
Duale Hochschule Baden-Württemberg
- Standort Stuttgart -

Fakultät für Technik

# Studienarbeit "Entwicklung einer Running-App für Android"

**im Studiengang "International Applied Computer Science"**

**Thema:** Entwicklung einer Running-App für Android

**Autoren:** Roland Weinreich
roland.weinreich@gmail.com
MatNr. 5885985

Franz Flintzer
fflintzer@gmail.com
MatNr. 8677472

**Version vom:** 5. Juni 2015

**Betreuer:** Prof. Dr. Karl Friedrich Gebhardt

# Zusammenfassung

# Abstract

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis

# Listingverzeichnis

# Abkürzungsverzeichnis

CMS ..... Content Management System
CSS ..... Cascading Style Sheets
ERM ..... Entity Relationship Modell
GNU ..... GNU is not Unix
GPL ..... GNU General Public License
GUI ..... Graphical User Interface
HTML ..... Hypertext Markup Language
IM ..... Instant Message
JS ..... JavaScript
JSON ..... JavaScript Object Notation
KPI ..... Key Performance Indicator
LGPL ..... GNU Lesser General Public License
OCR ..... Optical Character Recognition
RSS ..... Really Simple Syndication
SQL ..... Structured Query Language
TDD ..... Test-driven development
UGC ..... User Generated Content
WWW ..... World Wide Web
XMPP ..... Extensible Messaging and Presence Protocol

# 1 Einleitung

Als duale Studenten an unterschiedlichen Praxis- und Theoriestandorten gibt es längere Zeiträume in denen man Familie und Freunde nicht sehen kann. Das gemeinsame Joggen in der Freizeit fällt dann natürlich auch weg. Dazu kommen verschiedene Arbeitszeiten und teils sogar verschiedene Zeitzonen, die ein Treffen schon zeitlich nicht zulassen würden.

Besonders fehlen dann die Motivation überhaupt Sport zu machen sowie der gegenseite Ansporn durchzuhalten oder das Tempo zu steigern.

Genau hier möchten wir mit unserer Android-App ansetzen, die es Läufern erlaubt, zeitlich und vom Ort unabhänigig gegeneinander anzutreten.

