



# Projektbericht Skatenight-App

Projektseminar „Android-Programmierung“ im WS 14/15

Bernd Eissing, Pascal Otto, Daniel Papoutzis,  
Tristan Rust, Richard Schulze, Martin Wrodarczyk

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
<b>2</b>	<b>Vorgehensweise</b>	<b>3</b>
<b>3</b>	<b>Features der Apps</b>	<b>3</b>
<b>4</b>	<b>Das Android-SDK</b>	<b>3</b>
<b>5</b>	<b>Server-Backend</b>	<b>3</b>
5.1	Anforderungen . . . . .	3
5.2	Server, Keys und Build-Varianten . . . . .	4
5.3	Klassenstruktur der Skatenight API . . . . .	5
5.4	Datenmodell . . . . .	6
5.5	Verfügbare API-Aufrufe . . . . .	7
5.6	Google Cloud Messaging . . . . .	7
5.7	Feldberechnung . . . . .	7
<b>6</b>	<b>Frontend</b>	<b>7</b>
<b>7</b>	<b>Testing</b>	<b>7</b>

# 1 Einleitung

## 2 Vorgehensweise

von Tristan, Richard

## 3 Features der Apps

von Daniel

## 4 Das Android-SDK

von Pascal

## 5 Server-Backend

### 5.1 Anforderungen

von Richard

Zur Kommunikation der Apps, aber auch zur zentralen Speicherung der Veranstaltungen, Routen, Nutzern und Nutzergruppen ist ein Backend, im folgenden auch Skatenight API genannt, notwendig, das eine öffentlich erreichbare Schnittstelle zur Verfügung stellt. Neben der Speicherung von Daten ist aber auch notwendig, dass der Server eigene Berechnungen durchführen kann. So wird zum Beispiel das Feld, das den Benutzern in der User-App angezeigt wird, zentral auf dem Server berechnet und von den User-Apps abgerufen (siehe Abschnitt 5.7).

Eine weitere wichtige Anforderung war für uns, dass der Server kostenlos nutzbar ist und nach Möglichkeit bereits ein Grundgerüst bietet, über das aus Android-Apps möglichst einfach auf die Server-API zugreifen können. Wir wollten Hauptaugenmerk auf die Entwicklung der Apps legen und die Zeit zur Entwicklung des Backends möglichst gering halten. Das Google App Engine Projekt bietet diese Möglichkeit, da es von den realen Servern vollkommen abstrahiert. Zusätzlich dazu werden einige nützliche APIs angeboten, die unter anderem Funktionalität zur Speicherung von Daten bereitstellen und die Definition von Kommunikations-Endpunkten, sowie die Kommunikation vom Server zum Handy ermöglichen.

## 5.2 Server, Keys und Build-Varianten

von Richard

Während der Entwicklung der Apps ist uns schnell aufgefallen, dass aufgrund der verschiedenen Features, die jeweils auf eigenen Branches entwickelt wurden, auch der Server von mehr als einer Person angepasst wurde. Damit einher ging, dass zwischenzeitlich verschiedene Versionen des Servers notwendig waren, damit jede Teilgruppe des Teams das eigene Feature implementieren konnte, ohne die Entwicklung anderer Teammitglieder zu stören. Aus diesem Grund haben wir mehrere Projekte bei der Google App Engine angelegt, zwischen denen relativ einfach gewechselt werden kann. In unserem Projekt in den Modulen „app“ und „veranstalterapp“ gibt es entsprechende Build-Varianten, mit denen der gewünschte Server ausgewählt werden kann. Da es im Modul „SkatenightBackend“ nicht die Möglichkeit gibt Build-Varianten zu definieren, gibt es dort die Befehle „applyProductionserverConfig“, „applyTestserverConfig“ und „applyJenkinsserverConfig“. Diese kopieren jeweils aus den entsprechenden Unterordnern des src-Ordners die Konfigurationsdateien, die für einen Server-Wechsel notwendig sind, in den main-Ordner. Bei einem Serverwechsel wird die „Constants.java“ und die Projekt-ID in der Datei „appengine-web.xml“ ausgetauscht. Wenn anschließend der Befehl „appengineUpdate“ zur Aktualisierung des Servers aufgerufen wird, wird der Server aktualisiert, dessen Projekt-ID in der „appengine-web.xml“ steht. Die folgende Tabelle listet noch einmal die Details zu den einzelnen Google App Engine Projekten und auch die notwendigen Keys auf, mit denen sich die Apps gegenüber dem Backend und anderen Diensten identifizieren.

