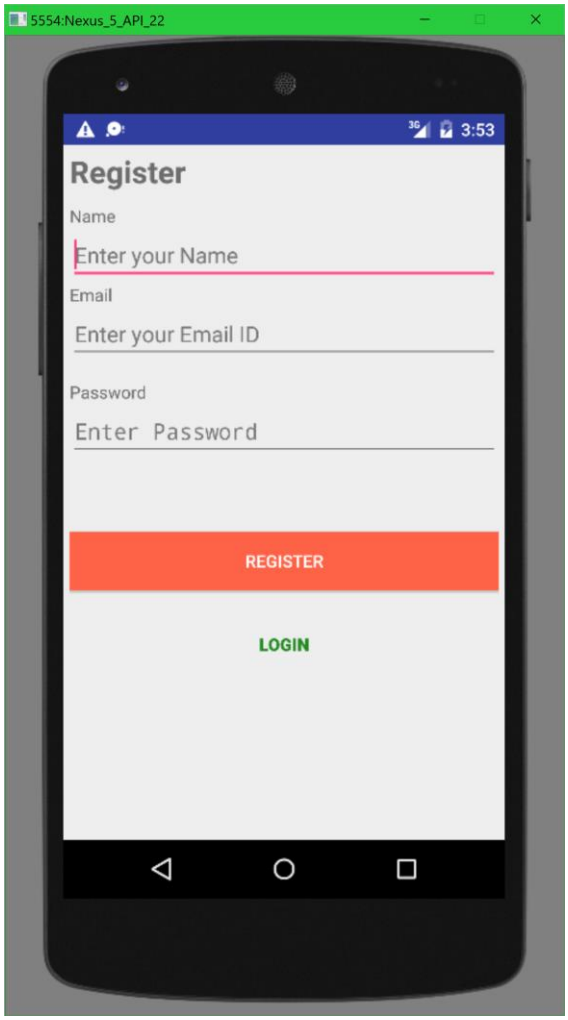
Anlegen der Variablen

Folgenden Variablen wurden in der *strings.xml* Datei erstellt:

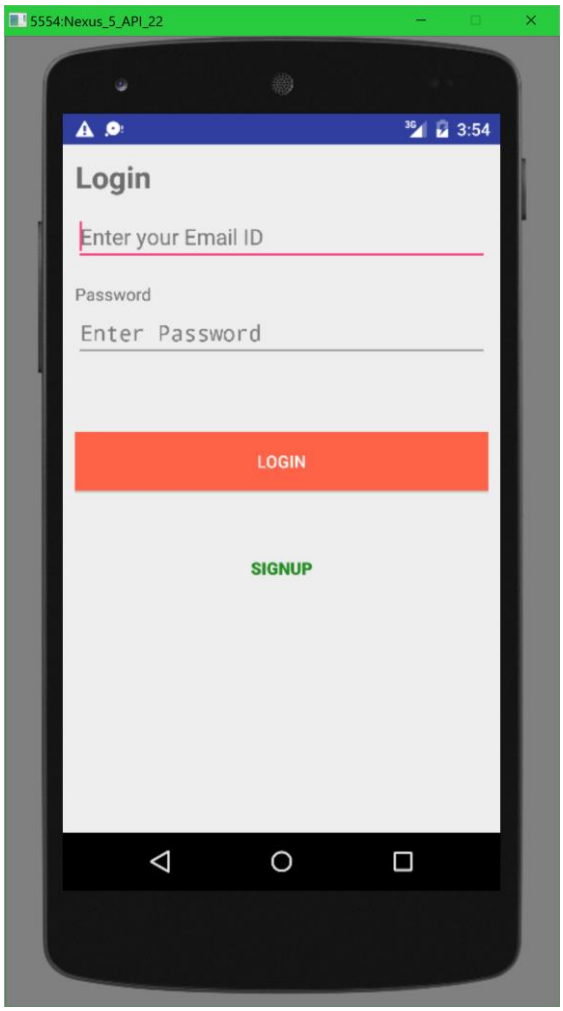
```
<string name="app_name">DEZSYS11-AndroidRstFulWS</string>
<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
<string name="title_activity_login">Login</string>
<string name="title_activity_home">HomeActivity</string>
<string name="register_title">Register</string>
<string name="login_title">Login</string>
<string name="name">Name</string>
<string name="email">Email</string>
<string name="pwd">Password</string>
```

