HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

IndoorGuide4Android

Bachelor-Thesis

Abteilung Informatik

Hochschule für Technik Rapperswil

Frühjahrssemester 2010

Autoren:	Christoph Egger, Adrian Geiter
Betreuer:	Prof. Stefan F. Keller, HSR, Institut für Software
Projektpartner:	Reto Senn, bitforge AG, Rapperswil
Experte:	Claude Eisenhut, Eisenhut Informatik AG, Burgdorf
Gegenleser:	Prof. Andreas Steffen, HSR

Impressum

Christoph Egger
Informatikstudent

Adrian Geiter
Informatikstudent

Prof. Stefan F. Keller
Institut für Software

HSR Hochschule für Technik Rapperswil
Oberseestrasse 10
Postfach 1475
CH-8640 Rapperswil
Tel. +41 (0)55 222 41 11
Fax +41 (0)55 222 44 00
office@hsr.ch
www.hsr.ch

Projektwebsite: **http://wiki.hsr.ch/StefanKeller/wiki.cgi?IndoorGuide4Android2**

Developer Wiki & Repository: **http://dev.ifs.hsr.ch/indoorguide4android/**

Erklärung der Eigenständigkeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde.
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Ort, Datum: Rapperswil, 17.06.2010

Name, Unterschrift: Adrian Geiter 

Christoph Egger 

Änderungsgeschichte

Datum	Version	Änderung	Autor
19.03.2010	0.0	Erstellung des Dokuments inkl. Inhaltsverzeichnis	A. Geiter
20.03.2010	0.1	Anforderungsspezifikation, Use Cases	C. Egger
13.04.2010	0.2	Projektmanagement	A. Geiter
15.04.2010	0.3	Evaluationen	C. Egger
17.04.2010	0.4	Rahmenbedingungen, Aufbau der Arbeit, Vision & Ziele	C. Egger
19.04.2010	0.5	Stand der Technik	C. Egger
21.04.2010	0.6	Risikoanalyse, Zeitauswertung	A. Geiter
25.04.2010	0.7	Komplette Textüberarbeitung	A. Geiter
05.05.2010	0.8	Erweiterung des Technischen Berichtes	A. Geiter
04.06.2010	0.9	Abstract	A. Geiter
08.06.2010	1.0	Resultate, Zielerreichung, Ausblick	A. Geiter
15.06.2010	1.1	Management Summary	A. Geiter
16.06.2010	1.2	Softwaredokumentation (Analyse, Design)	C. Egger
16.06.2010	1.3	Anleitungen	A. Geiter
17.06.2010	1.4	Implementationdetails	C. Egger
17.06.2010	1.5	Entscheidungen & Knackpunkte	C. Egger
17.06.2010	1.6	Glossary & Quellenverweis	A. Geiter
17.06.2010	1.7	Inhaltsverzeichnis & Bildverzeichnis	A. Geiter
17.06.2010	1.8	Finishing	A. Geiter

Freigabeliste

Datum	Version	Freigabe / Bemerkung	Person
21.04.2010	0.6	Anfrage für erstes Review des Dokuments	S. Keller
26.04.2010	0.7	Zwischenpräsentation mit Gegenleser	A. Steffen
15.06.2010	1.1	Review des Dokuments	S. Keller
17.06.2010	1.4	Review 2 des Dokuments	S. Keller
18.06.2010	1.8	Schlussabgabe	Alle

Abstract

Augmented Reality (AR) steht für eine zukunftsweisende Technik, die in Mobile-Applikationen immer häufiger eingesetzt wird. Dabei wird das Kamerabild mit kontextabhängigen Informationen überlagert. Kombiniert mit der Fähigkeit moderner Handys, sich lokalisieren zu können, entstehen daraus sehr attraktive und praktische Szenarios für Applikationen. Die am häufigsten eingesetzte Technologie für die Lokalisierung – GPS – stösst bei der Ortung innerhalb eines Gebäudes klar an seine Grenzen.

Im Rahmen dieser Bachelor-Thesis wurde eine AR-Applikation für Android Smartphones erstellt, welche auch innerhalb eines Gebäudes verwendet werden kann. Anhand von Signalen, die von Wireless Access Points empfangen werden, kann der aktuelle Standort berechnet werden. Diese Technologie wird durch den IndoorWPS Community Server - einem weiteren Projekt der Forschungsgruppe - zur Verfügung gestellt. Dabei werden zum Schutz der Privatsphäre zu keinem Zeitpunkt Informationen wie die eigene Position dem Server mitgeteilt. Der IndoorGuide4Android kann dadurch Points-of-Interest (POIs) in einem Gebäude (z.B. Museum, Campus oder Einkaufszentrum) anzeigen.

Die durch das Kamerabild gezeigte „Realität" wird mit diversen POI-Informationen und Links zu Multimediainhalten ergänzt. Bei horizontal gehaltenem Handy wird automatisch auf die Kartenansicht gewechselt, auf dem der Grundriss des entsprechenden Gebäudes mit den enthaltenen POIs und des aktuellen Standorts angezeigt wird.

Durch die Verwendung des Open Source Projekts OpenStreetMap können die POIs bequem vom PC aus übers Internet erfasst werden. Diese können dann direkt in den IndoorGuide4Android geladen werden. Um jedoch von der WLAN-Lokalisierung im Gebäude profitieren zu können, ist eine gewisse Vorbereitung und Infrastruktur des Anbieters nötig.

Management Summary

Ausgangslage

Grundidee

Die Grundidee für diese Arbeit ist es, die immer häufiger verwendete Technik „Augmented Reality (AR)" und Benutzung von Karten in den Indoor-Bereich zu bringen. Es existieren noch keine Applikationen mit dieser Funktionalität, welche in einem Gebäude funktionieren würde. Es soll ein IndoorGuide entstehen, der dem Benutzer Informationen über verschiedene Points-of-Interest (POIs) liefert.

Ziele

Ziel dieser Arbeit war es, einen Mobile Guide für die Android-Plattform zu entwickeln, welcher in einem Gebäude eingesetzt werden kann. Die wichtigsten funktionalen Anforderungen werden hier kurz erläutert:

- Die Position des Benutzers soll zuverlässig bestimmt werden.
- Daten sollen im Voraus heruntergeladen werden können.
- POIs sollen im Kamerabild sowie mittels Radar und auf der Karte angezeigt werden.
- Die Kartenansicht soll einen Plan des Gebäudegrundrisses mit dem aktuellen Standort und den POIs zeigen.
- POI-Informationen sollen angezeigt werden können (inkl. Links zu Multimediainhalten).

Bestehende Lösungsansätze

Wikitude World Browser: Dies ist ein mobiler AR-Browser für Android. Er beruht auf Wikipedia- und Qype-Artikeln, sowie Panoramio Photos, welche mit Geodaten verknüpft sind. Damit können weltweit ca. 350'000 Artikel nach Adresse oder GPS-Position durchsucht werden. Diese Applikation funktioniert nur mit GPS- und Internetverbindung.

RMaps: Diese Positionierungssoftware kann auf verschiedenen Karten die Position und POIs anzeigen. Dabei wird die Position anhand von GPS, WLAN oder Zellen des Mobilnetzwerks bestimmt.

IndoorWPS: Das Indoor Wireless Positioning System ist ein Navigationssystem für Innenräume auf der Basis von WLAN WiFi. Mit diesem System wird im IndoorGuide die Lokalisierung gemacht.

Vorgehen

Das Projektmanagement und die Softwareentwicklung wurden agil durchgeführt. Als Vorgehensmodell wurde Scrum eingesetzt. Durch dieses Vorgehen konnten Wünsche des Product Owners auch zu einem späteren Zeitpunkt des Projektes gut eingebracht werden. Nach jedem zweiwöchigen Sprint lag ein funktionsfähiges Release vor.

Risiken

Das Risiko bestand darin, dass Anforderungen nicht umgesetzt werden können, weil die eingesetzten Technologien diese Funktionalität noch nicht besitzen oder die Android-Plattform gewisse Schranken setzt. Diese Risiken waren hauptsächlich durch die Zusammenarbeit mit externen Projekten wie IndoorWPS, OpenStreetMap und OpenStreetMap-in-a-Box verbunden.

Involvierte Personen

Name	Rolle
Herr Stefan Keller (HSR)	Auftraggeber und Betreuer
Herr Reto Senn (bitforge AG)	Externer Projektberater
Herr Michael Klenk (HSR)	Mitarbeiter des IFS, Entwickler der *IndoorWPS*-Library
Herr Christoph Egger	Student, Entwickler des *IndoorGuide4Android*
Herr Adrian Geiter	Student, Entwickler des *IndoorGuide4Android*

Existierende Libraries

IndoorWPS: Die IndoorWPS-Library wurde als externe Komponente in den IndoorGuide eingebaut. Sie dient zur Bestimmung der aktuellen Position und wurde vom IFS der HSR entwickelt.

Ergebnisse

Resultat

In dieser Arbeit ist es gelungen, eine Android-Applikation zu erstellen, welche innerhalb eines Gebäudes POIs anzeigen kann. Die Lokalisierung wird mit IndoorWPS gemacht. Dabei werden die WLAN-Fingerprints und Karten vom IndoorWPS Community Server bezogen. Als POI-Provider wird OpenStreetMap-in-a-Box eingesetzt. Die folgende Abbildung soll die Zusammenhänge und Systemstruktur beschreiben.

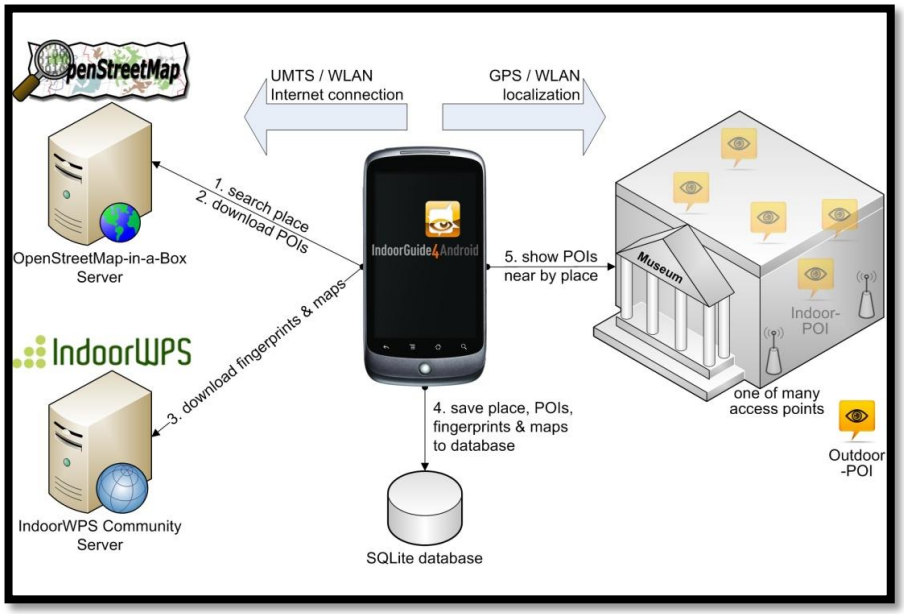


Bild 1: Systemstruktur IndoorGuide4Android

Bewertung

Der entwickelte IndoorGuide ist an verschiedensten Orten sinnvoll einsetzbar. Es ist gelungen, eine Applikation zu erstellen, welche als Guide in einem Gebäude wie beispielsweise einem Museum, Campus oder Einkaufszentrum eingesetzt werden kann. Dies ist momentan noch neuartig und existiert im Rahmen von AR- / Kartenapplikationen noch nicht.

Zielerreichung

Alle geforderten Anforderungen wurden erfüllt. Es gibt jedoch noch Erweiterungsmöglichkeiten, die aus Zeitgründen (noch) nicht realisiert wurden. Zudem können gewisse Teil-Anforderungen nur realisiert werden, wenn beim IndoorWPS gewisse Weiterentwicklungen erfolgen.

Abweichung

Das automatische Auswählen der richtigen Karte konnte nicht realisiert werden, da vom IndoorWPS (noch) keine Information über das Gebäude und Stockwerk, in dem sich der Benutzer befindet, geliefert wird.

Kosten

Da auf Open Source Libraries und Services zugegriffen wurde, entstanden keine Kosten in Form von Geld. Für einen möglichen Anbieter des IndoorGuides entstehen lediglich Kosten für die Lokalisierungsinfrastruktur mit Wireless Access Points. Zudem kostet es ihn natürlich Zeit für das Erfassen von Fingerprints, POIs und Karten.

Ausblick

Gelerntes

Bei der Durchführung dieses Projektes wurde in verschiedenen Bereichen einiges gelernt:

- Neue Technologien erlernt (Android, JSON, WFS)
- Erstellen einer Mobileapplikation
- Einblick in den Geoinformatik-Bereich
- Funktionsweise eines WLAN-Positionierungssystems
- Services wie OpenStreetMap, OpenStreetMap-in-a-Box, IndoorWPS, Geonames.org
- Durchführung eines Projektes mit der agilen Methodik Scrum
- Vertieftes Wissen im Software Engineering Bereich

Verbleibende Probleme

Ein grundsätzliches Problem liegt noch bei der Genauigkeit der Positionsbestimmung über IndoorWPS (oder allgemein WLAN). Zudem ist das Bitmap-Handling bei Android 2.1 eine allgemein bekannte Problematik, welche das Arbeiten mit grossen Bildern (in diesem Fall grossen Karten) schwierig macht. Offen bleibt auch noch das automatische Wählen der richtigen Karte, wenn mehrere möglich wären.

Was würde anders gemacht werden?

Würde eine weitere, ähnliche Arbeit durchgeführt werden, würde von den wichtigsten Funktionen zu einem früheren Zeitpunkt ein Prototyp erstellt werden. Dies wäre nützlich, um den Aufwand und die Machbarkeit dessen besser abschätzen zu können. Ansonsten würde das Projekt ähnlich durchgeführt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

TEIL I – Technischer Bericht

1. Einführung

1.1 Problemstellung

Die heutige Welt ist sehr stark technisiert und sucht immer weiter nach neuen technischen Möglichkeiten, um dem Benutzer noch mehr Komfort zu bieten. Ortsbestimmung und Navigation ist bei Apps für Mobiledevices nicht mehr wegzudenken.

Augmented Reality (AR) steht für eine zukunftsweisende Technik, die immer mehr Einzug in die Welt der Mobiledevices nimmt. Beispiele neuer Mobile-Apps zeigen, dass sich Entwickler immer häufiger dieser neuen Technologie bedienen. Kombiniert mit der Lokalisierung mittels GPS können daraus sehr attraktive und praktische Applikationen erstellt werden.

GPS hat aber auch ganz klar seine Grenzen. Die Ortung bricht innerhalb eines Gebäudes ab. Dass jedoch eine Lokalisierung kombiniert mit AR-Funktionalität, auch innerhalb von einem Gebäude funktionieren kann, soll diese Arbeit aufzeigen.

1.2 Aufgabenstellung[1]

Einführung
"Augmented Reality" (AR) ist eine neue Entwicklung, die auf Mobiles Phones (bzw. Smartphones) besonders zur Geltung kommt. Dabei wird das Kamerabild mit kontextabhängigen Informationen überlagert, beispielsweise mit Points of Interest (POIs), die als dreidimensionale Icons (Stecknadeln) an ihrem Ort schweben. Beispiele dafür sind Wikitude oder Layar. Innerhalb der AR gibt es verschiedenste Ansätze, darunter die Orientierung im Gebäude und in Anlagen ('indoor') - wie z.B. in einem Campus, Einkaufszentrum oder Museum.

Aufgabenstellung
Für diesen Ansatz soll auf Basis eines Android-Smartphones, das u.a. mit Kamera, Wifi, GPS und mit Kompass ausgestattet ist, ein Prototyp neu entworfen und realisiert werden. Es handelt sich dabei um ein mobiles Besucher-Informationssystem (Mobile Guide) innerhalb der HSR und auf dem HSR-Campus als AR-Applikation.
Zwei Anwendungsszenarien stehen im Vordergrund: Einerseits soll sich ein ortsunkundiger Besucher eines Campus' über die Räume in der Nähe informieren können. Er soll sich auch Hypertext und Multimedia-Inhalte zu naheliegenden Objekten anzeigen lassen können, dies mittels AR-Kamerabild und Karte.
Hier der Beginn der zwei Szenarien: "Ein externer Besucher kommt auf dem Campus der HSR an und sucht das Sitzungszimmer 1.227. Er richtet seine Kamera im Hauptgebäude 1 auf den Kiosk, es erscheint eine Infobox mit Öffnungszeiten und den in der Nähe befindlichen Räumen. Wenn er sein Handy waagerecht hält, erscheint ein Gebäudeplan (...)".
Im zweiten Szenario gibt es eine Job-Ausstellung im Hauptgebäude 1 mit Herstellern-Ständen. Der Ausstellungsbesucher richtet sein Handy auf einen Stand und erhält Informationen dazu. Weitere Details sollen innerhalb der Arbeit erarbeitet werden. Wichtiger Bestandteil der Arbeit ist auch der Entwurf von stimmigen Bildschirmdialogen.

[1] Zitat der originalen Aufgabenstellung für die Bachelorarbeit. Autor: Stefan Keller

Folgende Lieferdokumente sollen erstellt werden (deutsch wo nichts angegeben):

- Android-Applikation, User Interface englisch (Bedienungsanleitung ist nicht nötig).
- Vollständiger compilier-bereiter Sourcecode (englisch) sowie als Download gekennzeichnetes Zip-File mit ausführbarem Bytecode.
- Installationsanleitung, falls sinnvoll (englisch).
- Technischer Bericht (deutsch) und SW-Engineering-Dokumentation, z.T. englisch.
- Allfällige Dokumente gemäss Vorgaben der Abt. I. (Plakat, ‚Abstract'), deutsch.
- Dokumentierte Demo im Web (stichwortartig, bebildert, ggf. Video).

Hinweise

- Es wird in diesem Prototyp ein Not-Always-Online-Betrieb angenommen, was die Realisierung eines Cachings bedingt.
- Der Positionierungsdienst für 'indoor' basiert auf IndoorWPS, einem anderen HSR-Forschungsprojekt, das auf Basis von Wifi-Signalen arbeitet. Es ist abzuklären, inwiefern sich ohne grossen Aufwand auch das eingebaute GPS einbinden lässt. Bei der Wifi-Positionierung ist mit ungenauen, "umherspringenden" Koordinaten zu rechnen.
- Die POIs und Gebäudepläne werden vom Benutzer wenn „Online" (ggf. vorgängig) heruntergeladen (z.B. mittels Angabe des Gebäude- oder Ortsnamen-Suche).
- Die Arbeitsweise ist agil, wo sinnvoll mit Unit-Tests. Es wird Wert auf ausgetestete Software und einfache Installation gelegt.
- Für die erfolgreich abgeschlossene Arbeit werden 12 ECTS angerechnet, d.h. es wird eine Arbeitsleistung von mind. 360 Stunden pro Person erwartet.