Web-Client ID: Dient der Authentifizierung über OAuth2, das die gesicherten API-Funktionen, wie beispielsweise das Anlegen eines Events schützt.		
Key	Release	37947570052-dk3rjhgran1s38gscv6va2rmv2bei8r.apps.googleusercontent.com
	Debug	644721617929-unb9em0kl73b9evdv2h52ufn26fao20p.apps.googleusercontent.com
	Jenkins	1032268444653-7agre4q3eosqhlh92sq62hf5fan9jbv5.apps.googleusercontent.com
Client-ID der User-App: Dient der Identifikation der User-App am Backend.		
Key	Release	37947570052-g006o3ovfotnjqreltom6c7cbktm7dap.apps.googleusercontent.com
	Debug	644721617929-mk0do3jm5lec1dasijdj3220ot87gbn7.apps.googleusercontent.com
	Jenkins	1032268444653-kj1mpisvlrpl67e7db2fu2005aube7mu.apps.googleusercontent.com
Client-ID der Veranstalter-App: Dient der Identifikation der Veranstalter-App am Backend.		
Key	Release	37947570052-4ru7asfhnrmjmmqvj3qdp2rrp31fudf3.apps.googleusercontent.com
	Debug	644721617929-kbcha00vb3j30sh9at05sagm73iltqqo.apps.googleusercontent.com
	Jenkins	1032268444653-83ih5mp5mguh66c012u0sobqe4oqoupr.apps.googleusercontent.com
Google-Maps API-Key für die User-App: Wird zur Nutzung der Google-Maps API in der User-App benötigt.		
Key	Release	AIzaSyBKGVQHij0JcJgHyCnT0ZrLihbCn8Av2jg
	Debug	AIzaSyDQgEKF42jjm57x7kgnY7EI62CcXW_zDS0
	Jenkins	AIzaSyAQ47zRjgPRk1sxJYifd2URQdEnk5HvnRw
Google-Maps API-Key für die Veranstalter-App: Wird zur Nutzung der Google-Maps API in der Veranstalter-App benötigt.		
Key	Release	AIzaSyBedRi4A-p3LNQLy5lgOsBEDIdKuLbi2U
	Debug	AIzaSyAO4zGlaKexByZmEiFKLbo18Y_I-t9KbMc
	Jenkins	AIzaSyBes8RSzVdqcQY-vnj9GLWxCLBKAFcZzh98
Google Cloud Messaging API-Key: Dient der Registrierung beim GCM-Dienst von Google in der User-App.		
Key	Release	AIzaSyDO8mosWwYXjjGZ9besu9CZw1LDEEXrFXE
	Debug	AIzaSyCpwhxda1Lb5E61_fybZq2iSJgViZu3QNM
	Jenkins	AIzaSyBjeKzx_vzkUgQZdT83BB0BMD3gFebpet0

### 5.3 Klassenstruktur der Skatenight API

von Richard

Das Backend ist eine in Java geschriebene Anwendung, die sich im Projektordner in dem Modul „SkatenightBackend“ befindet. Das für die Apps zugängliche Interface wird in der Klasse „SkatenightServerEndpoint“ beschrieben. Alle darin implementierten öffentlichen Methoden sind über die generierten Client-Libraries aufrufbar.

Die Klasse „EventStartServlet“ wird über einen Cron-Job, der in der Datei „cron.xml“ im WEB-INF-Ordner des Moduls definiert ist, alle zwei Minuten aufgerufen. Ihre Aufgabe ist es nach Events zu suchen, die kürzlich begonnen haben und für die noch keine GCM-Nachricht an die teilnehmenden Benutzer versandt wurde. Problematisch war, dass Cron-Jobs bei einem Google App Engine Projekt nicht dynamisch zur Laufzeit erstellt werden können. Die zunächst angedachte Lösung für jedes Event einen eigenen Cron-

Job anzulegen, der genau zum Startzeitpunkt des Events aufgerufen wird, ist dadurch nicht möglich. Wir haben uns deshalb dafür entschieden, die Events mit einem Boolean zu markieren, wenn sie gestartet sind, sodass in der „doGet()“-Methode nach Events gesucht werden kann, für die diese Markierung noch nicht gesetzt wurde. Für jedes startende Event kann dann eine Liste der GCM-IDs der Benutzer erstellt werden, die über das Event benachrichtigt werden müssen. Gleichzeitig wird für die Benutzer auch die aktuelle Event-ID („currentEventId“) gesetzt, die für die Berechnung des Feldes relevant ist.