# 2 Architektur

## 2.1 Überblick

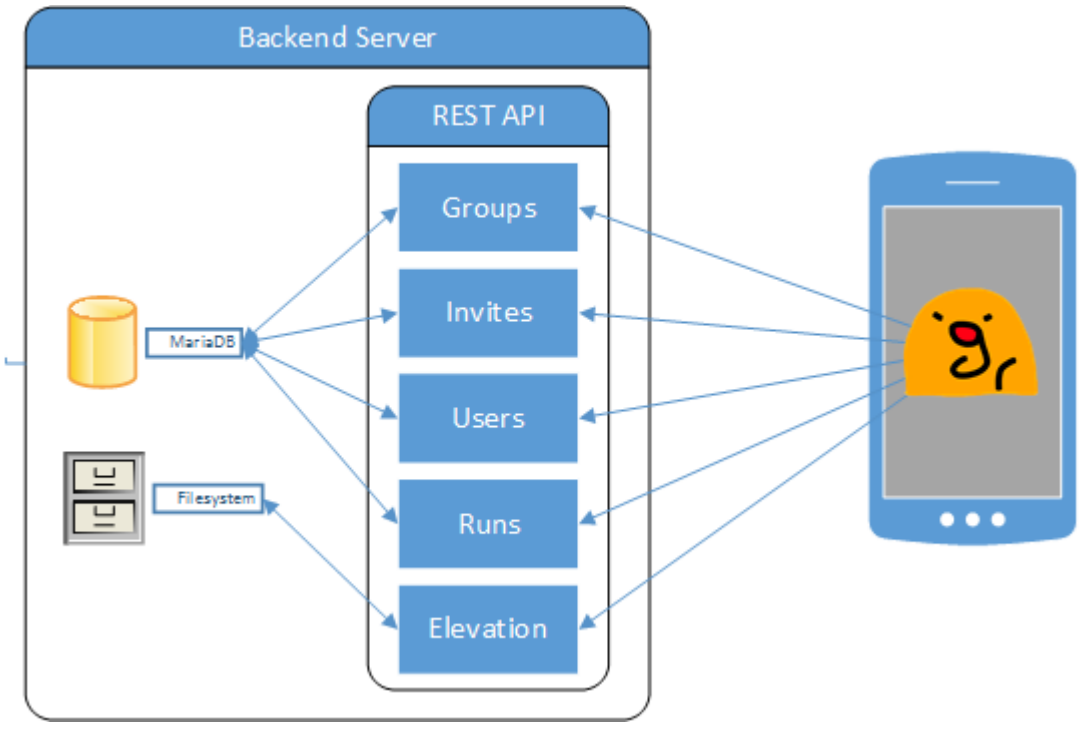


Abbildung 1: Kommunikation zwischen App und Backend-Server

Im folgenden erkläre ich die auf der Abbildung dargestellten Komponenten und warum diese für das Projekt gewählt wurden.

## 2.2 Backend Server

### 2.2.1 Datenbank

Als Datenbankverwaltungssystem haben wir uns entschieden “MariaDB” zu nutzen. MariaDB ist das Pendant zu “MySQL”, jedoch ist es schneller, sicherer und hat eine aktivere Community. Beide Teammitglieder könne bereits auf längere Erfahrung in MySQL zurückgreifen. Die Arbeit mit MariaDB war aufgrund des weitgehend identischen Befehlssatzes sehr unkompliziert.

### 2.2.2 REST API

Um der App eine geeignete Schnittstelle zu Daten über das Internet bereitzustellen haben wir uns für eine REST API entschieden. REST steht für “Representational State Transfer”, einem stetig wachsenden Architekturmodell für Client-Server-Kommunikation bei Web Services.

**REST Allgemein** REST ist zustandslos, d.h. jede Nachricht enthält genügend Informationen um den Inhalt zu verstehen. Das ist besonders wichtig mit Augenmerk auf Skalierbarkeit. So ist die Lastenverteilung der REST-Anfragen auf mehrere Systeme sehr einfach, da keine sitzungspezifischen Daten o.Ä. zwischen den Servern synchronisiert werden müssen. Das gleiche gilt natürlich auch für High-Availability-Systeme.

REST schöpft den HTTP Funktionsumfang weitgehend aus, um u.A. erweitertes Caching und eine hohe Selbsterklärungsfähigkeit zu garantieren.

So erlaubt die Operation “POST” es, neue Ressourcen zu erstellen.

“GET” ermöglicht es, Resourcen anzufragen. “GET” ist nullipotent, d.h. es werden unter Garantie keine Änderungen an der Ressource stattfinden.

Mit “PUT” kann man eine Ressource ersetzen, mit “DELETE” löschen. Diese beiden Operationen bezeichnet man als “idempotent”. Das bedeutet dass die Anfrage das gleiche Ergebnis produzieren wird, egal wie häufig diese wiederholt wird.

Jeder Ressource ist genau eine URL zugeordnet. Beispielsweise bekommt so der Client mit einer Anfrage mit der Operation “GET” auf “myserver.com/run” die Liste aller Läufe zurück, während die gleiche Anfrage auf “myserver.com/run/15” nur den Lauf mit der ID 15 zurückgibt.

Ein weitere Eigenschaft von REST ist es, dass die Kommunikationsform nicht standardisiert ist. So ist es möglich, anhand von Daten verpackt in JSON, HTML, XML oder auch in einem komplett eigenen Format zu kommunizieren.

