**Deckblatt**

**Inhaltsverzeichnis** Seite

1. Die Idee hinter der App

2. Das Grundprinzip der Tanzschul App

3. Die Activitys der Tanzschul App

3.1 Übersicht

3.2 ConnectedActivity

3.3 Funktionen und Informationsaustausch mit dem Server

3. Die Activitys der Admin App

3.1 Übersicht

3.2 Funktionen und Informationsaustausch mit dem Server

3 Server-Client

3.1 Server

3.1 Aufbau

3.1.1 Jetty-Servlettcontainer

3. Der Server

3.1 Die Server Struktur

3.2 Der Aufbau der SQL Datenbank im Server

3.3 Die JFrame zur Nutzer Interaktion

4. Der Client

4.1 Funktionen der Activitys

4.1 Rolle der ConnectedActivity-Klasse

4.2. Eine Übersicht über die Activitys

5 Erklärung des Request–Response Prinzips am Beispiel der LoginActivity

6 Verallgemeinerungen

7 Activitys

5.3 Request-Response Prinzip

6. Danksagung

7. Literaturverzeichnis

8. Eidesstattliche Erklärung

Verschlüsselung /Er Diagramm/ Zyklen/ neuer aufruf

**1. Die Idee hinter der App**

Ich habe, wie viele andere Jugendliche auch, an mehreren Kursen für Standardtänze

teilgenommen. Da alle Standardtänze jedoch Paartänze sind stehen viele Jugendliche vor dem

Problem einen passenden Partner des gegenteiligen Geschlechts zu finden.

In meiner Tanzschule verläuft die Vermittlung größtenteils über Freunde. Die Tanzschulen selber vermitteln über eine ausgehängte Pinnwand, an die Suchende einen Steckbrief anheften können auf den sie dann ihre Kontaktdaten notieren.

Diese Methode ist aus meiner Sicht jedoch äußerst veraltet und umständlich. Denn eine solche

Pinnwand weist an sich keine Ordnungsstruktur auf, zudem muss man sich um mit ihr zu

interreagieren in der Tanzschule befinden. Auch sind kurzfristige Änderungen an den Steckbriefen sind nahezu unmöglich.

Meine App integriert dieses „Pinnwand-Prinzip“ in eine Server-Client basierende App.

Dadurch kann die Zugänglichkeit und Effizienz der Vermittlung zwischen Tanzschülern erhöht werden.

**2. Das Grundprinzip der Tanzschulapp**

Jeder neue Nutzer muss sich vor der Nutzung der App registrieren und einen neuen Account erstellen. Mit diesem kann er sich dann sich dann einloggen um auf weitere Funktionen der App zugreifen. Jeder Nutzer hat ein eigenes Profil dass er nach seinen Vorstellungen anpassen kann. Wenn der User, Nutzern des anderen Geschlechts mitteilen möchte, dass er nach einen Tanzpartner für einen bestimmten Kurs sucht kann er sich für diesen suchend eintragen. Als Alternative dazu kann er sich Nutzer die das bereits getan haben nach Kursstufe und Wochentag sortiert vom Server abrufen. Anschließen kann er das Profil jedes ausgegeben Benutzers einsehen. Im Profil Tab werden alle vom fremden Nutzer angegeben Informationen angezeigt. Insofern man nun Kontakt aufnehmen möchte hat man mehrere verschiedene Möglichkeiten das zu tun darunter auch ein App eigener Chat.

Da die Kurse nicht in jeder Saison zum gleichen Zeitpunkt sattfinden muss es für die Inhaber einer Tanzschule, die meine App verwendet, möglich sein neue Kurse hinzuzufügen und veraltete zu löschen. Zu diesem Zweck habe ich noch eine Admin App programmiert.

**3. Die Activitys der Tanzschul App**

Die Tanzschul App weist für jede ihre Zustände eine eigene Activity auf, das hat mehrere Vorteile gegenüber einem Neuzeichnen der [GUI](https://developer.android.com/guide/topics/ui/index.html) (Graphical User Interface) innerhalb einer einzigen [Activity-Klasse](https://developer.android.com/reference/android/app/Activity.html), für jeden Wechsel des Zustands.