## **5.4 Datenmodell**

von Richard

Abbildung 1 zeigt das auf dem Server repräsentierte Datenmodell. Die Attribute, die mit einer dickeren Linie gestrichelt umrandet sind Listen von Objekten, die serialisiert werden und nicht als eigener Typ in der Datenbank existieren. Die Veranstalter werden als Hosts in der Datenbank abgespeichert, haben jedoch keine weiteren Beziehungen zu anderen Entitätsklassen.

## **5.5 Verfügbare API-Aufrufe**

von Richard

## **5.6 Google Cloud Messaging**

von Richard

## **5.7 Feldeberechnung**

von Pascal

# **6 Frontend**

# **7 Testing**

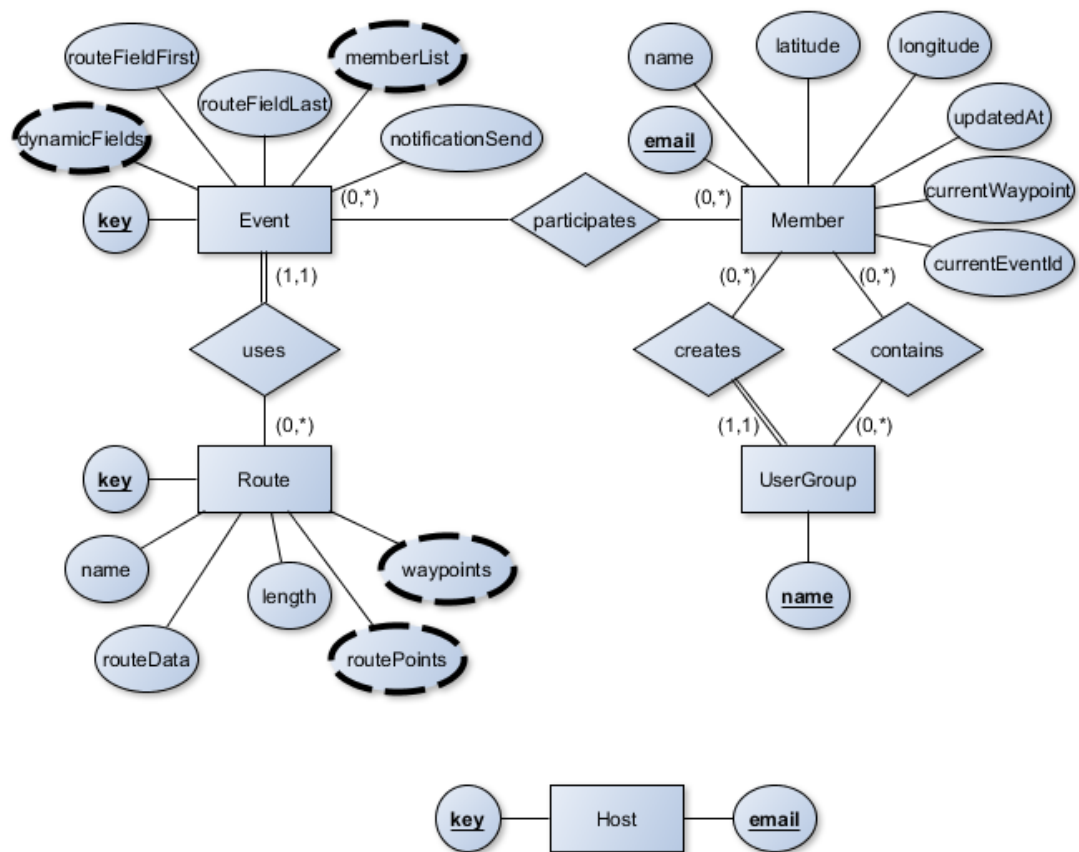


Abbildung 1: ER-Modell der Daten, die auf dem Server gespeichert werden