

**Duale Hochschule Baden-Württemberg Mannheim**



**Große Studienarbeit**

# **Location-based Services**

Theoretische Erarbeitung und prototypische Implementierung

Name: **Melanie Hammerschmidt, Victor Schwartz & Patrick Senneka**  
Matrikelnummer: 6620867, 2696414, 4858361  
Kurs: TINF12AI-BC  
Studiengang: Angewandte Informatik  
Studiengangsleiter: Prof. Dr. H. Hofmann  
Betreuer: Prof. Dr. H. Hofmann  
Semester: 5. - 6. Semester  
Datum: 01.06.2015



## Ehrenwörtliche Erklärung

Gemäß § 5 Abs. 3 der Studien- und Prüfungsordnung DHBW Technik vom 22.09.2011 versichere ich hiermit, die vorliegende Arbeit selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln verfasst zu haben.

01.06.2015

Datum

Melanie Hammerschmidt, Victor  
Schwartz & Patrick Senneka

## **Abstract**

Hier folgt das Abstract...

# Inhaltsverzeichnis

<b>Ehrenwörtliche Erklärung</b>	<b>III</b>
<b>Abstract</b>	<b>IV</b>
<b>Inhaltsverzeichnis</b>	<b>V</b>
<b>Abkürzungsverzeichnis</b>	<b>VII</b>
<b>Abbildungsverzeichnis</b>	<b>VIII</b>
<b>Tabellenverzeichnis</b>	<b>IX</b>
<b>Vorwort</b>	<b>X</b>
<b>1. Einleitung</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. LBS Definitionen . . . . .	2
1.3. Typen von LBS . . . . .	5
<b>2. Theoretische Grundlagen</b>	<b>8</b>
2.1. Standortarten . . . . .	8
2.2. Positionsbestimmung und Funktionsweise . . . . .	16
2.2.1. Kriterien für die Standortbestimmung . . . . .	17
2.2.2. Satellitengestützte Positionierung . . . . .	18
2.2.3. Positionierung in (Mobil)Funknetzen . . . . .	24
2.2.4. Positionsbestimmung in Gebäuden . . . . .	28
2.2.5. Kombination von Verfahren zur Positionsbestimmung . . . . .	30
<b>3. Anwendungsfälle für LBS</b>	<b>31</b>
3.1. Historie von LBS . . . . .	31
3.2. Hauptnutzer von LBS . . . . .	33
<b>4. Prototypische Umsetzung</b>	<b>38</b>
4.1. Anforderungen . . . . .	38
4.2. Architektur . . . . .	39
4.3. Technologien und Entscheidungen . . . . .	42
4.3.1. Phonegap und Apache Cordova . . . . .	48
4.3.2. Ionic . . . . .	48
4.3.3. HTML 5 . . . . .	48
4.3.4. CSS . . . . .	49
4.3.5. Angular-JS . . . . .	50
4.3.6. Kartenmaterial . . . . .	50
4.3.7. iBeacons . . . . .	65

<b>5. Implementierung</b>	<b>68</b>
5.1. App.js . . . . .	68
5.2. Controller und Services . . . . .	69
5.3. Applikationsaufbau . . . . .	70
5.3.1. Spielen-Tab . . . . .	71
5.3.2. Highscores-Tab . . . . .	80
5.3.3. Account-Tab . . . . .	81
5.3.4. Credits-Tab . . . . .	82
5.3.5. Übersicht . . . . .	83
5.3.6. Bewertung der Architektur . . . . .	83
5.4. Umsetzung iBeacons . . . . .	83
<b>6. Fazit</b>	<b>88</b>
6.1. Ausblick . . . . .	88
<b>Literatur</b>	<b>i</b>
<b>A. Appendix sections</b>	<b>iii</b>

## **Abkürzungsverzeichnis**

<b>App</b>	Applikation
<b>GPS</b>	Global Positioning System
<b>LBS</b>	Location-based Services

## Abbildungsverzeichnis

1.	Context-aware services . . . . .	4
2.	Magnetfeld der Erde . . . . .	9
3.	Kompassrose mit Gradzahlen . . . . .	10
4.	Postleitzahlgebiete in Deutschland . . . . .	11
5.	Längen- und Breitengrade der Erde . . . . .	13
6.	Infrastruktur zur Positionsbestimmung . . . . .	17
7.	Grundprinzip Satelliten Positionsbestimmung . . . . .	19
8.	Umlaufbahn von GPS Satelliten . . . . .	19
9.	GPS Basissationen . . . . .	22
10.	GPS Präzision . . . . .	23
11.	Präzision Differential GPS . . . . .	23
12.	GSM Abdeckung in Deutschland . . . . .	25
13.	Cell of Origin . . . . .	25
14.	Antennen Segment . . . . .	26
15.	Timing Advance mit Segment antenna . . . . .	27
16.	Uplink Time of Arrival . . . . .	27
17.	WLAN Messwerttabelle . . . . .	28
18.	SpotOn . . . . .	30
19.	Modell-View-Controller Pattern . . . . .	40
20.	Ablauf der Modell-View-Controller . . . . .	41
21.	Marktanteile mobiler Betriebssysteme . . . . .	43
22.	iBeacons der Firma Estimote . . . . .	67
23.	Ablauf Applikationsdarstellung . . . . .	68
24.	Tabaufbau . . . . .	70
25.	Tabaufbau Spielen-Tab . . . . .	71
26.	Model-View-Controller: Play-View . . . . .	72
27.	Model-View-Controller: Game-View . . . . .	73
28.	Model-View-Controller: Tasks-View . . . . .	77
29.	Model-View-Controller: Task-View . . . . .	78
30.	Tabaufbau Highscores-Tab . . . . .	80
31.	Model-View-Controller: Highscores-View . . . . .	80
32.	Tabaufbau Account-Tab . . . . .	81
33.	Model-View-Controller: Account-View . . . . .	81
34.	Tabaufbau Credits-Tab . . . . .	82
35.	Übersicht Controller-Services Verbindungen . . . . .	83



## Tabellenverzeichnis

1.	Bedeutung von Kartenmaterial . . . . .	51
----	--	----

**Vorwort**

---

# 1. Einleitung

## 1.1. Motivation

*Verfasst von Patrick Senneka*

Historisch gesehen war die Information über den eigenen Standort schon immer von großer Bedeutung. In Verbindung mit Daten, die auf den Standort bezogen sind, war der Mehrwert enorm. So konnte man z.B. in der Seefahrt mit einem durch Sterne bestimmten Standort in Verbindung mit einer Karte die Reiseroute so bestimmen, dass man Land erreicht bevor die Vorräte ausgehen.

Dieses noch recht weit hergeholte Beispiel über die Nutzung von Standort-bezogenen Daten konnte mit Hilfe von Technik auf die Informatik übertragen werden. Damit wurde der Begriff Location-based Services geprägt.

Dem Endverbraucher waren solche Services zunächst einmal nur über spezielle Geräte, wie Navigationssystem möglich. In den letzten Jahren wurden milliardenfach Smartphones mit GPS-Modulen und mobilem Internet verkauft. Solch ein Smartphone besitzen nun eine Mehrzahl der in Deutschland lebenden Menschen. [1] Durch die gerade erwähnten technischen Gegebenheiten der Smartphones ist die Nutzung von Location-based Services für den Nutzer ein Kinderspiel geworden.

Diese Arbeit beschäftigt sich mit einer theoretischen Erarbeitung zu Location-based Services und eine prototypische Implementierung einer LBS Smartphone Applikation.

## 1.2. LBS Definitionen

*Verfasst von Patrick Senneka*

Für diese Arbeit über Location-based Services ist es wichtig zu definieren, was unter diesem Begriff genau zu verstehen ist. Der erste Schritt dazu ist den Begriff ins deutsche zu übersetzen. Unter Location-based Services versteht man nach einer wörtlichen Übersetzung Standort-bezogene Dienste. Nach der deutschen Sprache wäre nun anzunehmen, dass es Dienste sind, die unter Zuhilfenahme des Standorts angeboten werden. In der Literatur gibt es allerdings nicht eine feste Definition, die mit dem ersten Location-based Service festgelegt wurde. Es ist vielmehr so, dass es mehrere Begrifflichkeiten gibt, die nicht das gleiche aussagen, aber dennoch so verwendet werden. Vergleiche hierzu Zitat 1.1 von Alex Küpper.

**Zitat 1.1:** „Although Location-based Services (LBSs) have been an issue in the field of mobile communication for many years, there exists neither a common definition nor a common terminology for them. For example, the terms location-based service, location-aware service, location-related service, and location service are often interchangeably used. [2, S.1]

**Übersetzung:** Obwohl Location-based Services (LBSs) schon seit vielen Jahren ein Thema sind, existiert weder eine einheitliche Definition noch eine einheitliche Begrifflichkeit. Zum Beispiel werden die Ausdrücke location-based service, location-aware service, location-related service and location service oft austauschbar verwendet.

Eine eindeutige Definition zu Location-based Services wurde von Jochen Schiller in seinem Buch „Location-Based Services“ aufgestellt. Vergleiche Definition 1.2.

**Definition 1.2:** „The term location-based services (LBS) is a recent concept that denotes applications integrating geographic location (i.e., spatial coordinates) with the general notion of services. “[3, S.1]

**Übersetzung:** Der Begriff Location-based Services (LBS) ist ein aktuelles Konzept, dass Applikationen, die geografische Standorte integrieren (z.B., räumliche Koordinaten) mit dem eigentlichen Gedanken eines Services vereint.

Obwohl die Definition für Location-based Services von Jochen Schniller klar und eindeutig erscheint, besteht nach der Aussage von Alex Küpper noch Klärungsbedarf bezüglich der andren oft synonym verwendeten Begriffen:

- location service
- location-based service

- location-aware service
- and location-related service

**location service**

Der Hauptunterschied zwischen Location Service und Location-based Services besteht in der Verarbeitung der Standortdaten. Bei einem Location Service wird der Standort eines Objekts (z.B eines Handys) bestimmt. Dieser ermittelte Standort wird dann als Service extern bereitgestellt. Die Standortdaten werden also erfasst und zur Verfügung gestellt, ohne diese zu verarbeiten. Der Unterschied zu einem Location-based Service besteht darin, dass bei diesem die Standortdaten weiterverarbeitet werden. Ein location service ist also einer der wichtigsten Bestandteile eines Location-based Service, der dabei die Standortdaten nicht extern bereitstellt, sondern zu Weiterverarbeitung einem Location-based Service zu Verfügung stellt. [2, S.1-2]

**location-based service**

Entspricht der Definition 1.2 nach Schiller.

**location-aware service**

<b>Definition 1.3:</b> „Generally, context-aware services are defined to be services that automatically adapt their behavior, for example, filtering or presenting information, to one or several parameters reflecting the context of a target. These parameters are termed context information.“[2, S.2]
--

<b>Übersetzung:</b> Im Allgemeinen sind context-aware services als solche definiert, dass sie automatisch ihr Verhalten, z.B. filtern oder darstellen von Informationen, an ein oder mehrere Parameter anpassen, die den Kontext des Ergebnisses widerspiegeln. Diese Parameter werden als Kontext Informationen bezeichnet.
--

Die in Definition 1.3 beschriebene Kontext Parameter können einerseits Daten sein, die direkt von einem Sensor wie zum Beispiel einem Thermometer oder Positionssensor sein. Diese Rohdaten bilden den Primären Kontext. Andererseits gibt es noch aus Rohdaten abgeleitete Kontextdaten. Diese entstehen durch Kombination und oder Filterung von Rohdaten. Das sind die Sekundär Kontextdaten. [2, S.2]

Die beispielhafte Aufzählung von Rohdaten zeigt, dass context-aware services location-aware services mit einschließen könne allerdings nicht müssen. Verdeutlicht wird das durch Abbildung 1.



Abbildung 1: Context-aware services  
[2, S. 2]

### location-related service

Location-related service ist ein noch breiter gefasster Begriff, als Location-aware service. Es ist als mobiler Service definiert, bei welchem Standorte eine große Bedeutung spielen. Vergleiche hierzu Definition 1.4.

**Definition 1.4:** „Therefore, mobile services relying on this importance of location will be referred to in the following as locationrelated services (LRS). Location Based Services (LBS) are a sub-category of LRS “[4, S.5]

**Übersetzung:** Mobile Services, die sich auf die Bedeutung von Standorten beziehen werden im folgenden als locationrelated services (LRS) referenziert. Location Based Services (LBS) sind eine Unterkategorie von LRS

### 1.3. Typen von LBS

*Verfasst von Patrick Senneka*

Location-based Services lassen sich grundsätzlich in zwei Kategorien einteilen. Diese unterscheiden sich in die Art und Weise, wann der eigene Standort bestimmt und genutzt wird.

In diesem Kapitel geht es darum, diese beiden Kategorien zu definieren und voneinander durch Vor und Nachteile abzugrenzen.

In den Büchern von Alexander Küpper und vielen Onlinequellen sind LBS in die Kategorien reaktive und proaktive-LBS unterteilt. Der Autor Jochen Schiller verwendet die Begrifflichkeit Pull- und Push Services, die allgemein für Services gültig ist. Hierbei entsprechen reaktive LBS den Pull Services, sowie die proaktiven LBS den Push-Services.

Im Weiteren wird die für Location-based Services spezifischere Notation, reaktive- und proaktive-LBS, von Alexander Küpper verwendet.

#### Reaktive LBS

Herr Küpper unterscheidet in seinem Buch „Location-Based Services - Fundamentals and Operation“ grundsätzlich zwischen reaktiven- und proaktiven-LBS. Diese Unterscheidung macht er an der Nutzung der Standortdaten von dem LBS fest. Erhält ein LBS den Standort nur dann wenn dieser vom Nutzer aktiviert ist, ist es ein reaktiver LBS.

Vergleiche hierzu Definition 1.5:

<b>Definition 1.5:</b> „LBSs can be classified into reactive and proactive LBSs. A reactive LBS is always explicitly activated by the user.“ [2, S.3]
---

<b>Übersetzung:</b> LBSs können in reaktive und proaktive LBSs klassifiziert werden. Ein reaktiver LBS wird immer ausdrücklich durch den Nutzer aktiviert.
--

Herr Schiller definiert die reaktiven LBS (Pull Services) dahingehend, dass eine Applikation nur Informationen aus einem Netzwerk oder über einen Standort erhebt, wenn diese aktiv genutzt wird.

Vergleiche hierzu Definition 1.6:

Bezugnehmend auf die zwei Definitionen über reaktive LBS wird dieser Begriff nochmals in Bezug auf eine Smartphone App, wie sie im Rahmen dieser Studienarbeit prototypisch erstellt wird, erläutert.

**Definition 1.6:** „Pull services, in contrast, mean that a user actively uses an application and, in this context, ”pulls information from the network. This information may be location-enhanced (e.g., where to find the nearest cinema) “[3, S.20]

**Übersetzung:** Pull-Dienste bedeuten, dass ein Nutzer eine Applikation aktiv nutzt, und in diesem Kontext Informationen von einem Netzwerk bezieht. Diese Informationen können in Verbindung zu einem Standort sein (z.B. wo das nächste Kino zu finden ist).

Spezielle Definition:

Reaktive LBS sind für Smartphone Apps, die nur während der Benutzung auf den Standort zugreifen und diesen verwenden. Während der Benutzung bedeutet dabei, dass die App geöffnet sein muss und nicht im Hintergrund läuft.

### Proaktive LBS

Im Vergleich zu reaktiven LBS zeichnen sich nach Alexander Küpper proaktive LBS dadurch aus, dass diese durch Standortereignisse automatisch aktiviert werden.

Vergleiche hierzu Definition 1.7 :

**Definition 1.7:** „Proactive LBSs, on the other hand, are automatically initialized as soon as a predefined location event occurs, for example, if the user enters, approaches, or leaves a certain point of interest or if he approaches, meets, or leaves another target. “[2, S.3]

**Übersetzung:** Proaktive LBSs werden automatisch initialisiert, sobald ein zuvor definiertes Standortevent auftritt. Dies kann zum Beispiel sein, dass ein Nutzer eine Sehenswürdigkeit betritt, sich dieser annähert oder diese verlässt. Oder wenn der Nutzer sich einem anderen Ziel nähert, es trifft oder es verlässt.

Proaktive LBS nach Jochen Schniller stellt klar, dass der Nutzer Informationen zu seinen Standorten erhält und das sogar ohne, diese Informationen aktiv oder bewusst anzufordern. Diesen Schritt übernimmt der proaktive LBS für den Nutzer.

Vergleiche hierzu Definition 1.8 :

In Bezug auf eine Smartphone App laufen proaktive LBS im Hintergrund mit. Sie erfassen und verarbeiten ständig den Standort des Smartphone Nutzers, ohne dass dieser solch eine Aktion angefordert hat. Tritt allerdings ein Standortereignis auf, tritt die Applikation in den Vordergrund und teilt dem Nutzer das Ergebnis mit.

Solch ein Ereignis kann zum Beispiel eine App zum Freunde treffen sein. Wenn man sich in der Nähe des Standorts eines Freundes befindet kann ein proaktiver LBS darüber informieren.



**Zitat 1.8:** „Push services imply that the user receives information as a result of his or her whereabouts without having to actively request it. The information may be sent to the user with prior consent (e.g., a subscription-based terror attack alert system) or without prior consent (e.g., an advertising welcome message sent to the user upon entering a new town). “[3, S.20]

**Übersetzung:** Push services implizieren, dass der Nutzer als Ergebnis zu seiner Position Informationen erhält, ohne diese aktiv anzufragen. Die Informationen können dem Nutzer mit seiner Zustimmung (z.B. ein abonniertes Terror Angriff Warnsystem) gesendet werden oder ohne vorherige Zustimmung (z.B. eine Willkommensnachricht beim betreten einer neuen Stadt).

Heutzutage fällt bei der Betrachtung mehreren Apps auf, dass diese hauptsächlich zur Gruppe der reaktiven LBS gehören. So stellen Kartendienste auf dem Handy erst Kartenmaterial zu Verfügung, wenn die App geöffnet ist.

Dem gegenüber gibt es allerdings schon proaktive LBS, die weit verbreitet sind. Dazu zählt zum Beispiel Google Now. Google Now bestimmt ständig die Standortdaten des Smartphones. Erst, wenn ein mit dem Standort in Verbindung stehendes Ereignis eintritt, wird der Nutzer den Service wahrnehmen. Das kann zum Beispiel abends sein, wenn der letzte Zug zwischen dem momentanen Standort und dem Zuhause in kürze abfährt.

## 2. Theoretische Grundlagen

### 2.1. Standortarten

*Verfasst von Victor Schwartz*

Betrachtet man den Titel dieser Arbeit „Location Based Services“ (Standort bezogene Dienste), wird schnell klar aus welchen Bestandteilen sich LBS zusammensetzt: der eigene Standort und ein Dienst, der diesen nutzt. Dieses Kapitel beschäftigt sich damit, wie ein Standort definiert werden kann.

Als einleitendes Beispiel kann man sich vorstellen, man steht mitten in einer groß Stadt wie Mannheim. Für einen selbst ist klar wo der eigene Standort ist. Will man diesen einem Freund mitteilen, kann dies schwierig sein. Deshalb gibt es definierte Arten seinen Standort bis auf wenige Zentimeter genau zu beschreiben. Dieses Kapitel ist in mehrere Abschnitte unterteilt. Jeder Abschnitt erläutert eine mögliche Beschreibung des eigenen Standortes. Zu beachten ist an dieser Stelle, dass es sich hierbei nicht um alle Möglichkeiten handelt einen Standort zu beschreiben. Es ist immer möglich eine individuelle Beschreibung des eigenen Standortes anzugeben.

#### **Standortbeschreibung mit einem Kompass**

Dieser Abschnitt beschreibt wie man mit Hilfe eines Kompasses seine Position bestimmen bzw. angeben kann. Hierfür wird erst vorgestellt wie ein Kompass funktioniert und welche Voraussetzungen zur Standortbestimmung gegeben sein müssen.

Bei einem Kompass handelt es sich um ein Gerät, dessen Aufgabe es ist, in die Richtung des Nordpol zu zeigen. Dieses Ziel wird durch das Erdmagnetfeld erreicht.

Das Erdmagnetfeld ist in Abbildung 2 zu sehen. Durch dieses Feld ist die komplette Erdkugel magnetisiert. In der Nähe des geografischen Südpols liegt der magnetische Nordpol. Umgekehrtes gilt für den geografischen und magnetischen Nord- bzw. Südpol. Mit Hilfe eines magnetisierten Metallstücks kann dieses Feld genutzt werden. Meist handelt es sich dabei um eine Nadel, die dann in Richtung Norden zeigt.

Bei einem Kompass kommt neben der Nadel auch eine Kompassrose (siehe Abbildung 3) zum Einsatz. Mit Hilfe dieser Rose lässt sich eine Gradzahl ablesen. Sie orientiert sich immer am Abstand von der angezeigten Richtung des Norpols. Diese Gradzahl kann für die Beschreibung des eigenen Standort genutzt werden. Wie dieses Verfahren funktioniert, wird im nächsten Abschnitt erklärt.

Standortbeschreibung mit zwei Fixpunkten Möchte man seinen geografischen Stand-



Abbildung 2: Magnetfeld der Erde  
[5]

punkt jemandem mitteilen oder sich diesen notieren, muss man ihn beschreiben. Hat man dafür nur einen Kompass zur Verfügung, kann man sich zwei Fixpunkte zur Hilfe nehmen um den eigenen Standort zu bestimmen. Anhand eines Beispiels soll dieses Verfahren erklärt werden. Befindet man sich auf einem Boot in der Nähe der Küste und möchte den Standort mit einem Kompass bestimmen, so sucht man sich zwei gut sichtbare Punkte an der Küste, dies können beispielsweise Leuchttürme oder eindeutig identifizierbare Felswände sein. Nun fixiert man den ersten Punkt mit dem Kompass und liest die exakte Grad Zahl auf der Kompassrose ab. Mit Hilfe der Angabe von Gradzahl und erstem Fixpunkt, kann der eigene Standort auf eine Linie differenziert werden. Um nun den Standort auf einen Punkt zu differenzieren fixiert man mit dem Kompass den zweiten Fixpunkt und liest ebenfalls die Grad Zahl ab. Zeichnet man diese Informationen in eine Karte ergibt sich ein Schnittpunkt von zwei Geraden. An dieser Stelle befindet man sich. Die Beschreibung des eigenen Standortes könnte beispielsweise wie folgt aussehen:

270 Grad von Leuchtturm X

250 Grad von Gebäude Y

Der Vorteil dieses Verfahren ist: Es kann überall angewendet werden, wo es Fixpunkte gibt. Nachteile dieser Methode sind: Es wird keine Höhe angegeben und es kann eine große Ungenauigkeit beim Ablesen des Kompasses entstehen.



Abbildung 3: Kompassrose mit Gradzahlen  
[6]

### Standortbeschreibung mittels einer Adresse

Im Gegensatz zur eben vorgestellten Standortbeschreibung, ist folgende wesentlich bekannter. Es handelt sich um die Anschrift bzw. Adresse. Bei dieser Beschreibung wird kein exakter Standort angegeben, sondern ein Haushalt. Dies geschieht über einen Namen, eine Straße mit Hausnummer und die Angabe einer Stadt, welche konkretisiert über die Postleitzahl angegeben wird. Eine Adresse kann in Deutschland beispielsweise wie folgt aus:

Max Mustermann  
Coblitzalle 53  
12345 Mannheim

#### 3. Zeile: Postleitzahl und Stadtname

Die letzte Zeile der Adresse enthält den Namen einer Stadt, sowie eine Postleitzahl. Bei der Angabe der Stadt handelt es sich um die größte Angabe innerhalb der Adresse. Mit Hilfe des Stadtnamens soll eine erste Orientierung innerhalb Deutschlands ermöglicht werden. Es handelt sich nicht nur um Großstädte wie Berlin, Frankfurt, Hamburg etc. sondern alle Gemeinden, welche den Titel Stadt tragen. Da manche Stadtnamen mehrfach vorkommen oder die Angabe zu ungenau ist, wird der Stadname mit der Postleitzahl (PLZ) definiert bzw. konkretisiert. Allein für die Stadt Mannheim gibt es über 15 verschiedenen Postleitzahlen. Diese sind teilweise jedem Stadtteil zugeordnet.

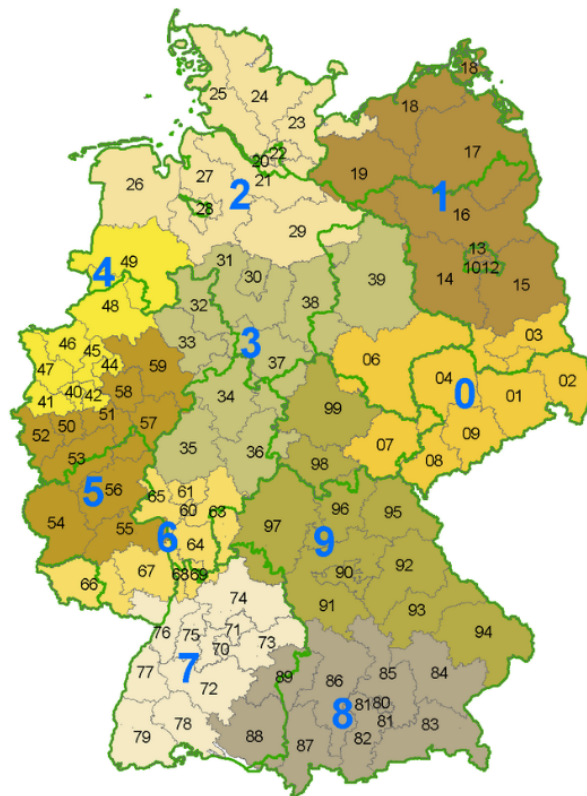


Abbildung 4: Postleitzahlgebiete in Deutschland  
[7]

Da die Angabe von einer Stadt und der dazugehörigen PLZ das Gebiet noch nicht weit genug einschränkt um einzelne Häuser bzw. Haushalte zu identifizieren, wird ein Straßename angegeben. Die europäischen Straßennamen sind meist mit Namenwörtern angegeben. Einige Beispiele hierfür sind: - Lindenstraße - Schillerstraße - Hauptstraße

Um ein Haus innerhalb einer Straße exakt definieren zu können bedient man sich einer Nummerierung - den sogenannten Hausnummern. Jedes Haus hat eine eigene Hausnummer. Da es häufig vorkommt, dass in einem Haus mehrere Parteien wohnen, wird neben den beschriebenen Bestandteilen einer Adresse noch ein Empfänger mit Vor- und Nachname oder nur Nachname angegeben. Diese Beschreibung findet auf der ganzen Welt Anwendung, allerdings nicht immer in diesem Format. Das beschriebene Format trifft größtenteils in Europa zu.