Erstellen der Layouts

In diesem Projekt gibt es drei Layouts. *register.xml* ist das Layout, welches verwendet wird wenn sich ein Benutzer registrieren möchte. Dieses ist außerdem der Startbildschirm, welcher erscheint, wenn die App gestartet wird.



login.xml ist das Layout, welches verwendet wird wenn sich ein Benutzer einloggen will.



Hat sie ein Benutzer erfolgreich eingeloggt, so bekommt er das *home.xml* Layout zu sehen.

Erstellen der Java-Klassen

Um mit den Layouts auch arbeiten zu können wurden einige Java Klassen erstellt. Die *Utility* Klasse beinhaltet Methoden, welche von anderen Klassen benötigt werden. Eine von diesen ist beispielsweiße das überprüfen einer korrekten email-Adresse.

HomeActivity kümmert sich um das Anzeigen des Bildschirmes, nachdem sich ein Benutzer erfolgreich angemeldet hat. *LoginActivity* kümmert sich, wie der Name schon vermuten lässt, um das einloggen von Benutzern. Die RegisterActivity Klasse wiederum ist für das registrieren von Benutzern verantwortlich. Anders als in der Vorlage vorgesehen habe ich dabei anstatt eines AsyncHttpResponseHandler einen TextHttpResponseHandler verwendet.
Um beim Login und beim Registrieren nicht einzeln die URL Adresse eingeben zu müssen, und um etwaige Änderungen des Codes zu vereinfachen wurde zuletzt noch eine Constants Klasse erzeugt. Diese beinhaltet neben IP-Addrese und Port, den Pfad, auf welchem das RestFul Webservice läuft.

Gradle Dependencies anpassen

Um die Java-Klassen Kompilierbar zu machen, musste eine Gradle-Abhängigkeit hinzugefügt weren. Dazu wurden der Android Asynchronous Http Client dem Gradle Build File hinzugefügt.

Anpassen des AndroidManifest.xml

Um beim Starten der App das gewünschte Fenster zu erhalten müssen die eben erstellten Activities, dem AndroidManifest hinzugefügt werden. Dabei habe ich die Einstellungen der Vorlage folgendermaßen angepasst, als dass ich das verwendete Package, sowie die Android SDK aktualisiert, und das zu verendende Android Icon geändert habe. Dabei war folgende Codezeile wichtig, um der App die Rechte um auf das Internet zuzugreifen zu erteilen.

```
<uses-permission android:name="android.permission.INTERNET" />
```

2.2 Ausführen des Projektes

Emulator

Mit dem Android AVD Manager lassen sich ganz einfach virtuelle Geräte erzeugen. In diesem Beispiel wurde ein Nexus 5 erzeugt.

