# Abschlussbericht

## 1 Zielbestimmung

Es soll ein System zur Tourenplanung für Touristik und Freizeitgestaltung designt werden.
Die Applikation soll den Namen “Tour2Go” tragen.

### 1.1 Muss-Kriterien

Der Anwender nutzt das System mit Hilfe einer nativen Android Applikation.
In der Applikation muss jeder Anwender ein Benutzerkonto erstellen oder sich mit einem bestehenden Benutzerkonto anmelden können.
Die persönlichen Daten müssen jeder Zeit vom Anwender bearbeitet werden können.
Der Kerninhalt der Applikation besteht darin, dass Anwender neue Touren erstellen können.
Für eigene, schon vorhandene Touren gibt es Funktionalitäten diese zu Laden und zu Bearbeiten.
Eine Tour setzt sich aus einer sortierten Reihenfolge von Wegpunkten zusammen, welche der Anwender der Tour hinzufügen und wieder entfernen kann.
Die Applikation zeigt dem Anwender seine aktuelle Position auf einer Karte, auf welcher auch die Wegpunkte der Tour abgebildet sind.
Die Verbindung der Wegpunkte wird als Tour visuell auf der Karte dargestellt.
Für die Angaben der Positionierung wird das GPS-Netz verwendet.
Beim Starten der Applikation sieht der Anwender eine Karte des aktuellen Gebiets mit seiner Position und kann sich öffentliche Touren von anderen Anwendern anschauen. Jedoch sind alle anderen Funktionalitäten ohne Registration nicht verfügbar.

### 1.2 Soll-Kriterien

Der Anwender bekommt in der Applikation eine Top 10 von “Points Of Interest” angezeigt, die aus externen Quellen stammen.
Die "Points Of Interest" sollen der aktuellen Tour als Wegpunkte hinzugefügt und entsprechend auch wieder entfernt werden können.
Der Anwender soll die Möglichkeit haben sich Details zu einem "Points Of Interest", wie Kontaktdaten, Öffnungszeiten, Bewertungen und Ähnliches, anzeigen zu lassen.
Über die Applikation sollen der aktuellen Tour auch Medien, wie Bilder oder Videos, als neuer Wegpunkt hinzugefügt werden können.

## 1.3 Kann-Kriterien

Mit Hilfe der “Passwort vergessen”-Funktionalität kann der Benutzer sein Passwort zurücksetzten lassen.

Die Anzeige der “Points Of Interests” lässt sich nach bestimmten Kriterien, wie z.B Entfernung, Kategorie oder Bewertung, vom Anwender selbst filtern.

Die Applikation ermöglicht das Teilen von Touren als Nachricht in sozialen Netzwerken, wie z.B. Facebook oder Google+.

Es können auch einem bestehenden Wegpunkt einer ausgewählten Tour Bilder oder Videos zugeordnet werden.

Anwender können ihre eigenen Touren für andere Anwender öffentlich zugänglich machen.

So haben diese die Möglichkeit sich öffentliche Touren anzusehen, aber diese nicht zu verändern.

Jedoch können die öffentlichen Touren kopiert werden und stehen so als eigene Tour für die Bearbeitung zur Verfügung.

Die Kalenderfunktionalität beinhaltet eine zeitliche Übersicht aller eigenen Touren.

Des Weiteren ermöglicht die Applikation das Live-Tracken einer Tour über eine separate Ansicht.

Für diese Funktionalität wird eine Auswertung visuell in Form von Diagrammen und einer Karte dargestellt bzw. eine Zusammenfasssung der wichtigsten Tourdaten ausgegeben.

## 1.4 Abgrenzungkriterien

Die Applikation bietet keine automatisierte Routenplanung anhand bestimmter Parameter.