Bei letzterem müssen nämlich mehrere Layouts innerhalb einer Klasse implementiert werden.

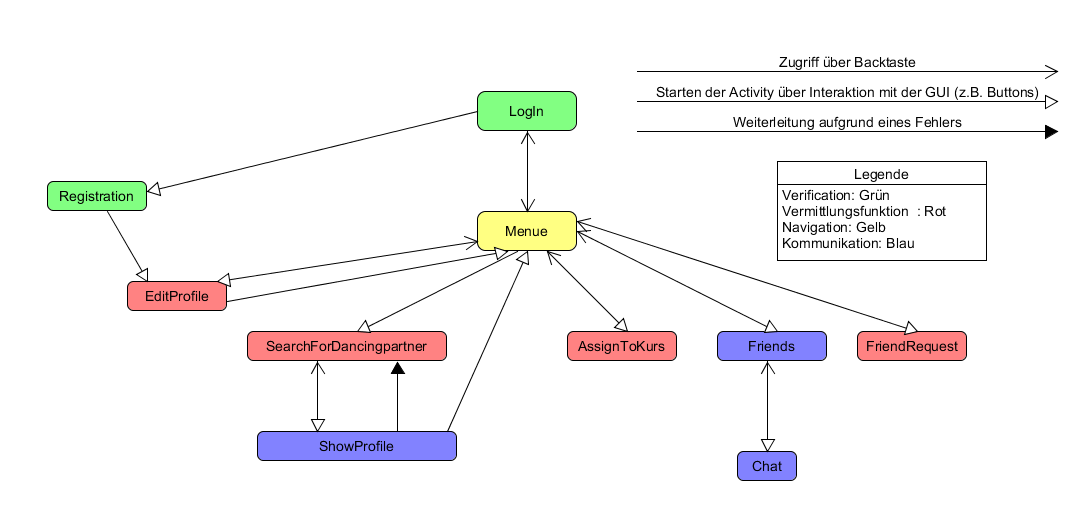
Das führt dazu, dass der Code insgesamt komplizierter gestaltet ist, um die verschiedenen Zustände der App umzusetzen und voneinander abzugrenzen zu können. Beispielsweise muss bei jedem Zustandswechsel unnötiger Ressourcenspeicher freigegeben und Verweise auf nichtmehr zeichenbare GUI-Elemente gelöscht werden.

Des Weiteren wäre die Umsetzung von mehreren Benutzeroberflächen in einer Klasse unnötig lang und würde somit die Codeverständlichkeit und Übersichtlichkeit stark verschlechtern.

Zu guter Letzt müssten bei der neuzeichnenden Variante sehr viele Attribute in der Activity-Klasse und so im Arbeitsspeicher des Geräts abgespeichert werden die nur in bestimmten Zuständen der App überhaupt Verwendung finden.

Beim Verwenden mehrerer Activitys ist die Abgrenzung untereinander deutlich einfacher. Denn in diesem Fall kann der vordefinierte [Lebenszyklus](https://developer.android.com/training/basics/activity-lifecycle/index.html) einer Activity verwendet werden. Dieser bring Methoden wie *onPause*, *onStop* oder *onDestroy* mit sich in denen nicht mehr verwendet Ressourcen freigegeben werden können. Diese werden aufgerufen, wenn die Activity, neugezeichnet, temporär in den Hintergrund verschoben oder beendet wird. Beim Starten einer neuen Activity können wichtige Informationen über Intents weitergegeben werden. Informationen die auch nach dem Beenden der App vorhanden sein sollen können extern z.B. in sog. [SharedPreferrences](https://developer.android.com/reference/android/content/SharedPreferences.html) gespeichert werden.

**3.1 Überblick über die Zustände der App**

****

Diese Grafik zeigt die in der Tanzschul App vorhandenen Aktivitas und so auch alle Zustände. Die Peile veranschaulichen, durch welche Art von User Interaktion der Zustandswechsel ausgelöst wurde und so auch eine neue Aktivitas in den Vordergrund rückt.