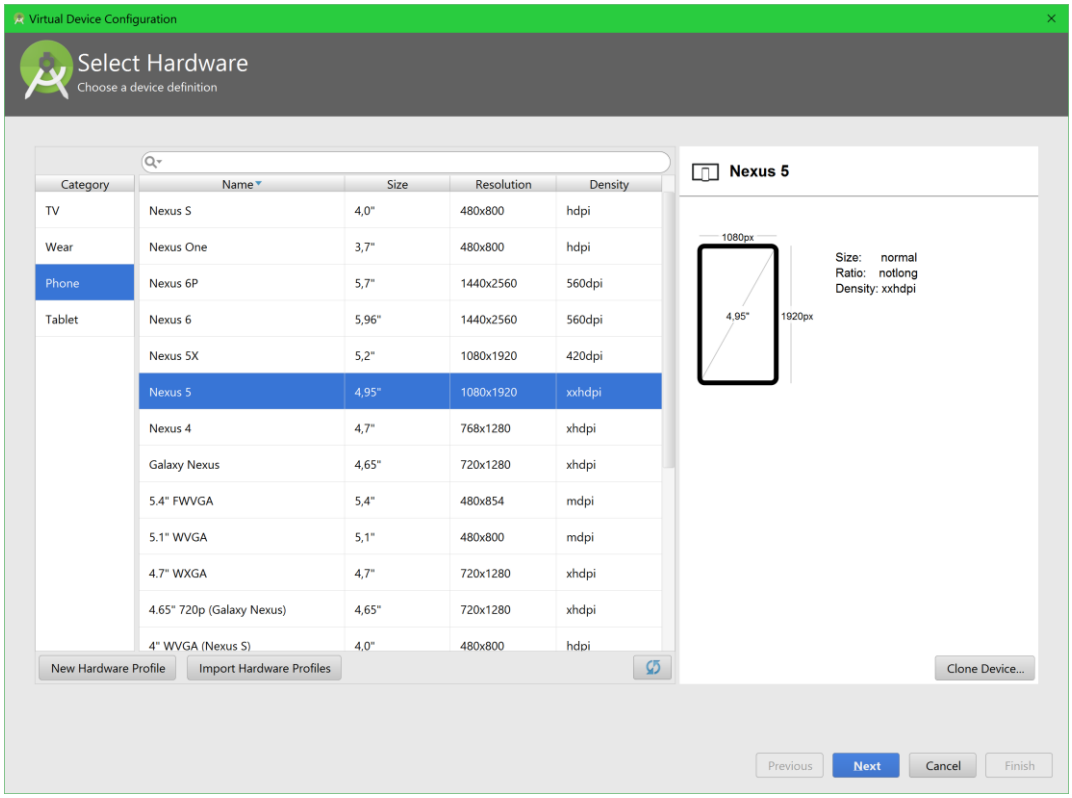


Abbildung 5: Anlegen eines neuen Emulators

Als System Image wurde Lollipop gewählt.