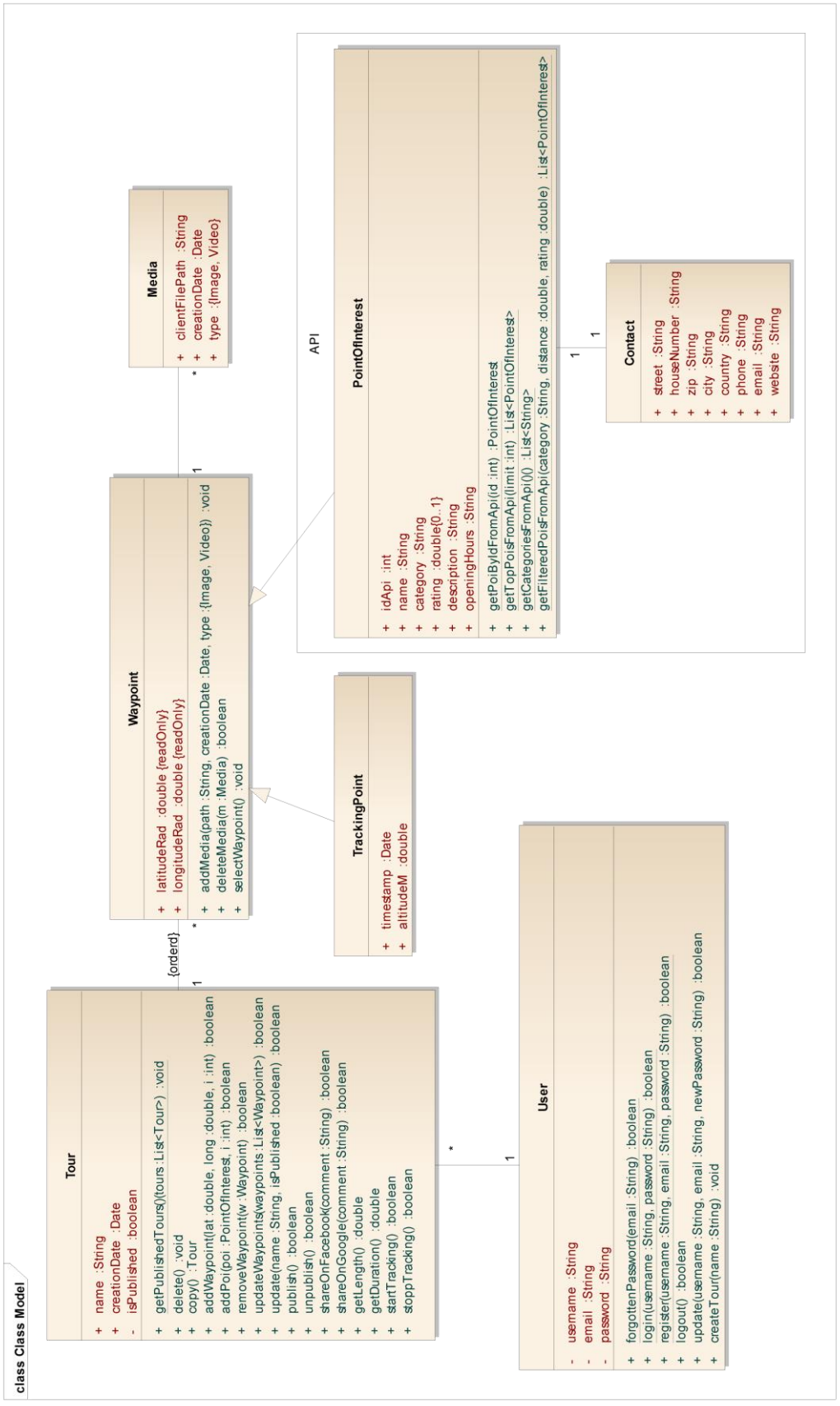
Die Applikation wird kein offizielles Verzeichnis für Sehenswürdigkeiten, öffentliche Gebäude etc. bereitstellen.

Die Applikation bietet keine Möglichkeit “Points Of Interest” zu bearbeiten oder dem System hinzuzufügen.