Randbedingungen, Infrastruktur, Termine und Beurteilung

- Randbedingungen Hardware/OS:
  - Android-Hardware
  - OS: Android/Java
- Software:
  - Java SE gem. Android
  - ANT, Eclipse IDE
- Termine und Beurteilung: gemäss Angaben auf www.i.hsr.ch.

1.3 Rahmenbedingungen

1.3.1 Administratives

Datum	Terminbeschreibung
22.02.2010	Beginn der BA
11.06.2010	Abgabe Abstract & A0-Poster an Betreuer
18.06.2010, 12:00	Abgabe des Berichts an Betreuer
18.06.2010, 12:00	Abgabe des A0-Posters im Studiensekretariat 6.113
25.06.2010	Präsentationen (HSR-Forum)
01.07.2010, 11:00	Mündliche BA-Prüfung

1.3.2 Devices / Hardware

Arbeitsplatzrechner	
Prozessor	Intel Core 2 Duo E6750, 2.66 GHz
Arbeitsspeicher	3.00 GB
Betriebssystem	Windows XP Professional, SP3

HTC Nexus One (Google-Phone)	
Prozessor	Qualcomm Snapdragon QSD8250, 1 GHz
Netzwerk	GSM, UMTS (GPRS Klasse 10, EDGE Klasse 10, HSDPA 7.2 Mbps, HSUPA 2 Mbps)
Bildschirm	3,7 Zoll, AMOLED, kapazitiver Touchscreen (480 x 800 Pixel, 16 Mio. Farben)
Kamera	5 Megapixel (2560 x 1920 Pixel)
Betriebssystem	Android 2.1

Bild 2: Nexus One

Motorola Milestone		
Prozessor	ARM Cortex A8, 550 MHz	 Bild 3: Motorola Milestone
Netzwerk	GSM, UMTS (GPRS Klasse 12, EDGE Klasse 12, HSDPA 10.2 Mbps, HSUPA 5.76 Mbps)	
Bildschirm	3,7 Zoll, TFT, kapazitiver Touchscreen (480 x 854 Pixel, 16 Mio. Farben)	
Kamera	5 Megapixel (2592 x 1944 Pixel)	
Betriebssystem	Android 2.0.1 -> später Update auf Android 2.1	

1.3.3 Entwicklungsumgebung

Software	Zweck	Beschreibung
Eclipse Java EE 3.5 (Galileo)	Entwicklung	IDE für die Entwicklung in Java 1.6. Quelle: www.eclipse.org
Tigris Subclipse 3.0	Repository	Eclipse-Plugin für die Synchronisation mit dem SVN-Repository. Quelle: subclipse.tigris.org
Android SDK 2.1	Entwicklung	Das Software Development Kit zur Android-Programmierung für die Plattform 2.1. Quelle: www.android.com
Android Development Toolkit 0.9	Entwicklung	Wird für die Entwicklung von Android-Apps gebraucht. Bietet Emulatoren (AVD), LogCat, File Explorer, usw. Quelle: www.android.com
JUnit 4.0	Testen	Testframework für TestCases und TestSuites. Quelle: www.junit.org

1.3.4 Organisatorische Software

Software	Zweck	Beschreibung
Microsoft Office 2007, SP2 (inkl. Visio 2007)	Dokumentation	Wird zum Schreiben der Dokumentation verwendet. Quelle: www.microsoft.com
SparxSystems Enterprise Architect 7.0	UML Modellierung	Wird zur UML Modellierung für die Dokumentation verwendet. Quelle: www.sparxsystems.com
Dropbox 0.7	Repository	Zur Synchronisation von Dokumenten. Quelle: www.dropbox.com

ScrumDesk 4.5	Projektmanage-ment	Dient zum Management der Scrum-Abläufe. Planung und Kontrolle der Sprints, Arbeitszu-teilung, ... Quelle: www.scrumdesk.com
Paymo 3.0	Zeiterfassung	Wird für die Zeiterfassung gebraucht. Bietet einen praktischen Desktop-Client, sowie eine gute Auswertungsmöglichkeit. Quelle: www.paymo.biz

1.4 Vorgehen, Aufbau der Arbeit

Zu Beginn der Arbeit waren noch keine Erfahrungen mit Handyprogrammierung vorhanden. Auch das Betriebssystem Android war noch unbekannt. Deshalb wurde beim Projektstart stark aufs Technologiestudium gesetzt.

Vertieft wurde das Thema, indem die vorausgehende Bachelor Thesis zum Thema *IndoorGuide4Android*[2] durchgelesen wurde. Zudem wurde das Buch *Hello, Android*[3] durchgearbeitet, welches grundlegendes Wissen über die Android-Programmierung vermittelte. Dies wurde dann parallel mit praktischen Beispielprogrammen vertieft. Darauf folgte ein vielseitiges Ausprobieren von Funktionen, die später im IndoorGuide eingesetzt werden sollten. Dazu gehörte das Ansteuern der Kamera und das Auslesen des Winkelsensors, das Integrieren des Browsers in eine Applikation, die Verwendung des Kompasses und weitere grundlegende Fertigkeiten.

Die Entscheidungen, wie und wann die geplanten Features eingebaut werden sollten, wurden agil getroffen. So wurde jedes Feature priorisiert und zum gegebenen Zeitpunkt im Sprint eingeplant und implementiert.

Abgeschlossen wurde die Arbeit, indem neben diesem Bericht auch noch Benutzeranleitungen für das Erfassen neuer Points-of-Interest (POIs), wie auch das Bedienen der Android-Applikation erstellt wurden. Ebenfalls wurde eine Wiki-Seite eingerichtet, auf der die Applikation heruntergeladen werden kann und zusätzliche Informationen und Verweise auf verwandte Arbeiten ersichtlich sind.

[2] Vorausgehende Bachelor Thesis: eprints.hsr.ch/81/1/IG4A_Dokumentation.pdf
[3] Hello, Android (Pragmatic Programmers): Introducing Google's Mobile Development Platform, 3rd Edition, Autor: Ed Burnette, ISBN-13: 978-1934356494, Version: 2010-2-25

2. Vision und Ziele

Die Vision dieser Bachelor Thesis ist es, eine AR-Applikation für Android Handys zu erstellen, welche als Guide innerhalb eines Gebäudes (indoor) alle interessanten Objekte (POIs) anzeigt. Die durch das Kamerabild gezeigte Realität soll dazu mit diversen POI-Informationen und Links zu Multimediainhalten überlagert werden.

Die POIs, welche sich in einem gewissen Umkreis befinden, sollen durch Anzeige im Kamerabild und Positionierung in einem Radar den Benutzer unterstützen, sich in der Umgebung zurechtzufinden. Wird das Handy horizontal gehalten, soll das Kamerabild durch eine Karte ersetzt werden, die den Grundriss des entsprechenden Gebäudes mit den enthaltenen POIs und der aktuellen Position anzeigt.

Der momentane Hype um AR ist in der Mobile Community gross. Dies soll ausgenutzt werden, um die bestehenden Applikationen mit einer Indoor-Anwendung zu ergänzen, welche noch einzigartig im AR Bereich ist. Der IndoorGuide4Android hat sein Einsatzgebiet vor allem an Orten, an denen kein GPS-Signal empfangen werden kann. Dies soll anhand von WLAN-Lokalisierung realisiert werden, bei dem anhand von Signalen von mehreren Wireless Access Points der aktuelle Standort berechnet werden kann.

Im Folgenden werden kurz einige realistische Einsatzmöglichkeiten vorgestellt.

2.1 Campus

Da sich das Finden von einem gewissen Hörsaal bei einem grösseren Campus, zum Beispiel einer Fachhochschule oder einer Universität, als schwierig erweisen kann, würde der *IndoorGuide* dem Studierenden unter die Arme greifen. Zudem gäbe es die Möglichkeit, nützliche Informationen, wie beispielsweise der Menüplan der Mensa, zur Verfügung zu stellen.

2.2 Museum

In einem Museum könnten Ausstellungsstücke als POIs markiert werden. So wäre es möglich, anhand der Karte einen Überblick über das Museum zu erlangen. Die Interesse erweckenden Ausstellungsstücke wären mit dem IndoorGuide schnell auffindbar. Den Ausstellungsstücken könnten zusätzliche Audio- oder Textnachrichten angehängt oder sogar Informationen in Form von Bildern oder Filmen zur Verfügung gestellt werden. Dies soll vor allem Personen ansprechen, die nicht gerne an Führungen teilnehmen oder Zusatzinformationen wünschen.

2.3 Messe

An Messen könnte der *IndoorGuide* ähnlich wie bei einem Museum eingesetzt werden. Hier gibt es jedoch noch eine andere interessante Anwendung, die beim Aufbau und der Einrichtung einer Messehalle zu tragen käme. Dies muss immer sehr schnell von statten gehen und viel Material muss von Lieferanten und Arbeitern an den richtigen Ort gebracht werden. Es ist nicht immer ganz einfach, den richtigen Sektor in einer riesigen Messehalle zu finden. Gerade solchen Lieferanten, die sich nicht auskennen und nur eine Sektornummer für die Auslieferung kennen, würden ihr Ziel mit einem IndoorGuide sicherlich schneller finden.

2.4 Firma

Firmengelände können sehr weitläufig und verwinkelt beziehungsweise unlogisch aufgebaut sein. Der *IndoorGuide* könnte auch hier Unterstützung geben, um sich auf einem fremden Firmengelände zurechtzufinden. Es sollte folglich problemlos möglich sein, bei einem Meeting oder Vorstellungsgespräch den richtigen Sitzungsraum zu finden. Eine weitere Einsatzmöglichkeit wäre das publizieren von interessanten Informationen, wenn die Firma einen Tag der offenen Tür veranstaltet.

2.5 Einkaufszentrum

In einem Einkaufszentrum gibt es oft das Problem, die gewünschten Läden zu finden oder überhaupt in Erfahrung zu bringen, welche Läden existieren. Hier könnte der *IndoorGuide* eine Übersicht liefern, mit der man sich schneller zurechtfindet. Ebenfalls könnte man Informationen wie beispielsweise Öffnungszeiten oder Sonderangebote zu den verschiedenen Läden zur Verfügung stellen.

2.6 Bibliothek

In Bibliotheken könnte der *IndoorGuide* eingesetzt werden, um das Finden von Regalen mit bestimmten Genres oder Autoren zu vereinfachen.

So gibt es noch viele weitere Einsatzmöglichkeiten, für den *IndoorGuide*. In dieser Arbeit soll die Grundlage für einen stabilen und brauchbaren *IndoorGuide* gelegt werden, der dann je nach Anwendungsgebiet weiter ausgebaut werden kann.

3. Stand der Technik

3.1 Bestehende Lösungsansätze

Es gibt bereits einige Applikationen, die auf AR setzen. Ebenfalls sind Lösungen vorhanden, die ähnlich wie der IndoorGuide aufgebaut sind und ähnliche Funktionalität bieten, jedoch für Outdoor-Szenarien mit GPS konzipiert sind.

Vier solcher Applikationen werden hier beschrieben und können als Quelle der Inspiration und Ideensuche dienen. Zudem wird als fünfter Lösungsansatz IndoorWPS beschrieben, welcher für die Lokalisierung eingesetzt werden soll.

3.1.1 Layar (Version 3.0.5) [4]

Bild 4: Layar

Der Layar Reality Browser (entwickelt in Holland) zeigt die reale Welt mit digitalen Informationen in Echtzeit an. Diese Technologie wird Augmented Reality genannt. Das durchs Handy gesehene Bild wird basierend zur Position überlagert.

Wie funktioniert Layar's Augmented Reality? Die Idee ist simpel: Layar funktioniert anhand der Kombination von Handykamera, Kompass und GPS Daten. Damit wir die Position und das Sichtfeld des Benutzers bestimmt. Auf diesen geografischen Koordinaten basierend werden Daten empfangen und das Kamerabild überlagert.

Layar bietet noch mehr, es gibt die Möglichkeit anhand des API von Layar einen eigenen Webserver zu erstellen und seine eigenen Daten in die Layar Applikation einzubinden.

3.1.2 Wikitude World Browser 4 (Version 8.08)[5]

Bild 5: Wikitude World Browser

Wikitude World Browser ist ein mobiler Augmented Reality (AR) Browser für die Android Plattform. Der auf Wikipedia- und Qype-Artikeln, sowie Panoramio Photos beruht, welche mit Geodaten verknüpft sind. Somit können weltweit ca. 350000 Artikel nach Adresse oder GPS-Position durchsucht werden. Es werden dabei über 200 Quellen („Worlds") einbezogen, welche Inhalte liefern. Diese Inhalte (User Contents) können dabei von Benutzern über www.wikitude.me hochgeladen werden.

Gefundene Artikel wie zum Beispiel Sehenswürdigkeiten in der Umgebung können auf einer Karten-, Satelliten-, Listen-, aber auch auf einer „Augmented Reality" Kameraansicht dargestellt werden. Dabei wird das Kamerabild mit computergenerierten Daten überlagert.

[4] Quelle Layar: www.layar.com
[5] Quelle Wikitude: www.wikitude.org

3.1.3 RMaps (Version 0.7.6)[6]

RMaps ist eine Positionierungssoftware, die den Benutzer mittels GPS, WLAN oder Zellen des Mobilenetzwerks ortet und Ihre Position dann auf der Karte anzeigt. Karten und POIs können von verschiedensten Providern wie zum Beispiel Google Maps oder OSM bezogen werden. Diese Kartenausschnitte werden jeweils bei Gebrauch online heruntergeladen (d.h. es muss immer eine Verbindungsmöglichkeit mit dem Internet bestehen). RMaps verfügt über eine Funktion, mit der die Karte immer korrekt ausgerichtet wird.

Bild 6: RMaps

Bild 7: Peak.AR

3.1.4 Peak.AR[7]

Mit Peak.AR ist es möglich, Bergspitzen im Kamerabild mit deren Namen und Höhe zu beschriften. Wird das Gerät waagerecht gehalten, öffnet sich die Karte mit dem aktuellen Standort und den umliegenden Bergspitzen, welche noch mit Zusatzinformationen beschriftet sind.

3.1.5 IndoorWPS[8]

Das Indoor Wireless Positioning System (IndoorWPS) ist ein Navigationssystem für Innenräume auf der Basis von WLAN WiFi. Das Projekt umfasst einerseits Applikationen und Libraries zur Positionsberechnung auf verschiedenen Betriebssystemen sowie eine Website mit Webservices zur gemeinsamen Nutzung von WLAN-Fingerprints. Die vom Institut für Software (IFS) zur Verfügung gestellte Website[9] dient dem Austausch von Fingerprints und Gebäudegrundrisskarten.

Im Kapitel „5.1.1 IndoorWPS" im Teil III wird genauer auf die Funktionsweise eingegangen.

Bild 8: IndoorWPS

[6] Quelle RMaps: code.google.com/p/robertprojects
[7] Quelle Peak.AR: peakar.salzburgresearch.at
[8] Quelle IndoorWPS: wiki.hsr.ch/StefanKeller/wiki.cgi?IndoorWPS
[9] IndoorWPS Community Server: labs.geometa.info/indoorwps

3.2 Realisierbarkeit und Herausforderungen

Für alle Komponenten, die im IndoorGuide realisiert werden sollen, gibt es schon einzelne Applikationen (siehe oben). Somit sollten die Ziele des IndoorGuides realisierbar sein. Die Herausforderung wird sein, die einzelnen Komponenten zusammenzuführen und für den Benutzer sinnvoll benutzbar zu machen. Einige der grundlegenden Herausforderungen werden hier kurz beschrieben.

3.2.1 Lokalisierung

Die Lokalisierung anhand von WLAN-Signalen ist noch nicht fertig ausgereift, bringt jedoch ein genügendes Resultat um eine erste Version des IndoorGuides zu erstellen. Die Problematik liegt in der Ungenauigkeit des aktuellen Standorts. Sie variiert je nach Anzahl Fingerprints und Access Points zwischen 4 - 10 Meter (Radius). Ebenfalls hat die Umgebung einen Einfluss auf das Signal und somit auf die Genauigkeit. Ein Problem in diesem Zusammenhang liegt darin, dass sich die errechnete Position ohne tatsächliche Positionsänderung immer wieder ändert („herumspringt").

Durch das Springen der aktuellen Position ist es schwierig, die POIs in der AR-Ansicht genau im Bild zu positionieren, dies würde den Benutzer irritieren. Mit diesem Problem haben Applikationen, die mit dem genaueren GPS-Signal und grösseren Distanzen arbeiten, weniger zu kämpfen. Es muss daher eine Darstellung gewählt werden, die für den Benutzer trotzdem übersichtlich und intuitiv ist. Zudem soll auf der Kartenansicht mit einem Ungenauigkeitsradius um den aktuellen Standort angedeutet werden, dass die Position nicht genau ist.

Eine zusätzliche Problematik bringen POIs, welche nahe beieinander liegen. Falls sie einen zu kleinen Abstand zueinander haben, kann nicht mehr eindeutig bestimmt werden, welcher POI näher ist. Somit muss dies beim Erfassen der POIs berücksichtigt oder beispielsweise mit Bilderkennung optimiert werden. In folgendem Bild soll dies verdeutlicht werden. Das Handy kann nicht eindeutig anzeigen, ob sich der Benutzer nun vor dem Männer- oder Frauen-WC befindet. Doch es wird erkannt, dass die Cafeteria und die Telefonkabine weiter entfernt sind als die WCs.

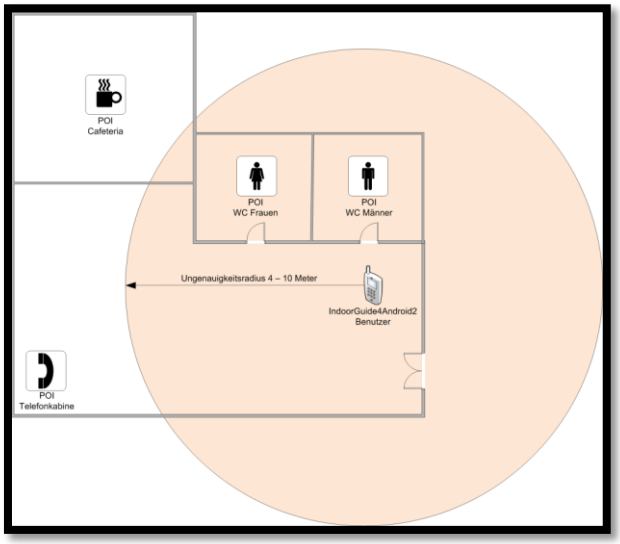


Bild 9: Lokalisierungsproblematik bei nahen POIs

3.2.2 Alternativen zur WLAN-Lokalisierung

Für diese Arbeit ist die externe Komponente IndoorWPS vorgegeben. Jedoch darf hier auch auf andere Ansätze der Lokalisierung oder Erkennung von POIs hingewiesen werden.