**Verwendung von Adressen** Die vorgestellte Beschreibung des eigenen Standortes mithilfe einer Adresse ist besonders wichtig für Brief- und Paketversandorganisationen. In Deutschland wurden in den letzten Jahren hauptsächlich mit der Deutschen Post AG Briefe verschickt. Sie ist darauf angewiesen, dass auf jedem Paket und Brief die Adresse im beschriebenen Format angegeben ist, damit eine Zustellung erfolgen kann. Desweiteren hat die Adresse mit der Entwicklung von Navigationsgeräten stark an Bedeutung dazu gewonnen. Im Jahr 2014 besaßen ca. 48 Prozent der deutschen Haushalte ein Navigationsgerät. Dort wird das Ziel hauptsächlich als Adresse angegeben. Auch im Alltag wird die Adresse meist zur Beschreibung des eigenen Standortes verwendet. Möchte man Freunde zu sich einladen, gibt man meist die Adresse an und mit einem Navigationsgerät oder einer Karte kann der Haushalt gefunden werden.

Vergleicht man eine Adresse mit der Standortbeschreibung des letzten Kapitels über einen Kompass, ist zu erkennen, dass man sich einen groben Überblick über den Standort mit Adresse viel leichter verschaffen kann. Die Koordinaten eines Kompasses kann man nur mit entsprechendem Fachwissen deuten.

Als Nachteile zählen die Genauigkeit und Abdeckung der Erde mit Adressen. Die Genauigkeit des eigenen Standortes kann mit einer Adresse nicht exakt angegeben werden. Während bei einer kleinen Einzimmerwohnung der Standort auf wenige Meter genau angegeben werden kann, kann sich dies auf mehrere Kilometer ausweiten, wenn die Adresse zu einem großen Anwesen gehört. Für manche Location Based Services ist es wichtig, dass diese auch in nicht besiedelten Gebieten verwendet werden können. Hierzu zählen beispielsweise LBS zum Wandern und Radfahren. Hierfür eignet sich die Angabe des eigenen Standortes mit einer Adresse nicht, da ein Standort nur auf der Minderheit

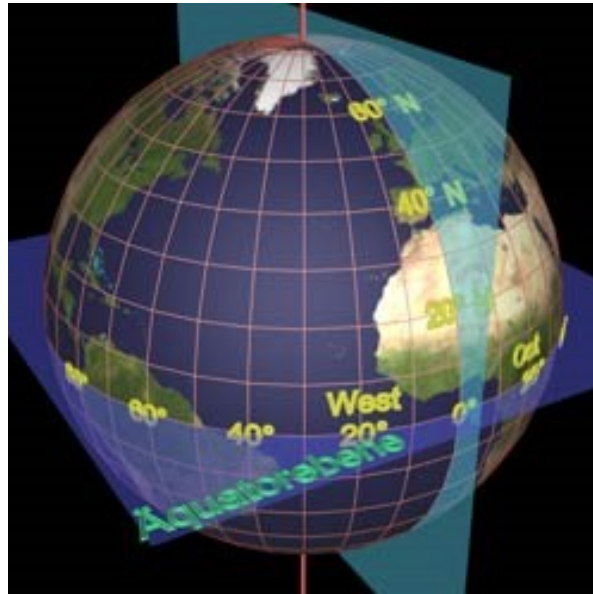


Abbildung 5: Längen- und Breitengrade der Erde  
[8]

der Erde mit einer Adresse angegeben werden kann. Im nächsten Abschnitt wird die Beschreibung des Standortes mithilfe von Längen- und Breitengraden erläutert.

### **Standortbeschreibung mit Längen- und Breitengraden**

Dieser Abschnitt erläutert die Standortbeschreibung mittels Längen- und Breitengraden. Hierfür kann man sich ein zweidimensionales Koordinatensystem vorstellen, welches den Nullpunkt im Zentrum besitzt. Dieses „Netz“ legt man bildlich gesprochen über den Globus. So kann für jeden Ort auf dem Globus eine exakte Beschreibung stattfinden.

#### Grundlagen für diese Standortbeschreibung

Beim Globus handelt es sich annähernd um eine Kugel. Diese Kugel dreht sich in ca. 24 Stunden um ihre eigene Achse. Die Achse kann man sich als gedachte Gerade vorstellen, welche an der untersten Stelle, dem Südpol eintritt und am obersten Punkt, dem Nordpol, wieder austritt. Stellt man sich nun die Achse als Strecke vom Nord- zum Südpol vor und halbiert diese, befindet man sich im ungefähren Mittelpunkt des Globus. Spannt man auf der Höhe der halbierten Achse einen Kreis um den Globus, so befindet man sich an der dicksten Stelle, dem Äquator. Dieser befindet sich im rechten Winkel zur Achse.

#### Breitengrade

Nachdem der Äquator im letzten Abschnitt definiert wurde, wird nun erklärt was Breitengrade sind. Um den gesamten Globus mit Graden zu beschreiben, hat man sich dazu

entschieden den Globus am Äquator in eine nördliche und eine südliche Halbkugel aufzuteilen. Der gesamte Globus ist in 180 Breitengrade aufgeteilt. Beim Äquator handelt es sich um den 0. Grad. In Abhängigkeit dieses Grades wird der Standort auf einem Breitengrad angegeben. Es gibt jeweils 90 Grade in die nördliche und südliche Richtung. Mithilfe dieser Angabe kann man seinen Standort auf eine Linie um den Globus beschränken. Die Stadt Mannheim beispielsweise liegt auf der Linie welche im 49 Grad Winkel zum Äquator steht auf der nördlichen Halbkugel. Im Vergleich hierzu liegt Sydney im 33 Grad Winkel zum Äquator auf der südlichen Halbkugel. Damit der eigene Standort nicht nur einer Linie beschrieben werden kann, welche eine Länge von 40.075km haben kann, gibt es Längengrade. Diese beiden Graden schneiden sich bei der Standortbeschreibung in einem Punkt und geben den Standort exakt an.

### Längengrade

Ähnlich wie bei Breitengraden handelt es sich bei Längengraden um gedachte Linien um den Globus. Längengrade stehen senkrecht zum Äquator und nicht parallel wie die Breitengrade. Bei diesen Graden ist zu beachten, dass sie immer den gleichen Umfang besitzen. Um den kompletten Globus abzudecken gibt es 360 Längengrade. Diese wurden aufgeteilt in westliche und östliche Grade mit jeweils 180 Grad. Für Längengrade gibt es keinen gegebenen Nullpunkt wie den Äquator. 1883 wurde sich darauf geeinigt den Nullmeridian (0. Längengrad) durch eine Sternwarte im englischen Greenwich laufen zu lassen. Seit dieser Einigung wird der Längengrad in Abhängigkeit des Winkels zur Stadt Greenwich angegeben. Für die Beispiele des letzten Abschnitts heißt das: Die Stadt Mannheim liegt auf dem 8. Längengrad östlich von Greenwich und Sydney auf dem 151. Grad östlich von Greenwich.

### Genauigkeit der Längen- und Breitengrade

Der Umfang des Globus beträgt ca. 40.000 km. Teilt man diesen nun durch 360 Längengrade erhält man einen Abstand von 111km zwischen zwei Längengraden. Das bedeutet, wenn der Längengrad lediglich mit einer Gradzahl angegeben ist, handelt es sich um eine Strecke von 111km auf dem der Standort liegen kann. Daher teilt man entweder ein Grad in 60 Minuten und eine Minute in 60 Sekunden, was eine Strecke von 30m entspricht oder man gibt zu einer Grad Zahl Dezimalstellen an. Bei der Angabe von 5 Dezimalstellen beträgt die Genauigkeit ca. 1 Meter. Durch die Krümmung der Erde, lässt sich diese Rechnung nicht auf Breitengrade übertragen.

Eine vollständige Angabe des Standortes könnte wie folgt aussehen:

49 Grad 29 Minuten N, 8 Grad 28 Minuten O

49.487459 N, 8.466039 O



Ein Nachteil dieser Standortbeschreibung ist, dass man sich nicht direkt ein Bild von der Position machen kann. Nur die wenigsten Menschen kennen sich so gut mit Koordinaten aus um sich direkt den Standort auf einer Karte vorzustellen.

Die Vorteile dieser Beschreibung liegen in der Abdeckung. Der gesamte Globus kann mit diesem Verfahren beschrieben werden. Es gibt keinen Ort, der nicht mit Längen- und Breitengraden beschrieben werden kann. [9]

## 2.2. Positionsbestimmung und Funktionsweise

*Verfasst von Patrick Senneka*

In diesem Kapitel wird zunächst einmal geklärt, was Positionsbestimmung ist. Nach Alex Küpper ist die Positionsbestimmung wie folgt definiert:

**Zitat 2.1:** „Positioning is a process to obtain the spatial position of a target “[2, S.121]  
**Übersetzung:** Positionsbestimmung ist ein Prozess, um die räumliche Position eines Ziels zu erhalten.

Mit räumlicher Position ist hierbei ein Standort gemeint, der zu einem geeigneten Bezugssystem bestimmt wird. Das bedeutet ein Position die sich auf das Bezugssystem „Weltkarte“ bezieht repräsentiert den geografischen Standort auf der Weltkarte. Die Position mit dem Bezugssystem eines bestimmten Gebäudes repräsentiert den Standort in dem Gebäude. Bsp.: Stockwerk 1 Raum 139b in der DHBW-Mannheim.

Ein weiteres Zitat bezüglich der Positionsbestimmung grenzt den Begriff Positionsbestimmung deutlich von Ortung ab. Dieser Unterschied soll hier auch aufgezeigt werden. Hierzu das Zitat der Webseite [www.itwissen.info](http://www.itwissen.info):

**Zitat 2.2:** „Die Begriffe Positionsbestimmung und Ortung werden häufig synonym benutzt; sie unterscheiden sich allerdings im Detail. So wird mit der Positionsbestimmung der Ort von Objekten oder Personen eindeutig in einem geografischen Koordinatensystem festgelegt. Sie bildet die Basis für die Ortung und wird dann zur Ortung, wenn Dritten die ermittelte Position mitgeteilt wird. “[10, Positionsbestimmung]

Das Zitat ist aussagekräftig und grenzt Positionsbestimmung und Ortung eindeutig voneinander ab.

In diesem Kapitel wird, wie es der Titel vorgibt, nur die Positionsbestimmung betrachtet. Dieser ist die Voraussetzung, um Ortung überhaupt durchzuführen. Allerdings wird im weiteren Teil dieser Arbeit verstärkt die Ortung betrachtet werden. Die Übertragung der Position spielt nämlich zur Bereitstellung eines location-based Services in nahezu allen Fällen eine große Rolle. Da die Positionsbestimmung für die Ortung benötigt wird, diese hier im Detail vorgestellt.

Bei einer Positionsbestimmung werden Messdaten gesammelt, um die eigenen Position festzulegen. Die Messdaten beziehen sich dabei immer auf festgelegte Fixpunkte, von welchen die Position schon bekannt ist. Diese Daten sind zum Beispiel Winkel, Geschwindigkeit und Entfernung.

In den kommenden Kapiteln werden unterschiedliche Methoden und Techniken zur Positionsbestimmung vorgestellt. Diese werden in drei Kategorien eingeordnet, die satellitengestützte Positionierung, Positionierung in Mobilfunknetzen und Positionsbestimmung in Gebäuden. Abbildung 6 gibt für die drei Kategorien einen Überblick.

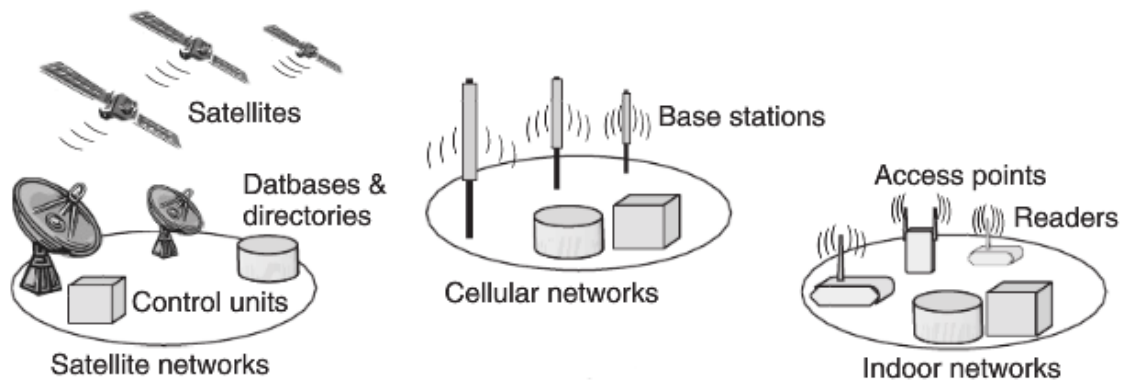


Abbildung 6: Infrastruktur zur Positionsbestimmung  
[2, S. 124]

### 2.2.1. Kriterien für die Standortbestimmung

*Verfasst von Patrick Senneka*

Kein Verfahren zur Positionsbestimmung ist perfekt. Deshalb werden die unterschiedlichen Methoden und Techniken zur Positionsbestimmung anhand von drei Qualitätskriterien betrachtet. Diese werden hier kurz erläutert.

#### 1. Bereich (Scope)

Ein Positionssystem ist immer auf einen Bereich bezogen, in dem eine Position theoretisch bestimmt werden kann. Dieser Bereich kann stark, von einem Raum bis zu einem weltweiten Bereich, variieren.

→ Ein großer Scope ist besser

#### 2. Abdeckung (Coverage)

Die Abdeckung eines Positionssystems kann maximal so groß sein, wie es der Bereich zulässt. Die Abdeckung ist die Teilmenge des Bereichs, in dem tatsächlich die Position bestimmt werden kann. So ist es beispielsweise bei einem Satelliten-

positionssystem nicht möglich eine Abdeckung in Gebäuden oder unter der Erde zu gewährleisten.

→ Eine große Abdeckung ist besser

### 3. Präzision (Precision)

Ein Positionssystem kann einen Standort nicht exakt, sondern mit einer gewissen Abweichung bestimmen. Diese Abweichung kann von Umwelteinflüssen abhängen und für eine Positionsbestimmung an der selben Stelle abweichende Ergebnisse liefern. Die Robustheit eines Positionssystem trägt somit auch zur Präzision bei.

→ Eine höhere Präzision ist besser

[3, S.183]

### 2.2.2. Satellitengestützte Positionierung

*Verfasst von Patrick Senneka*

Satellitengestützte Positionierung basiert, wie der Name schon vermuten lässt, auf Satelliten.

Das bekannteste System zur satellitengestützte Positionierung ist das Global Positioning System, dass mit GPS abgekürzt wird. Auf GPS wird nach einer Erläuterung über das generelle Funktionsprinzip von Satellitenpositionierung eingegangen.

Eine Grundvoraussetzung für Satellitengestützte Positionierung sind Satelliten, die sich in der Erdumlaufbahn befinden und elektromagnetische Wellen auf die Erde funken. Damit eine Position auf der Erdoberfläche mit Hilfe von Satelliten errechnet/bestimmt werden kann, muss auch ein Gerät vorhanden sein, dass die elektromagnetischen Wellen der Satelliten empfangen und verarbeiten kann.

Ist das gegeben kann prinzipiell die Position des Nutzers auf der Erdoberfläche anhand von der exakten Position von mindestens drei Satelliten ( $s_n$ ) und des Abstands zu diesen Satelliten ( $r_m$ ) bestimmt werden.

[3, S. 188]

In Abbildung 7 a) stellen die Kreise Positionen auf der Erde dar, an denen eine User-Position anhand des gemessenen Abstands ( $r_m$ ) möglich ist. In dieser Darstellung mit drei Satelliten ist die User-Position eindeutig an dem Schnittpunkt aller drei Kreise. In Abbildung 7 b) wird die Darstellung ins dreidimensionale überführt. Hierbei ist nun zu erkennen, dass die Kreise zu Kugeln geworden sind. Daraus resultierend gibt es nun

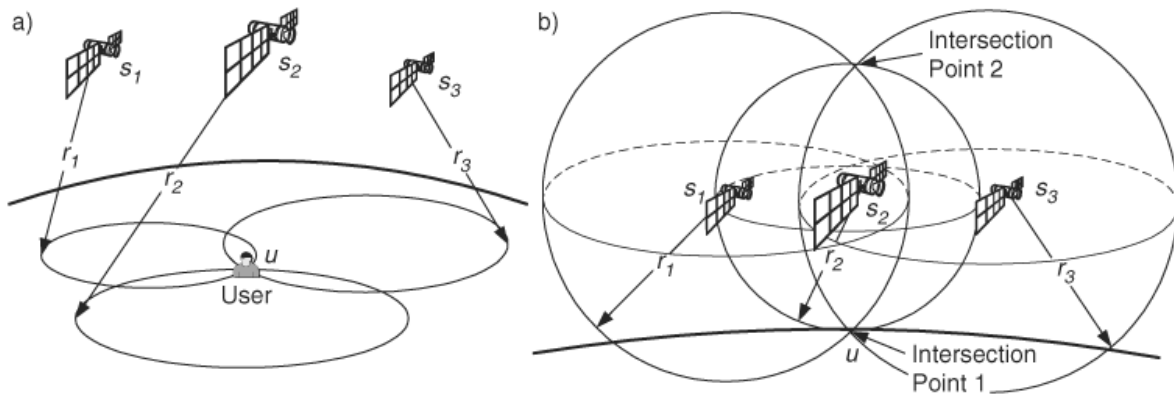


Abbildung 7: Grundprinzip Satelliten Positionsbestimmung  
[3, S. 188]

einen zweiten Schnittpunkt. Für die Positionsbestimmung gibt es nun zwei mögliche User-Positionen. Dieses Problem beseitigt man einfach, indem man die logisch wahrscheinlichere Position als richtige ansieht. In der Abbildung wäre das Intersection Point 1, da Intersection Point 2 im Weltall liegt und davon auszugehen ist, dass sich Menschen auf der Erdoberfläche befinden. [3, S. 188]

Es wurde nun erläutert, wie sich die Position ermitteln lässt, wenn dem Nutzer die Position der Satelliten und der Abstand zu diesen bekannt ist. Wie diese Werte ermittelt werden, wurde noch nicht erwähnt. Darauf wird nun eingegangen.

### Satelliten Positionen:

Satelliten kreisen um die Erde. Sie sind also Ständig in Bewegung.

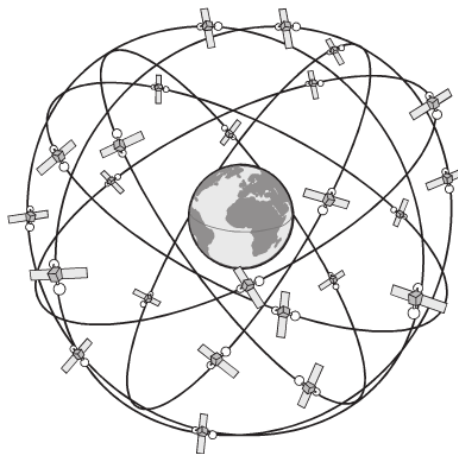


Abbildung 8: Umlaufbahn von GPS Satelliten  
[2, S. 164]

Da sie in zuvor festgelegten Umlaufbahnen kreisen, kann die Position eines Satelliten abhängig von der Zeit errechnet werden. Diese Berechnung kann ohne Probleme auf einem Smartphone stattfinden. Die einzigen Daten die dazu nötig sind, ist eine aktuelle Uhrzeit, sowie die Satellitennamen mit der dazugehörigen Umlaufbahn. Solche Daten befinden sich standardmäßig auf GPS fähigen Geräten. Bei Bedarf, durch den Ausfall oder das Hinzukommen neuer Satelliten, werden diese Daten aktualisiert. [3, S. 189]

### **Abstand zu den Satelliten:**

Der Abstand zu Satelliten wird anhand der Zeit berechnet, die das Signal vom Satelliten bis zum Empfänger benötigt. Diese Zeit wird dadurch ermittelt, dass vom Satelliten die aktuelle Zeit gesendet wird. Sobald diese beim Empfänger ankommt wird der Zeitunterschied  $\Delta t$  zur aktuellen Zeit ermittelt. Da sich elektromagnetische Wellen mit Lichtgeschwindigkeit  $c$  fortbewegen, beträgt die Geschwindigkeit näherungsweise  $300.000 \text{ km/s}$ . Mit der Formel  $r = c \Delta t$  lässt sich dann die Entfernung zum Satelliten ermitteln. Das Verfahren funktioniert allerdings nur, wenn die Uhren vom Satelliten und des Empfängers exakt synchronisiert sind. [3, S. 189]

Eine Abweichung der Uhren von nur  $1\mu\text{s}$  ergibt aufgrund der hohen Geschwindigkeit der elektromagnetischen Wellen eine Abweichung der bestimmten Position von ca. 300 Metern. [3, S. 189]

Diese Aussage verdeutlicht, wie wichtig es ist die Uhren exakt zu synchronisieren. Deshalb wird nun ein Verfahren zu Synchronisation von Uhren vorgestellt.

### **Verfahren zur Uhren Synchronisation:**

Jeder Satellit ist mit einer Atomuhr ausgestattet um immer die richtige Zeit zu senden. Eine Atomuhr in ein Smartphone einzubauen macht allein schon aus wirtschaftlicher Sicht keinen Sinn. Deshalb muss es ein Verfahren geben, mit dem die Uhren synchronisiert werden könne, oder der Unterschied zwischen den Uhren bestimmt werden kann. [3, S.189]

Das Grundprinzip dieses Verfahrens wird nun erläutert. Dazu werden zunächst einige Variablen definiert.

$t_s$  steht für die Systemzeit des Satelliten, an der das Signal gesendet wurde.

$\tilde{t}_{s=t_s+\delta t_s}$  ist die Zeit des Satelliten unter Berücksichtigung der Abweichung  $\delta t_s$  zur exakten Zeit.

$t_u$  steht für die Systemzeit der Users, zu der er das Signal empfangen hat.

Die Systemzeit des Users kann eine Abweichung zur exakten Zeit haben. Deshalb wird für die weitere Betrachtung  $\tilde{t}_{u=t_u+\delta t_u}$  unter Berücksichtigung der Abweichung  $\delta t_u$  zur exakten Zeit verwendet.

$\Delta \tilde{t} = \tilde{t}_u - \tilde{t}_s$  ist die gemessene Zeit zwischen dem Abschicken des Signals vom Satelliten bis zum Empfang des Signals beim User.

$c$  gibt die Lichtgeschwindigkeit an.

Im Abschnitt „Abstand zu den Satelliten“ wurde schon die Formel zur Berechnung des Abstands  $r = c \Delta t$  erwähnt. Diese basiert allerdings darauf, dass beide Uhren synchronisiert sind.

Hier wird diese Formel dahingehend angepasst, dass die Abweichungen  $\Delta t$  von den Systemzeiten berücksichtigt werden. Das ist dann der gemessene Abstand  $p$ :

“[3, S. 189 - 190]

”

$$p = c * \Delta \tilde{t} \tag{1}$$

$$= c * (\tilde{t}_u - \tilde{t}_s) \tag{2}$$

$$= c * ((t_u + \delta t_u) - (t_s + \delta t_s)) \tag{3}$$

$$= c * (t_u - t_s) + c * (\delta t_u - \delta t_s) \tag{4}$$

$$= r + c * (\delta t_u - \delta t_s) \tag{5}$$

“[3, S. 190]

Die Systemzeit des Satelliten ist bei dieser Betrachtung als exakt anzusehen, da Satelliten mit einer Atomuhr ausgestattet sind und von Bodenstationen kontrolliert werden.  $\delta t_s$  ist damit als 0 anzusehen.

Daraus erhalten wir  $p = r + c * \delta t_u$

$r$  kann hierbei durch die Koordinaten des Satelliten und des Users dargestellt werden:

$$p = \sqrt{(s_x - u_x)^2 + (s_y - u_y)^2 + (s_z - u_z)^2} + c * \delta t_u$$

Diese Gleichung enthält nun vier unbekannte Variablen  $u_x, u_y, u_z$  und  $\delta t_u$ .

Um diese Gleichung zu lösen benötigt man diese vier unbekannten Variablen. Jeder dieser Variable kann selbst wieder über eine Gleichung bestimmt werden. Um alle vier Gleichungen zu lösen sind dann vier Satelliten nötig.

Es gibt nun mehrere Möglichkeiten diese Lösung anzugehen. Es kann zum Beispiel durch ein Näherungsverfahren geschehen. Herr Schiller verweist in deiner Arbeit dabei auf die Möglichkeit eine Taylor Reihe.

Da es ab dieser Stelle nur noch eine mathematische Aufgabe ist die Lösung zu finden, und es dafür wiederum unterschiedliche Ansätze gibt, wird darauf nicht weiter eingegangen.

[3, S. 190]

### GPS

Das Global Position System wurde vom Amerikanischen Verteidigungsministerium entwickelt. Es gibt eine Variante, die nur dem Militär zugänglich ist und eine, die der Zivilbevölkerung frei zur Verfügung steht.

Das GPS System besteht auf 24 Satelliten, die um die Erde kreisen. Die Satelliten können über Basisstationen, die auf der ganzen Erde verteilt sind, ihre Uhren synchronisieren und ihre richtige Umlaufbahn beibehalten. [2, S. 162]



Abbildung 9: GPS Basissationen  
[2, S. 163]

Das GPS basiert grundsätzlich auf den zuvor vorgestellten Mechanismen zur Standortbestimmung.

Es gibt allerdings einen Unterschied in der militärischen und der zivilen Variante. Die militärische Variante bietet eine höhere Präzision, steht allerdings nur Ländern die der NATO angehören zur Verfügung.

Die zivile Variante bietet im Normalfall eine fast so hohe Präzision, wie die militärische, kann aber von der U.S. Army durch einen Mechanismus, der Selective Availability (SA) heißt deutlich ungenauer gemacht werden.



[3, S. 194 - 196]

Vergleiche hierzu Abbildung 10.

PPS steht für die militärische Variante wohingegen SPS für die zivile Variante des GPS steht.