**Spring Framework und Tomcat** Für die Realisierung der Webservices haben wir uns entschieden das Spring Framework zu nutzen. Ersteres wurde gewählt da es bei einem Teammitglied relevant für die Bachelorarbeit wird und sich so in das Framework eingearbeitet werden konnte.

Das quelloffene Framework erlaubt es u.A. skalierbare, performante, reichhaltige Web-Services aufzusetzen. Es ist möglich spezifische Sicherheitsmaßnahmen umzusetzen, JSON, XML o.Ä. automatisch zu serialisieren und deserialisieren, Anfragen zu validieren usw., d.h. es ist mehr als ausreichend für den Rahmen dieser Studienarbeit.

Die Programmierung im Spring-Framework erfolgt mit Java (mit zusätzlichen Annotationen) und XML. Es vereinfacht das Programmierprinzip Dependency Injection durch wahlweise das automatische Einlesen von hinterlegten XML-Dateien oder Annotationen im Java-Code. Auch das Programmierparadigma "Aspektorientierte Programmierung" wird dem Programmierer nahegelegt. Spring besteht aus vielen kleineren Frameworks, von denen einige Verwendung im Projekt gefunden haben:

- Sping Core - Das "Kern"-Framework.
- Spring Web - Um Web-Applikationen zu erstellen.
- Spring Web MVC - Fügt ein DispatcherServlet hinzu dass Anfragen an Handler/Controller weiterleitet und damit das MVC-Programmiermuster stark vereinfacht.[1].
- Spring Security - Die Sicherheitskomponente für das Spring Framework.

Als Application Server wurde Tomcat gewählt, da beide Teammitglieder diesen vorher bereits mehrfach aufsetzen durften und eventuell spätere Sicherheitsmaßnahmen wie das Einbinden eines Zertifikats um den HTTP-Verkehr über SSL zu verschlüsseln in wenigen Minuten möglich ist.

**JSON** Als Format für den Austausch von Daten zwischen Server und Client haben wir uns für JSON entschieden.

JSON weist im Vergleich zum stärksten Konkurrenten XML eine hohe Lesbarkeit auf und lässt sich sowohl auf dem Server als auch auf der App sehr einfach von und zu Java-Objekten serialisieren und deserialisieren.

Auf dem Server übernimmt diese Aufgabe Jackson, auf der App Gson.

**Maven** Das manuelle Einbinden von JARs in Java-Projekte stellt eine Herausforderung da. Hierbei müssen diese von vertrauenswürdigen Quellen manuell heruntergeladen werden, in den richtigen Ordner verschoben und importiert werden.

Um den Aufwand zu minimieren nutzen wir Maven, welche sich selbst um das Herunterladen und Einbinden der Java-Archive kümmert. Dabei ist auch das Aktualisieren der Pakete durch das Ändern einer einzigen Zeile in der Maven-Konfigurationsdatei möglich.

[1] Introduction to Spring Web MVC framework, vgl. [JHD+] [S.437]

### 2.2.3 App

Die App ist für Android 5.0 ab API 21 geschrieben.

Für das Betriebssystem Android haben wir uns entschieden da beide Teammitglieder ein Android-Smartphone besitzen. Da es sich bei dieser Studienarbeit um kein Projekt mit wirtschaftlicher Motivation handelt und die API einige Verbesserungen gegenüber älteren Versionen mitbringt hielt uns von der Entscheidung auch nicht die relativ geringe Verbreitung von 12,4% der APIs 21 und 22 ab[2] (Stand 1. Juni 2015).

# 3 Analyse

## 3.1 Einleitung

Die App soll es ermöglichen, einen Lauf mitzuschneiden, um ihn anschließend an eine Gruppe an Freunden zu veröffentlichen. Diesen ist es dann möglich, mit ihren Freunden zu laufen bzw. gegen diese anzutreten.

Um diese Funktionalität zu ermöglichen sind das Schreiben einer Android-App und das Aufzetzen eines Webserver nötig.