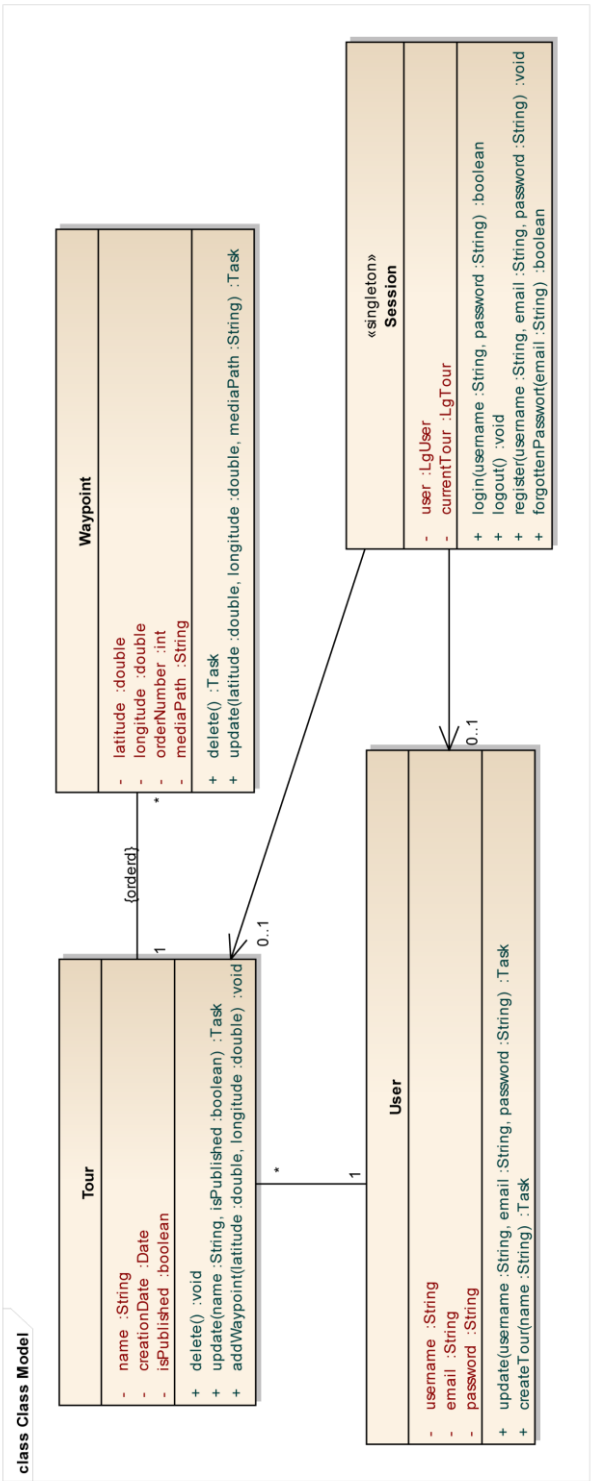
Die Anwendung nutzt ausschließlich das GPS-Netz zur Positionsbestimmung. Endgeräte, die das GLONASS oder COMPASS Netz nutzen sind nicht kompatibel.