Service	Horizontal Precision	Vertical Precision
PPS	22 m	27.7 m
SPS with SA	100 m	156 m
SPS without SA	25 m	43 m

Abbildung 10: GPS Präzision  
[3, S. 195]

Die Präzision von GPS kann durch Differential GPS nochmals verbessert werden. Dazu wird in einer Basisstation, zu der eine exakte Position vorhanden ist, die GPS Position ermittelt. Gibt es Abweichungen durch die Atmosphäre wird von der Basisstation eine Korrektur erstellt, die dann an den User weitergeleitet werden kann, um auch dessen Präzision zu verbessern. [3, S. 196]

Vergleiche hierzu Abbildung 11

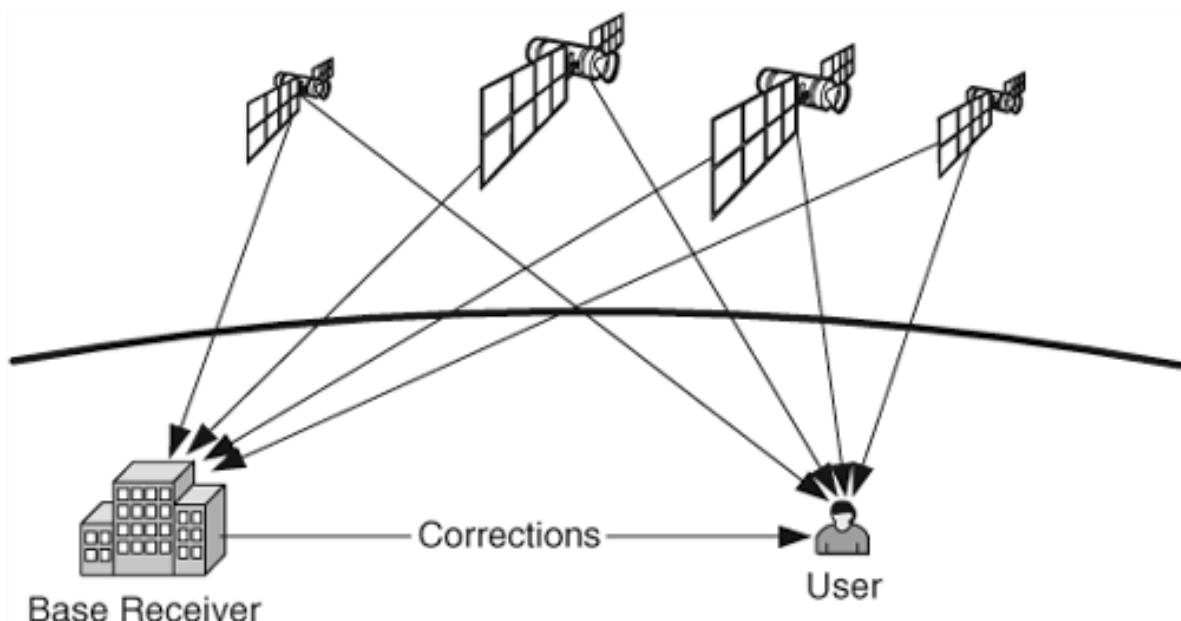


Abbildung 11: Präzision Differential GPS  
[3, S. 196]

Betrachtet man Satelliten Positionierung und im speziellen GPS kann man für die Qualitätskriterien die folgende Bewertung abgeben.

—> Bereich (Scope) Prinzipiell ist eine Standortbestimmung auf der ganzen Welt möglich.

—> Abdeckung (Coverage) Die Position kann überall auf der Erdoberfläche, mit einer Ausnahme, der Positionsbestimmung in Gebäuden, bestimmt werden. Das bietet eine sehr große Abdeckung.

—> Präzision (Precision) Eine hohe Präzision wird durch viele Satelliten und Korrektursignale ermöglicht. Die Position ist so robust gegen Umwelteinflüsse.

[3, S. 187]

Die Kosten zum Betreiben von Satelliten sind immens hoch. Es gibt allerdings auch andere Positionierungssysteme, die auf schon vorhandene Funknetze setzen. Diese werden nun vorgestellt.

### 2.2.3. Positionierung in (Mobil)Funknetzen

*Verfasst von Patrick Senneka*

Ein weit verbreitetes Funknetz ist das Mobilfunknetz. Es ist in über 190 Ländern verfügbar.[3, 206] Um zu verdeutlichen, wie groß die Abdeckung in Deutschland ist, wird die Netzabdeckung von drei großen Mobilfunkanbietern aufgezeigt. Vergleiche hierzu Abbildung 12.

Die Abbildungen zeigen eine nahezu vollständige Abdeckung Deutschlands durch das GSM Netz. Partiiell ist je nach Netzbetreiber die Abdeckung durch GSM nicht gegeben.

Zu Abbildung 12 ist noch anzumerken, dass es sich hierbei um Darstellungen der einzelnen Mobilfunkanbietern handelt. Es ist davon auszugehen, dass auf Grund von Marketingeinflüssen auf die Grafiken eine höhere Abdeckung dargestellt wird, als in der Realität vorhanden ist. Um einen groben Eindruck für die Abdeckung in Deutschland zu bekomme, sollten diese Abbildungen dennoch ausreichend sein.

Basierend auf GSM gibt es verschieden präzise Ansätze die Position eines Netzers über dessen Handy zu finden.

Der technisch einfachste aber auch unpräziseste Ansatz ist die Positionsbestimmung anhand des Funkmasten, an dem des Handy angemeldet ist. Das Prinzip dahinter ist, dass sich ein Handy immer mit dem stärksten Signal eines Funkmasten verbindet. Von

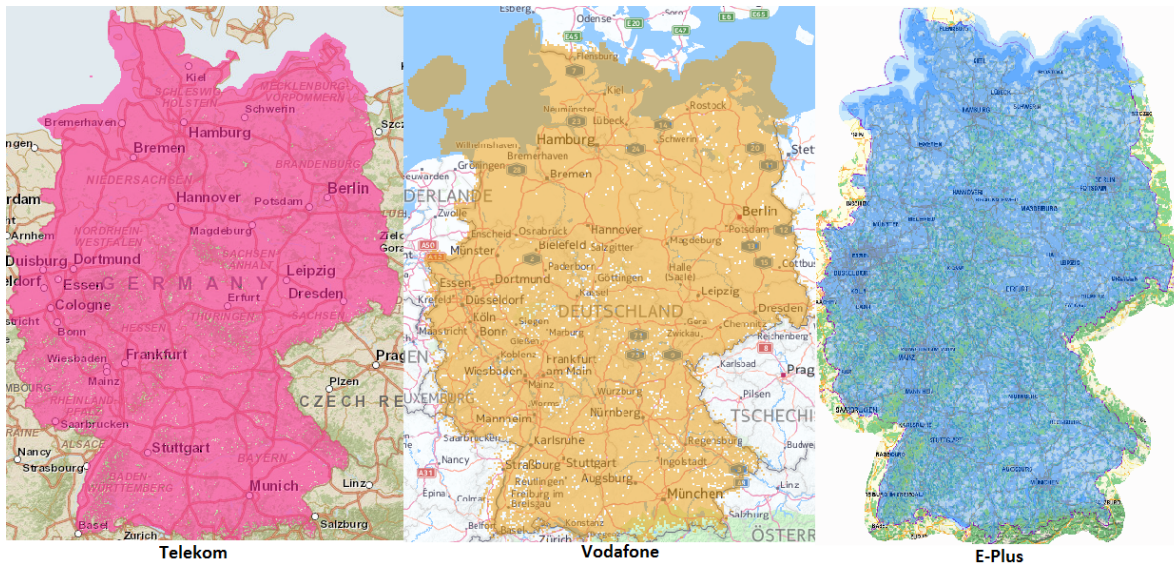


Abbildung 12: GSM Abdeckung in Deutschland  
[11] [12] [13]

dem Funkmast mit dem stärksten Signal ist auszugehen, dass es der nächstgelegene ist. Die Position, sowie ein Radius, in dem dessen Signal empfangen werden kann ist dem Betreiber eines jedem Funkmasten bekannt.

Sobald sich ein Handy mit einem Funkmasten verbindet, wird das in einer dezentralen Datenbank (Visitor Location Register) erfasst. Die Informationen aus der VLR Datenbank werden anschließend in die zentrale Datenbank des Betreibers des Funkmasten, dem Home Location Register, übertragen. Der Betreiber kann anhand seiner Positions-informationen des Funkmasten ermitteln, in welchem Bereich die Position des Handys bzw. des Nutzers ist. Vergleiche Abbildung 13.

[3, S. 207]

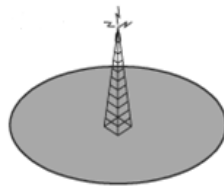


Abbildung 13: Cell of Origin  
[3, S. 209]

Die Präzision dieser Positionierung hängt von der Größe der Funkzelle ab. So ist in

Städten von einem Radius von ca. 1 km auszugehen, wohingegen in ländlicheren Regionen der Radius einer Funkzelle bis zu 35 km betragen kann. [3, S. 207]

Ein präziseres Verfahren wurde von der Firma Ericsson entwickelt und heißt Mobile Positioning System (MPS). [3, S. 207 - 208]

Dieses Verfahren besteht aus mehreren Teilverfahren, die hier nacheinander vorgestellt werden.

### 1. Cell of Global Identity

Dieses Verfahren entspricht den zuvor beschriebenen Positionsbestimmung anhand der Funkzelle. Vergleiche Abbildung 13. Diese Variante des MPS bildet die Grundlage für alle weiteren.

### 2. Segment antennas

Eine Funkzelle breitet sich meist  $360^\circ$  um einen Funkmasten aus. Dieser Bereich kann nicht von einer Antenne abgedeckt werden. Dafür werden normalerweise drei bis vier Antennen benötigt. Jede einzelne Antenne ist dann für einen geringeren Winkel zuständig ( $90^\circ$  oder  $120^\circ$ ).

Die Position eines Handys kann anhand der genutzten Antenne auf ein Segment der Funkzelle eingeschränkt werden. [3, S. 208]

Vergleich hierzu Abbildung 14.

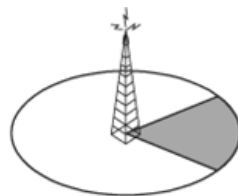


Abbildung 14: Antennen Segment  
[3, S. 209]

### 3. Timing Advance

In einer Funkzelle gibt es mehr als ein Gerät. Deshalb gibt es ein „Time-division multiplexing“, was jedem Gerät einen Zeitslot gibt, in dem es mit der Funkzelle kommunizieren kann. Dieses Verfahren berücksichtigt auch die Laufzeit des Signals von einem Handy zu einem Funkmasten. Die Informationen über die Laufzeit des Signals können dazu verwendet werden die Entfernung des Geräts zu einem Funkmasten zu bestimmen. Die Entfernung kann dabei in Abstufungen von ca. 555

Metern ermittelt werden. Unter Zuhilfenahme des Segment antenna Verfahrens kann die Präzision der Position so weiter verbessert werden. [3, S. 208]

Vergleiche hierzu Abbildung 15.

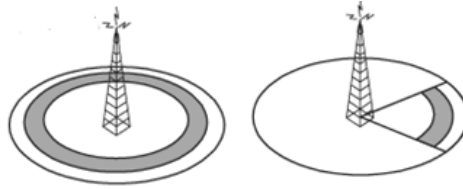


Abbildung 15: Timing Advance mit Segment antenna  
[3, S. 209]

#### 4. Uplink Time of Arrival

Befindet sich ein Handy in mindestens 4 Funknetzen kann zu allen verfügbaren Funknetzen nach dem „Timing Advance“ Verfahren die Entfernung bestimmt werden.

Ähnlich wie bei der Satelliten Positionierung ist es dann möglich eine genaue Position zu ermitteln. Diese ist dann auf 50 - 150 Meter genau. [3, S. 209]

Vergleiche Abbildung 16.

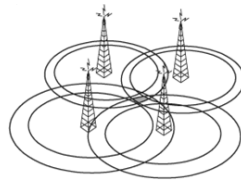


Abbildung 16: Uplink Time of Arrival  
[3, S. 209]

→ Bereich (Scope) Die Standortbestimmung ist in 190 Ländern möglich.

→ Abdeckung (Coverage) Die Abdeckung der Funknetze ist in den Ländern sehr hoch. Es ist sogar möglich in Gebäuden den Standort zu bestimmen.

→ Präzision (Precision) Die Präzision liegt im besten Fall bei 50 - 150 Metern, was für viele LBS schon ausreichend sein dürfte.

#### 2.2.4. Positionsbestimmung in Gebäuden

*Verfasst von Patrick Senneka*

Systeme zur Positionsbestimmung in Gebäuden haben eine vergleichsweise geringe Abdeckung. Sie bieten sich besonders für den Einsatz in Gebäuden an, da diese eine Geringe Fläche haben und da dort Verfahren wie GPS versagen.

Auf zwei Verfahren, die Positionsbestimmung in Gebäuden wird nun im genauer eingegangen. Zuerst wird die Positionsbestimmung anhand von WLAN Netzwerken dargestellt. Danach werden Radio Beacons genauer betrachtet.

### WLAN Netzwerke

Heutzutage sind in vielen Gebäuden WLAN-Netzwerke verfügbar. Ein gutes Beispiel dazu ist die DBHW-Mannheim. Dort ist in nahezu jedem Raum ein WLAN Accesspoint vorhanden.

Von Microsoft gibt es ein prototypisches Verfahren, welches anhand der Signalstärke aller verfügbaren WLAN Accespoints eine Positionierung in WLAN-Netzwerken ermöglicht.

Das Verfahren basiert darauf, dass man Messpunkte erstellt. Zu den erstellten Messpunkten ist die Position bekannt. Aus vielen Messpunkten ergibt sich dann eine Tabelle.

[3, S. 210]

Eine solche Tabelle sieht nach Herr Schillers Buch Location-based Services so aus:

$x/m$	$y/m$	$d/^\circ$	$SS_1/dBm$	$SS_2/dBm$	$SS_3/dBm$	$SS_4/dBm$
1.0	3.5	0	20	10	18	25
2.0	3.5	90	25	15	17	25
2.5	3.0	90	15	18	16	16
2.5	1.5	180	6	35	18	20
2.0	2.5	0	12	10	22	14

Abbildung 17: WLAN Messwerttabelle  
[3, S. 210]

Anhand solch einer Tabelle kann zur Positionierung die Signalstärke der einzelnen WLAN Accesspoints ermittelt werden und auf den am besten passenden Standort gemappt werden. So ist eine Positionierung in einem zuvor festgelegtem Raster möglich.

Das Problem an dem Verfahren ist, dass wenn zu wenig Accespoints vorhanden sind, wird die Position schnell ungenau. Außerdem sind Messungen zum Konfigurieren erforderlich.

Ein alternativer Ansatz wäre Triangulation. Das ist ähnlich zum dem Verfahren, das bei GPS vorgestellt wurde. Der Abstand zu den Accesspoint kann anhand er Signalstärke errechnet werden. Die Position von den Accesspoint ändert sich nicht und ist bekannt. Damit kann aus dem Schnittpunkt der Radium am die Accesspoints der eigene Standort ermittelt werden.

[3, S. 209-211]

→ Bereich (Scope) Die Reichweite eines WLAN Accesspoint beträgt maximal 100 Meter. In Deutschland und anderen Ländern ist WLAN nicht flächendeckend Verfügbar, sondern nur in wenigen Städten vorhanden. Der Bereich ist meist auf ein Gebäude beschränkt.

→ Abdeckung (Coverage) Die Abdeckung von WLAN Netzwerken entspricht ziemlich genau dem Scope.

→ Präzision (Precision) Die Präzision von WLAN Positionierung ist bis auf wenige Meter genau. Damit ist eine hohe Präzision gegeben.

### **Radio Beacons**

Radio Beacons sind kleine Geräte, die ständig elektromagnetische Wellen senden. Diese übermitteln Daten, wie die ID des Radio Beacons, bis zu einer Entfernung von 30 Metern.

Alleine durch die Lokalisation eines Beacons (Funkzelle) ist eine hohe Genauigkeit gegeben. Beacons können so konfiguriert werden, dass ihr Signal nur in einem Radius von ca. 5 Meter empfangen werden kann. Ermittelt ein Gerät (Handy) solch einen Beacon ist die eigene Position schon sehr präzise bestimmt.

Auch bei dieser Technologie ist es wieder möglich Signale von mehreren Beacons zur Positionsbestimmung zu verwenden. Dadurch wird die ermittelte Position genauer. Bei Beacons ist es sogar möglich eine 3D-Position zu ermitteln.

Ähnlich zu dem bei Satelliten Positionsbestimmung beschriebenem Verfahren ist es auch hier wieder möglich eine genaue Position zu ermitteln, wenn man Signale von mehreren Beacons empfängt. Das Verfahren beruht auf einer Abstandsmessung anhand der Time of Arrival von Signalen. Ein beispielhaftes Verfahren dafür heißt SpotOn. Nutz man SpotOn wird eine Präzision der Position von 3 Meter erreicht. Das ist ein sehr gutes Wert.

[3, S. 201 - 204]

Vergleiche Abbildung 18.

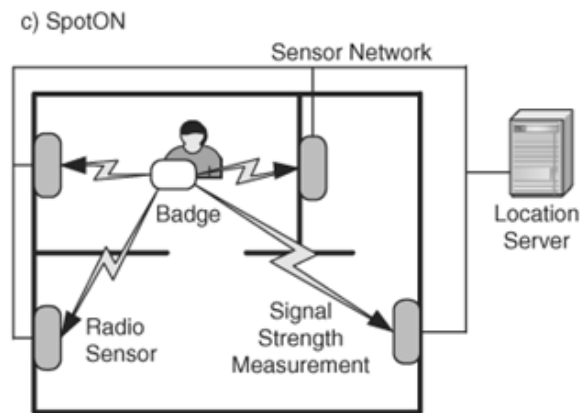


Abbildung 18: SpotOn  
[3, S. 201]

→ Bereich (Scope) Der Bereich kommt auf die Anzahl der Beacons an, ist aber als klein einzustufen.

→ Abdeckung (Coverage) Die Abdeckung in dem gegebenen Scope sollte 100 prozentig sein.

→ Präzision (Precision) Die Präzision ist mit 3 Metern sehr gut.

### 2.2.5. Kombination von Verfahren zur Positionsbestimmung

*Verfasst von Patrick Senneka*

Jedes der zuvor vorgestellten Verfahren hat Vor- und Nachteile, die hauptsächlich in der Präzision und der Abdeckung liegen. Mit Smartphones ist es möglich all die vorgestellten Verfahren zu nutzen, da diese fast immer mit GPS, WLAN und Bluetooth, sowie immer mit Mobilfunk ausgestattet sind. Bei diesen Möglichkeiten ist es naheliegend, dass man eine Kombination der Verfahren zur Positionsbestimmung verwendet, um stets die präziseste Position zu ermitteln.

Bei Cordova/Phonegap werden zur präzisen Positionsbestimmung die Verfahren GPS, IP, RFID, WLAN und GSM eingesetzt. [?] Die Verfahren laufen parallel, so dass jedes eine Position ermittelt und die Präzision dieser liefert. Der präziseste Standort wird dann für die App verwendet.



---

## 3. Anwendungsfälle für LBS

Dieses Kapitel zeigt auf, welche Funktionen und Möglichkeiten durch LBSs vorhanden sind. Unterteilt ist das Kapitel in mehrere Abschnitte. Der erste erläutert die Historie von LBSs, anschließend wird im Abschnitt "Theoretische Einsatzgebiete" aufgezeigt, welche Funktionen durch LBSs gegeben sind. Diese Funktionen werden anschließend im Abschnitt "Hauptnutzer von LBS" mit Beispielen aus der Umwelt erläutert.

### 3.1. Historie von LBS

*Verfasst von Victor Schwartz*

Erfunden wurden LBS von der U.S. Abwehrbehörde. Hierfür wurde das System Navstar entwickelt. Mit Hilfe von Satelliten kann die Position bis auf wenige Meter bestimmt werden. Bekannt geworden ist dieses System unter dem Namen Global Positioning System (GPS). Nutzer dieses Systems war das Militär. 1980 entschied man sich dazu, dass das System der Öffentlichkeit bereitzustellen. Ziel dieses Schrittes war es, Fortschritte in der Entwicklung von Satellitensystemen zu machen. Mit dieser Entscheidung wurden die Voraussetzungen geschaffen, LBS im privaten Umfeld nutzen zu können.[14]

Die Europäische Union entwickelte daraufhin mit der Europäischen Raumfahrt Behörde (ESA) ein eigenes System namens Galileo.

**Theoretische Einsatzgebiete** Die Autoren Allan J. Brimicombe und Chao Li unterscheiden in ihrem Buch "Location-Based Services and Geo-Information Engineering" [15, S.132] zehn verschiedene Einsatzgebiete. Einige davon werden in den folgenden Abschnitten erläutert:

- **Navigation** Seit der kostenfreien und öffentlichen Nutzung des GPS hat sich viel in der Navigationsbranche getan. Immer mehr Geräte verfügen über GPS-Empfänger, die eine Lokalisierung ermöglichen. Mit Hilfe des genauen Standortes ist es möglich, den Nutzer des Gerätes zu navigieren. Bei einer Navigation benötigt das System einen Start- und Zielpunkt. Das Gerät berechnet eine Route und informiert den Nutzer über Distanzen und Richtungsänderungen kurz bevor diese ausgeführt werden müssen, um der Route folgen zu können. Dies ist dem Gerät möglich, da es sich um einen proaktiven LBS handelt, der ständig den eigenen Standort abfragt.

Folgt der Nutzer den Anweisungen befindet er sich am Ende der Navigation am Ziel.

- Wegfindung

Im Gegensatz zur Navigation wird bei der Wegfindung nur bei der Planung der Route der Standort benötigt. Deshalb handelt es sich um einen reaktiven LBS. Eine Führung zum Ziel findet nicht statt. Im Umfeld von LBS wird bei einer Wegfindung der eigene Standort als Startpunkt gesetzt und ein Zielpunkt muss vom Nutzer angegeben werden. Ein Wegfindungsalgorithmus berechnet daraufhin eine Route. Diese kann beispielsweise auf einer Karte dargestellt werden oder jede Richtungsänderung wird mit einer Strecke in Textform aufgelistet. Moderne Wegfindungsprogramme erlauben die zusätzliche Angabe von Routenkriterien. Unter anderem kann die kürzeste Strecke favorisiert werden oder die schnellste.

- Echtzeit-Verfolgung

Ein weiteres Einsatzgebiet von LBS sind Verfolgungs- und Tracking-Systeme. Diese liefern in Echtzeit den Standort des Gerätes, welches den Empfänger enthält. Dies kann beispielsweise genutzt werden um einen Freund in einem schwer überschaubaren Gebiet zu finden. Derjenige der gefunden werden möchte, muss seinen Standort bestimmen lassen und diesen an den Suchenden übermitteln. Möglich ist dies voll automatisiert durch Apps bei Smartphones. Ein ähnlicher Anwendungsfall ist die Ortung der eigenen Kindes. Das Handy des Kindes sendet in regelmäßigen Abständen die Position an eine Webseite und die Eltern können sich den Standort über eine Karte betrachten. Diese LBS können sowohl als reaktiver bzw. proaktiver Dienst implementiert sein. Dies ist abhängig davon, ob der Standort dauerhaft gesendet wird oder nur einmalig.[16] [17]

- Elektronischer Handel

Im Zeitalter des Internets können viele Informationen und Aufgaben kostenlos im Internet abgerufen und bearbeitet werden. Einige Beispiele sind: Zeitung lesen, Recherchen durchführen, Musik hören, einkaufen. Daher gewinnt das Werbeschalten im Internet immer mehr an Bedeutung. Neben der personalisierten Werbung welche durch Nutzerdatenerfassung ermöglicht wird, spielt der Standort des Nutzers eine Rolle zum Schalten geeigneter Werbung. Neben Versand- und Online-Händlern gibt es viele Firmen, die ihre Produkte überwiegend in der Produktionsregion verkaufen. Für diese Händler ist Hyperlokale Werbung von großer Bedeutung. Dabei wird zuerst der Standort des Nutzers ermittelt. Je nachdem auf

was der Zugriff (GPS,WLAN) durch den Nutzer erlaubt ist, kann der Standort bis auf wenige Meter bestimmt werden. Anschließend kann gezielt Werbung über das mobile Internet geschaltet werden. Diese kann auf Webseiten oder in Applikationen angezeigt werden. [18] [19]

- User-solicited Informations (vom Nutzer gewünschte Informationen)  
Unter diese Kategorie fallen alle Anwendungen, die vom Nutzer für den geschäftlichen oder sozialen Gebrauch genutzt werden. Beispiele hierfür sind: Wetterprognosen, Zugverspätungen und Filmvorführungen.
- Ortsgebundene Tarife  
Am Anfang der 1990er Jahre wurden flächendeckend digitale Mobilfunknetze ausgebaut. Mit dieser Technik ist es mit einem Handy möglich von überall aus zu telefonieren. Abgerechnet wird pro telefonierter Minute bzw. Sekunde. Die Preise waren deutlich teurer im Vergleich zu Festnetztelefonen. Aus diesem Grund hat man ortsgebundene Tarife eingeführt. Über die Cell-ID, welche den ungefähren Standort des Handys mitteilt, können Anbieter günstigere Tarife anbieten. Bei Vertragsabschluss kann der Kunde seine Heimatadresse angeben. In einem definierten Radius beispielsweise 3km, wird dann beim Telefonieren über das Handy ein günstigerer Tarif berechnet. Ziel ist es, dem Kunden eine alternative für das Festnetz zu bieten. Je nach Anbieter kann bei Vertragsabschluss zusätzlich zur Handynummer eine Festnetznummer gegeben werden, welche nur aktiv wird wenn der Kunde sich in dem definierten Radius befindet. Bekannte Beispiele sind: Homezone des Anbieters O2, Vodafone Zuhause von Vodafone oder T-Mobile@home von Telekom. Im Laufe der Zeit haben sich diese Tarife nicht durchgesetzt und mit der Einführung von Flatrates haben sie immer mehr an Bedeutung verloren. [20]

## 3.2. Hauptnutzer von LBS

*Verfasst von Victor Schwartz*

LBS wurden erstmals vom amerikanischen Militär erfunden und genutzt. Nachdem die Services der Öffentlichkeit bereitgestellt wurden, führte dies zu immer mehr Anwendungsbereichen, beispielsweise zur Lokalisierung von Notrufen. In Europa findet dies über die Rufnummer „112“ statt, in Amerika „911“. Seit 1996 besteht in den USA eine Pflicht, bei einem Notruf den ungefähren Standort mitzusenden.