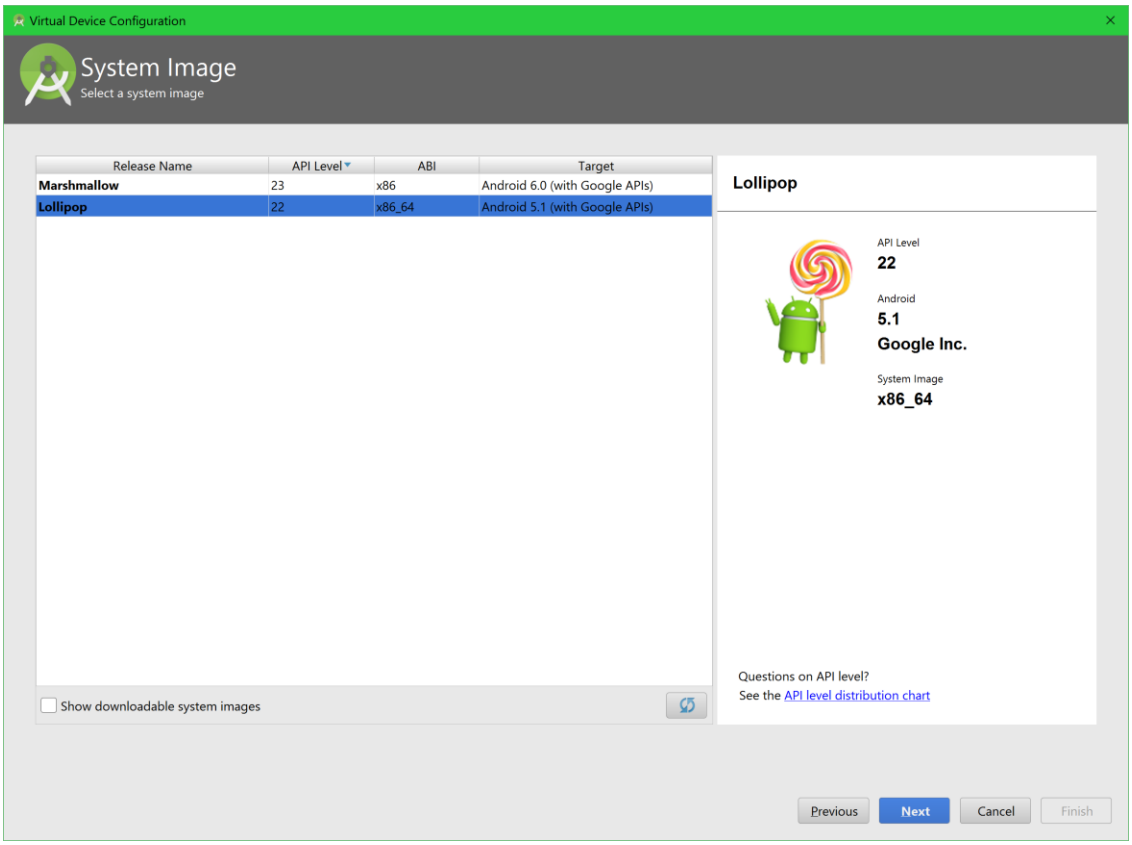


Abbildung 6: Wählen des Emulator Images

Die Standardeinstellungen wurde alle übernommen und das virtuelle gerät erzeugt.