Alle grün markierten Aktivitas sind dazu da, den User zu verifizieren. Jeder User hat eine individuelle ID die nur er selbst kennt, diese kann er bei den grün markierten Activitys zugesandt bekommen.

Die gelb markierte Activity dient zur Navigation zwischen den unterschiedlichen Funktionen der App. Zur direkten oder indirekten Interaktion zwischen Usern dienen die rot gefärbten Activitys.

**3.1.1 Die ConnectedActivity-Klasse**

Alle Activitys der Tanzschul App implementieren die abstrakte Klasse *ConnectedActivity.*

Die Methoden dieser Klasse dienen überwiegend zur Behandlung von Fehlern die bei dem versenden von Request auftreten können.

Dabei wird zwischen Verbindungsfehlern und Fehlercodes, die vom Server versandt werden, unterschieden da diese getrennt behandelt werden müssen.

Bei Verbindungs- und Eingabefehlern werden passende Fehlermeldungen ausgelesen. Wohingegen bei unbekannten oder fatalen Fehlern wie z.B. dem Versenden einer falschen Identifikationsnummer wird zum Login-Activity zurückgekehrt wird und alle Activitys aus Stack gelöscht werden, um weitere Folgefehler zu vermeiden.

Es bestand anfangs die Idee beide Arten von Fehlern in einer Methode zu vereinen. Jedoch wird die Methode *onConnectionError()*, die für das Behandeln von Verbindungsfehlern zuständig ist meist von den Kinderklassen übernommen wohingegen die *onError()* Methode in einigen Fällen zur individuellen Reaktion auf Fehlercodes überschrieben werden muss.

**3.1.1 Die LogIn-Activity**

Wie der Name bereits vermuten lässt, müssen sich die User über Activity einloggen. Nutzer die noch keinen Account besitzen werden über den „Registrieren“-Button zur Register-Activity weitergeleitet.

Im Austausch gegen eine passende Kombination von E-Mail und Passwort, rufen die User hier ihre ID vom Server ab. Sind die User dabei erfolgreich rückt die Menue-Activity in den Vordergrund. Nutzer die noch keinen Account besitzen werden über den „Registrieren“-Button zur Register-Activity weitergeleitet. Bei beiden Aktionen wird LogIn-Activity in den Hintergrund verschoben und die finish() Methode aufgerufen, diese teilt dem System mit, dass diese Activity nicht mehr gebrauch wird so, dass sie aus dem Arbeitsspeicher gelöscht wird sobald Android die Ressourcen für eine andere Aktion benötigt.

Die E-Mail- und Passwortkombination wird zudem auf Hakensetzung des Nutzers in sog. SharedPreferrences gespeichert werden (Beschreibung dazu!!).

**3.1.2 Die Menu-Activity**

Dabei ist zu beachten dass die Stufenbezeichnungen der Grundkurse von der Tanzschule Fischer Ingolstadt, an der ich einst selbst Schüler war übernommen wurden. Die höheren Stufen entsprechen dem einheitlichen Medaillensystem der [ADTV](http://www.adtv.de/) geprüften Tanzschulen. Der Wochentag bezeichnet den Tag in dem die wöchentlichen stattfindenden Kurse gehalten werden.

**3. Der Server**

Auf dem Server werden alle möglichen Daten über die User, Kurse, Admins und deren Beziehungen untereinander gespeichert. Öffentliche Daten werden auf Anfrage an die Nutzer gesendet.

**3.1 Die Serverstruktur**

**6. Danksagungen**

Abschließend möchte ich allen Seminarmitgliedern für ihre Offenheit und Hilfsbereitschaft danken. Insbesondere Tim Möschel der mir seine App als Serverseitiges Vorbild zeigte und Dominik Okwieka, Leander Dreier, Oskar Loeprecht und Klaus Hackner die mir jeder Zeit mit Rat beiseite standen.

**7. Literaturverzeichnis**

h

**8. Eidesstattliche Erklärung**

Ich erkläre, dass ich die Seminararbeit nicht ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis aufgeführten Quellen und Hilfsmittel verwendet habe.

Kösching, den 10. November 2016

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Abgabe nach den Ferien 8.11.16