Im Laufe der letzten Jahre wurden immer mehr Möglichkeiten geschaffen, mobil Telefone

zu lokalisieren und den Standort für beispielsweise Informationsdarstellung zu nutzen. Damit ergeben sich drei große Anwendungsbereich von LBS:

- Militär
- Lokalisierung von Notrufe
- Kommerzielle Nutzung für Privatpersonen und Unternehmen.

Im folgenden Abschnitt werden die genannten Hauptnutzer von LBS genauer erläutert und aufgezeigt wofür LBS verwendet wird.

#### LBS im Umfeld des Militärs

Den eigenen Standort zu kennen und beschreiben zu können ist nicht immer auf Anhieb möglich. Befindet man sich an einem Ort, bei dem viele Sehenswürdigkeiten oder bekannte Gegenstände wie beispielsweise Häuser, Parks, Straßen in der Nähe sind, fällt es einem meist einfacher den eigenen Standort einer anderen Person mitzuteilen, damit dieser einen findet.

Wesentlich schwieriger ist die Standortbestimmung, wenn man sich an einem Ort befindet, der sehr allgemein ist und keine Besonderheiten bzw. Identifikationsmerkmale aufweist. Oft befindet sich das Militär an solchen schwer zu definierenden Orten. Einige Beispiele für solche Orte sind: Wüsten, Wälder, Berge und Gebirge. Vermutlich war dies einer der Hauptgründe ein System zur Bestimmung des Standortes zu entwickeln.

Die Anwendung FBCB2 ist ein Beispiel für die Verwendung von mehreren Standorten. Sie wird bereits seit 10 Jahren vom amerikanischen Militär eingesetzt. FBCB2 ist die Abkürzung für „Force-Twenty-One Battle Command Brigade and Below“. Die Anwendung ist bei Panzerbrigaden im Einsatz. Auf einer Karte wird dem Nutzer angezeigt welche verbündeten Panzer in der Nähe sind. Zu jedem dieser Panzer werden weitere Informationen bereitgestellt. Mit diesem System braucht man keinen Kompass und keine Papierkarte mehr um sich einen Überblick zu verschaffen. [21]

Im Kriegsgeschehen allgemein nimmt die Bedeutung von LBS stark zu. Genutzt wird diese Technik unter anderem bei Lenkraketen. Der Befehlshaber braucht nur die Koordinaten anzugeben und die Rakete berechnet den optimalen Weg zum Ziel. Essentiell wichtig ist dabei für die Rakete zu jedem Zeitpunkt im Flug zu wissen an welchen Standort sie sich befindet um gegebenenfalls die Geschwindigkeit oder Höhe anzupassen.

Neben Lenkraketen gibt es immer mehr unbemannte Kriegsflugzeuge, sogenannte Drohnen. Gesteuert werden dies nicht aus dem Cockpit des Flugzeugs, sondern am Boden über einen Joystick. Der Joystick ist mit einem Computer verbunden und per Kame-

ra kann der „Pilot“ sehen, wohin er fliegt. Auch in diesem Anwendungsfall ist es von sehr großer Bedeutung, dass der Pilot jederzeit weiß, wo er sich befindet und in welche Richtung er fliegen muss. [22]

#### LBS zur Lokalisierung von Notrufen

Benötigt man schnelle Hilfe, dann ist es von großem Vorteil, wenn der Helfer schnellstmöglich an sein Ziel kommt. Voraussetzung dafür ist es, das Ziel zu kennen.

Deshalb werden LBS bei Notrufen verwendet. Die Idee dahinter ist es, den Standort des Anrufers an die Leitstelle zu übermitteln, die den Notruf entgegennimmt. Dies ist nur bei Notrufen von Mobiltelefonen möglich. Noch während des Anrufes kann ein Rettungswagen oder ein Einsatzwagen der Feuerwehr in Richtung des Anrufers aufbrechen. Hilfe ist bereits für den Anrufer unterwegs, während der Standort detailliert mitgeteilt wird.

Bereits 1996 wurde in Amerika ein Gesetz verabschiedet, das den Mobilfunkanbieter dazu verpflichtet, den ungefähren Standort des Anrufers bei einem Notruf mitzuteilen. 2003 wurde für Europa ein ähnliches Gesetz verabschiedet. Das amerikanische Gesetz wurde 2001 überarbeitet, und die Genauigkeit des Standortes muss nun zwischen 50-300 m liegen.

Aus einer Quelle von 2004 lässt sich entnehmen, dass in den USA ca. 33% (170.000 täglich) und in Europa 50-70% (220.000 täglich) der Notruf mit Hilfe eines Mobiltelefons getätigt werden. Man geht davon aus, dass mit dieser Technik ca. 5000 Menschenleben jährlich gerettet werden können.

Zur Ortung wird kein GPS-Modul benötigt, über sogenannten ID-Zellen kann der Standort des Anrufes bestimmt werden.[3]

#### Nutzung im kommerziellen Umfeld

**Praktische Einsatzgebiete** Nach einer Goldmedia-Analyse [23, S.9] verteilen sich die deutsche LBS-Marktstruktur 2014 auf 15 unterschiedliche Gebiete, einige werden in den folgenden Abschnitten erläutert:

- **Tourismus**

Mit Hilfe von LBS ist es möglich, auf einen Stadtführer aus Papier zu verzichten. Informationen können für Touristen in einer Applikation über das Smartphone bereitgestellt werden. So bekommt der Tourist immer genau die Informationen an-

gezeigt, die für die Gegenstände oder Gebäude in der Nähe relevant sind. Ebenfalls kann der Tourist seine eigene Stadttour bestreiten, da eine integrierte Navigation zu interessanten Punkten realisiert werden kann. Des Weiteren können POI „Points of Interest“ in Abhängigkeit vom eigenen Standort als mögliche Ziele vorgeschlagen werden. POI können beispielsweise Sehenswürdigkeiten, Museen, Restaurants oder Parkhäuser sein. Alle diese Informationen können mit einem Gerät bereitgestellt werden.

- **Beförderung und Verkehr**

Im Bereich öffentlicher Personennahverkehr ergeben sich durch LBS neue Möglichkeiten. Der Verkehrsverbund Rhein – Neckar beispielsweise bietet eine App an, welche unter anderem den Standort des Nutzers ermittelt und daraufhin die nächste Haltestelle in der Nähe als Start definiert und anzeigt, welche Linien der Busse und Straßenbahnen von dieser Haltestelle abfahren. Auch wird beschrieben wie man zu dieser Haltestelle gelangt.

Auch für Autofahrer kann LBS von Vorteil sein. In der App „Maps“ von Google kann der aktuelle Verkehr auf öffentlichen Straßen angezeigt werden. So ist es dem Fahrer möglich vor der Abfahrt zu überprüfen ob auf den geplanten Strecken ein Stau ist. Mit Hilfe von LBS werden anonymisierte Standortdaten an Google gesendet. Erkennt der Algorithmus, dass auf einer Autobahn viele Standortdaten mit geringer Geschwindigkeit vorhanden sind, wird auf der Karte ein Stau dargestellt und in die Routenplanung von Google Maps aufgenommen. Dies ist wiederum ein weiteres Einsatzgebiet von LBS. [24]

- **Navigation und Maps**

Bereits bei den theoretischen Einsatzgebieten wurde die Navigation und Wegfindung genauer erläutert. Zum Einsatz kommen diese Technologien in Handys mit GPS-Modul und Navigationsgeräten.

- **Gastronomie**

Im Gastronomiebereich finden LBS auch einen Anwendungsbereich. Beispielsweise können sich Nutzer über Restaurants in der Nähe informieren. Diese können auf einer Karte mit weiterführenden Informationen dargestellt werden. Informationen wie Öffnungszeiten, Art der Küche (chinesisches, deutsches, italienisches etc.) oder Bewertungen durch andere Nutzer. Bekannt aus dem TV sind aber auch Apps, die viele Lieferdienste in der Umgebung anzeigen. Anhand des Standortes wird dem

Nutzer eine Liste von Lieferdiensten in seiner Umgebung zusammengestellt. Zwei Beispiele sind: Lieferando und Lieferheld.

- Taxi, Carsharing und Bikesharing

In städtischen Gebieten braucht man für den Alltag nicht unbedingt ein Auto. Meist befinden sich viele Geschäfte in der Nähe oder sind über öffentliche Verkehrsmittel gut erreichbar. Trotzdem kommt es vor, dass ein Auto gebraucht wird. Zum Beispiel für größere Einkäufe wie Möbel oder schwere Gegenstände. Deshalb gibt es Carsharing-Anbieter. Über eine Internetseite kann sich ein Nutzer dort registrieren und sich dann stunden- oder minutenweise ein Auto mieten. Besonders ist hierbei, dass es keine expliziten Mietstationen gibt. Die Autos werden einfach in der Stadt geparkt. Möchte ein Kunde ein Auto mieten, kann er das in der Nähe parkende nehmen. An dieser Stelle spielt LBS eine wichtige Rolle. Die Autos sind mit Transpondern ausgestattet. Auf einer Karte des Nutzers werden alle Autos in der Nähe abgebildet. Mit dieser Technik ist es möglich schnell und einfach ein Auto des Vermieters zu finden. Ähnlich ist das Verfahren bei Bikesharing-Anbietern und Taxis. In diesem Fall sind die Fahrräder bzw. Taxis mit Transpondern ausgestattet und der Nutzer kann sich das nächstgelegene aussuchen.

- Sport

Sogenannte Sport-Apps nutzen ebenfalls LBS. Es handelt sich hierbei um Programme zur Aufzeichnung und Dokumentation sportlicher Aktivitäten. Fährt der Nutzer mit dem Fahrrad oder geht joggen, wird die zurückgelegte Strecke über den eigenen Standort ermittelt, ebenso die Geschwindigkeiten. Mit einem Computer oder Smartphone kann anschließend die eigenen Aktivitäten betrachtet werden.

## 4. Prototypische Umsetzung

Um theoretisches Wissen mit praktischer Umsetzung zu verbinden, wurde neben der Theoretischen Behandlung von LBS ein Location Based Service entwickelt. Da Smartphones einen immer größeren Stellenwert in der Gesellschaft haben und meist ein GPS-Modul verbaut ist, wurde sich dazu entschieden eine Applikation für Smartphones zu entwickeln. Im Vordergrund der Entwicklung steht nicht ein fertiges Produkt zu entwickeln, sondern einen Prototypen zu implementieren welcher mehrere Grundfunktionalitäten im LBS Umfeld enthält.

Damit der Prototyp nicht zu abstrakt wird, wurde sich dazu entschieden ein Spiel zu entwickeln.

Um die erlangten Kenntnisse im Bereich der Location Based Services zu vertiefen, wurde im Verlauf dieser Arbeit auch eine App entwickelt, die die Grundzüge der Positionierung nutzt.

Die App soll dem Nutzer ermöglichen auf spielerische Weise die Koordination zu verbessern. Das Spiel basiert auf der Funktionsweise einer Wünschelrute. Dem Spieler werden dabei, basierend auf seiner Startposition verschiedene Aufgaben gestellt. Diese sind in der Form gestellt, dass der Spieler eine ganz bestimmte Position innerhalb eines selbstgewählten Radius erreichen muss. Für eine erfolgreiche Annäherung an das Ziel bekommt der Spieler Punkte, für ein entfernen Minuspunkte.

Das Ziel des Spiels ist, bis auf eine Abweichung von einem Meter die Ziel-Position zu erreichen. Dafür ist eine ständige Verfolgung seiner Position nötig, über die der Spieler jederzeit Rückmeldung erhält und sich so, wie bei einer Wünschelrute, langsam an sein Ziel annähern kann.

### 4.1. Anforderungen

*Verfasst von Melanie Hammerschmidt*

Die Analyse im Vorfeld der Implementierung ergab die folgenden Anforderungen. Unterschieden wurden diese im Bezug auf Fachlichkeit und technischen Anspruch.

1. Fachliche Anforderungen
  - a) Grundeinstellungen für Spieler speichern
    - i. Name



- ii. Spielradius
    - iii. Startposition
  - b) Aufgaben erstellen (entsprechend Startposition eines Spielers)
    - i. Erreichen von definierten Orten/Koordinaten
      - A. entsprechend gewähltem Radius
      - B. steigender Schwierigkeitsgrad
    - ii. Erreichen einer gewissen Höhe
  - c) Aufgaben spielen
    - i. Prüfung der aktuellen Position
    - ii. Vergleich zwischen Aufgabe und aktueller Position
    - iii. ständige Rückmeldung (verbleibende Entfernung zum Ziel) an den Spieler
    - iv. Punkte neu berechnen (sammeln oder verlieren)
    - v. Punkte in Highscore speichern
    - vi. Möglichkeit, das Spiel zu unterbrechen und später wieder zu starten
2. Technische Anforderungen
- a) Plattformunabhängigkeit (vgl. Kapitel Technologien und Entscheidungen)
  - b) Benutzbarkeit
    - i. übersichtliche Steuerung
    - ii. Selbstbeschreibend
    - iii. Erwartungskonformität
    - iv. Fehlertoleranz
  - c) gute Animation der ”Wüschelroute”

## 4.2. Architektur

*Verfasst von Melanie Hammerschmidt*

Die App ist streng nach dem Konzept des Model-View-Controller aufgebaut.

Model-View-Controller ist ein Design Pattern, das häufig für heutige Programmierung

eingesetzt wird. Es setzt eine strikte Trennung von Teilkomponenten voraus, die für eine hohe Wiederverwendbarkeit und einen strukturierten Aufbau sorgen. Im Falle einer GUI-Anwendung trennt man bei Anwendung dieses Patter das Model, die View und den Controller. Das Pattern ist also lediglich eine Erweiterung der schon seit langem verbreiteten Trennung von Eingabe (Controller), Verarbeitung (Model) und Ausgabe (View).

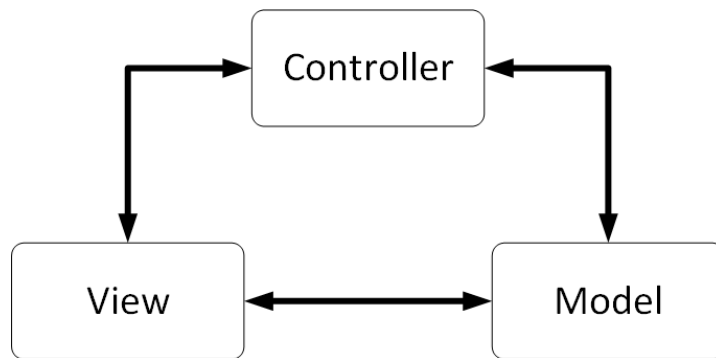


Abbildung 19: Modell-View-Controller Pattern

Das Model sorgt sich dabei allein um die Verarbeitung der Daten einer Anwendung. Alle Änderungen, die an den Daten ausgeführt werden sollen, finden im Model statt. Ein Model kann mit beliebig vielen Views verbunden werden und so Informationen austauschen.

Eine View kümmert sich um die graphische Darstellung der Anwendungsdaten, die es von seinem konkret zugewiesenen Model erhält.

Der Controller ist für die Möglichkeiten des Anwendungsnutzers zur Interaktion zuständig. Er übernimmt die Zuordnung von Benutzereingaben zu Funktionen des Models. Um diese Überleitung zu ermöglichen müssen sowohl Verbindungen zu Models als auch zu den Views existieren.

Entsprechend dieses Patterns ist die Architektur der hier beschriebenen App ebenfalls strukturiert gehalten. Einfach gesagt, besteht die App lediglich aus einer HTML-Seite, die immer wieder mit verändertem Inhalt dargestellt wird. Diese wird mithilfe von Javascript (Angular JS) ständig mit dem Inhalt einer neuen View gefüllt.

Für jede dargestellte Seite der App existiert eine View. Diese besteht ebenfalls wieder aus einer einfachen HTML-Seite, die allerdings über einen extra dafür definierten Controller mittels Variablen- und Funktionszuweisung verändert werden kann. Dieser Controller weißt auch die Methoden, die dann z.B. bei einem Button-Druck ausgeführt werden können, die entsprechende Verarbeitung zu.

Diese Verarbeitung findet in sogenannten Services statt. In jedem Controller besteht die Möglichkeit auf diese Services zuzugreifen. Sie bilden das Model in unserer Model-View-Controller-Darstellung. Sie enthalten die Daten und können sich sowohl mit dem lokalen Speicher der App als auch mit den Daten des Geräts verbinden. Außerdem können an dieser Stelle auch selbst definierte Werte und damit verbundene Funktionen hinterlegt werden, die jeder Controller aufrufen kann.

Um zwischen den einzelnen View wechseln zu können, wird immer ein aktueller Status der App gespeichert. Dieser Status entscheidet, welche View und welcher dazugehörige Controller in dem Moment verwendet werden. Über den Status der App kann man ganz leicht zwischen den Views unterscheiden und jeder Status hat seinen eigenen Historienstapel. Dadurch lassen sich vergangene Klicks leicht rückgängig machen.

Allgemein kann man zwei verschiedene Prozess-Ablaufarten unterscheiden.

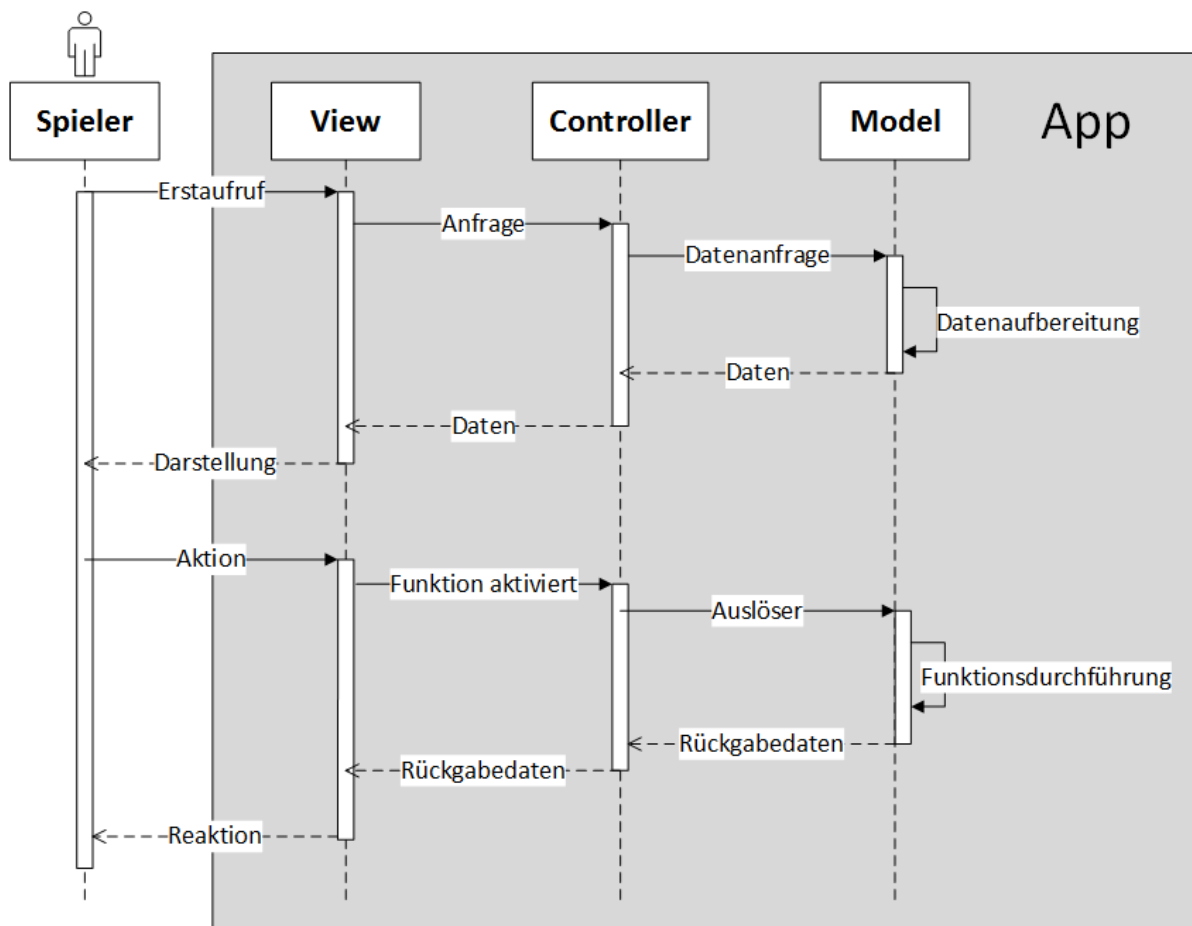


Abbildung 20: Ablauf der Modell-View-Controller

Die View kann Daten zur Anzeige anfordern, die sie selbst nicht besorgen kann oder der Nutzer aktiviert z.B. durch Klick auf einen Button eine Funktion, die der Controller (bzw. dann der Service) anbietet.

### 4.3. Technologien und Entscheidungen

*Verfasst von Victor Schwartz*

Aus den Anforderungen heraus wurde sich im Rahmen dieser Arbeit für diverse Technologien entschieden, welche eine Umsetzung der Anforderungen des Prototypen ermöglichen.

Eine Hauptanforderung für die prototypische Umsetzung einer LBS Applikation ist die Lauffähigkeit auf einem Smartphone. Diese Anforderung fällt nicht weiter auf, ist aber entscheidend für die Auswahl der zu nutzenden Technologien.

Möchte man eine Applikation für Smartphones entwickeln, muss man sich erst Gedanken darüber machen, welche Smartphones bzw. mobilen Betriebssysteme unterstützt werden sollen.

Unterschieden wird meist nur zwischen drei Betriebssystemen, da diese die größten Marktanteile in Deutschland besitzen (Siehe Abbildung).

Es handelt sich um folgende Betriebssysteme:

- Android
- iOS
- Windows

Um den Implementieraufwand geringer zu halten, wurde entschieden die Applikation auf den zwei Betriebssystemen mit dem größten Marktanteil lauffähig zu halten. Android und iOS haben gemeinsam einen Anteil von über 50 Prozent. Aus dieser Anforderung ergaben sich drei verschiedene Möglichkeiten zur Entwicklung:

**Implementierung als WebApp** Bei dieser Art von Programm handelt es sich um eine Webseite, die für mobile Geräte optimiert ist.

**Implementierung des Programm als native App** Bei einer nativen App handelt es sich um ein Programm, das speziell für ein mobiles Betriebssystem entwickelt wurde.

**Implementierung als HybridApp** Eine HybridApp ermöglicht es mit einem Quellcode mehrere App's erstellen zu lassen. Zur Verwendung kommen dabei Web-Technologien.

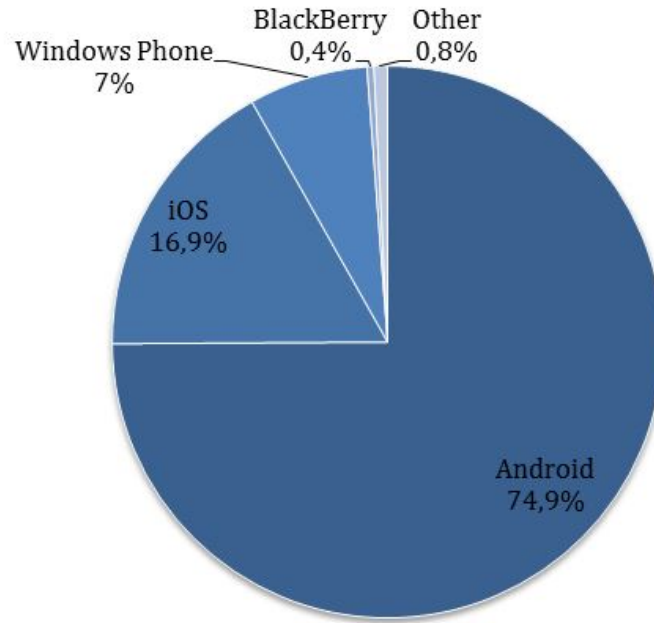


Abbildung 21: Marktanteile mobiler Betriebssysteme  
[25]

Diese drei Möglichkeiten sollen im folgenden Abschnitt anhand von definierten Kriterien bewertet werden.

Folgende Kriterien wurden für die Auswahl einer Methode betrachtet:

1. Zugriff auf Hardware des Mobiltelefons

Zur Implementierung einer LBS Applikation wird der eigene Standort benötigt. Um diesen zu bestimmen gibt es mehrere Verfahren (siehe Kapitel Standortbestimmung). Die dafür benötigten Informationen sind über die Hardware des Smartphones erreichbar. Beispielsweise kann der Standort über den eingebauten GPS-Empfänger bestimmt werden. Für die Implementierung der App ist es von Nöten, dass diese Informationen ausgelesen werden können.

2. Voraussetzungen und Vorkenntnisse

Da diese Studienarbeit innerhalb von wenigen Monaten entstanden ist und andere Projekte parallel bearbeitet werden mussten, ist ein Kriterium über welche Vorkenntnisse die Entwickler verfügen. Dieses Kriterium betrachtet die eingesetzten Programmiersprachen und technischen Geräte, die benötigt werden um eine LBS-App entwickeln zu können.

### 3. Aufwand

Dieses Kriterium ergänzt das vorhergehende. Lässt sich aber auch klar abgrenzen. Da die Zeit für diese Studienarbeit begrenzt ist, darf die Implementierung nicht zu viel Zeit in Anspruch nehmen, da sonst die Anforderungen nicht umgesetzt werden können. Betrachtet wird hierbei der Aufwand um die Anforderungen zu implementieren und die App betriebssystemunabhängig zu entwickeln.

### WebApp

Bei einer WebApp (Webapplikation) handelt es sich um eine Anwendung, die im Browser ausgeführt wird. Eine für Smartphones optimierte WebApp fokussiert sich darauf die Darstellung für kleine Bildschirme zu optimieren. Auch wird darauf geachtet, dass diese mit einem Touchscreen bedient werden kann. Ziel ist es, dass der Benutzer kaum einen Unterschied zu einer nativen App erkennt. Berechnungen werden bei dieser Variante meist serverseitig durchgeführt.

#### 1. Kriterium: Zugriff auf Hardware-Komponenten

Ein Nachteil der Webapplikation ist, dass es keine Standard-Hardwareschnittstellen für eine Browser-Anwendung gibt. Das bedeutet, eine WebApp kann nicht auf die Hardware zugreifen. Beispielsweise ist eine Standortbestimmung über GPS oder der Zugriff auf ein Bluetooth Modul nicht möglich.