RFID: VisiVis[10], ein Projekt der HSR, realisieren eine Lokalisierung anhand von RFID-Tags. Dort ist jedoch der grosse Nachteil, dass diese an den Gegenständen physikalisch angebracht werden müssen. Ebenfalls muss der Benutzer dabei ein RFID-Empfänger mit sich führen, um die nötigen Informationen zu bekommen.

Computer-Vision[11]: Microsoft hat den Weg der Computer-Vision Algorithmen eingeschlagen, bei der aufgenommene Bilder (eines Films oder der Kamera) real time analysiert werden. Sie werden mit einer Datenbank abgeglichen, in der vorgängig analysierte Bilder gespeichert wurden. Gibt es Übereinstimmungen, werden diese dann ins AR-Bild eingeblendet.

QR-Code: Eine andere Möglichkeit der Bilderkennung besteht darin, QR-Codes im Bild zu erkennen und damit Informationen über ein Objekt abzurufen. Die Schwierigkeit besteht dort jedoch noch darin, diese QR-Codes für den Bilderkennungsalgorithmus so schnell erkennbar zu machen, dass dem Benutzer echtes AR-Gefühl vermittelt wird. Ebenfalls müssten diese eine gewisse Grösse haben, damit sie auch auf grössere Distanz erkannt werden könnten.

3.2.3 Datenquelle

Eine weitere Herausforderung ist die Datenquelle. Die Daten der Fingerprints, POIs, Maps und Places müssen auf einem Server zur Verfügung gestellt werden können. Dieser Server soll zum einen die Anforderungen an die Daten erfüllen, zum anderen soll durch diese Arbeit kein eigener Server in Betrieb genommen, sondern eine vorhandene Infrastruktur genutzt werden.

Fingerprints und Maps: Für die Fingerprints und Maps ist die Datenquelle vorgegeben. Es soll der IndoorWPS Community Server benutzt werden, der alle nötigen Schnittstellen zur Verfügung stellt. Diese Infrastruktur besteht schon, muss unter Umständen jedoch erweitert werden. Eine Erweiterung sollte möglich sein, da die betreffenden Ansprechpersonen des IndoorWPS-Projekts an der HSR sind.

Places: Um die Daten eines Ortes herunterladen zu können, muss es eine Möglichkeit geben, um sogenannte Places zu suchen (z.B. Hochschule Rapperswil, Uni Zürich usw.). Für diese Suche wird ein Dienst benötigt, welcher einen Suchbegriff nach einem Ort oder POI entgegen nimmt und die Koordinate dessen zurückgibt. Für diesen Dienst gibt es verschiedene Anbieter. GeoNames.org, OpenStreetMap oder OpenStreetMap-in-a-Box wären mögliche Anbieter eines solchen Dienstes. Die Herausforderung besteht darin, den richtigen auszuwählen, wobei die Unterschiede nur klein sind.

POIs: Points-of-Interest sind die wichtigsten Daten in dieser Applikation. Dabei sollen diese speziell für den Gebrauch im Indoor-Bereich ausgestattet werden. Zudem geht aus der Aufgabenstellung hervor, dass sie Links zu Multimediainhalten speichern sollen. Es soll ein passender Dienst gefunden werden, welcher diese Ansprüche abdeckt. Beispielsweise wäre OpenStreet-

[10] Quelle VisiVis: www.ifs.hsr.ch/VisiVis.6057.0.html
[11] Quelle Computer-Vision: www.technologyreview.com/computing/22218/page1

Map ein solcher Dienst, welcher auch das Speichern von selbst definierten Attributen erlaubt. Da dort noch (fast) kein Gebrauch von Indoor-POIs gemacht wird, müssten diese neu modelliert werden. Da trotz wenigen Einschränkungen bei der Attributdefinierung ein gewisser „Standard" eingehalten werden soll, wird diese Aufgabe einige wichtige Entscheide beinhalten. Ein anderer möglicher Dienst wäre OpenStreetMap-in-a-Box. Bei diesem ist die Attributwahl zwar eingeschränkter, jedoch gäbe es eine gewisse Mitsprachemöglichkeit, da dieses Projekt in einer anderen Bachelorarbeit bearbeitet wird.

3.2.4 Kartenauswahl

Die Auswahl der korrekten Karte stellt eine Schwierigkeit dar, da es an einer Koordinate mehrere mögliche Karten geben kann. Dies kommt daher, dass in einen Gebäude meist mehrere Stockwerke existieren, die alle ihre eigene Karte haben. Dies könnte man einfach lösen, wenn bekannt wäre, in welchem Stockwerk sich der Benutzer befindet.

Ein anderer Problempunkt stellt die Überlappung der Karten dar. Durch das Drehen beim georeferenzieren von Karten entstehen meistens leere Flächen ausserhalb des eigentlichen Gebäudeplans (siehe Bild). Hauptsächlich durch diese Flächen entstehen Überlagerungen mehrerer Karten. Es kann jedoch rein vom Kartenbild her nicht entschieden werden, ob der Benutzer nun innerhalb des gezeichneten Gebäudes steht oder in einer dieser leeren Flächen. Sind nun mehrere Karten überlagert, ist es schwierig, die richtige davon auszuwählen. Diese Problematik soll auf eine intelligente Art und Weise gelöst werden, so dass dem Benutzer trotzdem die richtige Karte angezeigt werden kann.

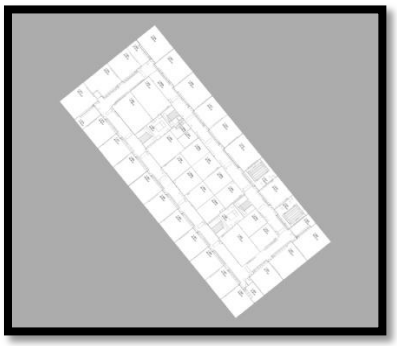


Bild 10: georeferenzierte Karte

4. Evaluation

Um die Bachelor-Thesis zu realisieren, gibt es verschiedene Lösungsmöglichkeiten zu diversen Komponenten der Applikation. In diesem Kapitel soll anhand von verschiedenen Bewertungsmatrizen die am besten geeignete Lösung evaluiert werden.

Den Lösungsansätzen wird anhand eines Kriterienkatalogs zu jedem Kriterium ein Gewicht zugeordnet, welches die Relevanz darstellt. Die Gewichte variieren im Bereich von 1-3 wobei 1 eher vernachlässigbar ist und 3 ein sehr wichtiges Kriterium darstellt.

Ein Lösungsansatz kann folgende Punkte erreichen: 0 = nicht erfüllt, 1 = mittelmässig erfüllt, 2 = gut erfüllt. Die mit dem Gewicht multiplizierte Punktzahl ergibt kumuliert mit den anderen Punkten die Gesamtpunktzahl. Der Lösungsansatz mit der höchsten Punktzahl wird dann in der Bachelor Thesis umgesetzt.

4.1 Webservice Responsetype

Es gilt zu evaluieren, welches Datenformat für die HTTP-Response bei einer Webserviceanfrage geeignet ist. Zwei Varianten wurden eruiert, **XML** und **JSON**.

4.1.1 Kriterienkatalog und Gewichtung

Kriterium (Beschreibung)	Gewicht (1-3)
Overhead: Um den teuren Datentransfer bei Mobilegeräten zu minimieren, soll der Response-Overhead so klein wie möglich gehalten werden.	3
Komplexität: Es sollte nicht unnötig viel Zeit investiert werden müssen, die Technologie zu erlernen. Kein zu grosser Implementierungsaufwand für gewünschte Funktionalität.	1
Funktionalitäten: Das Parsen, Bearbeiten und Verwenden des Datenformats sollte so einfach und so schnell wie möglich zu bewerkstelligen sein.	2

4.1.2 XML

Kriterium	Details	Gew.	Punkte	Total
Overhead	Der Overhead bei XML ist sehr gross.	3	0	0
Komplexität	XML ist einfach zu verstehen und bei den Entwicklern des IndoorGuides schon gut bekannt.	1	2	2
Funktionalität	Das Parsen von XML ist bestens bekannt jedoch auch „relativ" aufwändig und langsam.	2	1	2
			Total:	4

4.1.3 JSON

Kriterium	Details	Gew.	Punkte	Total
Overhead	Overhead existiert bei JSON praktisch nicht und ist deshalb sehr gut geeignet.	3	2	6
Komplexität	JSON wurde von den Entwicklern noch nie verwendet. Dies sollte jedoch nicht zu viel Zeit fürs Technologiestudium beanspruchen.	1	1	1
Funktionalität	JSON ist kompakt und schnell, das Parsen ist ebenfalls weniger aufwändig.	2	2	4
			Total:	**11**

4.1.4 Schlussfolgerung

Die Wahl fällt auf JSON. Obwohl sich das Entwicklerteam damit noch nicht auskennt, überwiegen die Vorteile. Da bei dieser Applikation vor allem Wert auf die Performance und ein minimaler Datentransfer gelegt werden soll, eignet sich diese Technologie am besten.

4.2 Geonames Provider

Es gilt zu evaluieren, mittels welches Services das Geoname-Lookup (Auffinden einer Koordinate zu einem Namen) realisiert werden soll. Dazu wurden drei Varianten eruiert, **GeoNames.org**, **OpenStreetMap** (OSM) und **OpenStreetMap-in-a-Box** (OSM-in-a-Box).

4.2.1 Kriterienkatalog und Gewichtung

Kriterium (Beschreibung)	Gewicht (1-3)
Unabhängigkeit: Die Applikation sollte möglichst unabhängig von fremden Serviceprovidern sein.	2
Performanz: Der Service sollte die Requests möglichst schnell verarbeiten und beantworten.	3
Verfügbarkeit: Der Service sollte so wenige Downtimes (Wartungsarbeiten, Ausfall) wie möglich haben.	1
Komplexität: Der Service sollte möglichst trivial zu benutzen sein.	1
Funktionalitäten: Der Service sollte flexibel sein und eine grosse Variation an Datenformaten bieten.	2

4.2.2 GeoNames.org

Kriterium	Details	Gew.	Punkte	Total
Unabhängigkeit	Die Unabhängigkeit ist bei diesem Service leider nicht gegeben.	2	0	0
Performanz	GeoNames ist ein „kleineres" Projekt und daher eher langsam, da grosse Auslastung auf dem Server herrscht.	3	1	3
Verfügbarkeit	Der Service kann durch Wartungsarbeiten, Fehler oder Überlastung temporär nicht mehr verfügbar sein. Ebenfalls liegen zum Beispiel Wartungszeitfenster nicht in unserer Hand.	1	1	1
Komplexität	Abfragen via WebService sind einfach zu realisieren. Gutes Wiki und Dokumentation vorhanden.	1	2	2
Funktionalität	GeoNames bietet XML und JSON als Datenformate. Ebenfalls wird eine grosse Anzahl vorbereiteter WebServices angeboten, dessen Funktionalitäten jedoch für diese Applikation nicht nötig sind.	2	2	4
			Total:	**10**

4.2.3 OpenStreetMap (OSM)

Kriterium	Details	Gew.	Punkte	Total
Unabhängigkeit	Die Unabhängigkeit ist bei diesem Service leider nicht gegeben.	2	0	0
Performanz	OSM ist ein grosses Projekt, hat jedoch auch eine grosse Auslastung.	3	1	3
Verfügbarkeit	Der Service kann durch Wartungsarbeiten, Fehler oder Überlastung temporär nicht mehr verfügbar sein. Ebenfalls liegen zum Beispiel Wartungszeitfenster nicht in unserer Hand.	1	1	1
Komplexität	Sehr gutes Wiki in dem das meiste beschrieben ist.	1	2	2
Funktionalität	OSM bietet XML. Auch eine grosse Anzahl vorbereiteter WebServices mit nützlichen Funktionalitäten.	2	2	4
			Total:	**10**

4.2.4 OpenStreetMap-in-a-Box

Kriterium	Details	Gew.	Punkte	Total
Unabhängigkeit	Sehr grosse Unabhängigkeit, da der Server von der HSR betrieben wird. Daten werden von OSM gespiegelt. Lokal auf dem Server kann alles konfiguriert werden.	2	2	4
Performanz	Da der Server noch nicht ausgelastet ist, ist die momentane Performance hoch. Dies kann sich jedoch in Zukunft ändern.	3	2	6
Verfügbarkeit	Sollten Störungen auftreten, können diese von der HSR direkt behoben werden. Wartungsarbeiten können geplant durchgeführt werden.	1	2	2
Komplexität	Klare Datenstruktur, jedoch noch nicht so ausführlich beschrieben. Durch direkte Ansprechpersonen vom OSM-in-a-Box Projekt ist Unterstützung vorhanden.	1	2	2
Funktionalität	OSM-in-a-Box bietet XML und JSON.	2	2	4
			Total:	**18**

4.2.5 Schlussfolgerung

Die Wahl fällt auf OpenStreetMap-in-a-Box. Die Faktoren Unabhängigkeit und Performanz sind am wichtigsten. Da der Server durch die HSR betrieben wird, können Anpassungswünsche angebracht werden und bei Problemen sind die Ansprechpersonen direkt vor Ort.

4.3 POI-Provider

Es gilt zu evaluieren, mittels welches Services die POIs heruntergeladen werden sollen. Die Kriterien und Detailbeschreibungen sind ähnlich wie in der vorhergehenden Evaluation (Kapitel „4.2 Genoames Provider"). Zwei Varianten wurden eruiert, **OpenStreetMap** (OSM) und **OpenStreetMap-in-a-Box** (OSM-in-a-Box).

4.3.1 OpenStreetMap

Kriterium	Details	Gew.	Punkte	Total
Unabhängigkeit	Die Unabhängigkeit ist bei diesem Service leider nicht gegeben.	2	0	0
Performanz	OSM ist ein grosses Projekt, hat jedoch auch eine grosse Auslastung.	3	1	3
Verfügbarkeit	Der Service kann durch Wartungsarbeiten, Fehler oder Überlastung temporär nicht mehr verfügbar sein. Ebenfalls liegen zum Beispiel Wartungszeitfenster nicht in unserer Hand.	1	1	1
Komplexität	Sehr gutes Wiki in dem das meiste beschrieben ist. Leider macht die nicht definierte Attributwahl das ganze etwas komplexer.	1	1	1
Funktionalität	OSM bietet XML. Auch eine grosse Anzahl vorbereiteter WebServices mit nützlichen Funktionalitäten.	2	2	4
			Total:	9

4.3.2 OpenStreetMap-in-a-Box

Kriterium	Details	Gew.	Punkte	Total
Unabhängigkeit	Sehr grosse Unabhängigkeit, da der Server von der HSR betrieben wird. Daten werden von OSM gespiegelt. Lokal auf dem Server kann alles konfiguriert werden.	2	2	4
Performanz	Da der Server noch nicht ausgelastet ist, ist die momentane Performance hoch. Dies kann sich jedoch in Zukunft ändern.	3	2	6
Verfügbarkeit	Sollten Störungen auftreten, können diese von der HSR direkt behoben werden. Wartungsarbeiten können geplant durchgeführt werden.	1	2	2
Komplexität	Klare Datenstruktur, jedoch noch nicht so ausführlich beschrieben. Durch direkte Ansprechpersonen vom OSM-in-a-Box Projekt ist Unterstützung vorhanden.	1	2	2
Funktionalität	OSM-in-a-Box bietet XML und JSON.	2	2	4
			Total:	18

4.3.3 Schlussfolgerung

Die Wahl für den POI-Provider fällt hier konsequenterweise aus denselben Gründen wie bei der vorhergehenden Evaluation (Kapitel „4.2 Geonames Provider") auf OSM-in-a-Box.

5. Umsetzungskonzept

In diesem Kapitel wird eine grobe Übersicht über das angewandte Lösungskonzept gegeben. Detailliertere Informationen werden im Teil III beschrieben. Auch im Kapitel „6.2 Zielerreichung" wird kurz beschrieben, wie die Anforderungen erfüllt wurden.

5.1 Datenquellen

Der IndoorGuide arbeitet mit verschiedenen Datenquellen. Hier wird kurz erklärt, für was welche eingesetzt und wie darauf zugegriffen wird. Anleitungen zum Erfassen von Daten sind im Kapitel „7. Anleitungen und Tutorials" im Teil III zu finden.

5.1.1 IndoorWPS Community Server

Auf diesem Server liegen alle Fingerprints und Karten. Beim Laden eines Places wird eine Anfrage an diesen Server geschickt, welcher alle Fingerprints und Karten aus einem gewissen Radius mit dem Place als Mittelpunkt zurückschickt. Über die Weboberfläche dieses Servers können die Daten erfasst werden. Zudem gibt es fürs Erfassen von Fingerprints zusätzlich den IndoorWPS-Client, mit welchem das Erfassen etwas leichter geht.

5.1.2 OpenStreetMap (OSM)

OSM dient als POI-Provider. Die Daten werden jedoch nicht direkt über OSM bezogen, sondern über den unten beschriebenen OSM-in-a-Box Server. Die POIs müssen auf dem OSM-Server als Indoor-POIs erfasst werden. Dies kann direkt auf der Website von OSM mit dem „Potlatch"-Editor gemacht werden. Etwas einfacher geht das Erfassen mit der Java-Applikation „JOSM".

5.1.3 OpenStreetMap-in-a-Box (OSM-in-a-Box)

Dieser Server spiegelt die Daten von des OSM-Servers, speichert jedoch nur die nötigen Informationen ab. Dieser Server wurde hauptsächlich ausgewählt, da er WFS[12]-Abfragen über JSON unterstützt. Dieses Format war wichtig, um den teuren Datentransfer möglichst klein zu halten. JSON eignet sich dabei deshalb so gut, weil praktisch kein Overhead entsteht. Auf diesem Server werden keine Daten vom Benutzer selbst verändert, sondern lediglich die POI-Daten heruntergeladen. Zudem wird auch die Suche nach einem Place darüber abgewickelt. Die Reaktionszeit ist schneller als bei OSM und der IndoorGuide hat direkt nicht noch mit einem dritten Server (OSM) zu tun.

[12] Quelle WFS: http://www.opengeospatial.org/standards/wfs

5.2 Lokalisierung

Die Position wird mit zwei Services bestimmt. Die Auswahl, wann welcher Service benutzt wird, hängt von der Verfügbarkeit, der Genauigkeit und den Einstellungen ab.