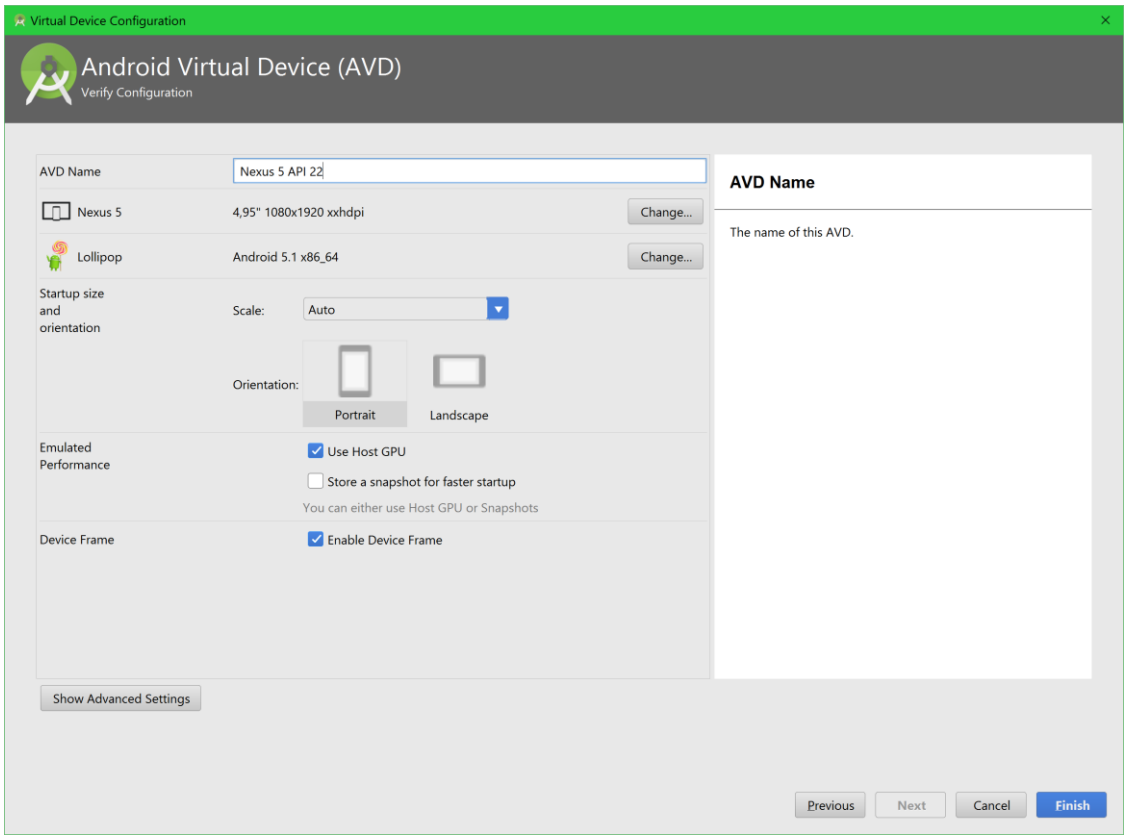
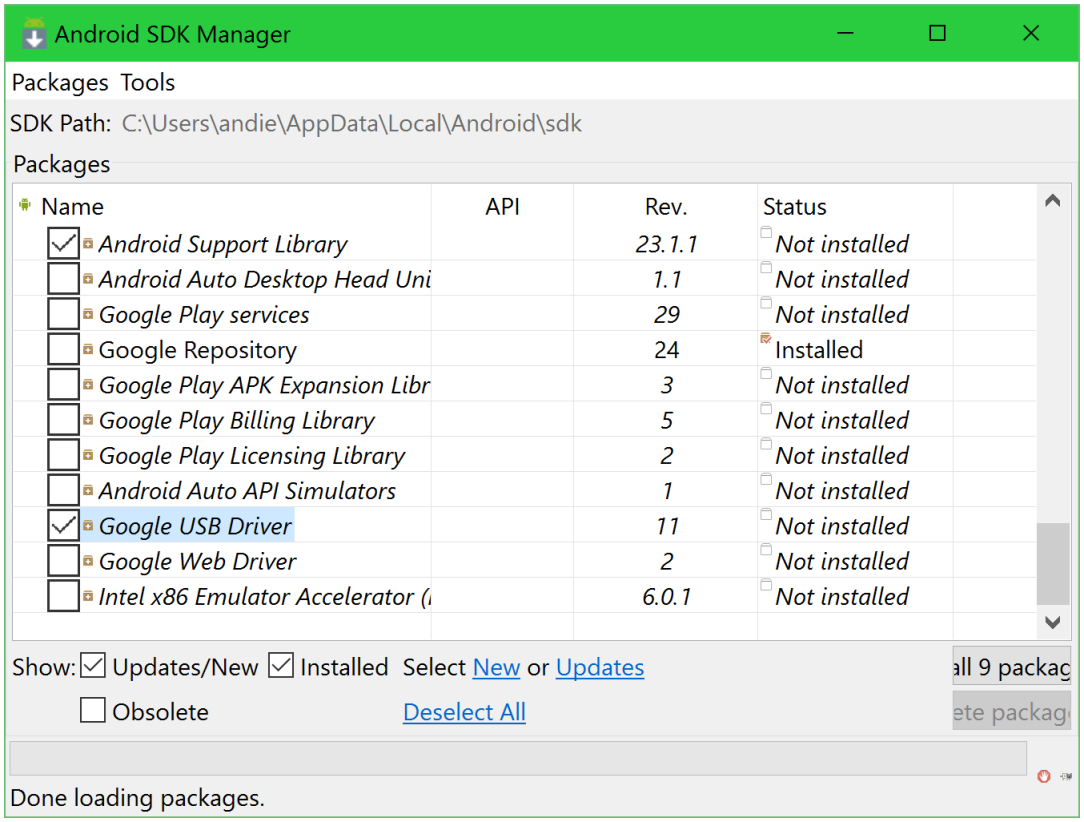


Abbildung 7: Einstellungen des Emulators

Die App kann nun im Android Studio gestartet werden. Dabei öffnet sich ein Fenster, welches fragt auf welchem gerät die App installiert werden soll. Hier wird der eben erstellte Emulator gewählt, und nach kurzer Installationszeit startet die App automatisch.