#### 2. Kriterium: Voraussetzungen und Vorkenntnisse

Da es sich um eine Webseite handelt, kommen die bekannten Technologien zur Entwicklung zum Einsatz. Hierzu zählen: HTML Hypertext Markup Language, Cascading Stylesheets und JavaScript.

Durch die an der dualen Hochschule erworbenen Kenntnisse in den Vorlesungen Web Engineering und Software Engineering, sind erweiterte Vorkenntnisse über diese Technologien bei allen Entwicklern dieser Arbeit vorhanden und stellen kein Problem dar. Zur Entwicklung reicht ein Editor sowie ein Internetbrowser.

#### 3. Kriterium: Aufwand

Betrachtet man die aufgezählten Technologien und Werkzeuge, so ist der Aufwand für die Entwicklung einer LBS-WebApp gering. Auch die Plattformunabhängigkeit stellt für eine WebApp keine Schwierigkeiten dar, da die genannten Web Technologien standardisiert sind. Allerdings fehlen definierte Schnittstellen für die Hardware-Ansteuerung. Der Aufwand, eine Schnittstelle zu entwickeln, würde den Rahmen

dieser Studienarbeit überschreiten. Selbst wenn eine Schnittstellen-Implementierung gelingt, so wäre diese nicht geräte- und damit plattformunabhängig.

### **Native App**

Bei nativen Apps handelt es sich um Programme, welche für ein spezifisches, mobiles Betriebssystem optimiert und lauffähig sind. Eine Ausführung auf einem Computer oder anderem Betriebssystem ist, wenn überhaupt, nur schwer umsetzbar. Installiert werden kann dies über einen betriebssystemabhängigen Marktplatz. Wie bereits in der Anforderungen erwähnt, soll die Applikation auf iOS und Android Geräten verfügbar sein, weshalb die zu untersuchenden Kriterien für beide Betriebssysteme betrachtet werden.

#### 1. Kriterium: Zugriff auf Hardware-Komponenten

Sowohl Android wie auch iOS unterstützen einen Zugriff auf die Hardware. Dieser erfolgt über APIs (Applikation Programming Interface) und bietet alle mit dem Gerät zur Verfügung stehenden Funktionalitäten an.

#### 2. Kriterium: Voraussetzungen und Vorkenntnisse

Android Entwicklung: Die Programmiersprache für native Android Apps ist Java. Durch die Vorlesung Java Programmierung sind erweiterte Kenntnisse vorhanden. Zur komfortablen Entwicklung werden eine Entwicklungsumgebung (z.B. Eclipse), das aktuelle Java SDK (Software Development Kit) und Android SDK benötigt. Alle diese Komponenten sind kostenfrei verfügbar.

iOS Entwicklung: Die Programmiersprache für native iOS Apps ist Objective-C. Zwar sind Kenntnisse über die Sprache C von zwei Entwicklern vorhanden, allerdings ist die Sprach C nur bedingt mit Objective-C vergleichbar. Es sind auf Grund dessen nur minimale Kenntnisse vorhanden.

Zur Implementierung einer iOS App wird neben einem Apple Computer diverse Software benötigt. Hierzu zählt die Entwicklungsumgebung Xcode. Möchte man die App auf einem Gerät testen, so wird ein Apple Developer Account, sowie das IOS Developer Programm benötigt, dieses ist kostenpflichtig.

#### 3. Kriterium: Aufwand

Möchte man eine App für beide Betriebssysteme erstellen, so muss jede App einzeln geschrieben werden. Es kann kein Code wiederverwendet werden, da sich die Programmiersprachen unterscheiden. Ebenfalls sind andere APIs vorhanden, was die Entwicklung erschwert. Der Aufwand ist hierbei so hoch wie für zwei Apps.

### **Hybrid App**

Laut Duden ist die Definition für den Begriff Hybrid "gemischt". Dies ist auch bei solchen Apps der Fall. Mit bekannten Web-Technologien wird der Code für die Apps geschrieben. Ein Framework übernimmt im Anschluss daran, die Generierung der nativen Apps.

#### 1. Kriterium: Zugriff auf Hardware-Komponenten

Es gibt verschiedene Frameworks um eine Hybride App zu entwickeln, deshalb kann man nicht pauschal sagen welche Zugriffe auf Hardware-Komponenten möglich sind. Allerdings bieten bekannte Frameworks wie Angular UI, Intel XDK oder Adobe Phonegap eine direkte Ansteuerung an Komponenten an. Hierzu zählen beispielsweise Kamera, Kompass, Beschleunigungssensor. Realisiert wird der Zugriff über APIs. Neben dem mitgelieferten APIs der Frameworks gibt es Plug-Ins welche von anderen Entwicklern zur Verfügung gestellt werden. Damit haben Hybrid Apps Zugriff auf eine Vielzahl von Hardware-Komponenten.

#### 2. Kriterium: Voraussetzungen und Vorkenntnisse

Entwickelt werden Hybrid Apps mit Web Technologien. Diese sind bereits bei Kriterium 2 von WebApps behandelt worden, ebenfalls die Vorkenntnisse.

Zu den Voraussetzungen zählt die Installation von Software. Ein Framework wird benötigt. Diese unterscheiden sich in den bereitgestellten Funktionalitäten. Zur Entwicklung mit Web-Technologien wird mindestens ein Editor benötigt. Damit das Framework native Apps generieren kann, wird für die diversen Betriebssysteme SDKs benötigt. Außerdem ist es hilfreich ein Gerät pro Betriebssystem zu besitzen, um die App zu testen. Ein andere Möglichkeit ist der Einsatz eines Emulators, der ein Smartphone auf dem PC simuliert.

Alle diese Werkzeuge können kostenlos heruntergeladen werden.

#### 3. Kriterium: Aufwand

Möchte man eine App für verschiedene Betriebssysteme entwickeln und den Aufwand gering halten, eignen sich dafür Hybrid Apps. Neben der eigentlichen Entwicklung ist es mit Aufwand verbunden alle Werkzeuge zu installieren und zu konfigurieren. Hat man diese Hürde geschafft, ist der Aufwand gering zu bewerten. Ist die Implementierung abgeschlossen, muss dem Framework mitgeteilt werden, welche Apps gebildet werden sollen und der Rest funktioniert automatisch. Der Gesamtaufwand für die Entwicklung einer Hybrid App ist niedrig bis mittel einzustufen.



## Zusammenfassung und Fazit

Fasst man die gewonnen Erkenntnisse über die Technologien zusammen ergibt sich folgende Tabelle:

Zugriff auf Hardware	Web App Schlecht, „nur auf wenige Hardware Komponenten kann zugegriffen werden“	Native App Vorteilhaft, da auf alle Hardware Komponenten zugegriffen werden kann
Vorraussetzungen und Vorkenntnisse	durch, die Vorlesung "Web Engineering" sind ausreichend Kenntnisse vorhanden	Keine besonderen Kenntnisse erforderlich
Aufwand	Gering, da eine Webseite auf jedes Betriebssystem lauffähig ist	Hoch, da für jedes Betriebssystem eine eigene App implementiert werden muss

Betrachtet man diese Tabelle wird deutlich, dass eine Möglichkeit am besten zu dieser Studienarbeit passt, die Umsetzung des Prototypen als Hybrid App. Eine Hybrid App vereint die Vorteile der WebApp mit den Vorteilen einer Native App. Durch den schlechten Hardware Zugriff der Web App ist die Möglichkeit ungeeignet. Bei einer nativen App ist der Aufwand zu groß, da für jedes Betriebssystem eine eigene App implementiert werden muss. Durch Plug-Ins bietet eine Hybride App viele Möglichkeiten, die Hardware von Smartphones anzusteuern. Über ein Framework kann in wenigen Minuten der geschriebene Quellcode für mehrere Betriebssysteme genutzt werden um eine App zu generieren. Aus diesen Gründen ist der Prototyp als Hybride App entwickelt worden. Verwendet wurde ein Framework von Adobe, dieses wird im nächsten Abschnitt erläutert. Darauf aufbauend findet eine kurze Vorstellung der Web Technologien statt.

TODO: An dieser Stelle wird eine Zusammenfassung der Erkenntnisse sowie ein Fazit hinzugefügt.

Gliederung:

Zusammenfassung

-Jeweils Vor- und Nachteile der einzelnen Technologien werden aufgezeigt

Das Fazit dieser Gegenüberstellung ist, dass eine Hybride Applikation genau die geforderten Vorteile der Betriebssystemunabhängigkeit und einfachen Nutzbarkeit bietet. Der zu erstellende LBS-Prototyp wird demnach als Hybrid-App entwickelt werden.

In diesem Zusammenhang sind auch die Technologien Apache Cordova, Ionic, HTML5, CSS und Angular-JS relevant. Außerdem ist es wichtig, die verwendbaren Kartenmaterialien zu unterscheiden und zu bewerten, sowie die in Kapitel XXX verwendete Technologie der iBeacons.

### 4.3.1. Phonegap und Apache Cordova

*Verfasst von Melanie Hammerschmidt*

PhoneGap entstand als Framework zur Entwicklung von Apps auf Basis von Web-Technologien wie HTML, CSS und JavaScript. 2011 wurde die Firma Nitobi Software, die PhoneGap einst entwickelte, von Adobe Systems aufgekauft, die das Produkt weiter unter dem Namen Apache Cordova fokussierten.

Im Rahmen der hier vorgestellten Arbeit wurde Cordova genutzt, um eine hybride App zu entwickeln. Da der Vorteil des Frameworks vor allem aus seiner Wiederverwendbarkeit besteht. Der geschriebene HTML-, CSS- und JavaScript-Code kann nämlich über Cordova entsprechend der verschiedenen Mobilplattformen unterschiedlich transformiert werden, um die entsprechende Applikation im richtigen Format zu erstellen. Unterstützt werden dabei die verbreitetsten Betriebssysteme Android und iOS, aber auch weitere wie Windows Phones, Blackberrys oder sogar FirefoxOS. Damit ist die fertige App später für alle zuvor vorgestellten Betriebssysteme verfügbar.

### 4.3.2. Ionic

*Verfasst von Melanie Hammerschmidt*

Ein weiteres genutztes Framework ist Ionic. Es handelt sich dabei um ein HTML5-Framework, das sich um das Benutzerinterface einer Web-Applikation kümmert. Der Fokus des Frameworks liegt dabei speziell auf mobilen Endgeräten. Ionic bietet, in Zusammenarbeit mit Angular JS, einige UI-Komponenten, die eine Oberfläche leicht in modernem Licht erscheinen lassen.

### 4.3.3. HTML5

*Verfasst von ??*

Durch die Entscheidung für Cordova und Ionic ist die Verwendung von HTML5 vorgegeben. Um eine Hybride App über Cordova erstellen lassen zu können, müssen die Texte in HTML-Seiten beschrieben werden, damit sie dann bei der Erstellung interpretiert werden können. In diesem Abschnitt wird die Technologie HTML5 im Allgemeinen erläutert.

HTML steht für Hypertext Markup Language (Übersetzt: Hypertext Auszeichnungssprache). Es handelt sich um eine textbasierte Auszeichnungssprache zur Strukturierung

von Elementen auf einer Webseite. Über Tags werden Elemente differenziert und identifiziert. Für diverse Anwendungsfälle gibt es HTML-Tags wie beispielsweise: Überschriften, Absätze, Tabellen, Grafikelemente. Diese Tags werden mit den Zeichen `<` und `>` umschlossen. Mit der Beschreibung `<TagName>` wird ein Tag geöffnet. Die meisten geöffneten Tags müssen mit dem Element `</TagName>` geschlossen werden.

HTML liegt aktuell in der 5. Version vor, weshalb der Name HTML5 gewählt wurde. Im Laufe der Zeit (1992- bis heute) wurden immer mehr Funktionen hinzugefügt. In den Anfängen von HTML war es nur möglich Text darzustellen. Mittlerweile gehören Bilder und Farben zum Standard, mit HTML5 können mittlerweile auch komplexe Animationen erstellt werden. Durch die Standardisierung von HTML durch das W3C können nahezu alle Browser in HTML5 entwickelte Webseiten anzeigen. Diese Funktionalitäten werden für den Prototypen verwendet.

#### 4.3.4. CSS

*Verfasst von ??*

CSS steht für Cascading Style Sheets (gestufte Gestaltungsbögen). Es handelt sich um eine Gestaltungssprache welche unter anderen in Verbindung mit HTML genutzt wird. CSS werden genutzt um Design für Webseiten zu gestalten. Zu einem Design können mit CSS Farben, Formen, Schriftarten und vieles mehr gehören. Durch die Trennung von Text mit HTML und Design durch CSS ist es möglich mehrere Webseiten mit ähnlichem oder gleichen Design zu gestalten. Realisiert wird dies mit einer CSS-Datei. In einer CSS Datei sind meist mehrere Klassen vorhanden. Jede Klasse besitzt verschiedene Eigenschaften wie Farbe, Schriftart oder Schriftgröße. Möchte man diese Klassen in einem HTML Dokument verwenden, so verknüpft man die CSS-Datei mit einem speziellen Tag mit der HTML Seite und kann dann beispielsweise Textabschnitten oder Überschriften die definierten Klassen zuweisen.

TODO!!!

<http://www.w3.org/standards/webdesign/htmlcss>

#### 4.3.5. Angular-JS

*Verfasst von Melanie Hammerschmidt*

Angular JS ist ein Framework, das voll auf JavaScript aufbaut und komplett klientenseitig ausgeführt wird.

Angular JS ist das Framework, welches die „Zwei-Wege-Datenbindung“ [?] unterstützt, die in unserem Prototypen so häufig verwendet wird. Es können dabei Daten sowohl von der dargestellten Seite übergeben sowie aufgenommen und dargestellt werden. Dieser gezielte Datenaustausch macht die Applikation enorm flexibel im Aufbau und in der Nutzung und zusätzlich die Architektur deutlich einfacher und strukturierter.

Außerdem werden durch Angular JS die Trennung von Darstellungsschicht (View) und Datenschichten (Controller und Model) unterstützt. Diese Trennung ist im Verlauf der Entwicklung von großer Bedeutung (vgl. Architektur). Angular JS bietet also kurz gesagt viele weitere Alternativen zu den herkömmlichen HTML-Tags. Nähere Implementierungsdetails zu Views, Controllern und Models folgen in den Kapiteln zur Architektur und Implementierung.

### 4.3.6. Kartenmaterial

*Verfasst von Patrick Senneka*

Kartenmaterial im Browser bzw. einer Hybrid-App ist ein essentieller Bestandteil von Location based Services. Durch eine Positionsbestimmung alleine erhält man nur Daten die für den Nutzer nicht anschaulich sind. Diese liegen normalerweise als geografische Koordinaten vor, die in geografischer Breite und geografischer Länge angegeben werden. Eine Beispielposition soll die Bedeutung von Kartenmaterial für den Nutzer von Location based Services verdeutlichen.

Als Beispiel hierfür wurde die Position der DHBW Mannheim in der Coblitzallee gewählt. Hierbei werden die geografischen Koordinaten, eine Adresse und ein Kartenausschnitt in einer Tabelle gegenübergestellt. Siehe hierzu Tabelle 1.

In der Tabelle sind verschiedenen Ortsdaten zur Verfügung gestellt, die jeweils Vor- und Nachteile aufweisen.

Die geographischen Koordinaten geben die Position am genauesten an, sind aber für fast keine Nutzer einer App anschaulich und bieten deshalb keinen großen Nutzen.

Die Adresse ist im Alltag am geläufigsten und somit für Nutzer am verständlichsten.


Geographische Koordinaten	Adresse	Kartenausschnitt
49°28'27.6"N 8°32'03.9"E	Duale Hochschule Baden-Württemberg Mannheim Coblitzallee 1-9 68163 Mannheim (Neustheim)	

Tabelle 1: Bedeutung von Kartenmaterial

Allerdings ist die Angabe nicht so genau wie die geographischen Koordinaten, denn die Angabe Hausnummer 1-9 gibt einen relativ großen Bereich an.

Die Vorteile eines Kartenausschnitts sind, dass die Detaillierung vom Nutzer angepasst werden kann. Des Weiteren werden viele grafische Informationen angezeigt, wie zum Beispiel den eigene Standort, an denen sich ein Nutzer orientieren kann. Der Nachteil dieser Variante ist, dass die Kartenausschnitte eine Abhängigkeit von externen Quellen und einem höheren Programmieraufwand mit sich bringen.

Auf Smartphones gehört Kartenmaterial und dessen Integration in Apps mittlerweile zum Standard, an welchen sich Nutzer gewöhnt haben. Und über die Hälfte der Deutschen, 44 Millionen nutzen schon ein Smartphone. Die Tendenz ist dabei steigend. Dies ist das Ergebnis einer Bitkom Studie. [1] Aus diesem Grund sollte auch Kartenmaterial in die Location based Services App integriert werden, welche die Autoren bei dieser Studienarbeit prototypisch entwickeln.

Mögliche Quellen für das Kartenmaterial sind „Google Maps“, „Bing Maps“ und „Open Street Maps“.

Da Kartenmaterial eine zentrale Quelle der App darstellen wird, ist die Auswahl eines Anbieters von großer Bedeutung. Aus diesem Grund wird hier eine genaue Analyse durchgeführt, welches Kartenmaterial sich am besten für diese App eignet.

### Anforderungen an das Kartenmaterial

Die Anforderungen an interaktives Kartenmaterial, bezüglich der in dieser Studienarbeit

entwickelten App, lassen sich in zwei Gruppen einteilen, die funktionalen und nichtfunktionalen Anforderungen.

Die nichtfunktionalen Anforderungen sind:

1. Kostenlose Abfragen
2. Ohne Account nutzbar
3. Gute Dokumentation mit Codebeispielen

Die funktionalen Anforderungen an das interaktive Kartenmaterial sind:

1. JavaScript API
2. Unterstütze Browser
3. Eigenen Standort anzeigen
4. Markierungen auf der Karte setzen
5. Markierungen bündeln (optional)

Bevor „Google Maps“, „Bing Maps“ und „Open Street Map“ bezüglich der Anforderungen untersucht werden, müssen diese genauer spezifiziert werden.

Zuerst widmen wir uns den nichtfunktionalen Anforderungen.

1. Kostenlose Abfragen

Da es sich bei der Implementierung um einen Prototypen für diese Studienarbeit handelt und dieser nicht kommerziell verwendet werden soll, sollen auch die Abfragen (map-loads) kostenlos sein. Zudem sollten genug kostenlose Abfragen zur Verfügung stehen. Bei drei Entwicklern und Tests über die Dauer der Studienarbeit (8-9 Monate) darf das Kontingent der kostenlosen Abfragen nicht aufgebraucht sein.

2. Ohne Account nutzbar

Die Nutzung ohne Account vereinfacht den Einstieg für das Kartenmaterial und sollte deshalb gewährleistet sein.

3. Gute Dokumentation mit Codebeispielen

Eine gute Dokumentation der API des Kartenmaterials mit vielen Codebeispielen erleichtert den Einstieg deutlich. Da die Teammitglieder dieser Studienarbeit noch keine Erfahrung mit Kartenmaterial haben, ist dies eine besonders wichtige Anforderung.

Als nächstes werden die funktionalen Anforderungen genauer erläutert.

1. JavaScript API

Eine JavaScript API ist essentiell wichtig, da der Prototyp mit HTML, CSS und Java Script entwickelt wird.

2. Unterstützte Browser

Als unterstützte Browser sollten die Browser von den drei großen Smartphone-Betriebssystemen Android, IOS und Windows Phone unterstützt werden, um sich die Möglichkeit offen zu halten die prototypische App auch auf anderen Plattformen zu nutzen.

3. Eigenen Standort anzeigen

Der eigene Standort muss grafisch auf einer interaktiven Karte angezeigt werden. Das Zentrieren des Kartenmaterials auf den eigenen Standort soll zudem möglich sein.

4. Markierungen auf der Karte setzen

Eigenen Markierungen müssen auf der Karte gesetzt werden könne. Dies muss grafisch erfolgen, denn es ist für den Prototypen besonders wichtig zu veranschaulichen, in welchem Bereich sich Ziele befinden.

5. Markierungen bündeln (optional)

Markierungen auf der Karte sollten gebündelt werden, wenn der Zoom-Faktor zu klein wird. Dies soll der Übersichtlichkeit bei vielen Markierungen auf der Karte dienen. Hierbei soll der Radius, in dem Markierungen gebündelt werden, eingestellt werden können.

Diese Anforderung ist als optional gekennzeichnet, da eine Bündelung erst bei vielen Zielen zu Übersichtlichkeit nötig ist und zunächst einmal mit wenig Zielen entwickelt wird.

## **Google Maps**

Seit 2005 gehört zu dem Produktportfolio von dem Suchmaschinenriesen Google ein Internet-Kartendienst namens Google Maps. Er gehört zu den verbreitetsten und erfolgreichsten Internet-Kartendiensten der Welt. [10, Lexikon Google Maps][26, S.88]

Zunächst wird Google Maps anhand der nichtfunktionalen Anforderungen bewertet, danach werden die funktionalen Anforderungen betrachtet.

### **Nichtfunktionale Anforderungen**

#### 1. Kostenlose Abfragen

Die Google Maps API steht generell als kostenloser Dienst zur Verfügung. Dieser darf in Webseiten, sowie mobile Apps eingebaut werden. Als Voraussetzung für die kostenlose Nutzung gilt allerdings, dass die eigene Webseite oder mobile App für alle Endnutzer kostenlos ist und öffentlich zugänglich sein muss.

Unter der kostenlosen Lizenz dürfen bis zu 25.000 Kartenladevorgänge pro Tag erfolgen. Ein Kartenladevorgang ist als initiales Laden der Karte definiert. Das bedeutet, dass Nutzerinteraktion mit der Karten nicht als erneuter Ladevorgang gewertet wird. In den Nutzungsbedingungen von Google wird darauf verwiesen, dass eine Sperre von mehr als 25.000 Abfragen pro Tag erst dann durchgeführt wird, wenn die Begrenzung mehr als 90 Tage in Folge überschritten werden sollte. Diese Auslegung erscheint sehr nutzerfreundlich. Für diese Studienarbeit sollten die 25.000 Abfragen pro Tag in jedem Fall völlig ausreichen. [27, Nutzungsbedingungen][27, Lizenzierung]

#### 2. Ohne Account nutzbar

Seit dem die Google Maps JavaScript API in der Version 3 vorliegt ist die Nutzung ohne einen Schlüssel möglich. Das bedeutet, dass keine Registrierung bzw. Anmeldung zum Nutzen nötig ist. Diese Praxis erleichtert es den Einstieg in Google Maps, da man sofort starten kann. [28]

#### 3. Gute Dokumentation mit Codebeispielen

Google Maps bietet ein Entwicklerhandbuch für die JavaScript API v3, dass sowohl ausführlich ist, als auch viele Beispiele bietet. Zudem ist das Benutzerhandbuch in deutsch abrufbar, was das Erarbeiten und Nachlesen vereinfacht. Zum ausführlichen Entwicklerhandbuch gibt es nochmals fast 150 Beispiele, zu denen der jeweils passende Output (Karte) angezeigt wird.[27, Documentation]

Neben der Dokumentation von Google selbst gibt es auch andere Tutorials im Netz. Eines davon ist von der Webseite [www.w3schools.com](http://www.w3schools.com), dass versucht von Grund auf eine Einführung in Google Maps zu geben.

### **funktionale Anforderungen**

#### 1. JavaScript API

Mit der Google Maps JavaScript API Version 3 bietet Google eine JavaScript API die sich sehr leicht in Webseiten integrieren lässt.



Bei einer HTML-Webseite muss lediglich ein Script Tag eingefügt werden, in dem als Quelle (src) die Google Maps API zu finden ist. Vergleiche Listing Zeile 1. Danach können mit JavaScript Karten von Google Maps erstellt werden. Vergleiche Listing Zeile 2 - 12.

```
1 <script src="https://maps.googleapis.com/maps/api/js?v=3.exp"></script>
2   <script>
3     var map;
4     function initialize() {
5       var mapOptions = {
6         zoom: 8,
7         center: new google.maps.LatLng(-34.397, 150.644)
8       };
9       map = new google.maps.Map(document.getElementById('map-canvas'),
10        mapOptions);
11     }
12     google.maps.event.addDomListener(window, 'load', initialize);
13   </script>
```

[27, Codebeispiel Simple Map]

## 2. Unterstütze Browser

Von der Google Maps JavaScript API 3 werden fast alle gängigen Browser unterstützt. Diese sind:

„IE 7.0 und höher (Windows)

Firefox 3.0 und höher (Windows, Mac OS X und Linux)

Safari 4 und höher (Mac OS X und iOS)

Chrome (Windows, Mac OS X und Linux)

Android

BlackBerry 6

Dolfin 2.0 und höher (Samsung Bada) “[27]

Die offizielle Unterstützung des Windows Phone Internetexplorers fehlt hierbei.

## 3. Eigenen Standort anzeigen

Den eigenen Standort kann man, sofern dieser bestimmt werden konnte (Listing Zeile 1 - 5), mit der Google Maps API auf der Karte anzeigen. Dies kann man zum

Beispiel mit einer Informationsbox, die man mit „new google.maps.Infowindow“ auf dem eigenen Standort erstellt, erfolgen. (Listing Zeile 7 - 11) Zur besseren Veranschaulichung wird die Karten dann noch auf die eigene Position zentriert (Listing Zeile 13)

```
1  // Try HTML5 geolocation
2  if(navigator.geolocation) {
3      navigator.geolocation.getCurrentPosition(function(position) {
4          var pos = new google.maps.LatLng(position.coords.latitude,
5                                             position.coords.longitude);
6
7          var infowindow = new google.maps.InfoWindow({
8              map: map,
9              position: pos,
10             content: 'Location found using HTML5.'
11         });
12
13         map.setCenter(pos);
14     }
```

[27, Codebeispiel Geolocation]

### 4. Markierungen auf der Karte setzen

Markierungen könne mit der API sehr leicht gesetzt werden. Neben der Position der Markierung muss noch eine Referenz auf die Google Maps Karte, sowie ein Name bei der Erstellung angegeben werden. Vergleiche Listing:

```
1  var marker = new google.maps.Marker({
2      position: new google.maps.LatLng(-25.363882,131.044922),
3      map: map,
4      title: 'Hello World!'
5  });
```

[27, Codebeispiel Simple Markers]

Den Markierungen auf der Karte können allerdings auch Bilder mit dem Attribut „icon“ zugeordnet werden. Des Weiteren kann man mit dem Attribut „draggable“ einstellen, ob man die Markierung verschieben kann oder nicht.