5.2.1 IndoorWPS

Anhand dieses Services wird die Lokalisierung über WLAN gemacht. Die Genauigkeit liegt bei 4 – 10 Meter. Dabei ist es unter anderem davon abhängig, wie viele Access Points vorhanden und wie diese verteilt sind. Wo und wie viele Fingerprints erfasst wurden ist ebenfalls ausschlaggebend. Die Fingerprints sollten möglichst in den Ecken von Räumen gemacht werden, um eine gute Mittelung zu erreichen. Da beim heutigen Entwicklungsstand vom IndoorWPS noch keine brauchbare Genauigkeit, sowie Informationen über Gebäude und Stockwerk zurückgegeben werden, konnten gewisse Features noch nicht implementiert werden.

5.2.2 GPS

Als zweite Lokalisierungsmöglichkeit wurde GPS implementiert. Dieses muss jedoch zuerst in der Android-Konfiguration aktiviert sein und der Benutzer muss es bei den Settings im IndoorGuide noch zusätzlich aktivieren. Dies ist nötig, da GPS standardmässig ausgeschaltet sein soll, um Strom zu sparen. Falls GPS eingeschaltet ist, wird auf diesen Service gewechselt, sobald die Genauigkeit besser als 15m beträgt und die Koordinate nicht veraltet ist.

Beim Starten des IndoorGuides wird ein Update-Thread gestartet, welcher in regelmässigen Abständen (momentan alle 2 Sekunden) eine Location holt. Falls weder IndoorWPS noch GPS ihren Dienst anbieten, wird eine „leere" Location mit Latitude 0.0 und Longitude 0.0 zurückgegeben.

5.3 Datenhaltung

Die heruntergeladenen Daten werden in zwei SQLite-Datenbanken gespeichert. Zwei sind es aus dem Grund, weil die IndoorWPS-Library die Daten selbständig verwaltet. In dieser Datenbank sind nur die Fingerprints gespeichert. In der Datenbank des IndoorGuides sind die POIs, Karten und Places gespeichert. Diese Datenbank musste auf die SD-Card ausgelagert werden, da sie den zur Verfügung stehenden internen Speicherplatz überschritt. Die Datenbank ist hauptsächlich wegen den Karten relativ gross geworden. Dabei handelt es sich während den Tests mit dem Place „Hochschule Rapperswil" um ca. 4 MB, wobei eine Karte mit 500 KB noch zu gross und nicht optimal war. Weitere Details zur Datenhaltung sind im Kapitel „3.2 Datenbank-Modell" im Teil III beschrieben.

6. Resultate und Ausblick

6.1 Schlussfolgerung

In dieser Arbeit wurde ein Produkt entwickelt, welches die gestellten Anforderungen erfüllt, jedoch noch reichlich Spielraum für Erweiterungen zulässt. Für den Anwendungsbereich im Gebäude ist diese AR-Applikation mit WLAN-Lokalisierung momentan noch einzigartig. Die Problematik der Ungenauigkeit, die bis 10 Meter sein kann und die Gegebenheit, dass sich die POIs in naher Distanz befinden, haben spezielle Ansprüche ans User Interface gestellt. Es ist jedoch gelungen, ein schönes und intuitiv zu bedienendes User Interface zu gestalten, welches die erwähnte Problematik vor dem Benutzer so gut wie möglich versteckt. Durch das Anzeigen der POIs in einer distanzsortierten Liste springen sie nicht in der AR-Ansicht herum und das Bild bleibt für den Benutzer ruhig und übersichtlich.

AR- und Kartenansicht kombiniert mit der Lokalisierung im Gebäude stellt eine Neuheit dar und kann in vielen Anwendungsgebieten nützlich eingesetzt werden. Mehr dazu ist im Kapitel „2. Vision und Ziele" im Teil I zu finden.

Was an dieser Stelle jedoch auch erwähnt werden muss, ist, dass der Anbieter gewisse Vorbereitungen treffen muss, damit das System auch sinnvoll benutzt werden kann. Es ist eine gut abgedeckte WLAN-Infrastruktur notwendig, die für eine gute Lokalisierung ausschlaggebend ist. Ebenfalls müssen genügend Fingerprints erfasst werden. Schliesslich müssen noch die POIs erfasst und die Karten auf den Server geladen werden. Nach dieser Arbeit wird der IndoorGuide4Android einen Mehrwert für grosse und komplexe Gebäude bringen.

6.2 Zielerreichung

In folgender Aufzählung werden rückblickend die funktionalen und nicht-funktionalen Anforderungen betrachtet und deren Zielerreichung in einem kurzen Resultatbericht beschrieben. Fast alle Anforderungen konnten erfüllt werden, die niedrig priorisierte Anforderung F13 wurde aus Zeitgründen zurückgestellt und drei Anforderungen (F09, F11, F12) konnten wegen fehlender Funktionalität beim IndoorWPS nicht vollständig, jedoch zufriedenstellend erfüllt werden. Die nicht-funktionale Anforderung NF03 konnte aus technischen Gründen abhängig von der Android-Plattform nur teilweise erfüllt werden. Die funktionalen und nicht-funktionalen Anforderungen sind im Kapitel "2. Anforderungsspezifikation" im Teil III zu finden.

ID	Anforderung	Resultat	✓
F01	Die Position des Benutzers soll zuverlässig bestimmt werden.	Die Positionierung des Benutzers wird mit dem IndoorWPS gemacht, welcher im jetzigen Entwicklungsstadium noch eine Ungenauigkeit von 4-10 Metern mit sich bringt. Diese konnte durch eine grössere Anzahl Fingerprints noch etwas verbessert werden. Durch die Berechnung der Position über den Durchschnitt einer grösseren Anzahl Messungen, konnte erreicht werden, dass sie stabiler bleibt (weniger herumspringt).	
F02	Ein Place soll über eine Suchfunktion gefunden werden.	Es wurden zwei verschiedene Suchfunktionen implementiert. Die eine geht auf den Suchbegriff des Benutzers, wobei dieser einen Ort oder ein POI (z.B. Hochschule Rapperswil) suchen kann. Es werden alle Suchresultate des eingegebenen Strings zurückgeliefert. Die andere Suchfunktion sucht automatisch anhand der aktuellen Koordinate, ob in der Umgebung Fingerprints, POIs und Karten vorhanden sind. Diese Funktion kann der Benutzer beispielsweise verwenden, wenn er an einem Ort ist, an dem er weiss, dass IndoorGuide-Daten vorhanden sind. Die Koordinate wird dann normalerweise über GPS bezogen.	
F03	Daten sollen im Voraus heruntergeladen werden können.	Es ist anhand der oben genannten Suchfunktion möglich, einen Place im Voraus herunterzuladen und zu speichern. Dies ermöglicht den Offline-Betrieb am Ort, an dem der IndoorGuide eingesetzt werden soll.	
F04	Die heruntergeladenen Daten sollen persistiert werden.	Um die Daten wie Fingerprints, POIs, Karten und Places auf dem Gerät zu speichern, wird eine SQLite Datenbank verwendet. Diese ist aus Platzgründen auf der SD-Card abgelegt.	
F05	Beim Starten der Applikation soll der zuletzt geladene Place geöffnet werden.	Beim Auswählen eines Places wird in der Datenbank ein Attribut gesetzt, welcher Place ausgewählt wurde. Durch diese Methode kann beim späteren Neustart der Applikation der zuletzt geladene Place geöffnet werden.	
F06	Zwischen AR- und Kartenansicht soll gewechselt werden können.	Bei vertikal gehaltenem Handy wird die AR-Ansicht eingeblendet. Bei horizontaler Lage die Kartenansicht. Um bei kleinsten Lageänderungen das herumspringen zwischen den Ansichten zu vermeiden, wurde eine Hysterese eingebaut. Dieser ist bei ca. 30° und hilft auch dabei, die Messages zu minimieren, die vom Sensor an den Applikationshandler geschickt werden.	

F07	POIs sollen im Kamerabild sowie mittels Radar und auf der Karte angezeigt werden.	Die POIs werden in der AR-Ansicht in einer distanzsortierten Liste gezeigt. Die Liste ist notwendig, da durch die Ungenauigkeit der Position und der relativen Nähe der POIs zum Betrachter diese nicht am richtigen Ort im Kamerabild platziert werden können (wie es von anderen AR-Applikationen bekannt ist, die mit viel grösseren Distanzen arbeiten). Die POIs in der näheren Umgebung werden in einem Radar als beschriftete Punkte dargestellt. Auf der Karte werden sie am richtigen Ort platziert.	
F08	Die POIs sollen aufgrund ihrer realen Position angezeigt werden.	Die POIs werden im Radar in proportional korrektem Abstand und Winkel zueinander angezeigt. Zudem sind sie anhand des Kompasses richtig ausgerichtet. Auf der Karte werden sie am richtigen Ort platziert.	
F09	POIs sollen gefiltert angezeigt werden.	Die POIs werden nach dem aktuell gewählten Place gefiltert. Dies schränkt die Anzahl POIs schon mal ein. Ein Vorteil der Darstellung in einer Liste ist, dass sie nach Distanz sortiert sind. D.h. dass die interessanten POIs immer zuoberst erscheinen. Das Filtern nach Gebäude oder Stockwerk, konnte noch nicht realisiert werden, da der IndoorWPS-Dienst die Gebäude- und Stockwerkinformation noch nicht anbietet. Dies ist jedoch eine mögliche Erweiterung, sobald das IndoorWPS-Projekt diesen Stand erreicht hat.	
F10	Die Kartenansicht soll einen Plan des Gebäudegrundrisses zeigen.	Die Karten können auf den IndoorWPS Community Server hochgeladen werden. Die Karte muss dabei zuerst georeferenziert und mit dem entsprechenden KML-File abgespeichert werden. Beim Herunterladen der Karte auf den IndoorGuide wird dann die Information der Eckkoordinaten und Ausrichtung der Karte als Information in einem XML mitgeliefert. Mit dieser Information kann der Gebäudegrundriss in der Kartenansicht richtig dargestellt werden.	
F11	Die Karten sollen je nach Ort, an dem sich der Benutzer befindet, gewechselt werden.	Nach dem Auswählen eines Places werden die Karten zur Auswahl angezeigt, welche beim aktuellen Standort des Benutzers verfügbar sind. Beispielsweise werden alle Stockwerkkarten des Gebäudes angezeigt, in dem sich der Benutzer befindet. Eine automatische Auswahl der richtigen Karte ist nicht möglich, da momentan noch keine Stockwerkinformation vom IndoorWPS-Dienst angeboten wird.	

F12	Die eigene Position soll in der Kartenansicht angezeigt werden.	Die eigene Position wird auf der Karte angezeigt. Beim Anzeigen der Karte wird auf diesen Punkt zentriert. Danach wird die Karte nicht automatisch bei Veränderung des Standorts verschoben, da es beim herumspringen der Position zu unruhig und verwirrend wirken würde. Um die eigene Position gibt es einen Ungenauigkeitskreis, der statisch gesetzt ist. Die dynamische Änderung ist noch nicht möglich, da die von IndoorWPS zurückgelieferten Genauigkeitswerte noch nicht brauchbar sind. Dies ist jedoch eine mögliche Erweiterung, sobald das IndoorWPS-Projekt diesen Stand erreicht hat.	
F13	Die Karte soll sich automatisch ausrichten.	Die Implementation für das automatische Ausrichten der Karte mit Hilfe des Kompasses wurde angefangen. Die mit niedriger Priorität versehene Anforderung konnte aus Zeitgründen nicht fertig gestellt werden. Sie wird nachträglich noch zu Ende entwickelt werden.	
F14	POI-Informationen sollen angezeigt werden können.	Der IndoorGuide wurde so aufgebaut, dass beim Anwählen eines POIs (in der Liste oder auf der Karte) Informationen darüber in einem neuen Fenster angezeigt werden. Dieses ist auch mit Buttons zu den Multimediainhalten und Websites versehen. Wenn keine Links hinterlegt sind, wird der Button in grau (deaktiviert) dargestellt.	
F15	Multimediainhalte / Websites eines POIs sollen geöffnet werden können.	Beim Anwählen eines Buttons zu einem Multimediainhalt oder einer Website wird diese im externen Browser geöffnet. Zudem besteht bei YouTube-Filmen die Auswahlmöglichkeit, diese in der YouTube-Applikation anzuschauen.	
F16	Ein Kompass soll die Richtung angeben.	Es wurde am Rand des Radars ein „N" eingebaut, welches immer nach Norden zeigt.	
F17	Lokalisierung soll ausserhalb des Gebäudes mit GPS gemacht werden.	GPS wurde als Alternative zur WLAN-Lokalisierung eingebaut. Wenn diese in den Einstellungen aktiviert ist, wird automatisch auf GPS-Koordinaten gewechselt, sofern diese eine gewisse Genauigkeit und Aktualität erreichen. Der aktive Dienst wird dem Benutzer mit einem Statussymbol angezeigt.	
F18	Der Benutzer soll die Applikation nach seinen Bedürfnissen konfigurieren können.	Mit der Möglichkeit, selbst Einstellungen vornehmen zu können, kann der Benutzer den IndoorGuide seinen Wünschen anpassen (z.B. GPS, den Radar und die Statusbenachrichtigungen ein- bzw. auszuschalten).	

NF01	User Interface	Das User Interface wurde durchgehend und schön gestaltet. Die Buttons, Listeneinträge oder sonstigen Objekte können gut ausgewählt werden, sodass eine gute Benutzbarkeit entstand. Ebenfalls sind die Abläufe logisch und intuitiv zu bedienen. Die Fehlermeldungen und sonstigen Meldungen informieren den Benutzer über die aktiven Vorgänge oder Probleme. Zusätzlich sind die Fehlermeldungen mit Detailinformationen versehen, die dem Entwickler helfen, den Fehler zu finden.	
NF02	Dokumentation	In der Dokumentation wurden die wesentlichen Entscheidungen und wichtigsten Klassen und Methoden beschrieben.	
NF03	Testing	Das automatische JUnit-Testing unter Android stellte sich als relativ schwierig heraus, da Activities nur beschränkt getestet werden können. Diese sind von Benutzeraktionen und Hardwarebedingten Events abhängig. Wo möglich wurden JUnit-Tests gemacht. Zudem wurde mit Systemtests unter verschiedenen Umständen und Szenarios die Funktionalität getestet.	
NF04	Teamgeist	Das Arbeitsklima im Team war angenehm und es wurde in die gleiche Richtung gearbeitet. Zudem gab es keine grösseren Auseinandersetzungen, was die Arbeitsaufteilung oder die Entscheidungen anbelangt.	
NF05	Meetings	Es wurden regelmässig Meetings mit dem Betreuer durchgeführt, bei denen er auf den aktuellsten Stand gebracht wurde und seine Wünsche einbringen konnte. Zudem konnten wertvolle Inputs geholt werden.	
NF06	Code	Der Code sollte verständlich geschrieben sein. Zudem wurden komplizierte Stellen mit Kommentaren versehen. Der ganze Code ist mit JavaDoc in Englisch dokumentiert worden.	

6.3 Ausblick: Weiterentwicklung

Der IndoorGuide4Android kann mit dem jetzigen Entwicklungsstand ohne weiteres verwendet werden. Trotzdem gibt es einige Möglichkeiten zur Weiterentwicklung. Diese werden im Kapitel „6. Resultate und Weiterentwicklung" im Teil III ausführlich erläutert. Hier nur eine kurze Auflistung der weiteren Möglichkeiten.

6.3.1 Genauigkeitsangabe

Dem Benutzer könnte angezeigt werden, wie genau seine Lokalisierung geschätzt wird. Dies wäre bei GPS kein Problem, bei IndoorWPS jedoch noch in Entwicklung.

6.3.2 Fingerprints anzeigen

Ein Feature für den Anbieter eines IndoorGuides wäre, wenn man auf der Karte die Fingerprints sichtbar machen könnte. Dies würde das Erfassen von Fingerprints unterstützen.

6.3.3 POIs filtern

Dem Benutzer könnte die Möglichkeit angeboten werden, die POI-Liste zu filtern. Eine solche Weiterentwicklung würde Übersicht bei grosser Anzahl POIs schaffen. Das Filtern nach Gebäude oder Stockwerk ist jedoch nur möglich, wenn der IndoorWPS erweitert würde. Dort müsste zur zurückgegebenen Location noch die Gebäude- und Stockwerkinformation mitgegeben werden.

6.3.4 Wegdistanzen berechnen

Momentan wird nur die Luftdistanz zu einem POI berechnet. Eine Weiterentwicklung wäre das Berechnen der wirklichen Wegdistanz. Dies würde wahrscheinlich das Implementieren von Routeninformationen bedingen.

6.3.5 POIs vom Handy aus erfassen

Für den Anbieter wäre es eine angenehme Sache, wenn er vor Ort beim POI, diesen erfassen und auf den Server laden könnte. Evt. könnte dies mit der Funktion verbunden werden, an dieser Stelle direkt auch einen Fingerprint zu setzten.

6.3.6 POIs mit Notizen versehen

Damit POIs noch interaktiver genutzt werden könnten, wäre das anbringen von Notizen eine spannende Weiterentwicklung. Dies würde z.B. in einem Museum ermöglichen, dass die Besucher Notizen mit ihren Kommentaren hinterlassen könnten.

6.4 Persönliche Berichte

6.4.1 Christoph Egger

Noch vor Antritt der Arbeit war ich etwas skeptisch gegenüber dem Themenbereich dieser Bachelorarbeit. Dies vor allem bezüglich der mir unbekannten Augmented Reality Technik und dem georeferenzieren, beziehungsweise der Verwaltung von Geodaten. Die Erfahrung, mit Android auf einem Smartphone zu programmieren, reizte mich jedoch sehr.

Zu Beginn habe ich mich dann intensiv mit der Aufgabenstellung und den daraus resultierenden Aufgaben beschäftigt. Folglich lernte ich die Augmented Reality Technik kennen und schätzen. Von den ersten Eindrücken durch die existierenden AR-Applikationen „Layar" und „Wikitude" war ich begeistert. Nun verstand ich die Zusammenhänge der uns gestellten Aufgabe und war überaus motiviert eine Indoorapplikation mit diesen Fähigkeiten zu kreieren. Als ich dann mit den ersten Schritten auf Android, begann war ich der Arbeit vollends verfallen. Android ist sehr angenehm zu programmieren und ich fand mich schnell zurecht.

Die Zusammenarbeit mit anderen Partien (IndoorWPS und OSM-in-a-box) macht die Arbeit noch abwechslungsreicher als sie sowieso schon war. Es gab vieles zu kommunizieren und einiges musste auf unsere Bedürfnisse angepasst werden. Die Zusammenarbeit hat jedoch gut funktioniert und war sehr konstruktiv.