## 3.2 Funktionale Anforderungen

### 3.2.1 Webserver

Der Webserver stellt eine Schnittstelle bereit, die einem Klienten nach der Authentifizierung folgende Funktionalitäten bereitstellt:

- Nutzer abzufragen, anzulegen und zu löschen,
- Läufe abzufragen, anzulegen und zu löschen,
- Gruppen abzufragen, anzulegen und zu löschen,
- Nutzer zu Gruppen hinzuzufügen oder zu entfernen,
- Freunde einzuladen und
- Kacheln für Höhendaten anzufordern und herunterzuladen.

### 3.2.2 Android App

Die Android App soll dem Nutzer folgenden Funktionalitäten bereitstellen:

- Anmeldung beim Webserver
- Registrierung beim Webserver

[2] Dashboards, vgl. [AOSb]

- Abmeldung vom Webserver
- Laufen ohne währenddessen gegen Freunde anzutreten
- Laufen um währenddessen die Position der Freunden mitverfolgen zu können
- Eintreten in Gruppen, Austreten aus Gruppen und Erstellen von Gruppen
- Einladen appfremder Nutzer in eine Gruppe
- Einladen anderer Nutzer der App in eine Gruppe
- Weitergabe des Administratorprivilegs innerhalb der Gruppe durch den Gruppenadministrator

## 3.3 Nichtfunktionale Anforderungen

An den Webserver werden folgende Anforderungen gestellt:

- Die Authentifizierung muss sicher sein, sodass das Passwort des Nutzers und seine persönlichen Daten sicher vor Fremdzugriff und Manipulation durch Unbefugte sind.
- Der Server muss innerhalb von 5s Sekunden reagieren.
- Mobile Datenbeschränkungen müssen bedacht werden, sowohl in Bezug auf Geschwindigkeit als auch Datenmengen.

### 3.3.1 Usability

Die App muss einfach zu bedienen sein. Wir richten uns allgemein an Sportler und Sportinteressierte. Außer grundlegenden Kenntnissen zur Nutzung von Android-Apps können wir keine Annahmen zu den technischen Fähigkeiten des Nutzers machen.

Teile der App sind für die Nutzung beim Joggen. Auch hierbei muss es dem Nutzer möglich sein relevante Informationen zu erkennen und sicher mit der App zu interagieren.

### 3.3.2 Sicherheit

Um Nutzer zu authentifizieren ist eine Nutzername/Passwort-Vergabe notwendig. Diese müssen ausreichend gesichert sein, um unauthorisierte Zugriffe auf Ressourcen zu verhindern.

Außerdem muss die Sicherheit des Gerätes zu jedem Zeitpunkt gewährleistet sein, d.h. die Einschleusung über die App von Schadcode darf nicht möglich sein.

## 3.4 Use Cases

## 3.5 Komponenten der Webservers

## 3.6 Komponenten der Android App

Die folgenden Abschnitte befasst sich mit der Analyse der Komponenten der Ghostrunner App. Notwendig sind für die Aufzeichnung des Laufes beispielsweise eine Möglichkeit der Positions- und Höhenbestimmung. Es muss außerdem eine Möglichkeit der Authentifizierung und Kommunikation mit dem Webserver geben. Die Android Plattform wird auf mögliche Lösungen untersucht, welche hier beschrieben werden.

### 3.6.1 Kommunikation mit dem Webserver

Für die Kommunikation mit dem Webserver muss eine Internetverbindung zur Verfügung stehen. Diese ist bei allen Geräten, auf denen unsere App laufen soll Standard. Programmatisch wird also nur ein HTTP-Client benötigt, der von der Android-Plattform bereitgestellt wird. Er ermöglicht, dass HTTP-Anfragen wie GET und POST gestellt werden und Informationen in HTTP-Header und Body übergeben werden können. Beim Warten auf Antworten vom Webserver kommt es automatisch zu nicht vorhersehbaren Verzögerungen. Android erlaubt es aus diesem Grund nicht, diese Netzwerkanfragen im Main Thread der App zu starten, weil sonst während der Wartezeiten die komplette Benutzeroberfläche einfriert. Bei den Standard HTTP-Clients von Android kann die Netzwerkanfrage durch die Benutzung der Klasse AsyncTask in einen separaten Thread ausgelagert werden. [AOSa]