### 5. Markierungen bündeln (optional)

Markierungen können in der Standard JavaScript API in Version 3 nicht gebündelt werden.

Mit einer zusätzlichen Bibliothek von Google kann diese Funktionalität allerdings

ergänzt werden. Die Bibliothek heißt „google-maps-utility-library-v3 “. Mit Hilfe dieser Bibliothek kann ein „Markercluster “erstellt werden, dass Markierungen bei einer gewissen Zoomstufe bündelt. Hierbei wird dem Markercluster ein Array der Markierungen, eine Referenz auf die Karte, sowie Markercluster-Einstellungen.

```
1 var mcOptions = {gridSize: 50, maxZoom: 15};
2 var markers = [...]; // Create the markers you want to add and
    collect them into a array.
3 var mc = new MarkerClusterer(map, markers, mcOptions);
```

[29]

### **Bing Maps**

Bing Maps ist der Kartendienst des Softwarekonzerns Microsoft. Neben Kartenmaterial bietet der internet-basierte Dienst auch Satellitenbilder und Luftaufnahmen.

### **Nichtfunktionale Anforderungen**

#### 1. Kostenlose Abfragen

Bing Maps bietet für öffentlich zugängliche Webseiten, sowie für mobile Apps für Konsumenten ein kostenloses Kontingent von 125.000 Transaktionen pro Jahr. Will man dieses Kontingent von Transaktionen überschreiten, werden Kosten fällig. [30]

#### 2. Ohne Account nutzbar

Bing Maps ist nicht ohne einen Account nutzbar. Bevor man mit der dazugehörigen API entwickeln kann, ist es nötig zuerst eine Microsoft ID anzulegen, mit der man dann wiederum einen Entwickler Key für Bing Maps anfordern kann. Ein schneller Einstieg ist auf Grund von Registrierungen nicht möglich. Zudem gibt es unterschiedliche Lizenzmodelle, die zwischen Webseite und mobile App unterscheiden, was zur Folge hat, dass man für eine neue Plattform einen neuen Key benötigt. [26] [30]

#### 3. Gute Dokumentation mit Codebeispielen

Bing Maps bietet eine ausführliche Dokumentation für die Bing Maps AJAX Control Version 7.0. Dabei werden die Klassen beschrieben und fast jede mit einem Beispiel verdeutlicht. Zusätzlich gibt es über 200 Codebeispiele, die man direkt im Browser ausprobieren und editieren kann. Diese Beispiele kann man auch ohne Account (Key) nutzen.

### **funktionale Anforderungen**

#### 1. JavaScript API

Bing Maps bietet eine JavaScript API, die „Bing Maps AJAX Control Version 7.0“. Es ist möglich diese einfach in einem script-Tag in eine Webseite einzubinden (Listing Zeile 1). Danach kann man eine Karte erstellen, indem man ein neues Map Objekt erzeugt. (Listing Zeile 6)

```
1 <script type="text/javascript" src="http://ecn.dev.virtualearth.  
    net/mapcontrol/mapcontrol.ashx?v=7.0"></script>  
2     <script type="text/javascript">  
3         var map = null;  
4         function getMap()  
5         {  
6             map = new Microsoft.Maps.Map(document.getElementById(''  
                myMap'), {credentials: 'Your Bing Maps Key'});  
7         }  
8     </script>
```

[31, Codebeispiel CreateMap1]

### 2. Unterstütze Browser

Die JavaScript API von Bing Maps wird laut Microsoft von fast allen Browsern unterstützt.

Die unterstützen Desktop Browser sind:

„

- Internet Explorer 7.0 and later
- Firefox 3.6 and later
- Safari 5 and later
- Google Chrome

Die unterstützen mobilen Browser sind:

- Internet Explorer Mobile Browser
- Apple iPhone Browser
- Google Android Browser
- Research in Motion (RIM) BlackBerry Browser

“[32]

Alle relevanten Browser für Smartphones werden unterstützt.

### 3. Eigenen Standort anzeigen

Der eigene Standort kann bei Bing Maps mit einem „GeoLocationProvider“ ermittelt werden. Dieser erhält eine Referenz auf das Kartenobjekt, in dem der Standort angezeigt werden soll (Listing Zeile 2). Mit der Funktion „getCurrentPosition“ kann dann der Standort des Nutzers ermittelt werden.

```
1 map.entities.clear();
2 var geoLocationProvider = new Microsoft.Maps.GeoLocationProvider(
    map);
3 geoLocationProvider.getCurrentPosition();
```

[31, Codebeispiel GetUserLocation1]

### 4. Markierungen auf der Karte setzen

Mit Bing Maps ist es möglich Markierungen auf der Karte zu platzieren. Dieser werden hierbei PushPins genannt. Diese Pushpins werden zum Anzeigen dem Map Objekt übergeben (Listing Zeile 3). Eine nachträgliche Positionsänderung der gesetzten Pushpins ist einfach möglich, indem man diesem eine neue Position zuweist (Listing Zeile 4).

```
1 map.entities.clear();
2 var pushpin= new Microsoft.Maps.Pushpin(map.getCenter(), null);
3 map.entities.push(pushpin);
4 pushpin.setLocation(new Microsoft.Maps.Location(47.5, -122.33));
```

[31, Pushpins7]

### 5. Markierungen bündeln (optional)

Microsoft bietet keine Funktion mit der man Markierungen(Pushpins) zusammenfassen kann. Um diese Funktionalität dennoch mit Bing Maps nutzen zu können, kann man auf Code von Drittherstellen zurückgreifen. Dieser Code kann allerdings in einer neuen Version von Bing Maps unbrauchbar sein und eine Weiterentwicklung ist nicht sehr wahrscheinlich, da hinter den Drittherstellen meist nur eine Person steht. [26, S. 92]

## **Open Street Maps**

Open Street Map ist die sogenannte „offenste aller Karten“ [26, S.92]. Das bedeutet, dass das Kartenmaterial von der OpenStreetMap Foundation jedem frei und kostenlos zu Verfügung gestellt wird. Das Kartenmaterial selbst kann von jedem Nutzer überarbeitet

werden oder es kann als Basis für neues Kartenmaterial dienen (Points of Interest). So steht es auch jedem frei Fehler zu finden und verbessertes Kartenmaterial einzureichen.

Mit Open Streetmap ist es sogar möglich seine eigenen Karten zu verwenden.

Da Open Streetmap nur Karten liefert und keine dazugehörigen API's ist man auf eine API von einer anderen Quelle angewiesen. Auf der Webseite [www.openstreetmap.org](http://www.openstreetmap.org) wird als JavaScript API LeafletJS verwendet. In der weiteren Betrachtung ist Open Streetmap immer unter der Verwendung von LeafletJS zu sehen.

### **Nichtfunktionale Anforderungen**

#### 1. Kostenlose Abfragen

Open Steet Map bietet jedem privaten oder Kommerziellen Nutzer die Möglichkeit auf das Kartenmaterial zuzugreifen. Abfragen sind dabei generell kostenlos. Des Weiteren ist die Anzahl der Abfragen weder für einen Tag, noch für ein Jahr begrenzt. [26]

#### 2. Ohne Account nutzbar

Für die Nutzung von Open Street Map ist weder eine Registrierung, noch ein Account erforderlich. Nach der Angabe der Ressource im Script Tag (Leaflet API) einer Webseite kann mit JavaScript auf das Kartenmaterial von Open Street Map zugegriffen werden. [33]

#### 3. Gute Dokumentation mit Codebeispielen

Eine gute Dokumentation ist durch die Webseite von Leaflet gegeben. Alle Funktionen werden dort übersichtlich beschrieben, sowie anhand von Beispielen erläutert.

Ein großer Vorteil der Dokumentation von Leaflet sind die Tutorials. Diese beschränken sich auf die wesentlichen Funktionen, wie beispielsweise die Nutzung von Leaflet unter mobilen Geräten. [33]

Bei schwierigen Aufgaben ist es durch Wiki-Beiträge nahezu immer diese zu lösen. [26]

### **funktionale Anforderungen**

#### 1. JavaScript API

Die OpenStreetMap Foundation selbst bietet nur Kartenmaterial an. Eine JavaScript API von der OpenStreetMap Foundation ist nicht vorhanden. Allerdings gibt es Bibliotheken, die von anderen Projekten veröffentlicht wurden, die diese Funk-

tionalitäten abbilden. Die bekannteste dürfte „Leaflet “sein. Leaflet wird nämlich auch auf der Webseite von Open Steet Map, „[www.openstreetmap.org](http://www.openstreetmap.org)“, genutzt.

Die API von Fremdprojekten haben zum Nachteil, dass nicht mehr alles aus einer Hand kommt. [26]

## 2. Unterstütze Browser

Leaflet unterstützt Browser nahezu alle Browser. Generell werden Browser unterstützt, die unter der HTML-Rendering-Engine WebKit laufen, was schon fast alle Browser abdeckt. Die unterstützten Browser sind im einzelnen:

„On Desktop

Chrome,

Firefox,

Safari 5+,

Opera 12+,

IE 7–11

On Mobile

Safari for iOS 3–7+,

Android browser 2.2+, 3.1+, 4+,

Chrome for Android 4+ and iOS,

Firefox for Android,

Other WebKit browsers (webOS, Blackberry 7+, etc.),

IE10/11 for Win8 devices “[33]

Alle relevanten Browser von Smartphones werden unterstützt.

## 3. Eigenen Standort anzeigen

Der Eigene Standort lässt sich mit Leaflet mit einer einzigen Funktion bestimmen. Vergleiche hierzu Listing Zeile 17.

Um den Standort anzuzeigen ist zunächst ein Kartenobjekt nötig. Dieses wird in Zeile 1 des Listings initialisiert. Um die eigene Position auf der Karte grafisch anzeigen zu können ist die Funktion „onLocationFound“, im Listing Zeile 3 bis 8, nötig. Diese Funktion wird aktiviert, wenn der eigene Standort gefunden wurde.

Dann wird die Genauigkeit des Standorts ermittelt und grafisch über einen circle dargestellt (Zeile 4 und 7). Das Zentrum des Standorts wird durch einen Marker (Zeile 5) dargestellt.

Wenn der Standort nicht bestimmt werden konnte wird dem Nutzer mit der Funktion „onLocationError“ eine Fehlermeldung angezeigt.

```
1 var map = L.map('map');
2
3 function onLocationFound(e) {
4     var radius = e.accuracy / 2;
5     L.marker(e.latlng).addTo(map)
6         .bindPopup("You are within " + radius + " meters from this
7                     point").openPopup();
8     L.circle(e.latlng, radius).addTo(map);
9 }
10 function onLocationError(e) {
11     alert(e.message);
12 }
13
14 map.on('locationfound', onLocationFound);
15 map.on('locationerror', onLocationError);
16
17 map.locate({setView: true, maxZoom: 16});
```

[33]

Dieses Beispiel zeigt gut, wie die Leaflet JavaScript API aufgebaut ist. Hier ist alle Funktion separat und können einzeln genutzt werden. So kann man als Programmierer genau die Funktionalitäten nutzen, die man braucht. In diesem Beispiel ist es auch Möglich die Karte stets auf den eigenen Standort zu zentrieren und diesen nicht zusätzlich grafisch anzuzeigen.

#### 4. Markierungen auf der Karte setzen

Markierungen können mit der Funktion „marker(latitude, longitude)“ erstellt werden und der Karte hinzugefügt werden. Ein Beispiel dafür ist schon im Listing der Funktionalen Anforderung 3 in Zeile 5 bis 6 enthalten.

#### 5. Markierungen bündeln (optional)

Mit LeafletJS lassen sich Marker nicht bündeln. Diese Funktionalität kann durch



das einbinden eines weiteren Projekts, „Leaflet.markercluster“ ergänzt werden. [26, S.92]

Das Markercluster kann eine Sammlung von Markern auf der Karte gebündelt anzeigen. Die Marker können mit „.addLayer()“ zum Markercluster hinzugefügt werden. Dieses wird wiederum als Layer zur Karte hinzugefügt. (Listing Zeile 6)

```
1 var markers = new L.MarkerClusterGroup();
2
3 markers.addLayer(L.marker([175.3107, -37.7784]));
4 // add more markers here...
5
6 map.addLayer(markers);
```

[33]

### **Fazit Kartenmaterial**

Um ein Fazit zu ziehen, welches Kartenmaterial zum Einsatz in der Location-based Services App kommt, wird hierbei zuerst auf die Vor und Nachteile der einzelnen Anbieter eingegangen.

#### **Google**

Vorteile:

- Alles kommt aus einer Hand
- Keine Registrierung nötig
- Gute und ausführliche Dokumentation mit vielen Beispielen

Nachteile:

- Lizenzkosten fallen an, wenn 25000 Abfragen pro Tag häufig überschritten werden
- Der Internetexplorer für Windows Phone wird offiziell nicht unterstützt

#### **Bing**

Vorteile:

- Gute und ausführliche Dokumentation mit vielen Beispielen
- Alle relevanten Browser werden unterstützt

Nachteile:

- Cluster nur mit externer Bibliothek nutzbar

- Registrierung ist erforderlich
- Unterschiedliche Lizenzmodelle für Web und mobile Anwendungen
- Anzahl der kostenlosen Abfragen ist sehr begrenzt

### **Open Street Map**

Vorteile:

- Unbegrenzt viele kostenlose Abfragen
- Individualisierbarkeit(eigenes Kartenmaterial)
- Alle relevanten Browser werden unterstützt
- Gute und ausführliche Dokumentation mit vielen Beispielen
- Verständliche API Struktur

Nachteile:

- Man ist auf mindestens 3 Hersteller bzw. Quellen angewiesen

#### Bewertung der nichtfunktionale Anforderungen:

1. Kostenlose Abfragen:  
OpenStreetMap bietet unlimitiert viele kostenlose Abfragen und kann damit überzeugen.
2. Ohne Account nutzbar:  
OpenStreetMap und Google Maps können ohne Account genutzt werden.
3. Gute Dokumentation mit Codebeispielen:  
Alle drei Anbieter bieten ausführliche Dokumentationen mit Tutorials an.

Bei den nichtfunktionalen Anforderungen schneidet Open Steet Maps am besten ab.

#### Bewertung der funktionalen Anforderungen:

1. JavaScript API:  
Alle Anbieter bieten eine JavaScript API und überzeugen damit. Die Kapselung der Funktionen von OpenStreetMap scheint aber für die Autoren am sinnvollsten gewählt zu sein.
2. Unterstützte Browser:  
Google Maps unterstützt den Internet Explorer für Windows Phone Geräte offiziell nicht. Bing Maps und OpenStreetMap können mit den unterstützten Browsern überzeugen.

## 3. Eigenen Standort anzeigen:

Alle drei Anbieter überzeugen hierbei gleichermaßen.

## 4. Markierungen auf der Karte setzen:

Auch hierbei überzeugen alle drei Anbieter gleichermaßen.

## 5. Markierungen bündeln:

Google Maps liefert alles aus einer Hand sowohl bei Bing Maps, als auch bei OpenStreetMap muss auf Bibliotheken dritter zurückgegriffen werden.

Durch die fehlende offizielle Unterstützung von Google Maps für den Windows Phone Browser ist es nicht für diese Studienarbeit geeignet, die Möglichkeit zu haben eine plattformübergreifende mobile LBS App zu entwickeln.

Im Vergleich der Vor und Nachteile und den nichtfunktionalen Anforderungen überzeugt OpenStreetMaps bei den gewählten Anforderungen deutlich mehr als Bing Maps.

Das Fazit ist, dass sich OpenStreetMaps am besten für den Einsatzzweck dieser Studienarbeit und der damit verbundenen Location-based Services App eignet.

#### 4.3.7. iBeacons

*Verfasst von Victor Schwartz*

Umsetzung der Anforderung X.X „Erkennen des Ziels mit einer Abweichung von einem Meter“.

Dieser Abschnitt erläutert die Umsetzung der Anforderung X.X. Wie bereits erläutert wird GPS zur Abstandsermittlung des Prototypen verwendet. Das bedeutet, während der Nutzer sich auf das Ziel zubewegt wird der Standort über GPS bestimmt. Bereits in Kapitel X ist die Genauigkeit von GPS Positionen angegeben. Sie liegt bei x Metern. Um mit dieser Genauigkeit erkennen zu können, ob sich ein Spieler am Ziel befindet wurde ein Radius von 10 Metern um den Zielpunkt gewählt. Nach einigen Praxistests wurde deutlich, dass dieser Radius nicht ausreicht. Die Testkandidaten konnten den Zielpunkt trotz des Radius nicht erreichen. Aus diesem Grund entschied man sich dazu die Genauigkeit am Zielpunkt deutlich zu erhöhen, mit iBeacons. In Kapitel X ist bereits auf RadioButtons eingegangen worden. Eine Unterkategorie bilden von diesen, bilden die iBeacons.

##### iBeacons

Bei iBeacons handelt es sich um eine neue Technologie zur Abstands bzw. Standortbe-

stimmung von Smartphones welche vom Hersteller Apple stammt. Auf Materieller Ebene betrachtet ist ein iBeacon ein kleines Gerät welches ständig Bluetooth Signale sendet. Die verwendete Technik ist Bluetooth Low Energy (BLE). Das Gerät besteht aus

„einer kleinen Platine mit Bluetooth-Sender und einer Knopfzelle. Beides steckt in einem Kunststoffgehäuse, das in der Größe zwischen einem dicken 2-Euro-Stück und einer PC-Maus rangiert.“[34]

Übertragen werden mit dem Signal nur eine eindeutige ID sowie drei Zahlen. Vorteil der BLE Technologie ist der geringe Stromverbrauch. Laut Hersteller soll ein iBeacon zwischen sechs und vierundzwanzig Monaten ein Signal senden können. Abhängig ist dies von der Signalstärke und dem Sendeintervall. Mit dem 2,4 GHz Signal werden Reichweiten von bis zu 50 Metern erreicht. Ein Smartphones welches das Signal empfängt kann einen ungefähren Abstand zum iBeacon ermitteln. Verwendet man mehrere iBeacons, kann über Triangulation ein genauer Standort bestimmt werden. Besonders an dieser Standortbestimmung ist, dass sie auch in Gebäuden funktioniert. Hier ist GPS teilweise gar nicht oder nur schlecht empfangbar. Daraus resultieren diverse Anwendungsbereich im folgenden sind ein paar aufgelistet:

„

- die Navigation und Präsentation von Informationen im Museum
- das Dirigieren von Bahnfahrern zum richtigen Bahnsteig und Wagen
- Rabattprogramme und Kundenkarten
- Abholbenachrichtigungen für vorbestellte Waren beim Betreten des Ladens
- die Automatisierung von Gebäudefunktionen wie Heizung, Licht und Jalousiestellung
- Hinweise auf die Stadion-Einlasskontrolle mit den kürzesten Wartezeiten
- Live-Umfragen unter Teilnehmern einer Vortragsveranstaltung
- kostenlose Lektüre einer Zeitschrift beim Aufenthalt in einem Café
- Bereitstellung der Tageskarte eines Restaurants auf dem Smartphone

“[34]

iBeacons der Firma Estimote

Empfangen werden können die Signale der iBeacons mit Bluetooth 4.0 Empfangsmo-



Abbildung 22: iBeacons der Firma Estimote  
[35]

dulen, welche in den aktuellen Smartphones verbaut sind. Dies funktioniert sowohl mit Apple sowie Android Geräten, was für diese Arbeit von großer Bedeutung ist.

## 5. Implementierung

Wie in Abschnitt 4.2 bereits erwähnt ist die Applikation dem Design-Pattern Model-View-Controller nachempfunden. Im Folgenden wird nun die konkrete Implementierung beschrieben und die Architektur mittels Codebeispielen konkretisiert.

Der allgemeine Ablauf der Anwendung sieht lediglich einen Aufruf der HTML-Seite `index.html` vor. In diese wird mittels Javascript der jeweilige Inhalt (eine View) geladen und mit Daten und Funktionen (Controller) versorgt.

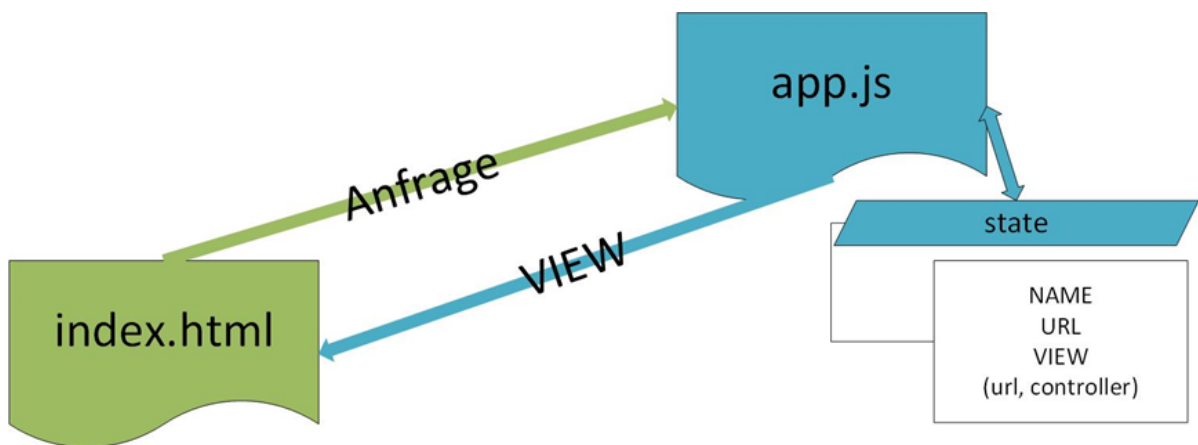


Abbildung 23: Ablauf Applikationsdarstellung

`App.js` ist eine Javascript-Datei, die unter Anderem die Zuordnung der Views zu ihrem jeweiligen Controller durchführt.

### 5.1. App.js

*Verfasst von Melanie Hammerschmidt*

In der `app.js` ist pro View ein Status (`state`) hinterlegt, der alle folgenden Informationen definiert:

- Name des Status
- URL (innerhalb der App)
- View-Bezeichnung (URL der View/Vorlage, Controllernamen)

Konfiguriert wird ein solcher Status beispielsweise wie folgt:

```
1 .state('tab.play', {
```

```
2     url: '/play',
3     views: {
4         'tab-play': {
5             templateUrl: 'templates/tab-play.html',
6             controller: 'PlayCtrl'
7         }
8     }
9 }
```

Die Informationen über die aktuell gewählte View gibt `app.js` an `index.html` zurück, welche sie nach außen dem Nutzer darstellt.

`App.js` wird bei jedem Start der Applikation zuerst geladen. Dabei findet auch immer eine Prüfung der aktuellen Internetverbindung statt. Internet wird für die Darstellung des Kartenmaterials benötigt. Die Konsequenz einer fehlenden Internetverbindung ist demnach ein Abbruch der Applikation.

Standardmäßig wird bei korrektem Startverhalten die `play`-View in die `index.html` geladen. Das ist eine ebenfalls in `app.js` definierte Defaulteinstellung.

Diese folgende Codezeile wird demnach dann aufgerufen, wenn der URL-Provider keine zu einem Status passende URL finden kann. Dann wird die URL auf die des `tab.play`-Status gesetzt.

```
1 $urlRouterProvider.otherwise('/tab/play');
```

## 5.2. Controller und Services

*Verfasst von Melanie Hammerschmidt*

Für jede View existiert ein Controller, der dann die alleinige Macht über Model-Aufrufe für die View hat. Der Begriff Model wird im Folgenden durch den Cordova-spezifischen Begriff Service ersetzt. Insgesamt wurden für die Applikation drei Services entwickelt:

- *Player Service*: Zuständig für die Behandlung von Spielerdaten
- *Location Service*: Zuständig für Positionierungsaufgaben und die Handhabung der Lokalisierung
- *Task Service*: Zuständig für Aufgabenerstellung und -management

Alle folgenden Statusbeschreibungen trennen View, Controller und Model bildlich voneinander.

### 5.3. Applikationsaufbau

*Verfasst von Melanie Hammerschmidt*

Die App besitzt einen getabhten Aufbau, das heißt es existiert eine Kopfzeile, die dem Nutzer vier verschiedene Tabs anbietet: Spielen, Highscores, Account und Credits.

Die angezeigten Tabs der Applikation sind ein Spezialstatus, der in `app.js` als abstrakt gekennzeichnet ist und ständig im Hintergrund aktiv ist. Die Tabs sind auch wieder als HTML-Template definiert.

```
1 .state('tab', {  
2   url: "/tab",  
3   abstract: true,  
4   templateUrl: "templates/tabs.html"  
5 })
```

Alle folgenden Tabs bauen auf diesem abstrakten Status auf.

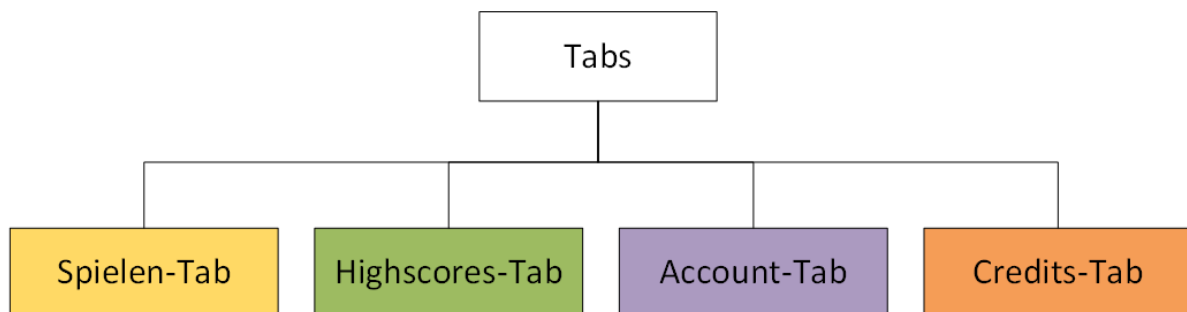


Abbildung 24: Tabaufbau

Jeder der Tabs wird im Folgenden näher beschrieben werden.



### 5.3.1. Spielen-Tab

*Verfasst von Melanie Hammerschmidt*

Der Spielen-Tab bietet dem Nutzer die Möglichkeit ein Spiel zu starten. Er gibt dafür einen Namen und einen Spielradius an. Unter dieser Kombination wird er später seinen Highscore dieses Spiels finden können.