Das Arbeiten mit Herrn Keller war für mich ebenfalls stets angenehm und konstruktiv. Herr Keller führte das Projekt gut und hatte massig kreative Ideen, welche das Projekt IndoorGuide weit vorantrieben. Das Zusammenspiel mit meinem Teamkollegen Adrian Geiter war stets angenehm und produktiv. Wir haben uns in den verschiedenen Aufgaben immer gut ergänzt.

Zum Abschluss kann ich mit Begeisterung sagen, dass es uns gelungen ist eine nützliche und funktionstüchtige Applikation zu erstellen, welche in diesem Rahmen noch nicht existiert.

6.4.2 Adrian Geiter

Ich war von Beginn weg motiviert für dieses Projekt, da ich selbst von Handys und deren Möglichkeiten begeistert bin. Ich sah auch den Nutzen hinter der Idee des IndoorGuides und konnte mich sehr schnell mit dem Thema anfreunden. Den Einblick in die Geoinformatik, zusammen mit der Lokalisierungsthematik, fand ich sehr interessant. Die Arbeit mit Android war für mich völliges Neuland, in der Benutzung wie auch in der Entwicklung. Doch das Einarbeiten war viel leichter als zuerst erwartet. Ich kann mir momentan gut vorstellen, auch weiterhin in der Entwicklung für Android Geräte tätig zu sein.

Dieses Projekt war für mich das erste agil durchgeführte Projekt. Die Auswahl von Scrum für dieses Projekt fand ich optimal und habe gerne mit dieser Methodik gearbeitet. Im Bereich Software Engineering lernte ich vor allem den bewussten Umgang mit Ressourcen. Da auf einem Handy die Ressourcen knapper als auf einem PC sind, musste sparsam damit umgegangen werden. Wir stiessen oft an Grenzen mit dem internen Speicher und mussten auch die langsamere und teure Internetverbindung beachten.

Das Arbeitsklima empfand ich als angenehm und konstruktiv. Teilweise hatte ich das Gefühl, dass unser Betreuer, Herr Keller, uns mit seinen vielen Ideen und Vorstellungen überfordern würde. Wir mussten ihn oder uns selbst ab und zu auf den Boden der Realität zurückholen. Trotzdem fand ich seine kreativen Ideen wertvoll für dieses Projekt. Es freute mich besonders zu sehen, wie er sich auch selbst mit dem Thema auseinander setzte und uns viele Hinweise zu ähnlichen Applikationen oder Themen schickte. Ich denke, ein so motivierter und engagierter Betreuer trägt viel für eine gelungene Arbeit bei. Die Zusammenarbeit mit meinem Teamkollegen Christoph Egger war ebenfalls gut. Wir waren nicht immer gleicher Meinung, konnten uns jedoch immer einigen. Unsere unterschiedlichen Stärken ergänzten sich gut und ich würde wieder ein Projekt mit ihm zusammen durchführen.

In einem Satz zusammengefasst: Dieses Projekt forderte mich heraus, war lehrreich und ich bin begeistert von dem erreichten Resultat.

6.5 Dank

An folgende Personen wollen wir unseren Dank für die Unterstützung aussprechen.

Prof. Stefan Keller (HSR): Wir danken Ihnen für die Unterstützung während des Projekts. Es war angenehm mit Ihnen zusammen zu arbeiten und haben Ihre Gedankenanstösse geschätzt.

Reto Senn (bitforge AG): Vielen Dank für deine Unterstützung beim User Interface. Wir konnten deine Ratschläge jeweils gut in die Arbeit einfliessen lassen.

Michael Klenk (HSR): Danke für deine Hilfe und Flexibilität beim integrieren der IndoorWPS-Library. Auch danken wir dir, dass du kurzfristige Änderungen und Wünsche ernst genommen und umgesetzt hast.

Andreas Meier & Joram Zimmermann (HSR Studenten): Danke für eure Unterstützung beim OSM-in-a-Box Projekt. Wir waren froh, dass wir ein Mitspracherecht bei der Datenstruktur für die Indoor-POIs hatten und dass ihr dies so umsetzen konntet.

Robert Geiter (geiterkonzept): Vielen Dank für das super Logo, welches du uns kreiert hast. Es gefällt uns sehr und macht einen professionellen Eindruck.

TEIL II – Projektmanagement

1. Projektorganisation

Name	Kontaktdaten	Rolle
Herr Stefan Keller (HSR)	vertraulich	Auftraggeber und Betreuer
Herr Reto Senn (bitforge AG)	vertraulich	Externer Projektberater
Herr Michael Klenk (HSR)	vertraulich	Mitarbeiter des IFS, Entwickler der *IndoorWPS*-Library
Herr Christoph Egger	vertraulich	Student, Entwickler des *IndoorGuide4Android*
Herr Adrian Geiter	vertraulich	Student, Scrum Master, Entwickler des *IndoorGuide4Android*

Die Studierenden teilen sich die Arbeiten gleichmässig untereinander auf. Beide sind jeweils über die Arbeiten des anderen informiert und kennen die Funktionsweise sowie den Code. Dies ermöglicht bei Ausfall eines Studierenden, dass der andere seine Arbeiten weiterführen kann.

Der Scrum Master ist für die Projektplanung, Aufgabenverteilung und die Leitung der Scrum Mettings zuständig. Diese Rolle wird von Adrian Geiter übernommen.

2. Besprechungen

Für dieses Projekt wurde die agile Software Entwicklungsmethode Scrum eingesetzt, bei welcher nachfolgende Arten von Besprechungen existieren. Zudem wurde auf Wunsch des Betreuers wöchentlich eine Besprechung durchgeführt.

Besprechung	Beschreibung
Sprint Planning	Zu Beginn eines Sprints wurde dieses Meeting durchgeführt. Dabei wurde die Planung für die nächste Entwicklungsphase vorgenommen, d.h. Stories aus dem Backlog dem nächsten Sprint zugeordnet.
Daily Scrum	Täglich durchgeführtes Meeting mit allen Entwicklern. 15 min Austausch über aktuellen Stand der Arbeit und aufgetretene Probleme.
Sprint Review	Nach einem Sprint wurde eine Kurzpräsentation aller neu entwickelten Funktionen durchgeführt. Dieses wurde auf Wunsch des Betreuers jede Woche beim Meeting mit ihm durchgeführt und nicht nur alle zwei Wochen nach einem Sprint.
Sprint Retrospective	Nach einem Sprint wurde rückblickend zusammengetragen, was positiv und negativ am vergangenen Sprint war und diente zur Verbesserung der Arbeitsqualität.
Meeting mit Betreuer	Wöchentlich wurde ein Meeting mit dem Betreuer gemacht. Dieses diente dazu, den Betreuer über den Stand der Arbeit zu informieren, allfällige Probleme zu besprechen und neue Ideen, Tipps und Gedanken des Betreuers in die Arbeit einfliessen zu lassen. Die Sitzungsprotokolle sind auf der beigelegten CD zu finden.

3. Projektvorgehen

3.1 Agiles Vorgehen

Die Aufgabenstellung für dieses Projekt verlangt eine agile Vorgehensweise. Dies aus dem Grund, einen flexiblen und schlanken Softwareentwicklungsprozess zu benutzen, welcher eine lauffähige Software gegenüber umfangreicher Dokumentation bevorzugt. Es soll mehr auf die zu erreichenden Ziele fokussiert werden. Als Grundlage dafür gelten das agile Manifest[13] und die 12 Prinzipien dahinter.

3.2 Scrum

Aus den verschiedenen agilen Vorgehensmodellen wurde für dieses Projekt Scrum ausgewählt. Dieses Modell eignet sich besonders gut, weil es das Ziel verfolgt, durch Reduzierung von unnötigem Management-Ballast möglichst schnell zu ausführbarer Software zu kommen. Mit Hilfe von lauffähigen Zwischenversionen wird Feedback beim Auftraggeber eingeholt, welches in die Weiterentwicklung mit einbezogen wird. Ebenfalls eignet sich diese Methode, da bei Beginn des Projektes noch nicht alle Details und Features, welche zu implementieren sind, bekannt waren.

Die Iterationen werden bei Scrum *Sprints* genannt. Die Sprints werden in diesem Projekt auf zwei Wochen ausgelegt. Zu Beginn eines Sprints werden die am höchsten Priorisierten Features aus dem Product Backlog dem Sprint zugeordnet. Die grob beschriebenen Features des Product Backlog werden feiner ausgearbeitet und im Sprint Backlog festgehalten. Während eines Sprints werden täglich kurze Meetings abgehalten, damit alle Entwickler den aktuellen Stand des Projekts mitbekommen. Nach einem Sprint wird die nutzbare Software während dem Sprint Review dem Product Owner vorgeführt. Schlussendlich wird noch ein team-internes Sprint Retrospective durchgeführt, bei welchem sich das Team über Erfahrungen des vergangenen Sprints und möglichen Verbesserungen des Entwicklungsprozesses reflektiert. Dann wird der nächste Sprint geplant und möglicherweise der Product Backlog mit gewonnenen Erkenntnissen und evt. neuen Features ergänzt und überarbeite.

[13] Quelle agiles Manifest: www.agilemanifesto.org

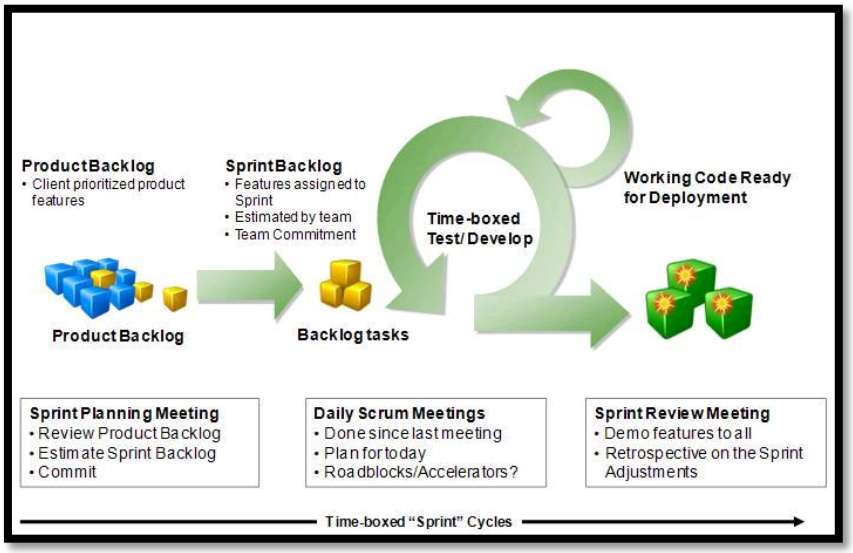


Bild 11: Scrum Prozess, Quelle: www.1doto.com

3.3 ScrumDesk

Für die Planung der Sprints wurde die Software ScrumDesk[14] (Version 4.5.0) eingesetzt. Sie bietet umfangreiche Funktionen und ist ideal für Entwicklerteams, die mit der Scrum-Methodik arbeiten. Alle Teammitglieder und der Product Owner haben Zugriff auf den aktuellen Stand des Projekts und können die ihnen zugewiesenen Stories bearbeiten.

Die Stories wurden vom ScrumMaster im Backlog erfasst und mit Prioritäten versehen. Vor jedem Sprint konnten die am höchsten priorisierten Stories ausgewählt und einem Teammitglied zugewiesen werden.

Die Software kann von Teams bis fünf Personen gratis verwenden werden.

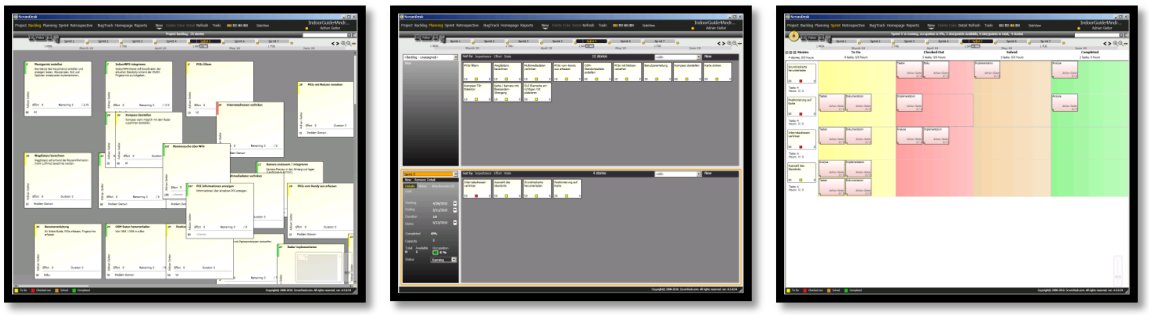


Bild 12: ScrumDesk, Product Backlog (links), Sprintplanung (Mitte), Sprint Backlog (rechts)

[14] Quelle ScrumDesk: www.scrumdesk.com

3.4 Product Backlog

Der Product Backlog enthält die Stories (Features) des ganzen Projekts. Vor jedem Sprint werden diese Stories neu bewertet und priorisiert. Bestehende Elemente können dabei entfernt, sowie neue hinzugefügt werden. Hoch priorisierte Features werden von den Entwicklern in den Sprint Backlog übernommen.

In der folgenden Auflistung sind alle Stories ganz kurz beschrieben. Die ausführliche Beschreibung ist im Sprint Backlog zu finden. Da sich die Prioritäten ständig änderten, wurden sie hier nicht aufgelistet.

ID	Story	Beschreibung	Sprint	✓
01	Layout entwerfen	Layout aller Views, Dialoge und Menüs auf Papier entwerfen.	Sprint1	
02	Erste Views und Hauptmenü erstellen	Das Hauptactivity und die grundlegenden Views erstellen. Ebenfalls ein Hauptmenü implementieren.	Sprint1	
03	IndoorWPS integrieren	IndoorWPS-Library im IndoorGuide integrieren. Sie soll Koordinaten des aktuellen Standorts anhand der WLAN-Fingerprints zurückgeben.	Sprint1	
04	Kamera integrieren	Kamera-Preview in den Hintergrund der AR-Ansicht legen. Dies im Landscape- wie auch im Portrait-Modus des Handys.	Sprint2	
05	Fingerprints laden	Alle vorhandenen Fingerprints vom IndoorWPS Community Server in die SQLite-Datenbank laden.	Sprint2	
06	Aktuelle Standort-Koordinate anzeigen	Den aktuellen Standort auf einfache Art anzeigen lassen. Dies anhand der Lokalisierung von IndoorWPS.	Sprint2	
07	Suche eines Places implementieren	Die Suche nach einem bestimmten Ort (Place) implementieren. Das Resultat soll eine Koordinate sein.	Sprint2	
08	POIs modellieren	POIs so modellieren, dass sie von OSM akzeptiert werden und den gewünschten Nutzen für den IndoorGuide erfüllen.	Sprint3	
09	OSM-in-a-Box Datenstruktur ausarbeiten	Absprache mit OSM-in-a-Box Team, welche Daten für den IndoorGuide relevant sind. Wünsche an die Datenbankstruktur anbringen.	Sprint3	
10	POIs herunterladen	Daten der POIs von OSM-in-a-Box herunterladen. Dies soll mit der WFS- & JSON-Technologie geschehen.	Sprint3	

11	Radar implementieren	Radar implementieren, welcher die POIs in der näheren Umgebung anzeigt. Diese sollen anhand des Kompasses richtig ausgerichtet werden.	Sprint3	
12	Moduswechsel (AR- / Kartenansicht)	Je nach Lage des Geräts soll eine andere Ansicht eingeblendet werden (waagrecht = Karte, senkrecht = Kamera -> AR).	Sprint3	
13	POIs in AR auflisten	POIs in einer Liste auf dem Display anzeigen und nach Distanz sortieren.	Sprint4	
14	Auf neue IndoorWPS-Version umstellen	Da es eine grosse Umstellung in der IndoorWPS-Library gab, muss diese neu integriert werden.	Sprint4	
15	Suche nach Places mit WFS und JSON	Die aktuelle Suche nach Places auf OSM, welche ein XML liefert, durch eine neue WFS Abfrage auf OSM-in-a-Box ersetzen, welche ein JSON liefern soll.	Sprint4	
16	POI-Informationen anzeigen	Die POIs sollen mit Detailangaben angezeigt werden können, wenn sie angewählt werden.	Sprint4	
17	Datenspeicherung	Die Daten wie Places, POIs und Maps sollen in der SQLite-Datenbank gespeichert werden. Datenbank erstellen und die nötigen DB-Manager implementieren.	Sprint4	
18	Multimediadaten anzeigen	Die Links zu Multimediadaten und Websites im Browser öffnen.	Sprint5	
19	Auswahl des Places implementieren	Die Auswahl des Places (nach der Suche) und das anschliessende Herunterladen der Fingerprints und POIs implementieren.	Sprint5	
20	Kompass darstellen	Kompass in irgendwelcher Weise beim Radar darstellen.	Sprint5	
21	Layout sauber umsetzen	Durchgehendes Layout umsetzen und passende Icons suchen.	Sprint5	
22	Kartenansicht implementieren	View für das Anzeigen der Karte implementieren. Dazu Karte unverzerrt darstellen und verschiebbar machen.	Sprint5	
23	Position auf der Karte berechnen	Aktueller Standort und die Position der POIs für die Karte berechnen und positionieren.	Sprint6	
24	POIs und Standort auf Karte anzeigen	Overlay mit POIs und aktuellem Standort erstellen. Beim Anwählen der POIs sollten die POI-Informationen angezeigt werden.	Sprint6	
25	GPS implementieren	Positionierung über GPS implementieren. Dazu soll bei einer brauchbaren Position automatisch in diesen Modus gewechselt werden.	Sprint6	