Vereinfacht werden kann dieser Schritt durch die Benutzung des Android Asynchronous HTTP Client. Die quelloffene Bibliothek bietet einfache funktionen für die HTTP-Anfragen GET, POST, PUT und DELETE, die für die Kommunikation mit RESTful-APIs notwendig sind und empfängt Antworten vom Server automatisch asynchron, wodurch der UI-Thread nicht blockiert wird. [Smi]

### 3.6.2 Anmeldung und Authentifizierung

Ein weiterer wichtiger Aspekt ist die Authentifizierung beim Webserver. Nach einer Anmeldung mit Nutzername und Passwort wird vom Server ein Sicherheitstoken erstellt, das bis zu seinem ebenfalls vom Server definierten Ablaufdatum von Client benutzt werden kann, um sich anzumelden. So kann bei einem eventuellen Angriff auf die Netzwerkkommunikation nur noch ein zeitlich, nicht mehr die vollständigen Es macht Sinn, diese Authentifizierung an einer einzigen Stelle im Client zu behandeln.

### 3.6.3 Positionsbestimmung

Für eine genaue Laufanalyse muss zu jedem Zeitpunkt die Position des Läufers bekannt sein, denn daraus lassen sich Geschwindigkeit und gelaufene Strecke berechnen. Die Positionsbestimmung erfolgt über das eingebaute GPS-Modul des Smartphones, dessen Genauigkeit in Städten durch WLAN-basierte Ortung und GSM-Ortung durch das Mobilfunknetz verbessert werden kann. Die Module zur Ortung können über das Package android.location der Android API angesprochen werden. Eine Positionsbestimmung ist also mit Boardmitteln eines Android Smartphones ohne Umwege möglich. Eine neuere und verbesserte Möglichkeit bietet die Google Location Services API. Sie ist Teil der Google Play Services, also auf allen Android Geräten verfügbar, die diesen und andere Google Dienste installiert haben, was einem Großteil der Geräte entspricht. Die verschiedenen Ortungsmöglichkeiten werden hier optimiert und zusammengeführt. Programmierer können auf einfache Weise Positionsupdates anfordern und diese über verschiedene Parameter bezüglich Genauigkeit, Updatehäufigkeit und Akkuverbrauch seinen Bedürfnissen anpassen. [AOSc]

### 3.6.4 Höhenbestimmung

Zusätzlich zu Positions- und Zeitmessung wird für unsere Anwendung noch eine weitere Variable - die Steigung - benötigt. Wenn Läufer verschiedene Strecken laufen, muss diese berücksichtigt werden, um einen fairen Vergleich zu schaffen. Dazu soll später Läufern, die eine größere Steigung überwinden mussten als andere ein Zeitbonus zugesprochen werden.

Für die Bestimmung der Höhe gibt es verschiedene Möglichkeiten, die im Folgenden kurz erläutert werden.

**Höhenbestimmung per GPS** Grundsätzlich ist per GPS eine dreidimensionale Positionsbestimmung möglich, also neben der horizontalen Positionierung auch eine vertikale Höhenmessung. Diese kann jedoch aufgrund der für horizontale Positionsbestimmung optimierten Positionierung der GPS-Satelliten starken Schwankungen unterliegen. Insbesondere in Städten kommt hierzu das Problem der Reflektion des GPS Signals an hohen Gebäuden, durch die die Positionsgenauigkeit weiter beeinträchtigt wird. Die Beeinträchtigung ist horizontal weniger signifikant als vertikal, und wird durch die Nutzung der im vorherigen Abschnitt genannten zusätzlichen Positionierungsmethoden noch weiter optimiert. Vertikal ist eine solche Optimierung nicht möglich. Noch größer ist die Fehlersignifikanz bei dem Versuch, die Höhe über Normalnull zu berechnen, die von dem vom GPS-System genutzten rein mathematischen Ellipsoid WGS-84 abweicht. Durch die Umrechnung vergrößern sich entsprechende Fehler. [Gla06] Letzeres ist allerdings für unsere Arbeit aber nicht relevant, da wir keine genaue Höhe, sondern nur die relativen Unterschiede für die Steigungsberechnung benötigen. Trotz-