Die View, die dahinter steht ist anfangs die Play-View, welche prinzipiell im Spielen-Tab als Startseite der Applikation angezeigt wird. Im Verlauf des Spiels wird der Nutzer auf verschiedene View weitergeleitet, die im Folgenden näher beschrieben werden.

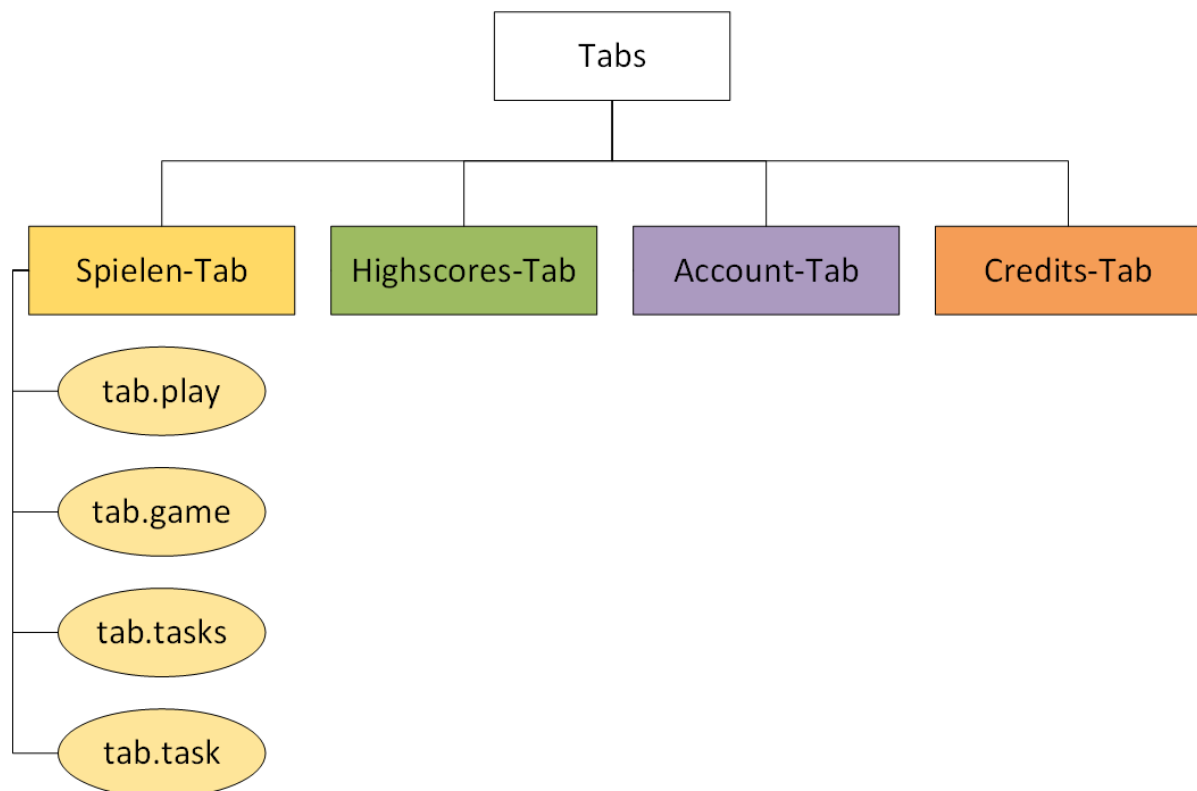


Abbildung 25: Tabaufbau Spielen-Tab

**Play-View** Die Play-View hat ein Textfeld, das den Spielernamen übernehmen kann und eine Selectbox, die den Radius auf zwei bis fünf Kilometern beschränkt. Bei fehlender Eingabe wird auf drei Kilometer als Defaultwert gesetzt. Außerdem bietet die View einen Button, der das Spiel startet.

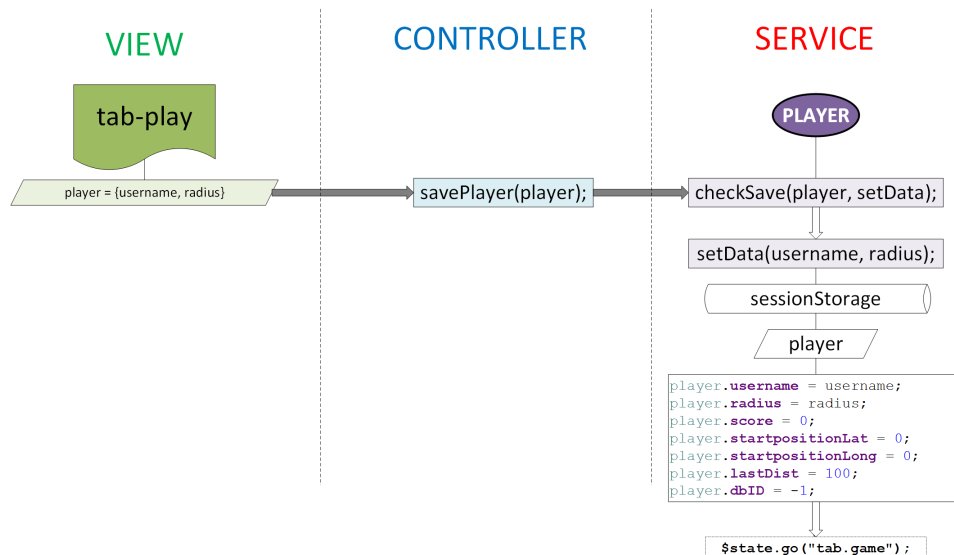


Abbildung 26: Model-View-Controller: Play-View

Der Play-View wurde über `app.js` der Controller 'PlayCtrl' zugewiesen, der wiederum Zugang zum Service 'Player' besitzt. Der *Player Service* kapselt alle Funktionen, die mit den Nutzerdaten im Zusammenhang stehen.

Schon während der Eingabe der benötigten Informationen (Name und Radius) wird ein Javascript-Objekt 'player' generiert, das diese Daten enthält. Mit Druck auf den Button 'Los gehts' wird dieses Objekt an die Funktion 'savePlayer' des Controllers übergeben. Der Controller leitet bei Aufruf dieser Funktion die Objektdaten Name und Radius an den dazugehörigen Service, der zuerst auf gültige Befüllung des Spielernamen prüft und bei Befüllung diesen speichert. Der dafür genutzte Speicher ist der `sessionStorage` des Geräts, der für die Dauer der Session die Daten des Spielers als Objekt mit den folgenden Eigenschaften speichert:

1. Spielername und Radius: beides über Nutzer eingegeben und über Controller an Service zur Speicherung übergeben
2. Score: Punktestand des Spielers
3. StartpositionLat: Breitengrad der Startposition
4. startpositionLong: Längengrad der Startposition

5. lastDist: letzte Distanz zum Ziel für Verlaufsvergleich
6. dbID: ID des Spielers im Spielerarray des Sessionstorage
7. lastUpdate: Zeitpunkt der letzten Änderung

Die noch nicht existierenden Werte werden mit Defaultwerten gefüllt.

Wenn der Spielername nicht angegeben wurde erhält der Nutzer eine Aufforderung und das Spielerobjekt wird nicht weitergeleitet/gespeichert. Bei erfolgreichem Speichern hingegen erfolgt über den Service eine Weiterleitung in den Status 'tab-game' bzw. auf die Game-View.

**Game-View** Die Game-View liegt ebenfalls im Tab 'Spielen' und ist für die aktuelle Standortbestimmung - also die Bestimmung des Startpunkts des Nutzers zuständig.

Sie besitzt anfänglich lediglich einen Button, über den man die Positionsbestimmung anstoßen kann. Bei fehlerfreier Ermittlung wird dem Nutzer eine Open Street Map mit seiner Position und dem eingezeichneten Radius angezeigt, sodass er einen Überblick über sein 'Spielfeld' bekommt. Außerdem wird ein weiterleitender Button eingeblendet, über den der Spieler die Spielmöglichkeiten in seinem berechneten Umfeld angeboten erhält. Bei Fehlern bei der Positionsbestimmung erhält der Nutzer anstelle der Karte und des Button schriftliche Rückmeldung über die Art seines Fehlers.

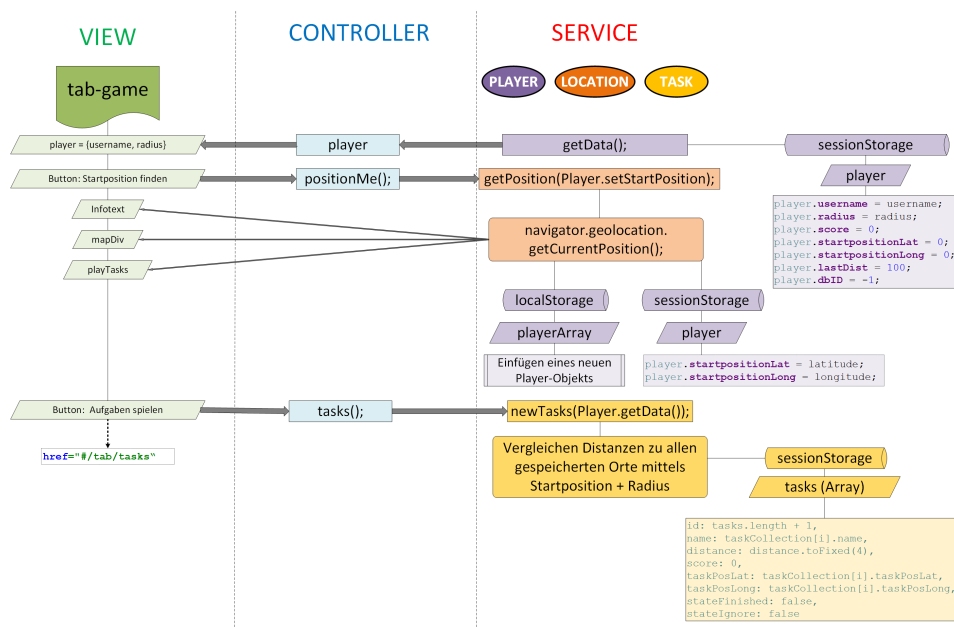


Abbildung 27: Model-View-Controller: Game-View

Die Game-View ist an den Controller 'GameCtrl' gebunden, welcher auf drei verschiedene Services zugreifen kann: *Player*, *Location* und *Task*.

Der *Player Service* kümmert sich, wie erwähnt um die Nutzerdaten. Der *Location Service* wurde entwickelt, um die Positionsbestimmung und die Behandlung der Koordinatendaten zu kapseln. Der *Task Service* hingegen gewährleistet die Generierung, Verfolgung und Speicherung der Aufgaben (Tasks) der ganzen Applikation.

Die Nutzung der Applikation sieht eine Bestimmung der Startposition eines Spielers vor Erstellung der Aufgaben vor. Der Spieler erhält auf der Game-View eine kurze Übersicht seiner bisher gemachten Angaben (Spielernamen und Radius). Diese werden vom Controller vor dem endgültigen Laden der View beim *Player Service* angefragt (`getData()`). Der Service muss dafür auf den Sessionstorage des Geräts zugreifen und das 'player'-Objekt auslesen.

Nach dem Laden der View kann der Nutzer dann über den Button 'Startposition finden' seine aktuelle Position bestimmen lassen. Mit Klick auf den Button wird über den Controller, bzw. über seine Funktion 'positionMe', die Funktion des *Location Service* aufgerufen werden ('getPosition'). Diese Funktion ist nicht nur für die eigentliche Bestimmung der Koordinaten, sondern auch für die Speicherung jener Daten verantwortlich.

Der Befehl '`navigator.geolocation.getCurrentPosition()`' ist der Weg, um auf die Geolokationsfunktionen von Cordova, bzw. des Geräts zuzugreifen. Nach diesem Aufruf werden die Felder der Game-View (Infotext, mapDiv und playTasks) entsprechend gefüllt und bei erfolgreicher Durchführung die Startpositionsdaten des Spielers im Sessionstorage aktualisiert und der Spieler in das Spielerarray des Localstorage integriert. Der Localstorage des Geräts enthält im Vergleich zum Sessionstorage zu jeder Zeit seine Daten und wird nicht automatisch gelöscht. Das Array enthält also alle Spieler, die dann später auch im Highscore aufgelistet werden können.

Wenn es bei der Positionierung zu Problemen kommt, erhält der Nutzer Rückmeldung nachdem die Timeout-Zeit von 30 Sekunden abgelaufen ist.

Bei erfolgreicher Positionierung werden die Koordinaten über das mapDiv, einem Bereich der Game-View, in einer Open Street Map dargestellt (vgl. Kapitel XXX). Außerdem wird in diesem Fall der Button 'Aufgaben spielen' eingeblendet.

Wird dieser Button daraufhin gedrückt, startet der Controller die Erstellung der Aufgaben für den Spieler.

In dieser prototypischen Implementierung ist nur eine fixe Anzahl von Aufgaben imple-

mentiert, die von den Entwicklern frei bestimmt wurden und in einem Array gespeichert sind.

Jede Aufgabe in dieser Sammlung hat folgende Informationen:

1. id: Identifikationsnummer der Aufgabe
2. name: schriftliche Bezeichnung der Aufgabe
3. taskPosLat: Aufgaben-Längengrad
4. taskPosLong: Aufgaben-Breitengrad

Folgende Aufgaben sind im Prototypen implementiert. Es ist zu beachten, dass dies lediglich ein Auszug des Array ist, welches außerdem auch beliebig erweiterbar wäre.

```
1      var taskCollection = [  
2          {  
3              id: 0,  
4              name: "Mannheim Hbf",  
5              taskPosLat: 49.479904895467314,  
6              taskPosLong: 8.470357013095054  
7          },  
8          {  
9              id: 1,  
10             name: "Mannheim Universitaet",  
11             taskPosLat: 49.48373966279436,  
12             taskPosLong: 8.46222996711731  
13         },  
14         {  
15             id: 2,  
16             name: "Mannheim Wasserturm",  
17             taskPosLat: 49.48336089947494,  
18             taskPosLong: 8.477369150878872  
19         },  
20         {  
21             id: 3,  
22             name: "Mannheim Neckar",  
23             taskPosLat: 49.490776620594524,  
24             taskPosLong: 8.482561907531704  
25         },  
26         ...  
27     ]
```

Der Controller ruft die Erstellung der Aufgaben für den Spieler über die Funktion 'new-

Tasks' des *Task Service* auf.

Der Service berechnet daraufhin für jede Aufgabe in seiner Sammlung die Entfernung, die der Spieler zu der jeweiligen Position hätte und speichert sie in ein Spieler-Aufgabenarray, wenn die Distanz kleiner oder gleich dem vom Spieler gewählten Radius ist.

Die Berechnung der Distanz erfolgt über die Funktion 'getDistance', welche im Folgenden aufgeführt ist.

```
1      function getDistance(lat1, lon1, lat2, lon2) {
2          var R = 6371; // Radius der Erde in Kilometern
3          var dLat = deg2rad(lat2 - lat1);
4          var dLon = deg2rad(lon2 - lon1);
5          var a =
6              Math.sin(dLat / 2) * Math.sin(dLat / 2) +
7              Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
8              Math.sin(dLon / 2) * Math.sin(dLon / 2)
9              ;
10         var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
11         var d = R * c;
12         return d;
13     }
```

[?]

```
1      function deg2rad(deg) {
2          return deg * (Math.PI / 180)
3      }
```

[?]

Diese Funktion gibt die Distanz zwischen zwei Positionen in Kilometern zurück.

Jede Aufgabe, die im Spielerradius liegt, wird mit folgenden Informationen im Spieler-Aufgabenarray hinterlegt.

- id: Aufgabenidentifikationsnummer
- name: Name der Aufgabe
- distance: Entfernung von Startposition mit vier Nachkommastellen
- score: Punktestand des Spielers
- taskPosLat: Aufgaben-Breitengrad
- taskPosLong: Aufgaben-Längengrad

- stateFinished: Statusanzeige, ob Aufgabe erledigt wurde
- stateIgnore: Statusanzeige, ob Aufgabe ignoriert werden soll (im Prototypen noch ungenutzt, daher standardmäßig auf false gesetzt)

Am Ende wird das Spieler-Aufgabenarray ebenfalls im Sessionstorage des Geräts gespeichert. Die Applikation ist so gestaltet, dass eine Session einem Spiel mit einer Aufgabenliste entspricht, bei der Punkte gewonnen werden können. Ein Neustart der Applikation bedeutet eine neue Aufgabenliste für den Spieler.

Nach Erstellung der Aufgabenliste für den Nutzer wird dieser auf die Tasks-View weitergeleitet, die diese Liste anzeigen wird.

**Tasks-View** Die Tasks-View besteht selbst lediglich aus einer Liste von Buttons, die die einzelnen Aufgaben repräsentieren und einer Gesamtpunkteanzeige für den Spieler.

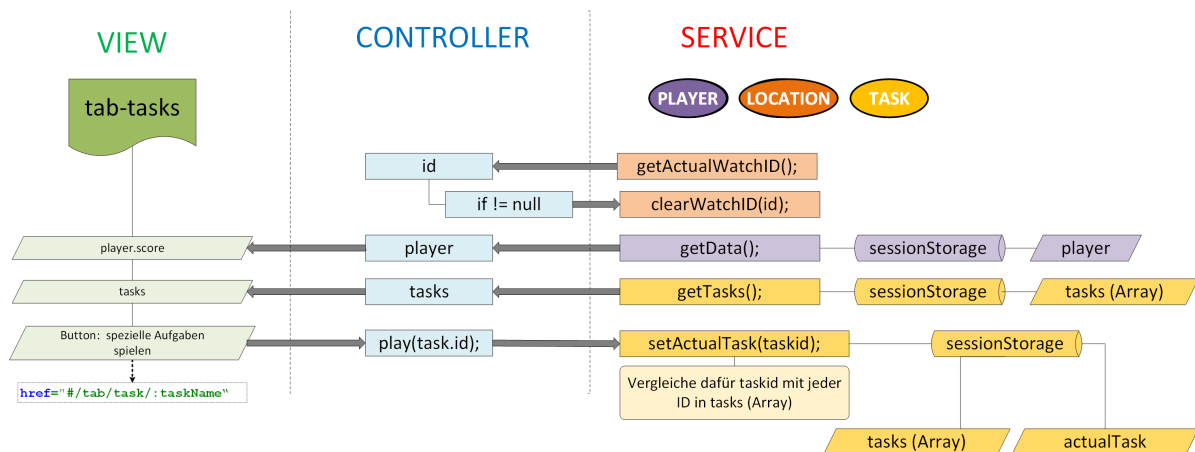


Abbildung 28: Model-View-Controller: Tasks-View

Der Tasks-View wurde über `app.js` der Controller 'TasksCtrl' zugewiesen, der wiederum Zugang zum Service 'Player' und 'Task' besitzt. Der 'Player'-Service kapselt alle Funktionen, die mit den Nutzerdaten im Zusammenhang stehen und 'Task' hat die volle Übersicht über die Aufgaben des Spielers.

Dem Nutzer werden seine aktuell erreichten Punkte angezeigt, die über den *Player Service* vom Controller angefragt werden. Genauso bereitet der Controller die Liste der Aufgaben, die er vom *Task Service* erhält für den Nutzer auf indem er eine Liste von Elementen (Button) anzeigt, die zu der jeweiligen Aufgaben führen sollen. Diese Ermittlungen erfolgen vor dem Laden der View.

Ebenfalls vor dem Anzeigen der HTML-Seite prüft der Controller noch, ob bereits die Position des Geräts geprüft wird. Es könnte ja auch sein, dass der Nutzer die Aufgabe, die er gerade spielt abbricht und auf die Seite mit der übersichtlichen Aufgabenliste zurückkehrt. In diesem Fall wird die sogenannte Watch-ID, die mit der Positionsüberwachung vergeben und im Sessionstorage gespeichert wurde, gelöscht (`clearWatchID(id)`).

Der Nutzer hat dann nach dem Laden der View die Möglichkeit eine Aufgabe auszuwählen, indem er auf einen der Buttons drückt. Damit stößt er im Controller die Funktion 'play' an, der er (über die Auswahl des Button) die entsprechende Aufgaben-ID übergibt. Der Controller startet diese Aufgabe über den *Task Service*, bzw. seine Funktion 'setActualTask(taskID)' während der Nutzer an die Task-View weitergeleitet wird.

**Task-View** Die Task-View ist die Übersichtsseite für eine Aufgabe. Dem Spieler wird die Startentfernung, sein aktueller Punktestand für diese eine Aufgabe genannt. Außerdem bekommt er, wie bei einer Wünschelrute einen Hinweis, ob er sich vom Ziel entfernt oder ob er sich ihm nähert. Dieser Hinweis erscheint nur, wenn er sich auch bewegt, das heißt, wenn sich der Abstand zum Ziel ändert. Ab diesem Moment findet die Positionierung durch die Applikation durchgängig statt.

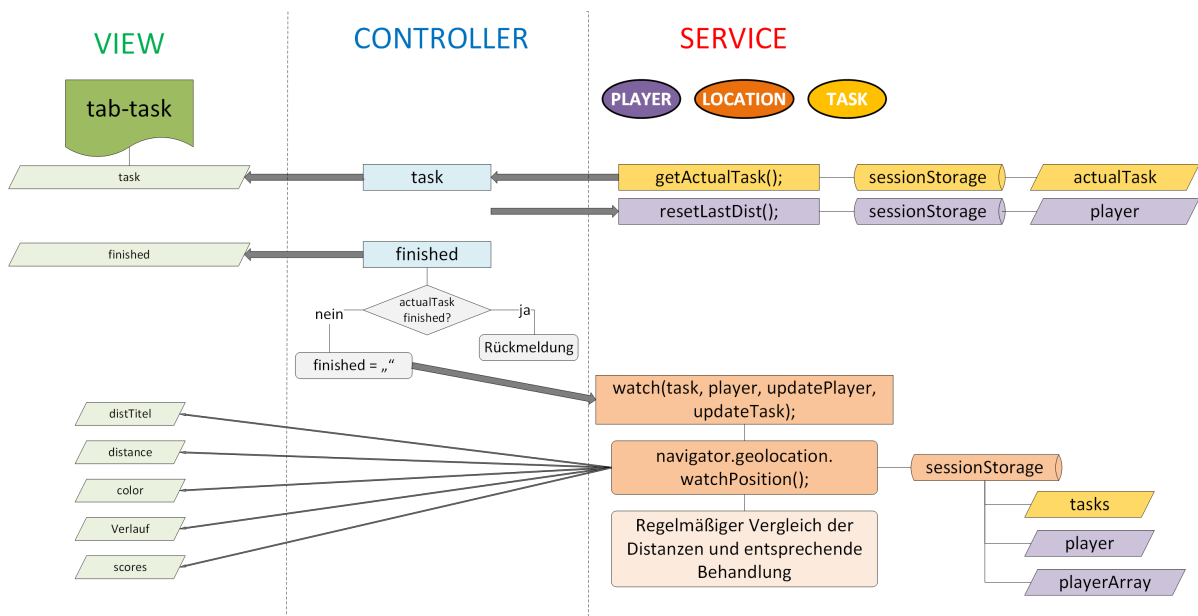


Abbildung 29: Model-View-Controller: Task-View



Der Task-View wurde über `app.js` der Controller `'TaskCtrl'` zugeordnet. Dieser sorgt dafür, dass die Informationen über die Aufgabe angezeigt werden. Dafür hat er Zugriff auf den *Task Service*, bei dem er über die Task-ID alles über die Aufgabe erfahren kann. Die aktuelle Task-ID wurde zuvor vom *Task Service* im Sessionstorage abgelegt und kann jederzeit eingesehen (`'getActualTask'`-Funktion) und bei Bedarf auch wieder gelöscht werden.

Die Ermittlung der aktuellen Aufgabe und ihrer Informationen findet ebenfalls wieder vor dem Laden der View statt. Genauso muss vor dem Anzeigen auch die letzte gemessene Distanz zurückgesetzt werden. Dafür ist der *Player Service* zuständig, da es sich bei der Distanz um einen spielerbezogenen Wert handelt. Dafür ruft der Controller die Funktion `'resetLastDist'` auf, über die der *Player Service* diese Angabe des Spielers auf die Startdistanz der Aufgabe initialisiert.

Wenn der Controller dann feststellt, dass die Aufgabe noch nicht zu Ende gespielt wurde (Prüfung des `stateFinished`), kann das Spiel beginnen und der *Location Service* startet die Funktion `'watch'`. Wenn das Spiel bereits erfolgreich durchlaufen wurde, wird dies dem Nutzer über eine entsprechende Meldung (`finished`) mitgeteilt, ansonsten bleibt die Variable ohne Textinhalt.

Die `'watch'`-Funktion des *Location Service* nutzt wie auch bei der Startpositionsbestimmung den `'navigator.geolocation'`. Dieser besitzt eine Funktion `'watchPosition'`, die eine ID zurückgibt (im Sessionstorage hinterlegt) und solange die Position bei Änderung ermittelt bis die ID gelöscht wird (`clearWatch(ID)`).

Der Ablauf bei erfolgreicher Positionsbestimmung besagt, dass dem Nutzer jeweils seine neue Distanz, die Tendenz über eine farbliche Daumengeste, sowie die neue Punktzahl entsprechend Annäherung/Entfernung angezeigt werden (`distTitel`, `distance`, `color`, `Verlauf`, `scores`). Bei Annäherung bekommt der Spieler einen Punkt dazu bei Entfernung einen Punkt abgezogen.

Wenn die Distanz kleiner als zehn Meter beträgt, gilt die Aufgabe als absolviert. Dafür bekommt der Spieler zehn Punkte und die Watch-ID wird gelöscht. Außerdem werden in diesem Fall die Aufgabe, bzw. der Score für diese Aufgabe und die Gesamtpunkte für den Spieler aktualisiert und gespeichert.

Bei Fehlern bei der Positionsbestimmung erfolgt eine Fehlermeldung nachdem die Timeout-Zeit von 60 Sekunden abgelaufen ist. Die Zeit ist bei der `'watch'`-Funktion doppelt so hoch eingestellt, wie bei der Erstbestimmung. Die Erstbestimmung muss für den Nutzer ersichtlich schnell geschehen. Wenn er erst einmal eine Aufgabe verfolgt, kann es schnell

ler mal passieren, dass das Gerät die Position verliert. Sie sollte jedoch sehr robust gegen 'GPS-Löcher' sein und daher wurde die Zeit in dem Fall sehr hoch gewählt, sodass es zu keinem Aufgabenabbruch kommt. In diesem Fall wird natürlich auch die Watch-ID gelöscht.