# 2 Diagramme

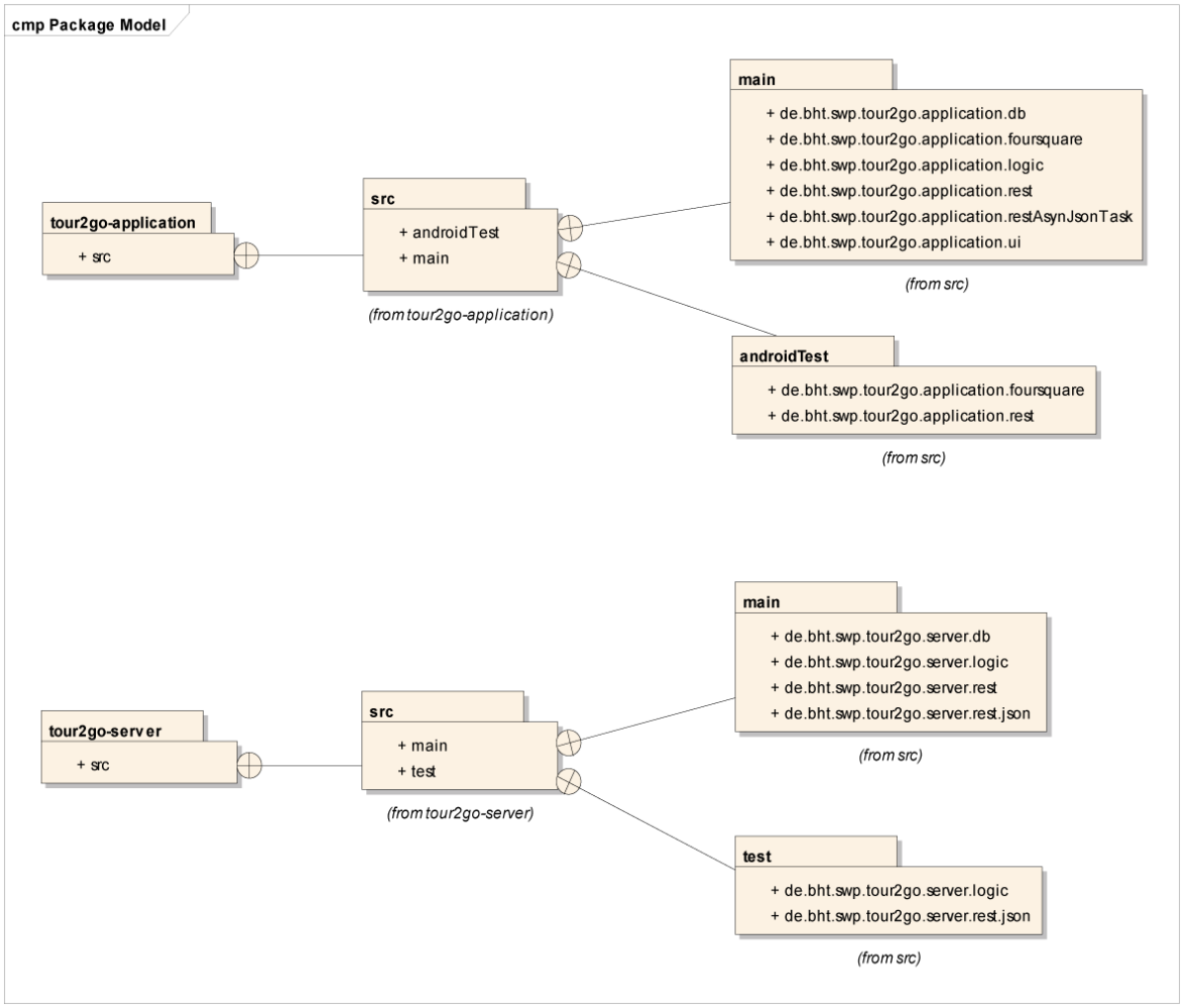
## 2.1 Klassendiagramm (Anforderung)