26	Places speichern / laden	Places in Datenbank speichern. Ebenfalls sollen sie geladen werden können. Bei Programmstart soll automatisch der letzte Place geladen werden.	Sprint6	
27	Aktueller Place suchen	Place anhand der aktuellen Koordinate (von GPS) suchen und entsprechend die Daten herunterladen.	Sprint6	
28	POIs nach Umkreis filtern	POIs in der Liste filtern, dass nur die angezeigt werden, welche zum geladenen Place gehören.	Sprint6	
29	Progress-Dialoge einbauen	Bei länger dauernden Vorgängen sollen Progress-Dialoge eingeblendet werden.	Sprint6	
30	Karten herunterladen	Alle Maps aus einem bestimmten Range beim IndoorWPS Community Server herunterladen und in der SQLite-Datenbank speichern.	Sprint7	
31	Karten auswählen	Menü erstellen, in dem man verfügbare Karten auswählen kann.	Sprint7	
32	DB auf SD-Card auslagern	Da der kleine interne Speicher, der zur Verfügung steht, bei den Kartendaten nicht ausreicht, soll die Datenbank auf die SD-Card verschoben werden.	Sprint7	
33	Settings implementieren	Settings implementieren (z.B. GPS oder Radar ein- / ausschalten)	Sprint7	
34	Compass-Tilt-Detector implementieren	Detektor implementieren, der eine Meldung ausgibt, wenn der Kompass / Lagesensor zu ungenau ist und neu kalibriert werden muss.	Sprint7	
35	Exceptionhandling umsetzen	Exceptions sollen dem Benutzer sinnvoll dargestellt werden.	Sprint7	
36	Statussymbol anzeigen	Statussymbol anzeigen, welches dem Benutzer zeigt, über welchen Dienst die aktuelle Location berechnet wird (oder keine Location geliefert wird).	Sprint7	
37	Karte drehen	Karte immer nach Norden ausrichten. Falls dies nicht möglich wäre, sollte ein Nordpfeil angezeigt werden.	-	
38	Genauigkeitskreis anzeigen	Auf der Karte soll um den aktuellen Standort ein Genauigkeitskreis gezeigt werden, welcher sich dynamisch anpasst.	-	
39	Fingerprints auf Karte anzeigen	Für das Erfassen und Analysieren der Fingerprints wäre es hilfreich, diese auf der Karte (wie die POIs) anzuzeigen. Dies sollte unbedingt standardmässig ausgeschaltet sein.	-	

40	Benutzeranleitung	Benutzeranleitung in den IndoorGuide integrieren.	-	
41	AR- & Kartenansicht mit fliessendem Übergang	Beim Wechsel von AR- & Kartenansicht soll ein fliessender (animierter) Übergang gemacht werden.	-	
42	POIs filtern	POIs sollen nicht nur nach Places gefiltert werden, sondern auch nach weiteren Attributen wie z.B. nach Building.	-	
43	Wegdistanz berechnen	Distanz von aktuellem Standort zu einem POI anhand von echten Wegdistanzen berechnen (nicht Luftlinie).	-	
44	POIs vom Handy aus erfassen	Um das Erfassen von POIs zu vereinfachen, sollen sie direkt vom Handy aus erfasst und dem Server übermittelt werden können.	-	
45	POIs mit Notizen versehen	Um den Benutzern die Möglichkeit zu geben, Notizen / Kommentare zu hinterlassen, sollen diese an einen POI angeheftet und dem Server übermittelt werden können.	-	

3.5 Sprint Backlogs

Die Sprints dauern jeweils 2 Wochen. Die Tätigkeiten werden vor dem Sprint definiert, indem die höchst priorisierten Stories aus dem Product Backlog ausgewählt werden. Nach einem Sprint wird jeweils ein ausführbares Release generiert.

In den folgenden Auflistungen werden die Stories der Sprints genauer beschrieben als im Backlog. Zudem ist die dafür aufgewendete Arbeitszeit angegeben (in Stunden).

3.5.1 Sprint 1 (03.03. - 16.03.2010)

ID	Story	Beschreibung	Dauer	✓
01	Layout entwerfen	Layout aller Views, Dialoge und Menüs auf Papier entwerfen. Ebenfalls den Ablauf festhalten, den ein Benutzer machen muss, wenn er den IndoorGuide benutzen will.	1.48	
02	Erste Views und Hauptmenü erstellen	Das Hauptactivity „IndoorGuide" und die grundlegenden Views für den IndoorGuide, den Aboutscreen und die Settings erstellen. Ebenfalls ein Hauptmenü mit About, Settings und Exit implementieren und mit deren Views verknüpfen.	2.25	
03	IndoorWPS integrieren	IndoorWPS-Library im IndoorGuide integrieren. Sie soll Koordinaten des aktuellen Standorts anhand der WLAN-Fingerprints zurückgeben.	22.28	

3.5.2 Sprint 2 (17.03. - 30.03.2010)

ID	Story	Beschreibung	Dauer	✓
04	Kamera integrieren	Kamera-Preview in den Hintergrund der AR-Ansicht legen. Dies im Landscape- wie auch im Portrait-Modus des Handys.	11.15	
05	Fingerprints laden	Vorläufig sollen alle vorhandenen Fingerprints vom IndoorWPS Community Server in die SQLite-Datenbank geladen werden. Später sollen die Fingerprints dann auf den ausgewählten Place beschränkt werden. Die Datenbank wird von der IndoorWPS-Library erstellt und verwaltet.	16.3	
06	Aktuelle Standort-Koordinate anzeigen	Den aktuellen Standort vom IndoorWPS beziehen und auf einfache Art anzeigen lassen. Die Anzeige kann als Text geschehen. Zudem die Genauigkeit der Koordinaten testen.	1.93	
07	Suche eines Places implementieren	Die Suche nach einem bestimmten Ort (Place) implementieren. Das Resultat soll eine Koordinate sein. Zuerst soll der Dienst evaluiert werden. Die Suche soll mit Ortschaften und POIs (z.B. Hochschule Rapperswil) funktionieren.	20.82	

3.5.3 Sprint 3 (31.03. - 13.04.2010)

ID	Story	Beschreibung	Dauer	✓
08	POIs modellieren	POIs so modellieren, dass sie von OSM akzeptiert werden und den gewünschten Nutzen für den IndoorGuide erfüllen. Da es momentan noch keine offiziellen Indoor-Daten bei OSM gibt, muss ein Modell gefunden werden, das den Ansprüchen für den IndoorGuide genügt und dennoch nicht zu proprietär ist.	3.42	
09	OSM-in-a-Box Datenstruktur ausarbeiten	Absprache mit OSM-in-a-Box Team, welche Daten für den IndoorGuide relevant sind. Wünsche an die Datenbankstruktur anbringen. Die Datenstruktur soll die neu modellierten OSM-Indoor-POIs wiederspiegeln können, sodass bei einer Anfrage NUR als Indoor-POI gekennzeichnete POIs zurückgegeben werden.	3.2	
10	POIs herunterladen	Daten der POIs von OSM-in-a-Box herunterladen. Dies soll mit der WFS- & JSON-Technologie geschehen. Dazu soll die WFS-Abfrage so formuliert werden, dass die Datenauswahl schon auf dem Server möglichst eingeschränkt und nur das wichtigste dem Client übertragen wird.	6.8	

11	Radar implementieren	Radar implementieren, welcher die POIs in der näheren Umgebung anzeigt. Diese sollen anhand des Kompasses richtig ausgerichtet werden. Es soll ausprobiert werden, ob es möglich ist, gewisse POIs zu beschriften.	15.03	
12	Moduswechsel (AR- & Kartenansicht)	Je nach Lage des Geräts soll eine andere Ansicht eingeblendet werden (waagrecht = Karte, senkrecht = Kamera -> AR). Es muss herausgefunden werden, ab welchem Winkel die Umstellung zu vollführen ist, damit es nicht störend wirkt.	16.03	

3.5.4 Sprint 4 (14.04. - 27.04.2010)

ID	Story	Beschreibung	Dauer	✓
13	POIs in AR auflisten	POIs in einer Liste auf dem Display anzeigen und nach Distanz sortieren. Man soll die POIs in der Liste auswählen können, ebenfalls muss die Liste scrollbar sein. Angezeigt werden soll der Name, eine abgekürzte Beschreibung (z.B. erste paar Worte der Beschreibung) und die Distanz in Luftlinie. Die Liste soll dynamisch bei jedem Standortwechsel neu nach Distanz sortiert werden.	3:14	
14	Auf neue IndoorWPS-Version umstellen	Da es eine grosse Umstellung in der IndoorWPS-Library gab, muss diese neu integriert werden. Die Integration der Library wurde vereinfacht. Es wurden auch weitere Funktionen hinzugefügt.	1:18	
15	Suche nach Places mit WFS und JSON	Durch das Arbeiten mit OSM-in-a-Box ergeben sich auch bei der Suche nach Places neue Möglichkeiten. Die Evaluation hat ergeben, dass die aktuelle Suche nach Places auf OSM (XML-Response), durch eine neue WFS Abfrage auf OSM-in-a-Box ersetzt werden soll. Bei dieser Anfrage soll eine JSON-Response zurückgegeben werden.	9:32	
16	POI-Informationen anzeigen	Die POIs sollen mit Detailangaben angezeigt werden, wenn sie angewählt werden. In der Detailansicht sollen alle gespeicherten Informationen des POIs und zusätzlich die Luftdistanz zum aktuellen Standort ersichtlich sein. Die Beschreibung soll scrollbar sein, da diese auch mit viel Text gefüllt werden kann. Die Multimedialinks sollen als Buttons eingebunden werden.	7:00	

17	Datenspeicherung	Die Daten wie Places, POIs und Maps sollen in der SQLite-Datenbank gespeichert werden. Datenbank entwerfen und Create-Statements erstellen. Ebenfalls für jedes Datenobjekt einen DB-Manager erstellen, welcher das Speichern und Laden der Daten übernimmt.	9:15	

3.5.5 Sprint 5 (28.04. - 11.05.2010)

ID	Story	Beschreibung	Dauer	✓
18	Multimediadaten anzeigen	Die Links zu Multimediadaten wie Bilder, Video und Audio, sowie Websites und Wikipediaartikel sollen geöffnet werden. Entweder im externen Browser oder integriert in der POI-Information-Ansicht.	2.47	
19	Auswahl des Places implementieren	Die Auswahl des Places (nach der Suche) und das anschliessende Herunterladen der Fingerprints und POIs implementieren. Dazu sollen nur die Daten heruntergeladen werden, die in einem gewissen Umkreis des Places liegen (z.B. 1 km).	5.2	
20	Kompass darstellen	Kompass in irgendwelcher Weise beim Radar darstellen. Eine Möglichkeit wäre, Norden als „N" darzustellen und um den Kompass kreisen zu lassen.	3.0	
21	Layout sauber umsetzen	Das Layout verbessern und einheitlich gestalten. Dazu sollen passende Icons gesucht und die Farben passend gewählt werden. Durch Drittpersonen ein Feedback zum Layout einholen und dieses in die Entwicklung einfliessen lassen.	16.82	
22	Kartenansicht implementieren	View für das Anzeigen der Karte implementieren. Dazu soll die georeferenzierte Karte in „voller Grösse" und unverzerrt dargestellt werden. Sie soll mit dem Finger verschiebbar sein.	18.58	

3.5.6 Sprint 6 (12.05. - 25.05.2010)

ID	Story	Beschreibung	Dauer	✓
23	Position auf der Karte berechnen	Aktueller Standort und die Position der POIs für die Karte berechnen und positionieren. Für die Berechnung müssen die XML-Daten der Karte zur Hilfe genommen werden, in der die Eckkoordinaten des Bildes angegeben sind. Dies muss dann in Pixel umgerechnet werden.	18.4	
24	POIs und Standort auf Karte anzeigen	Die POIs und der aktuelle Standort sollen als Overlay auf die Karte gelegt werden. Wichtig zu beachten ist, dass beim Verschieben der Karte die Elemente fix auf der Karte bleiben. Beim Anwählen der POIs sollen die POI-Informationen angezeigt werden. Es soll die gleiche Activity wie bei der Auswahl aus der Liste angezeigt werden.	30.27	
25	GPS implementieren	Positionierung über GPS implementieren. Dazu soll automatisch in diesen Modus gewechselt werden, wenn man eine brauchbare Position erhält. Wichtig ist, dass GPS in den Einstellungen aktiviert / deaktiviert werden kann. Ebenfalls soll darauf geachtet werden, dass bei zu ungenauen GPS-Koordinaten auf IndoorWPS gewechselt werden soll.	5.5	
26	Places speichern / laden	Die Places sollen in der Datenbank gespeichert und geladen werden können. Es muss zusätzlich ein Attribut gespeichert werden, welches definiert, ob der Place gewählt wurde. Bei Programmstart soll automatisch der letzte aktive Place geladen werden.	24.63	
27	Aktueller Place suchen	Place anhand der aktuellen Koordinate suchen. Dies kann über GPS oder wenn vorhanden über IndoorWPS geschehen. Falls eine Koordinate gefunden wird, sollen entsprechend die Daten herunterladen, ansonsten eine Fehlermeldung ausgegeben werden. Da der automatisch gesuchte Place keinen Namen besitzen würde, soll der Benutzer selbst einen Namen definieren können.	12.0	
28	POIs nach Umkreis filtern	Die POIs in der Liste sollen so gefiltert werden, dass nur die angezeigt werden, welche zum geladenen Place gehören. Dies soll wie beim Herunterladen mit einem gewissen Radius um den Place gemacht werden.	3.4	

29	Progress-Dialoge einbauen	Bei länger dauernden Vorgängen sollen Progress-Dialoge eingeblendet werden. Dies ist beispielsweise beim Herunterladen der Daten oder beim Suchen nach einem Place (nach Suchbegriff oder aktueller Koordinate) notwendig.	4.92	

3.5.7 Sprint 7 (26.05. - 08.06.2010)

ID	Story	Beschreibung	Dauer	✓
30	Karten herunterladen	Alle Karten aus einem bestimmten Range beim IndoorWPS Community Server herunterladen und in der SQLite-Datenbank speichern. In einem ersten Schritt müssen die zur Karte gehörenden XML-Dateien heruntergeladen und geparst werden. Darin ist dann der nötige Link zum Bild zu finden. Im XML stehen weitere wichtige Informationen wie z.B. die Eckkoordinaten und der Floor. Da es grössere Datenmengen sind, muss hier besonders auf die Performance geachtet werden.	41.63	
31	Karten auswählen	Es soll ein Menü erstellt werden, in dem man verfügbare Karten auswählen kann. Dies ist z.B. beim längeren Drücken auf die Kartenansicht möglich oder direkt beim ersten Anzeigen der Karte. Im Menü sollen alle gefundenen Karten erscheinen, die mit der aktuellen Koordinate übereinstimmen (also Karten des Gebäudes, in dem man sich befindet).	8.8	
32	DB auf SD-Card auslagern	Da sich im letzten Sprint herausgestellt hat, dass der interne Speicher zu wenig Platz für die Datenbank mit gespeicherten Karten hat, muss die Datenbank auf die SD-Card ausgelagert werden. Falls dies nicht funktioniert, wäre noch eine Möglichkeit, die Karten in einem eigenen Ordner auf der SD-Card zu speichern und in der Datenbank lediglich den Pfad zur Karte zu speichern.	13.76	
33	Settings implementieren	Um dem Benutzer die Möglichkeit zu geben, den IndoorGuide in beschränkter Weise seinen Wünschen zu gestalten, sollen Einstellungsmöglichkeiten implementiert werden. Dazu gehören z.B. GPS oder Radar ein- / ausschalten.	2.05	

34	Compass-Tilt-Detector implementieren	Da der Kompass bei den meisten Handys relativ störanfällig ist, soll ein Detektor implementiert werden. Dieser soll eine Meldung ausgeben, wenn der Kompass / Lagesensor zu ungenau ist und neu kalibriert werden muss.	2.08	
35	Exceptionhandling umsetzen	Exceptions sollen dem Benutzer weitergeleitet und sinnvoll dargestellt werden. Dazu ist zu beachten, dass ein benutzerfreundlicher Text zu verwenden ist, mit dem der Benutzer etwas anfangen und einem Techniker helfen kann.	7.18	
36	Statussymbol anzeigen	Es soll ein Statussymbol angezeigt werden, welches dem Benutzer zeigt, über welchen Dienst die aktuelle Koordinate bezogen wird. Vor allem ist es wichtig für den Benutzer zu wissen, wenn keine Koordinate zurückgegeben wurde.	5.2	

4. Zeitplanung

4.1 Vorgeschriebene Arbeitszeit

Gemäss Vorgabe der Hochschule Rapperswil muss pro ECTS-Punkt 30 Arbeitsstunden geleistet werden. Die Bachelorarbeit ergibt bei erfolgreichem Abschluss 12 ECTS-Punkte, woraus eine minimale Arbeitszeit von 360 Stunden pro Student resultiert.

4.2 Terminplanung

Datum	Beschreibung
22.02.2010	Beginn der BA
24.02.2010	**Kickoff Meeting** (14:00)
03.03.2010	**Start Sprint 1,** Meeting mit Betreuer und Reto Senn, Bitforge (14:00)
10.03.2010	Meeting mit Betreuer (14:00)
17.03.2010	**Start Sprint 2,** Meeting mit Betreuer (14:00)
24.03.2010	Meeting mit Betreuer (15:30)
31.03.2010	**Start Sprint 3,** Meeting mit Betreuer (14:00)
14.04.2010	**Start Sprint 4,** Meeting mit Betreuer (14:00)
23.04.2010	Meeting mit Betreuer (13:30)
28.04.2010	**Start Sprint 5,** Zwischenpräsentation (09:00)
05.05.2010	Meeting mit Betreuer (14:00)

07.05.2010	Meeting mit Reto Senn, Bitforge (14:00)
12.05.2010	**Start Sprint 6**
20.05.2010	Meeting mit Betreuer (15:00)
26.05.2010	**Start Sprint 7**
27.05.2010	Meeting mit Betreuer (15:00)
04.06.2010	Meeting mit Betreuer (14:00)
10.06.2010	Meeting mit Betreuer (15:00)
11.06.2010	Abgabe Abstract & Ao-Poster an Betreuer
18.06.2010, 12:00	**Abgabe** des Berichts an Betreuer und des Ao-Posters im Studiensekretariat
01.07.2010, 11:00	Mündliche BA-Prüfung
25.06.2010	Präsentationen (HSR-Forum)

4.3 Projektphasen

22.02. – 02.03.2010	03.03. – 16.03.2010	17.03. – 30.03.2010	31.03. – 13.04.2010	14.04. – 27.04.2010
Vorbereitungs-arbeiten	Sprint 1	Sprint 2	Sprint 3	Sprint 4

28.04. – 11.05.2010	12.05. – 25.05.2010	26.05. – 08.06.2010	09.06. – 18.06.2010
Sprint 5	Sprint 6	Sprint 7	Abschlussarbeiten / Reservezeit

4.3.1 Vorbereitungsphase

In der ersten Phase werden Vorbereitungen fürs Projekt getroffen. Die Aufgabenstellung wird zusammen mit dem Betreuer (Product Owner) genau definiert und Projektmanagementvorbereitungen gemacht, die noch nicht direkt mit der Software zu tun haben.