### 5.3.2. Highscores-Tab

*Verfasst von Melanie Hammerschmidt*

Der Highscores-Tab besteht im Gegensatz zum Spielen-Tab lediglich aus einer View - der Highscores-View.

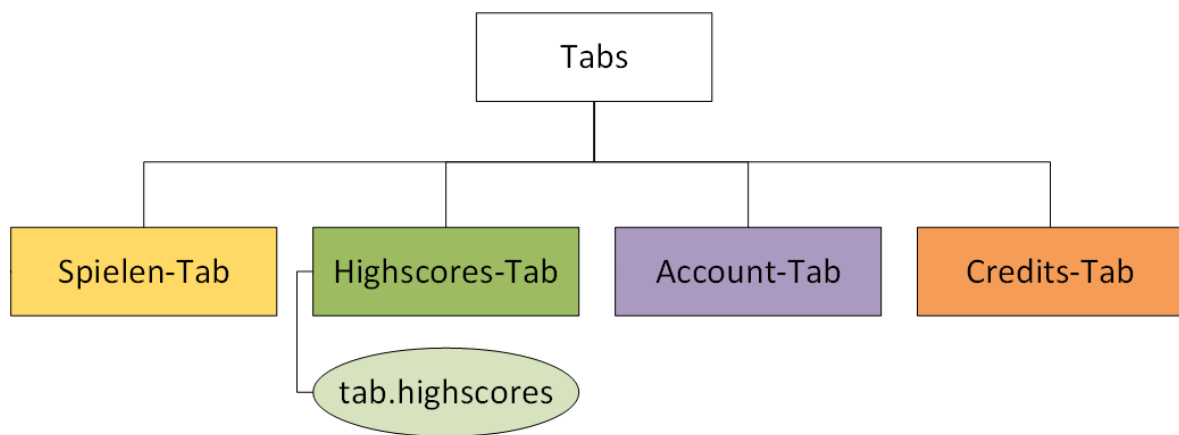


Abbildung 30: Tabaufbau Highscores-Tab

**Highscores-View** Diese View ist die Übersichtsseite für alle gespeicherten Highscores.

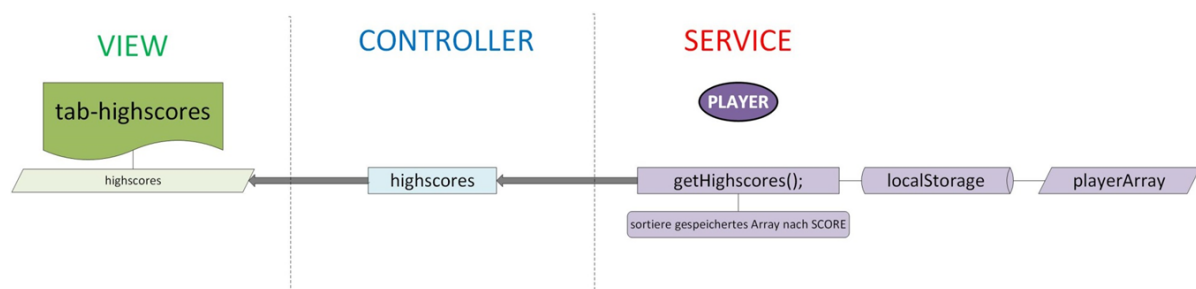


Abbildung 31: Model-View-Controller: Highscores-View

Der Highscores-View wurde über app.js der Controller 'HighscoresCtrl' zugewiesen, der Zugriff auf den *Player Service* hat.

Der Controller ist lediglich für die Anzeige von Highscore-Daten zuständig und besitzt daher keine Funktionen, die über die View aufgerufen werden können. Allerdings lädt er vor der Anzeige der View die Highscore-Daten, die der *Player Service* über die Funktion 'getHighscores' anbietet.

Der *Player Service* lädt dafür das Spielerarray, welches im Localstorage des Geräts hinterlegt wurde und sortiert es absteigend. Die Highscore-View zeigt dann lediglich die Spieler des Arrays in dieser Reihenfolge an und gibt die Hintergrundinformationen Spielernamen, Radius, Zeitpunkt der letzten Änderung und Punktzahl mit an.

### 5.3.3. Account-Tab

*Verfasst von Melanie Hammerschmidt*

Der Account-Tab besteht ebenfalls aus einer einzigen View - der Account-View.

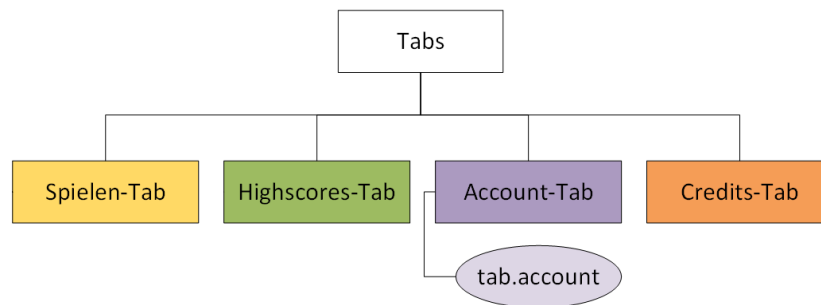


Abbildung 32: Tabaufbau Account-Tab

**Account-View** Die View ist für verschiedene Einstellungen gedacht, die der Spieler der Applikation geben können soll. Im Prototypen wurde an dieser Stelle der Reset-Button zum Zurücksetzen der Highscores implementiert.

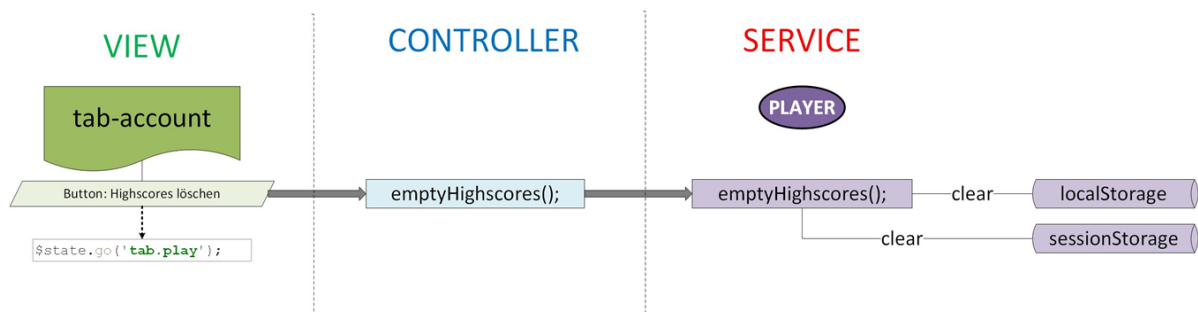


Abbildung 33: Model-View-Controller: Account-View

Der Account-View wurde über `app.js` der Controller `'AccountCtrl'` zugewiesen, der Zugriff auf den *Player Service* hat.

Mit Klick auf den Button 'Highscores löschen' werden über den Controller bzw. über den *Player Service* alle Daten aus dem lokalen und dem Sessionstorage gelöscht (`'emptyHighscores'`-Funktion). Im Anschluss daran wird der Nutzer auf die Startseite (Play-View) zurückgeleitet.

#### 5.3.4. Credits-Tab

*Verfasst von Melanie Hammerschmidt*

Der Credits-Tab besteht ebenfalls aus einer einzigen View - der Credits-View.

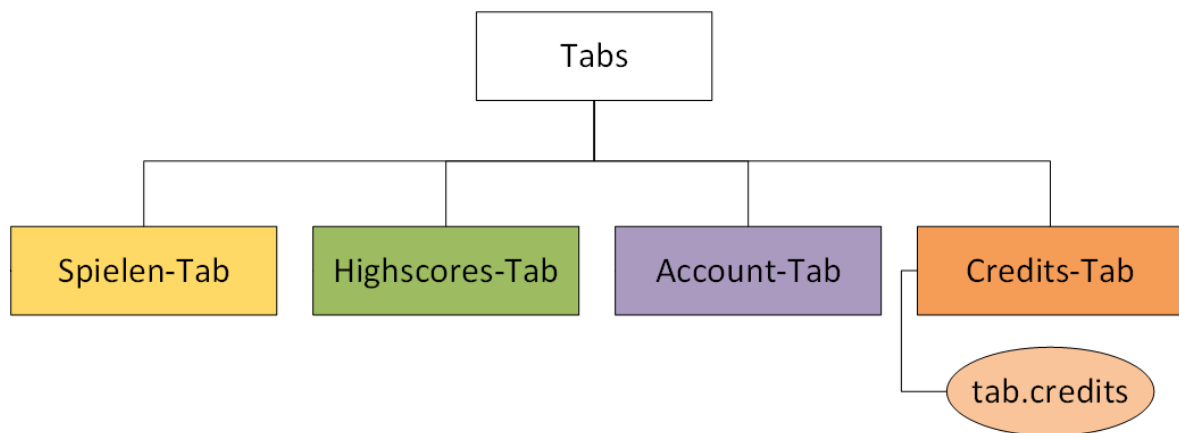


Abbildung 34: Tabaufbau Credits-Tab

**Credits-View** Diese View ist besonders. Ihr wurde zwar ein Controller (`'CreditsCtrl'`) zugewiesen. Da die Danksagungsseite jedoch keinerlei Funktionalität benötigt und auch keine Daten vor dem Laden erhalten muss, kann man sie als einfache HTML-Seite betrachten, die textuellen Inhalt im Original wiedergibt. Es existiert demnach auch kein Ablaufplan.

### 5.3.5. Übersicht

*Verfasst von Melanie Hammerschmidt*

Die drei Services *Player*, *Location* und *Task* werden je nach Aufgabenstellung von den Controllern benötigt. Jeder Funktionsaufruf eines Service braucht dafür zuerst eine feste Verbindung vom Controller zu dem Service. Die Abhängigkeitsstruktur der Applikation sieht so aus:

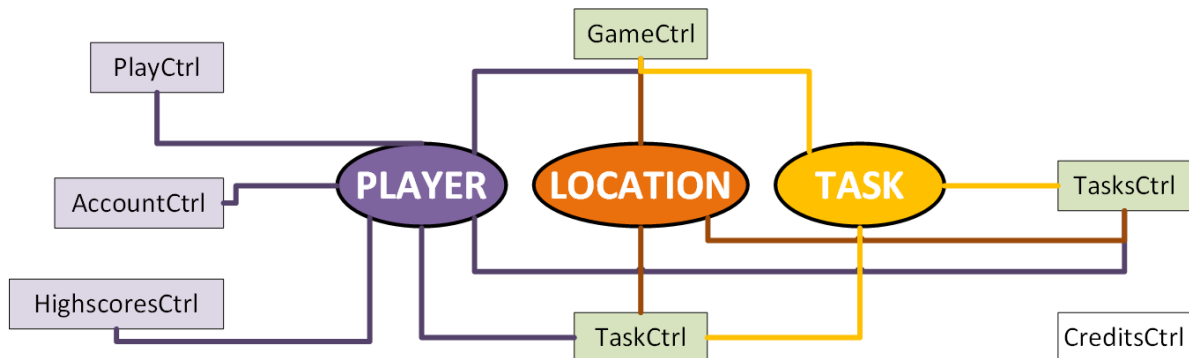


Abbildung 35: Übersicht Controller-Services Verbindungen

Wie in den vorherigen Abschnitten beschrieben existiert ein reger Informationsaustausch zwischen den Views, den Controllern und Services. Dieser ist symptomatisch für die Model-View-Controller-Architektur, wie bereits in Kapitel 4.2 beschrieben und sorgt für größtmögliche Unabhängigkeit zwischen den Elementen, da lediglich die Schnittstellen(Funktionsparameter) von Bedeutung sind und somit auch große Wiederverwertbarkeit gegeben ist.

### 5.3.6. Bewertung der Architektur

*Verfasst von Melanie Hammerschmidt*

**TODO:** Rückbezug der fertigen Architektur zu den gemachten Anforderungen (fachlich + technisch) + Bewertung.

## 5.4. Umsetzung iBeacons

*Verfasst von Victor Schwartz*

Ziel der Umsetzung in dieser Arbeit ist, dass die definierten Zielpunkte der App mit jeweils einem iBeacon ausgestattet werden. Damit diese die Genauigkeit der Entfer-

nung verbessern. Hierfür muss die App Signale des iBeacons empfangen. Deshalb läuft während der gesamten Ortungsphase im Hintergrund ein iBeacon Scanner. Sobald dieser Scanner den gewünschten iBeacon entdeckt hat wird die Distanz zum Ziel durch den iBeacon bestimmt und nicht per GPS, dadurch wird eine höhere Genauigkeit im Zielgebiet erreicht.

Um dies zu implementieren, muss das Zielpunkt Objekt angepasst werden. Standortobjekte benötigen neben Namen und Positionen noch eine iBeacon ID sowie Major und Minor Werte.

Mit den neuen Werten sind alle Voraussetzungen geschaffen um einen iBeacon Scanner zu implementieren. Hierfür wird das Plug-In "Cordova / Phonegap iBeacon plugin" von Peter Metz verwendet. Es ist open Source und kann von Git Hub heruntergeladen werden.

Das Plug-In enthält Schnittstellen und Funktionen zum finden von iBeacons. Hierbei werden zwei unterschiedliche Funktionalitäten geboten.

1. Monitoring von iBeacons Beim Monitoring wird ständig überprüft ob das Smartphone welches das Plug-In nutzt in einen Bereich eintritt, indem ein iBeacon Signale sendet. Ebenfalls wird überprüft ob das Smartphone den Bereich verlässt. Diese Funktion wird in dieser Arbeit genutzt um dem User anzuzeigen, dass bei Eintritt in eine iBeacon Zone nicht mehr GPS genutzt wird, sondern die Abstandsbestimmung mit Hilfe des iBeacons erfolgt.

2. Ranging von iBeacons Während Monitoring nur den Aus- bzw. Eintritt in eine Zone überprüft, ermittelt das iBeacon Ranging alle in der Nähe befindlichen iBeacons und gibt eine ungefähre Distanz zu diesen an. Diese Funktionalität wird genutzt um die Distanz zum Zielort in der App darzustellen.[?]

Im folgenden wird die Umsetzung anhand von Quellcode Beispielen erläutert:

Installation Damit das Plug-In genutzt werden kann, muss es erst über die Console des Betriebssystems installiert werden.

Hierfür navigiert man in der Console zum Hauptordner der App. Dort führt man den Befehl:

```
1 cordova plugin add https://github.com/petermetz/cordova-plugin-ibeacon.git
```

aus. Draufhin erfolgt die Automatische Installation im Ordner "plugin" der App und es kann auf die Schnittstellen zugegriffen werden.

Der allgemeine Ablauf besteht aus mehreren Schritten:

1. Schritt Anlegen eines Arrays mit iBeacon Informationen
2. Schritt Hauptfunktion aufrufen
3. Schritt Warten auf einen iBeacon in Reichweite
4. Schritt Gefundenes iBeacon-Objekt in Variable schreiben
5. Schritt Daten aus Objekt auslesen und anzeigen

Diese Schritte werden im folgenden mit Quellcode-Listings aus der App erläutert:

#### 1. Anlegen eines Arrays mit iBeacon Information

Wie bereits in Kapitel XX erläutert, senden Beacons mehrere Informationen in einem definierten Zeitabstand. Hierzu zählt die UUID, eine eindeutige Zahlen und Buchstabenkombination sowie eine Major und Minor Nummer. Mit Hilfe dieser Daten lässt sich ein Beacon identifizieren. Da jedes Ziel mit einem iBeacon ausgestattet werden soll, muss die App hierfür jeweils diese Daten kennen. Aus diesem Grund gibt es ein Objekt mit diesen Daten.

```
1 var mRegions =  
2     [  
3         {  
4             id: 'region1',  
5             uuid: '95F428B1-4A3A-4E39-B086-21BFF38DEB6D',  
6             major: 0,  
7             minor: 304  
8         }  
9     ];
```

#### 2. Hauptfunktion aufrufen

Damit nach dem iBeacon durch die App gesucht wird, muss 'Ranging' und 'Monitoring' gestartet werden. Dieses überwacht ob sich der iBeacon in Reichweite befindet. Hierfür wird die Funktion 'startMonitoringAndRangingRegion' aufgerufen. Beim Aufruf dieser Funktion wird das Objekt mit den iBeacon Daten 'mRegions' mit übergeben. Mit diesen Daten wird ein neues Objekt vom Typ BeaconRegion erstellt. Dieses Objekt wird von den Funktionen `cordova.plugins.locationManager.startRangingBeaconsInRegion(beaconRegion)`

und `cordova.plugins.locationManager.startMonitoringForRegion(beaconRegion)` verwendet um das Ranging und Monitoring nach diesem iBeacon zu starten.

```
1 function startMonitoringAndRangingRegion(region, errorCallback)
2     {
3         // Create a region object.
4
5         var beaconRegion = new cordova.plugins.locationManager.
            BeaconRegion(
6             region.id,
7             region.uuid,
8             region.major,
9             region.minor);
10
11         // Start ranging.
12         cordova.plugins.locationManager.startRangingBeaconsInRegion(
            beaconRegion)
13             .fail(errorCallback)
14             .done();
15
16         // Start monitoring.
17         cordova.plugins.locationManager.startMonitoringForRegion(
            beaconRegion)
18             .fail(errorCallback)
19             .done();
20     }
```

### 3. Warten auf iBeacon in Reichweite

Die im vorherigen Schritt beschriebene Funktion lässt die App im Hintergrund nach dem iBeacon suchen. Befindet sich der gesuchte iBeacon in Reichweite und das Bluetooth Modul des Smartphones empfängt Signale wird folgende Funktion aufgerufen:

```
1 function onDidRangeBeaconsInRegion(result)
2     {
3         globalBeacons = result.beacons;
4     }
```

Diese Funktion enthält die empfangenen Informationen des iBeacons in der Variable 'result'. Damit die Informationen global verwendet werden können, werden diese in die Variable 'globalBeacons' geschrieben. Der Vorteil ist, im Gesamten Dokument kann nun auf diese Informationen zugegriffen werden. Somit ist es möglich eine Fallunterscheidung



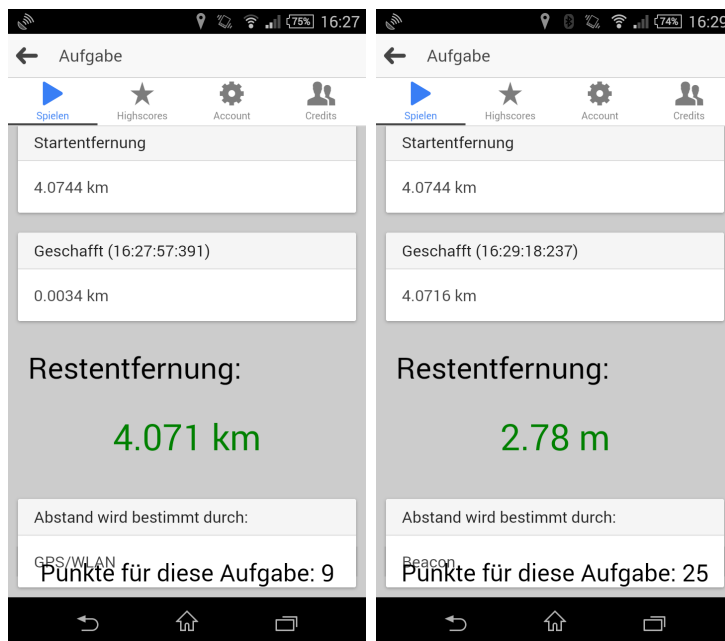
zu treffen ob die Entfernung für das Spiel per GPS/Wlan bestimmt oder über einen iBeacon ermittelt werden kann.

#### 4. Daten aus dem Objekt auslesen

Nachdem die Daten des iBeacons im Objekt `globalBeacons` abgespeichert wurden kann auf die übermittelte Distanz wie folgt zugegriffen werden:

```
1 dist = globalBeacons.accuracy;
```

Mit diesen Funktionen und Daten kann eine Fallunterscheidung erfolgen ob die Distanz per GPS/Wlan oder mit den Daten des iBeacon bereit gestellt wird. Mit einer zusätzlichen Variable 'beaconfound' wird die Fallunterscheidung vereinfacht. Diese Variable ist entweder 'false' oder 'true', je nachdem ob der gesuchte iBeacon in Reichweite ist oder nicht. Die beiden folgenden Screenshots zeigen auf, welche Unterscheidungen und Veränderungen in der App angezeigt werden, wenn GPS/Wlan oder iBeacons verwendet werden.



Auf beiden Bildern (XX und XX) ist die App abgebildet. Sie zeigt jeweils an wie weit das Ziel noch entfernt ist. Im linken Bild wird die Entfernung mit Hilfe von GPS bestimmt. Im rechten Bild findet die Bestimmung der Entfernung per Beacon statt. Per Fallunterscheidung wird das Anzeigefeld angepasst. Damit weiß der Nutzer, mit welchem System der Abstand bestimmt wird. Des Weiteren wird die Entfernung nicht in km angegeben sondern in Metern, da die Entfernungsbestimmung mit einem iBeacon wesentlich genauer ist als mit GPS.

TODO: An dieser Stelle wird mit Code-Listings die Umsetzung dieser Schritte  
detailliert erläutert werden.

---

## **6. Fazit**

### **6.1. Ausblick**

*Verfasst von ??*

## Literatur

- [1] ??, [http://www.bitkom.org/de/presse/8477\\_81896.aspx](http://www.bitkom.org/de/presse/8477_81896.aspx), 2015.
- [2] Kuepper, A., *Location-Based Services - Fundamentals and Operation*, John Wiley and Sons, New York, 2005.
- [3] Schiller, J. und Voisard, A., *Location-Based Services -*, Elsevier, Amsterdam, 1. aufl. edition, 2004.
- [4] Jens Strueker, Stefan Sackmann, D. E. I. P., Location-related services for mobile commerce – an analysis of the potentials for small and medium-sized enterprises, Technischer bericht, Albert-Ludwigs-Universität Freiburg, 2003.
- [5] ??, <http://www.leifiphysik.de/themenbereiche/magnetisches-feld-spule/ausblick>, 2015.
- [6] ??, [http://www.siebert.aero/images/product\\_images/super\\_images/16076\\_Haftfolie\\_137583.jpg](http://www.siebert.aero/images/product_images/super_images/16076_Haftfolie_137583.jpg), 2015.
- [7] ??, <http://de.wikipedia.org/wiki/Postleitzahl>, 2015.
- [8] ??, <http://www.kowoma.de/gps/geo/laengenbreitengrad.htm>, 2015.
- [9] ??, <http://www.kowoma.de/gps/geo/laengenbreitengrad.htm>, 2015.
- [10] GmbH, D. B., [www.itwissen.info](http://www.itwissen.info), 2015.
- [11] ??, [https://www.t-mobile.de/netzausbau/0,25250,15400-\\_,00.html](https://www.t-mobile.de/netzausbau/0,25250,15400-_,00.html), 2015.
- [12] ??, <http://www.vodafone.de/privat/hilfe-support/netzabdeckung.html>, 2015.
- [13] ??, <http://geoinfo.eplus.de/evinternet/>, 2015.
- [14] ??, <http://gpso.de/technik/navstar.html>, 2015.
- [15] Brimicombe, A. und Li, C., *Location-Based Services and Geo-Information Engineering*, John Wiley + Sons Ltd., 2009.
- [16] ??, <https://play.google.com/store/apps/details?id=com.fsp.android.friendlocator&hl=de>, 2015.
- [17] ??, [http://www.focus.de/digital/experten/asfour/ortung-per-smartphone-wie-eltern-stets-wissen-wo-ihre-kinder-gerade-sind\\_id\\_3336081.html](http://www.focus.de/digital/experten/asfour/ortung-per-smartphone-wie-eltern-stets-wissen-wo-ihre-kinder-gerade-sind_id_3336081.html), 2015.
- [18] ??, [http://www.mittelstandswiki.de/wissen/Hyperlokale\\_Werbung](http://www.mittelstandswiki.de/wissen/Hyperlokale_Werbung), 2015.
- [19] ??, <http://www.google.de/adwords/express/>, 2015.
- [20] Baß, C., Einsatzmöglichkeiten und lösungsansätze zur bereitstellung und nutzung von location based services, Technischer bericht.
- [21] ??, <https://www.badgermapping.com/blog/location-based-services-first-for-the-military-now>, 2015.
- [22] ??, <http://www.focus.de/panorama/welt/best-of-playboy/menschen-und->

- storys/tid-31979/high-tech-gegen-wuestenkrieger-exekution-per-joystick-der-drohnenkrieg-ist-laengst-realitaet\_aid\_1022225.html, 2015.
- [23] Goldhammer, P. D. K., Location-based services monitor 2014, Technischer bericht, Goldmedia GmbH Strategy Consulting, 2014.
- [24] ??, <http://www.pcwelt.de/ratgeber/Stau-Warnung-Google-Maps-Tomtom-Verkehrslage-Echtzeitverkehrsinformationen-373385.html>, 2015.
- [25] ??, <http://www.app-entwicklung.info/2014/12/mobile-betriebssysteme-verbreitung-und-marktanteile-mit-stand-dezember-2014/>, 2015.
- [26] Gruber, B., Javascript-apis fuer eigene kartenanwendungen im browser, ix Magazin fuer professionelle Informationstechnik (2015).
- [27] GoogleInc., <https://developers.google.com/maps/>, 2015.
- [28] ??, <http://googlegeodevelopers.blogspot.de/2010/03/introducing-new-google-geocoding-web.html>, 2015.
- [29] ??, <https://googlemaps.github.io/js-marker-clusterer/docs/examples.html>, 2015.
- [30] MicrosoftCorporation, <http://www.microsoft.com/maps/Licensing/licensing.aspx>, 2015.
- [31] MicrosoftCorporation, <https://www.bingmapsportal.com/Isdk/AjaxV7>, 2015.
- [32] ??, <https://msdn.microsoft.com/en-us/library/gg427618.aspx>, 2015.
- [33] ??, <http://leafletjs.com>, 2015.
- [34] ??, <http://www.golem.de/news/bluetooth-low-energy-ibeacon-ist-mehr-als-ein-leuchtfeuer-1403-105331.html>, 2015.
- [35] ??, <http://estimote.com/images/estimote-beacons-group.jpg>, 2015.

## **A. Appendix sections**