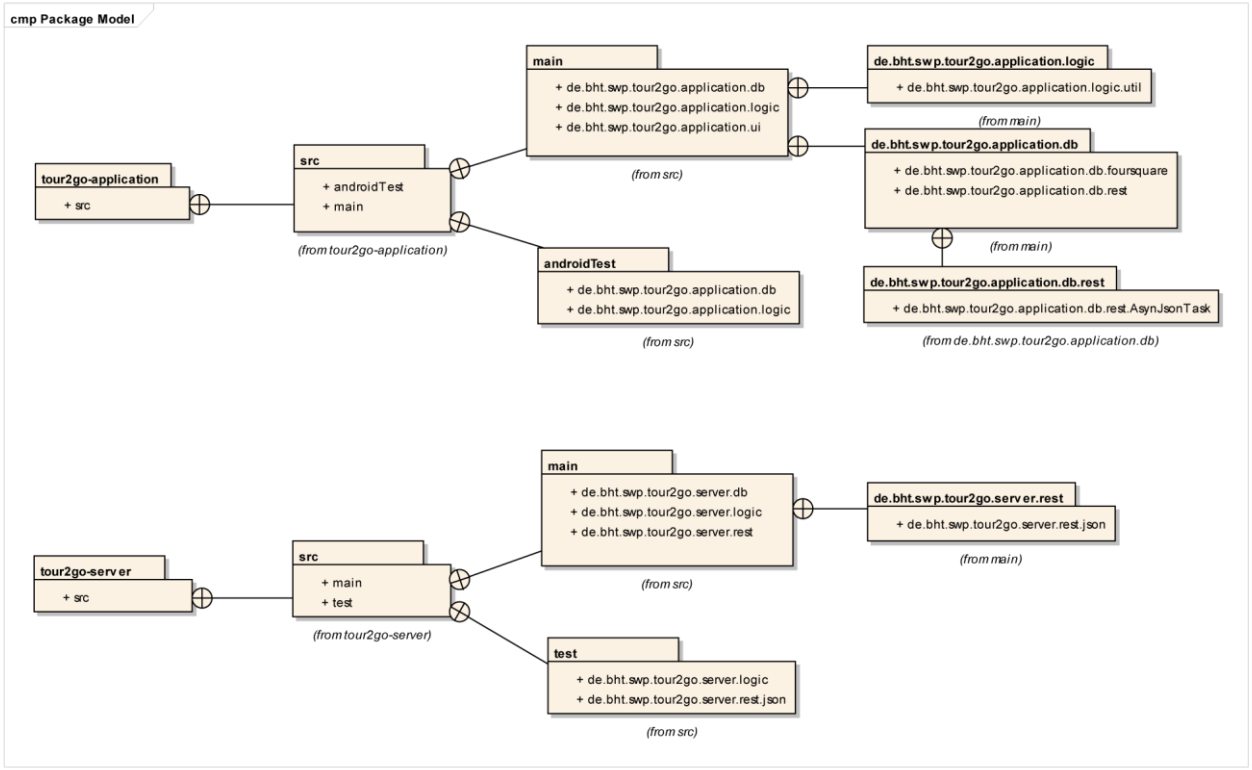
## 2.2 Klassendiagramm IST-Zustand



## 2.3 Package Diagramm (Anforderung)



## 2.4 Package Diagramm (IST-Zustand)



## 3 Technikeinarbeitungsplan

| Bereich | Technologie | Zuständigkeit | Backup |
|---|---|---|---|
| Oberfläche | Android SDK | Melanie Sommer | Christian Gehrke |
| | Google Maps API (Karte) * | Melanie Sommer | Christian Gehrke |
| | Foursquare API (Points Of Interest) | Robert Sarnighausen | Nils Kienöl |
| | | | |
| Persistenz | JPA (OR Mapper) | Christian Gehrke | Melanie Sommer |
| | | | |
| Server / | Jetty | Nils Kienöl | Robert Sarnighausen |
| Datenbank | REST-Server | Nils Kienöl | Robert Sarnighausen |
| | REST-Client | Robert Sarnighausen | Nils Kienöl |
| | | | |
| Projekt Hosting | Bitbucket | Robert Sarnighausen | Nils Kienöl |
| | | | |
| Entwicklungstools | Gradle | Robert Sarnighausen | alle |
| | Android Studio (Codestyle) | Melanie Sommer | alle |
| | | | |
| Dokumentation | | Melanie Sommer | |

# 4 IDE-, Build-, Ausführungsanleitung

## 4.1 4.1 IntelliJ & Android SDK Installation

### *Schritt 1*

- IntelliJ 13.1.3 runterladen unter http://www.jetbrains.com/idea/download/

### *Schritt 2*

- Android SDK runterladen unter http://developer.android.com/sdk/index.html

### *Schritt 3*

- IntelliJ starten → configure → Project Defaults → Project Structure → SDKs → + → Android SDK → Pfad zum Android SDK angeben