4.3.2 Sprint 1 – 7

Die zweiwöchigen Sprints starten jeweils an einem Mittwoch. Dies aus dem Grund, da die zwei Projektmitarbeiter jeweils am Montag und teilweise am Dienstag extern in einer Firma arbeiten. Meistens sind Sitzungen mit dem Product Owner auf den Mittwoch terminiert. Der Inhalt der Sprints ist dem Kapitel „3.5 Sprint Backlog" zu entnehmen.

4.3.3 Abschlussarbeiten / Reservezeit

In der letzten Phase werden keine neuen Features implementiert. Diese Phase dient dazu, den Bericht zu beenden, Anleitungen zu schreiben, das Plakat zu erstellen und allenfalls noch letzte Fehler an der Software zu beheben. Diese Phase dient auch als Reservezeit, die gebraucht wird, wenn ein Risiko eintrifft. Die Risiken sind im Kapitel „6. Risiko Management" im Teil II zu finden.

4.4 Meilensteine

Jeder Abschluss eines Sprints stellt einen Meilenstein dar, da ein ausführbares Programm in einem Release vorliegt. Die jeweiligen Features und Ziele eines solchen Meilensteins sind der Sprintplanung zu entnehmen (siehe Kapitel „3.5 Sprint Backlog").

Nr.	Datum	Meilenstein
MS01	24.02.2010	Projektstart – Kickoff-Meeting
MS02	16.03.2010	Abschluss Sprint 1
MS03	30.03.2010	Abschluss Sprint 2
MS04	13.04.2010	Abschluss Sprint 3
MS05	27.04.2010	Abschluss Sprint 4
MS06	11.05.2010	Abschluss Sprint 5
MS07	25.05.2010	Abschluss Sprint 6
MS08	08.06.2010	Abschluss Sprint 7
MS09	18.06.2010	Schlussabgabe

4.5 Releases

Release	Datum	Funktionalität
Release 0.1	16.03.2010	Hauptactivity, Menü, About, Settings, integrierter IndoorWPS
Release 0.2	30.03.2010	AR-Ansicht, Anzeige der aktuellen Koordinate, Suche nach einem Place
Release 0.3	13.04.2010	Wechsel zwischen AR- & Kartenansicht, Radar, POI-Download
Release 0.4	27.04.2010	POI-Liste, POI-Information-View, Datenspeicherung, WFS-Abfragen
Release 0.5	11.05.2010	Anzeige Multimediadaten, Kompass, Auswahl des Places, Kartenansicht
Release 0.6	25.05.2010	POIs & Standort auf Karte anzeigen, GPS, aktueller Place suchen, POIs filtern, Progress-Dialoge
Release 0.7	08.06.2010	Karten herunterladen & auswählen, Settings, Compass-Tilt-Detector, Statusanzeige, Exceptionhandling
Release 1.0	18.06.2010	Bugs gefixt, refactored Code

5. Auswertung der Arbeitszeiten

5.1 Arbeitsumfang

Gemäss Vorgabe der Hochschule Rapperswil muss pro ECTS-Punkt 30 Arbeitsstunden geleistet werden. Die Bachelorarbeit ergibt bei erfolgreichem Abschluss 12 ECTS-Punkte, woraus eine minimale Arbeitszeit von 360 Stunden pro Student resultiert. Dies ergibt eine Gesamtarbeitszeit von mindestens 720 Stunden. Eingeplant wurden 23 Arbeitsstunden pro Woche und Mitarbeiter (= 782 Stunden Gesamtarbeitszeit). Dies entspricht einem Plus von 62 Stunden (8.6 %) gegenüber der Minimalarbeitszeit. Schlussendlich wurden 832 Stunden aufgewendet. Was einem Plus von 112 Stunden (15.5 %) gegenüber der Minimalarbeitszeit entspricht.

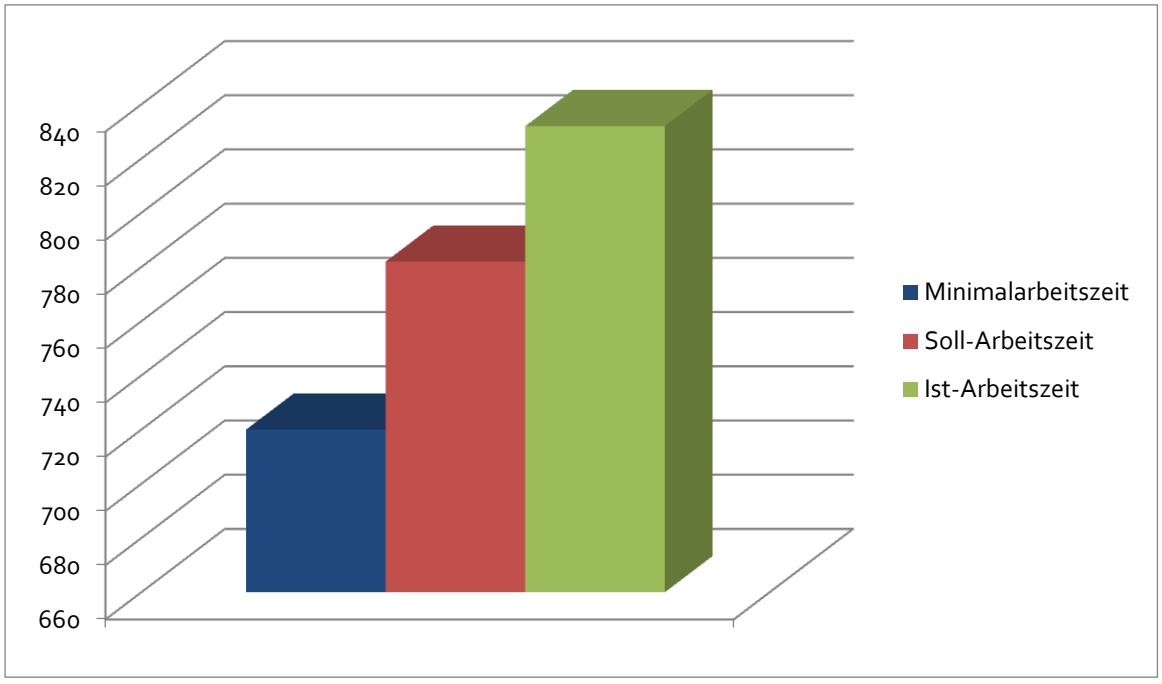


Bild 13: Arbeitsumfang

5.2 Wöchentliche Arbeitszeiten

In folgender Tabelle werden die geleisteten Arbeitsstunden pro Woche angezeigt. Die Soll-Arbeitszeit wurde von den Teammitgliedern auf 23 Stunden pro Person und Woche gesetzt (anstelle von den vorgeschriebenen ca. 21 Stunden).

Woche	Christoph Egger	Adrian Geiter
Woche 1 (22.02. – 28.02.2010)	12 Std. 55 Min.	13 Std. 52 Min.
Woche 2 (01.03. – 07.03.2010)	18 Std. 15 Min.	16 Std. 21 Min.
Woche 3 (08.03. – 14.03.2010)	14 Std. 55 Min.	22 Std. 31 Min.
Woche 4 (15.03. – 21.03.2010)	23 Std. 13 Min.	20 Std. 53 Min.
Woche 5 (22.03. – 28.03.2010)	19 Std. 51 Min.	23 Std. 38 Min.
Woche 6 (29.03. – 04.04.2010)	18 Std. 48 Min.	18 Std. 54 Min.
Woche 7 (05.04. – 11.04.2010)	22 Std. 8 Min.	25 Std. 41 Min.
Woche 8 (12.04. – 18.04.2010)	22 Std. 13 Min.	25 Std. 20 Min.
Woche 9 (19.04. – 25.04.2010)	12 Std. 37 Min.	24 Std. 2 Min.
Woche 10 (26.04. – 02.05.2010)	6 Std. 10 Min. *	23 Std. 47 Min.
Woche 11 (03.05. – 09.05.2010)	25 Std. 26 Min.	26 Std. 8 Min.
Woche 12 (10.05. – 16.05.2010)	21 Std. 43 Min.	10 Std. 31 Min. *
Woche 13 (17.05. – 23.05.2010)	32 Std. 38 Min.	32 Std. 15 Min.
Woche 14 (24.05. – 30.05.2010)	26 Std. 57 Min.	28 Std. 40 Min.
Woche 15 (31.05. – 06.06.2010)	37 Std. 10 Min.	34 Std. 41 Min.
Woche 16 (07.06. – 13.06.2010)	38 Std. 30 Min.	44 Std. 28 Min.
Woche 17 (14.06. – 16.06.2010)	44 Std. 52 Min.	41 Std. 50 Min.
Total:	**398 Std. 21 Min.**	**433 Std. 32 Min.**

* Private Abwesenheit

Bild 14: Wöchentliche Arbeitszeit

5.3 Projektphasen

5.3.1 Zeit pro Sprint

Die Sprints wurden mit 92 Stunden Soll-Zeit budgetiert. Dies teilt sich in 2 Wochen à 23 Stunden pro Teammitglied auf. Die Vorbereitungsphase und die Abschlussarbeiten werden hier auch erwähnt, obwohl sie keine Sprints sind. Diese dauern jeweils 23 Stunden pro Teammitglied.

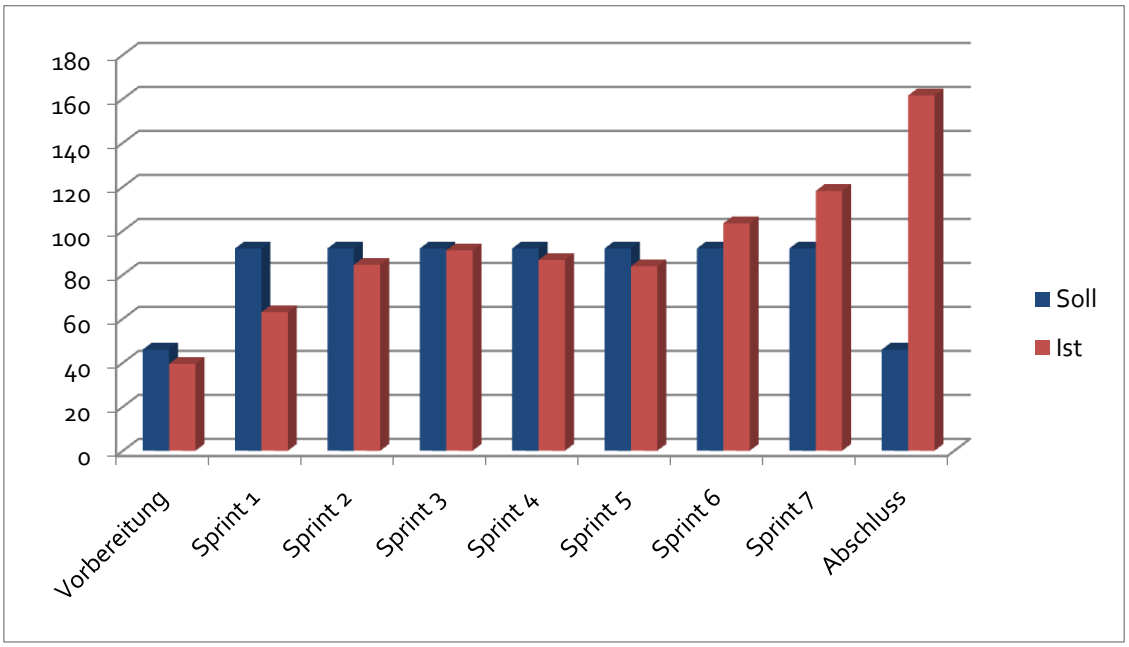


Bild 15: Zeit pro Sprint

5.3.2 Zeit pro Arbeitspaket

Jeder Sprint beinhaltet verschiedene Arbeitspakete, welche dann zusätzlich in verschiedene Aufgaben unterteilt sind. In folgenden Diagrammen ist zu sehen, für welche Arbeitspakete wie viel Zeit aufgewendet wurde.