dem können die möglichen starken Schwankungen der gemessenen Höhe für unsere Anwendung zum Problem werden.

#### Höhenbestimmung per Web-API

#### Höhenbestimmung per SRTM-Rohdaten

**Kombination verschiedener Methoden** Insbesondere durch Kombination von GPS und und Nutzung der SRTM-Daten sollten sich sehr genaue Ergebnisse erzielen lassen. Die Implementierung des Höhenbestimmungsmoduls sollte also flexibel genug sein, um das zuzulassen. Erreicht wird diese Flexibilität durch die Implementierung eines Interfaces ElevationService, das die Methode getElevation(double latitude, double longitude) besitzt. Durch Nutzung dieses Interfaces können verschiedene Services einfach ausgetauscht oder mit wenig zusätzlichem Programmieraufwand kombiniert werden. Hier muss nur noch ein Algorithmus entwickelt werden, der bestimmt, auf welche Weise verschiedene Höhendaten kombiniert werden. Für die Studienarbeit haben wir uns aber für die explizite Implementierung der SRTM-Methode entschieden, Veränderungen in späteren Versionen sind durch das Interface jederzeit ohne großen Aufwand möglich.

### 3.6.5 Aufzeichnen des Laufes

### 3.6.6 Frontend der Android App

### 3.6.7 Darstellung des Laufes

## 3.7 Analyse von Entitäten

# 4 Design

# 5 Implementierung

# 6 Testen

# 7 Anwendungsbeispiel

# 8 Ausblick

# 9 Fazit

# Literaturverzeichnis

[AOSa] AOSP: *Connecting to a Network*. http://developer.android.com/training/basics/network-ops/connecting.html

[AOSb] AOSP: *Dashboards*. http://http://developer.android.com/about/dashboards/index.html

[AOSc] AOSP: *Location Strategies*. http://developer.android.com/guide/topics/location/strategies.html

[Gla06] Gladstone, Philip: *Discussion of Vertical GPS Accuracy*. http://weather.gladstonefamily.net/gps_elevation.html. Version: 2006

[JHD+] Johnson, Rod ; Hoeller, Juergen ; Donald, Keith ; Sampaleanu, Colin ; Harrop, Rob ; Risberg, Thomas ; Arendsen, Alef ; Davison, Darren ; Kopylenko, Dmitriy ; Pollack, Mark ; Templier, Thierry ; Vervaet, Erwin ; Tung, Portia ; Hale, Ben ; Colyer, Adrian ; Lewis, John ; Leau, Costin ; Fisher, Mark ; Brannen, Sam ; Laddad, Ramnivas ; Poutsma, Arjen ; Beams, Chris ; Abedrabbo, Tareq ; Clement, Andy ; Syer, Dave ; Oliver Gierke, Rossen S. ; Webb, Phillip ; Winch, Rob ; Clozel, Brian ; Nicoll, Stephane ; Deleuze, Sebastien: *Spring Framework Reference Documentation*. http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

[Smi] Smith, James: *Android Asynchronous Http Client*. http://loopj.com/android-async-http

# Anhang

## Eidesstattliche Erklärung

# Eidesstattliche Erklärung zur <-Arbeit>

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

$Unterschrift$ : $Ort, Datum$ :