### *Schritt 4*

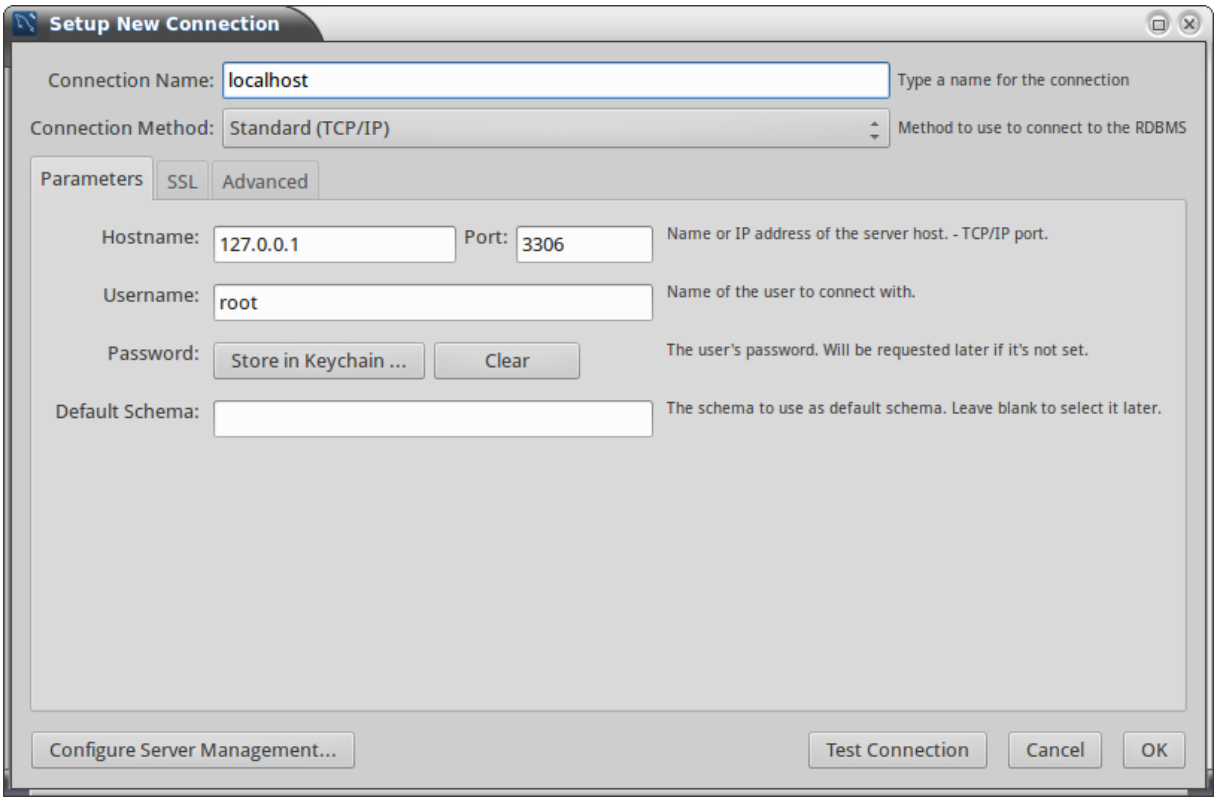
- Android SDK Manger starten → Haken bei Tools, Extras und Android 4.1.2(API 16) bis Android L (API 20 L, preview) setzen.
- "Accept License" klicken und dann "Install" → warten → Dialog schliessen
- Jetzt unter "Build Target" - "Android 4.2.2" wählen → OK → Apply → OK
- Import Project → root Ordner des Projekts wählen → OK → Import Project from external model → Gradle → OK → Finish

## 4.2 4.2 Xampp

- Xampp hier herunterladen: https://www.apachefriends.org/de/index.html
- Xampp installieren und Starten → im Tab "Manage Servers" wählen → "MySQL Database" wählen und rechts den "Start"- Button drücken

## 4.3 4.3 MySQL - Workbench

- herunterladen und installieren: http://www.mysql.de/products/workbench/
- MySQL Workbench starten und im Hauptscreen bei MySQL Connections auf das "+" Icon klicken.
- OK klicken
- rechts klicken auf die eben erstellte Connection → "Create new shema in the connected database" → bei name: "d01a7eac" eintragen → apply




## 4.4 4.4 Google Maps Key

- API Key anfordern wie auf https://developers.google.com/maps/documentation/javascript/tutorial#api_key beschrieben.

- Den erhaltenen Key im Projekt tour2go-project → tour2go-application → src → main → res → values → string.xml in Zeile 53 anstelle des alten einsetzen.