Mobiles Gerät

Um die App auf einem mobilen Gerät installieren zu können, mussten ein paar Einstellunge vorgenommen werden. Zuerst musste der SDK Manager gestartet werden und mit diesem Google USB Drive installiert werden. Zusätzlich dazu mussten noch Einstellungen direkt am mobilen Gerät geändert werden. So musste erstens der Entwicklermodus aktiviert, und anschließend USB-Debugging ermöglicht werden. Zusätzlich dazu musste in den USB Optionen der PTP Modus gewählt werden. Nachdem am Rechner der Treiber für da mobile Gerät aktualisiert wurde konnte die App auch auf mobilen Android Geräten ausgeführt werden. Dabei ist jedoch zu beachten, dass die IP-Adresse in der Constants Klasse auf die IP-Adresse des Rechners zu ändern ist!


Android SDK Manager
Packages Tools
SDK Path: C:\Users\andie\AppData\Local\Android\sdk
Packages
Name API Rev. Status
Android Support Library 23.1.1 Not installed
Android Auto Desktop Head Uni 1.1 Not installed
Google Play services 29 Not installed
Google Repository 24 Installed
Google Play APK Expansion Libr 3 Not installed
Google Play Billing Library 5 Not installed
Google Play Licensing Library 2 Not installed
Android Auto API Simulators 1 Not installed
Google USB Driver 11 Not installed
Google Web Driver 2 Not installed
Intel x86 Emulator Accelerator (6.0.1 Not installed
Show: Updates/New Installed Select New or Updates
Obsolete Deselect All
Done loading packages.

3 Probleme

Vorbereitung des Projektes

Um mit dem Projekt beginnen zu können wird eine Android SDK benötigt. Da diese jedoch vor dem Projektstart nicht auf dem Rechner vorhanden war, musste sie aus dem Internet heruntergeladen werden. Aufgrund der langsamen Downloadgeschwindigkeit in der Schule wurde der scheinbar einfachste Teil des Projektes zu einer Geduldsprobe.

Email pattern

Verbindung zu Host Localhost

Es gab Probleme damit, eine Verbindung vom Emulator zu dem Host localhost herzustellen. Dieses Problem konnte jedoch schnell behoben werden. [STAL]

4 Zeitaufzeichnung

Tätigkeit	Dauer
Projekt Vorbereitung	120min
Erstellen der mobilen Applikation	30min
Anbindung der mobilen Applikation an die Webservice-Schnittstelle	60min
Registrierung von Benutzern	15min
Login und Anzeige einer Willkommensnachricht	15min
Simulation bzw. Deployment auf mobilem Gerät	60min
Problembehandlung bzw. Fehlerbehebung	30min
Protokollierung	90min
Gesamt	7h

5 Quellen

[ANDR2]"Android Restful Webservice Tutorial – How to How to create RestFul Webservice in Jave"; Posted By Android Guru on May 14, 2014; online: http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-create-restful-webservice-in-java-part-2/
zuletzt besucht: 19:02.2016

[ANDR3]"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 besucht: 19:02.2016

[KALP] "Referenzimplementierung von DezSys09"; Paul Kalauner; online: https://github.com/pkalauner-tgm/dezsys09-java-webservices
zuletzt besucht: 19:02.2016

[ANDS] „Android Studio"
http://developer.android.com/sdk/index.html
zuletzt besucht: 19:02.2016

[STAL] „How to connect to my http://localhost web server from Android Emulator in Eclipse"; online:
http://stackoverflow.com/questions/5806220/how-to-connect-to-my-http-localhost-web-server-from-android-emulator-in-eclips
zuletzt besucht: 19:02.2016

[MOBD] "Building and Running from Android Studio" online:
http://developer.android.com/tools/building/building-studio.html
zuletzt besucht: 19:02.2016

[REPO] "Github Repository des Projektes", online:
https://github.com/aernhofer-tgm/DEZSYS11-AndroidRestFulWS2.git
zuletzt besucht: 20:02.2016