## 4.5 4.5 Unit Tests

- Auf Clientseite werden die Tests nicht automatisch während des Build Prozesses ausgeführt. Dies ist sinnvoll, da die Tests nicht in der JVM auf dem Rechner, sondern in der Dalvik VM auf einem Gerät/Emulator ausgeführt werden müssen.

- Zum Starten der Client Tests → Rechtsklick auf “de.bht.swp.tour2go.application.logic.LgTestSuite” oder „de.bht.swp.tour2go.application.db.DataProviderTest“ im Project Explorer → run (mit Android Logo!) → Device wählen (über USB-Verbindung oder Emulator)

## 4.6 4.6 Build

- Der Build der Anwendung erfolgt über die Ausführung des Tasks tour2go-application.
- Der Build der Serverseite erfolgt über die Ausführung des Tasks tour2go-server.

# 5 Programmierrichtlinien

Wir haben sowohl nach den in der Vorlesung vorgestellten, als auch nach den üblichen in Java geltenden Code-Konventionen gearbeitet. Zudem haben wir unsere Anwendung server- und clientseitig in einem 3-Schichten Paketmodell (db, lg, ui) strukturiert, um Zuständigkeiten abzugrenzen und den Zugriff auf kritische Methoden einzuschränken. So verhindern wir zum Beispiel, dass die Konstruktoren der Client-Logikklassen direkt aufgerufen und so nicht korrekt persistierte Daten erstellt werden können. Es wurden mit Transaction (serverseitig) und Task, bzw AsyncTask, (clientseitig) Klassen für die Transaktionssicherheit implementiert. Die Funktionalität, insbesondere komplexer und aufwändiger Methoden, wurde mittels JUnit-Tests sichergestellt.

# 6 Schlussbewertung

Mit der Entwicklung einer nativen Applikation unter Android haben wir nicht nur ein uns unbekanntes Framework als Schwerpunkt unseres Softwareprojektes gewählt, sondern uns gleichzeitig der Entwicklung auf einem bisher unbekannten Betriebssystem geöffnet. Dies war zum einen ein spannendes Themenfeld, zum anderen jedoch auch mit einigen unerwarteten Problemen bei der Umsetzung verbunden.

Während man für Android zwar in der uns vertrauten Programmiersprache Java entwickelt, bewegt man sich dennoch in einem Framework, welches sich von Abstraktionsgrad und Konzepten stark von dem unterscheidet, was bisher im Studium umgesetzt wurde. Exemplarisch sei hier zum Beispiel der Lebenszyklus der Komponenten einer Applikation genannt, mit dem ein starker Eingriff des Betriebssystems in den sequentiellen Ablauf des Programms einhergeht und so eine aufmerksame Konsistenzprüfung und Transaktionssicherheit der Daten nötig ist.

Als besondere Herausforderung haben sich auch Debugging und Fehlerbehandlung unter Android herausgestellt. Zum einen ist die Handhabung von ABD (Android Debugging Bridge) und ART (Android RunTime) hier sehr umständlich, zum anderen fehlt für eine gute globale Fehlerbehandlung der geeignete Anknüpfungspunkt, da geworfene Fehler schnell in der Komplexität des Frameworks verschwinden und nicht abgefangene Fehler die Applikation ohne Meldung/Stacktrace zum Absturz bringen.

Serverseitig haben wir zur Bearbeitung der Http-Anfragen mit JAX-RS auf ein Restful-Webservice Framework gesetzt, welches im Wesentlichen mit Annotationen konfiguriert wird. Hier war die korrekte De-/Serialisierung der Java-Klassen von und nach JSON von besonderer Schwierigkeit, u.a. dann wenn verschachtelte Klassen oder komplexe Daten (z.B. Timestamps) in die passenden Formate überführt werden mussten. Für das Persistieren der Daten in der Datenbank haben wir uns das Object Relational Mapping mittels JPA (Java Persistence Api) erarbeitet. Mit JPA müssen, im Gegensatz zum Hibernate-Framework, viele Datenbankoperationen und Darstellungen von Beziehungen der Entitätsklassen komplexer umgesetzt werden, was mit zusätzlichem Aufwand und Schwierigkeiten verbunden war.