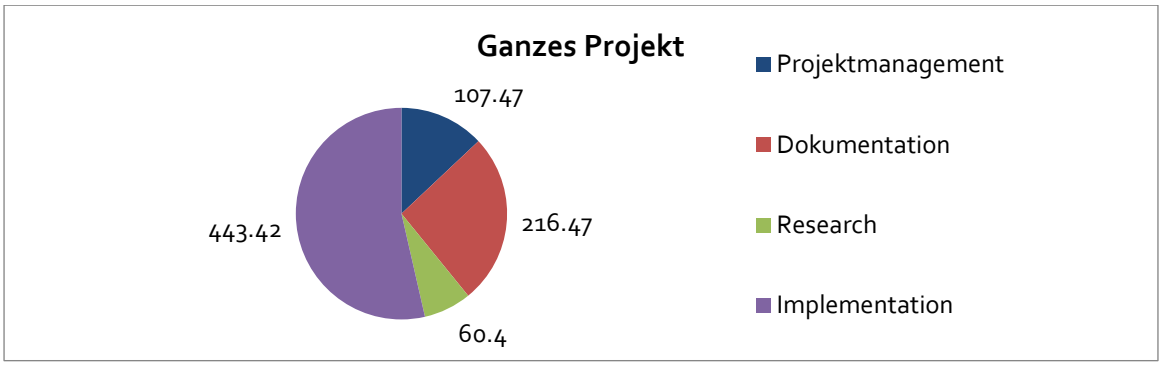


Bild 16: Arbeitspaket - ganzes Projekt

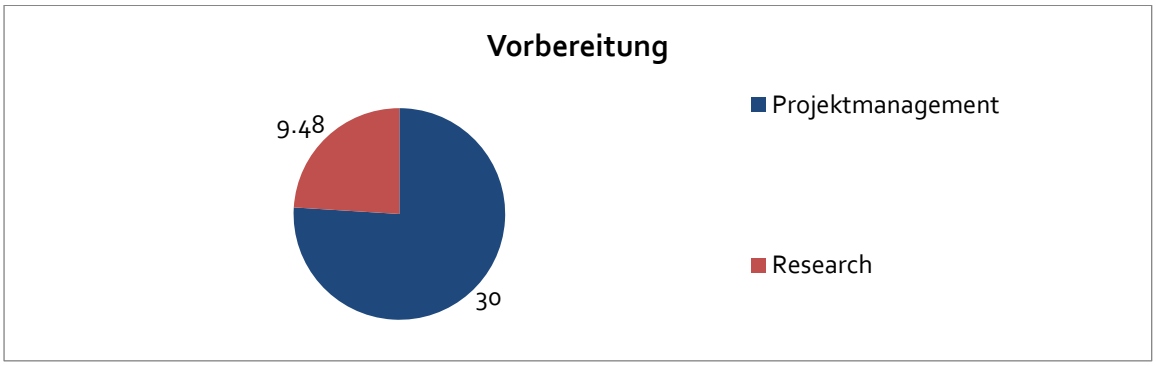


Bild 17: Arbeitspaket - Vorbereitung

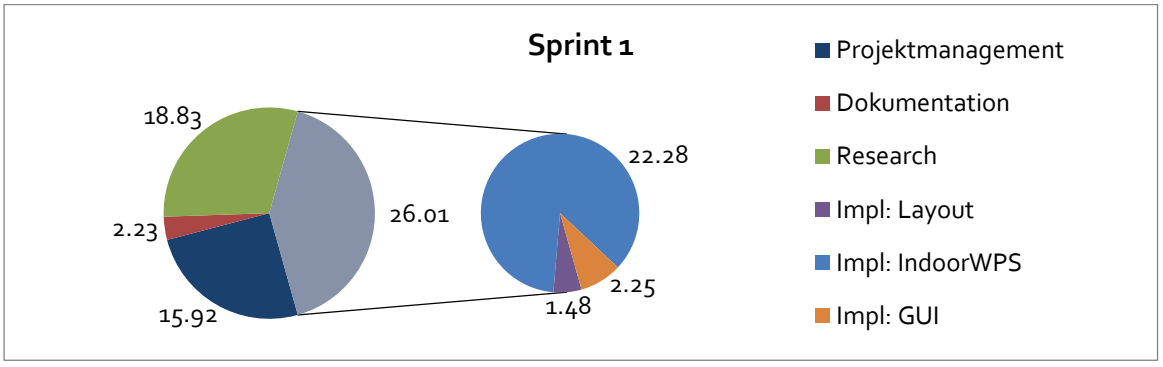


Bild 18: Arbeitspaket - Sprint 1

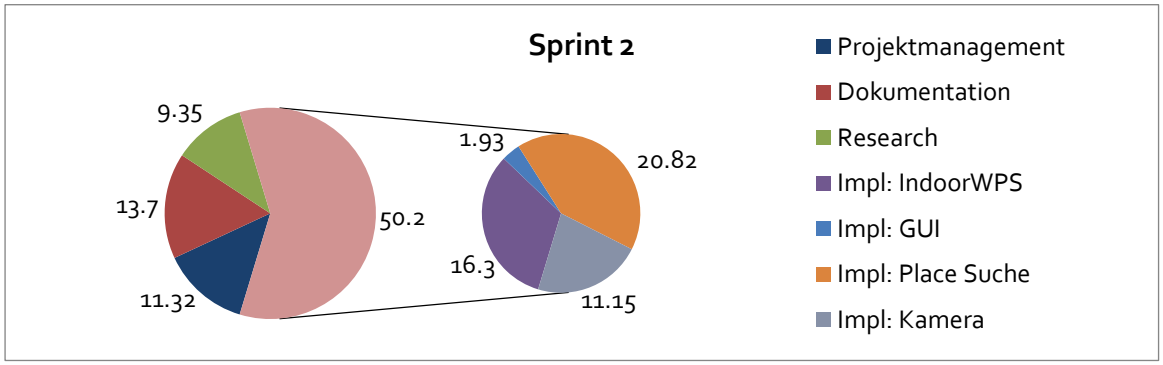


Bild 19: Arbeitspaket - Sprint 2

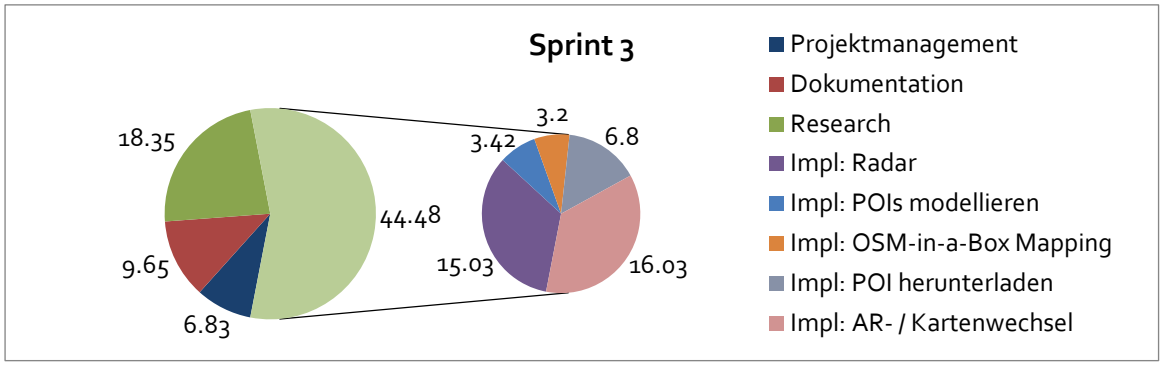


Bild 20: Arbeitspaket - Sprint 3

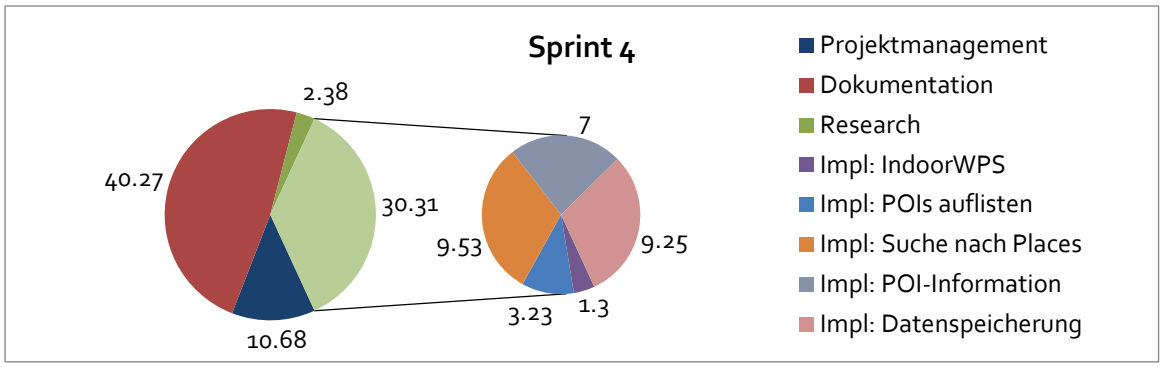


Bild 21: Arbeitspaket - Sprint 4

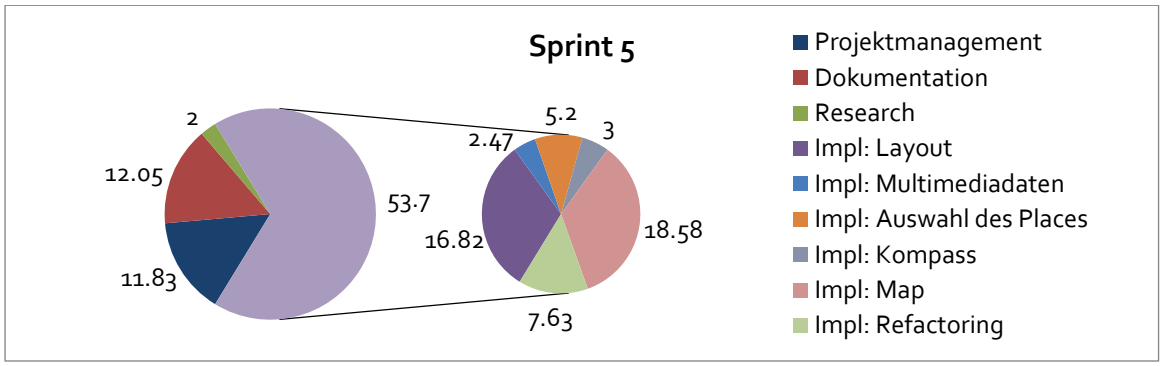


Bild 22: Arbeitspaket - Sprint 5

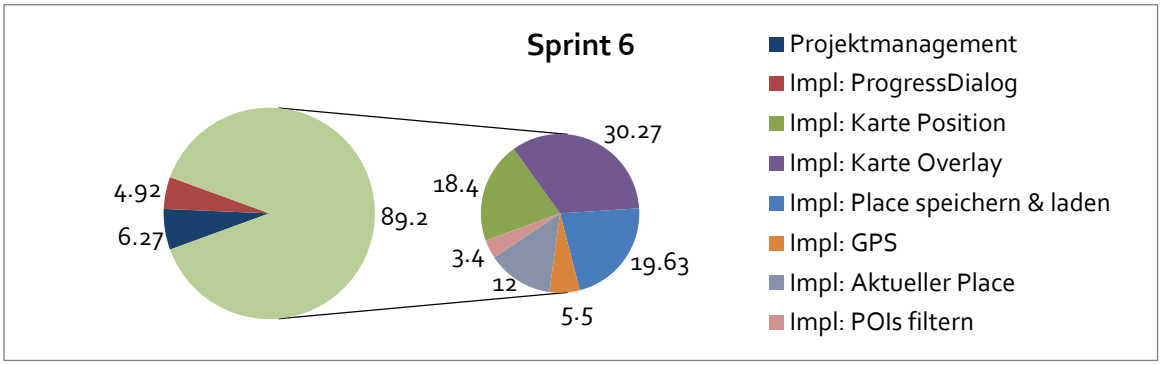


Bild 23: Arbeitspaket - Sprint 6

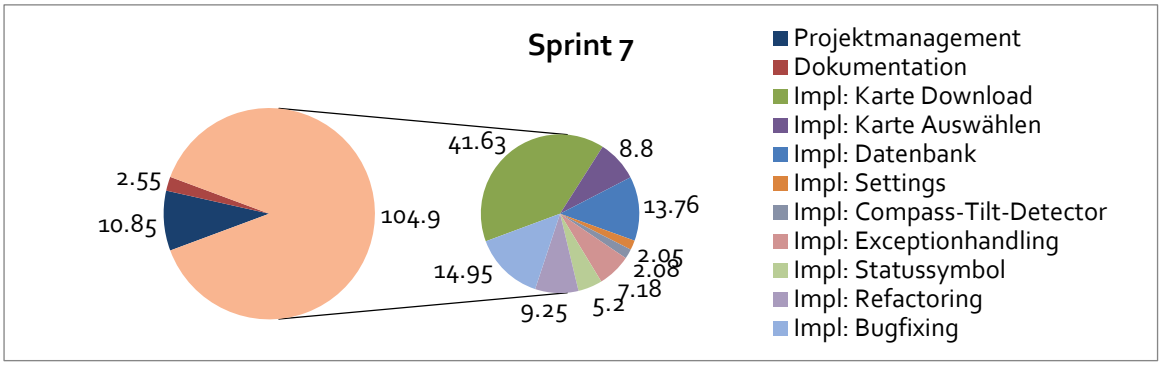


Bild 24: Arbeitspaket - Sprint 7

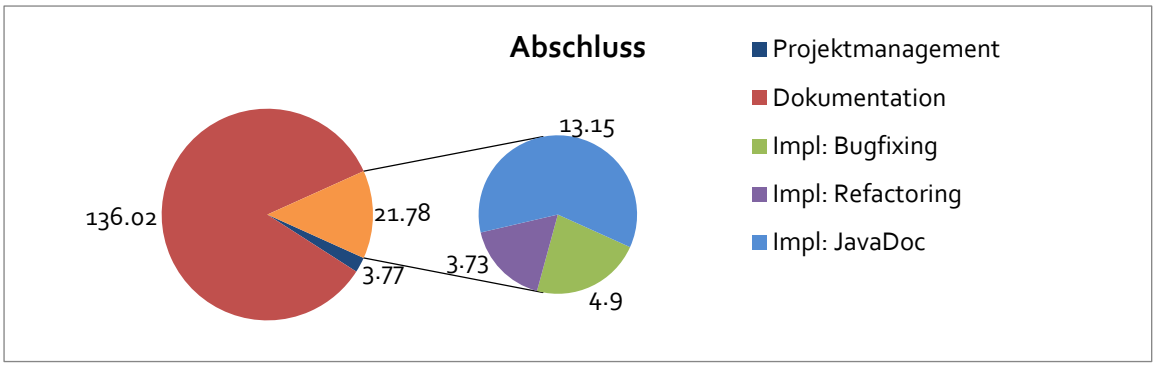


Bild 25: Arbeitspaket - Abschluss

6. Risiko Management

ID	Risiko	Massnahmen	Max. Schaden [h]	W'keit	Gewichtung [h]
R01	Ausfall eines Teammitglieds (Krankheit / Unfall)	Kontakt mit infizierten Personen vermeiden. Keine erhöhten gesundheitlichen Risiken eingehen.	50	20 %	10
R02	Team Probleme (Streit / Uneinigkeit)	Sich regelmässig nach Wohlbefinden erkundigen. Rücksicht auf die Meinung des anderen nehmen.	40	5 %	2
R03	Fehlende Motivation (-> reduzierte Leistungsfähigkeit)	Sich regelmässig nach Motivation erkundigen. Bei Unzufriedenheit mit gegenwärtigen Aufgaben nach Möglichkeit andere Themen zuweisen.	60	10 %	6
R04	Geringer Fortschritt (Entwicklung ist aufwändiger als erwartet)	Ziele eines Sprints reduzieren – Qualität vor Quantität.	80	15 %	12
R05	Ausfall Hardware	Redundanz (bei Ausfall von Schulrechner kann auf Notebook gewechselt werden, Handys sind zwei vorhanden).	10	5 %	0.5
R06	Datenverlust	Verwenden von Subversion und Dropbox. Daten zusätzlich lokal auf mehreren PCs.	100	5 %	5
R07	Höhere Gewalt (Katastrophe führt zu Datenverlust oder Arbeitsverhinderung)	Keine Massnahme möglich.	100	0.01 %	0.1
R08	Gewählte Technologie kann nicht verwendet werden	Evaluation durchführen, bei der die Brauchbarkeit der Technologie gut abgeklärt wird. Alternativen suchen.	60	15 %	9
R09	Mangelnde Softwarequalität	Testing, Reviews und Besprechungen mit Betreuer durchführen.	30	10 %	3
R10	Fehlentwicklung eines Kernelements	Fundierte Analyse und Besprechung der Kernelemente. Feedback von Betreuer einholen.	80	5 %	4
				Total Reserve:	**51.6**

7. Qualitätsmassnahmen

7.1 Sitzungsprotokolle

Bei Besprechungen mit dem Betreuer oder externen Kontaktpersonen, wird ein schriftliches Protokoll geführt. Dadurch sind Überlegungen und Entscheidungen für späteren Gebrauch und Nachvollziehbarkeit festgehalten. Die Sitzungsprotokolle sind auf der beigelegten CD zu finden.

7.2 Iteratives Vorgehen

Bei Scrum wird in Sprints (Iterationen) entwickelt. Diese Sprints dauern für dieses Projekt jeweils zwei Wochen. Nach einem Sprint werden die Ergebnisse besprochen und anhand der Rückmeldung die weiteren Aufgaben für den nächsten Sprint definiert.

7.3 Einsatz eines Versionierungssystems (SVN)

Für die Entwicklung des Programmcodes wurde ein SVN-Repository auf einem Server des IFS verwendet. Durch den Einsatz eines Versionierungssystems ist es allen beteiligten Personen jederzeit möglich, auf die aktuellen Daten zuzugreifen. Zudem hat man Zugriff auf ältere Versionsstände des Projekts. Ferner werden auch durch Unaufmerksamkeit oder andere Einflüsse verursachte Datenverluste reduziert.

7.4 ScrumDesk

Um die Projektfortschritte zu kontrollieren, wurde das Programm *ScrumDesk* eingesetzt. Dieses diente ebenso zur Planung der Sprints. Es ermöglichte, dass alle Teammitglieder, Entwickler wie auch der Betreuer (Product Owner) den aktuellen Stand des Projektes einsehen konnten.

7.5 JUnit-Tests

Um die Code-Qualität zu gewährleisten, resp. die Ausführbarkeit und Korrektheit, wurden JUnit-Tests implementiert, welche jeweils vor dem Commit ins SVN durchlaufen werden.

7.6 Programmierrichtlinien

Damit die Weiterentwicklung und Wartung des Codes vereinfacht wird, wurde die standardmässige Java-Formatierung von Eclipse verwendet. Ebenfalls wurde auf einheitliche und verständliche Namensgebung geachtet.

8. Arbeitsumgebung

Siehe Kapitel „1.3 Rahmenbedingungen" im Teil I.

TEIL III – SW-Projektdokumentation

1. Vision

Siehe Kapitel „2. Vision und Ziele" im Teil I.

2. Anforderungsspezifikation

2.1 Funktionale Anforderungen

2.1.1 Übersicht

ID	Anforderung	Priorität
F01	Die Position des Benutzers soll zuverlässig bestimmt werden.	sehr hoch
F02	Ein Place soll über eine Suchfunktion gefunden werden.	hoch
F03	Daten sollen im Voraus heruntergeladen werden können.	sehr hoch
F04	Die heruntergeladenen Daten sollen persistiert werden.	hoch
F05	Beim Starten der Applikation soll der zuletzt geladene Place geöffnet werden.	mittel
F06	Zwischen AR- und Kartenansicht soll gewechselt werden können.	hoch
F07	POIs sollen im Kamerabild sowie mittels Radar und auf der Karte angezeigt werden.	sehr hoch
F08	Die POIs sollen aufgrund ihrer realen Position angezeigt werden.	sehr hoch
F09	POIs sollen gefiltert angezeigt werden.	mittel
F10	Die Kartenansicht soll einen Plan des Gebäudegrundrisses zeigen.	sehr hoch
F11	Die Karten sollen je nach Ort, an dem sich der Benutzer befindet, gewechselt werden.	hoch
F12	Die eigene Position soll in der Kartenansicht angezeigt werden.	sehr hoch
F13	Die Karte soll sich automatisch ausrichten.	niedrig
F14	POI-Informationen sollen angezeigt werden können.	sehr hoch
F15	Multimediainhalte / Websites eines POIs sollen geöffnet werden können.	hoch
F16	Ein Kompass soll die Richtung angeben.	mittel
F17	Lokalisierung soll ausserhalb des Gebäudes mit GPS gemacht werden.	mittel
F18	Der Benutzer soll die Applikation nach seinen Bedürfnissen konfigurieren können.	niedrig

2.1.2 Detailiert

ID	Detailbeschreibung
F01	Der *IndoorGuide* benutzt für die Bestimmung seiner Position die vom IFS entwickelte *IndoorWPS*-Library. Das *IndoorWPS* lokalisiert die Position des Handys anhand von Fingerprints, welche bereits erfasst sind. Ein Fingerprint ist eine Momentaufnahme der WLAN-Umgebung. An einer bestimmten Position, wird später dieser Fingerprint wieder erkannt und liefert die aktuelle Position. Diese Lokalisierung soll zuverlässig funktionieren und eine genügend genaue Position liefern.
F02	Der Benutzer soll den Place auswählen können, von dem er die Daten herunterladen will. Dazu soll eine Suchfunktion implementiert werden, mit der nach Orten und POIs (z.B. Hochschule Rapperswil) gesucht werden kann.
F03	Eines der Konzepte des *IndoorGuides* geht davon aus, dass der Benutzer nicht permanent mit dem Internet verbunden ist. Aus diesem Grund muss es möglich sein, im Voraus Daten über den gewünschten Place herunterzuladen. Es soll also von zu Hause aus nach einem Place gesucht werden und die Daten dazu heruntergeladen werden können.
F04	Die heruntergeladenen Daten sollen auf dem Gerät persistent gespeichert werden. Bei einem erneuten Applikationsstart sollen die Daten nicht neu heruntergeladen werden müssen.
F05	Beim Starten des *IndoorGuides* soll der zuletzt geladene Place erscheinen. Der Zustand beim beenden soll also persistiert werden.
F06	Der *IndoorGuide* soll die Fähigkeit haben, auf eine intelligente Art und Weise zwischen der AR- und der Kartenansicht zu wechseln. Realisiert werden soll dies mittels der Informationen des internen Winkelsensors. Es soll eine Möglichkeit gefunden werden, die je nach Neigung des Handys die Ansicht wechselt. Falls das Handy horizontal liegt, soll die Kartenansicht und im gegenteiligen Fall die AR-Ansicht angezeigt werden.
F07	Der *IndoorGuide* soll aufgrund der von ihm ermittelten Daten die POIs stetig in der AR- und der Kartenansicht anzeigen können. Dabei sollen diese auf eine übersichtliche und praktische Art dargestellt werden. Die Kartenansicht soll die POIs ebenfalls anzeigen.
F08	In der AR-Ansicht sollen die POIs in einem Radar angezeigt werden. Dieser soll die Position der POIs im richtigen Abstand und Winkel zueinander anzeigen. In der Kartenansicht sollen die POIs an ihrer gegebenen realen Position angezeigt werden.
F09	Damit nicht übermässig viele POIs angezeigt werden, sollen sie sinnvoll gefiltert werden. Der Benutzer soll die Übersicht über viele POIs nicht verlieren.
F10	Bei der Kartenansicht soll ein Plan eines Gebäudegrundrisses angezeigt werden. Da keine offiziellen Karten von Gebäuden existieren, sollen diese als georeferenzierte Bilder im *IndoorGuide* angezeigt werden.
F11	Die Karte soll passend zum Gebäude, in dem man sich befindet, angezeigt werden. Bei mehreren Karten (z.B. mehrere Stockwerke, Überschneidungen von Karten) soll die richtige Karte angezeigt werden.

F12	Der *IndoorGuide* soll auf der Karte die eigene Position so genau wie möglich anzeigen. Es gibt bei der Messung der Position mittels *IndoorWPS* immer eine gewisse Lokalisierungsungenauigkeit. Diese Ungenauigkeit soll durch einen sich dynamisch ändernden Kreis um die eigene Position herum angezeigt werden.
F13	Damit der Benutzer sich mit der Karte besser zurechtfindet, soll sich diese anhand des Kompasses ausrichten.
F14	Um dem Benutzer Informationen über die POIs anzeigen zu können, sollen diese beim Anwählen preisgegeben werden. Es soll auch die Zusatzinformation der Distanz angezeigt werden.
F15	Die POIs sollen mit Links zu Multimediainhalten und Websites versehen werden können. Diese sollen im *IndoorGuide* geöffnet werden können.
F16	In der AR-Ansicht soll ein Kompass Norden anzeigen um dem Benutzer eine gewisse Orientierung zu geben.
F17	Da an den meisten Orten keine Lokalisierung über WLAN möglich ist, soll es möglich sein, dort auf GPS zurückzugreifen. Es soll automatisch der beste Dienst (GPS / WLAN) ausgewählt werden.
F18	Dem Benutzer soll es möglich sein, gewisse Einstellungen in der Applikation selbst vornehmen zu können. Zum einen muss sicher GPS, zum anderen Elemente im User Interface ausgeschaltet werden können.

2.2 Nicht-funktionale Anforderungen

ID	Anforderung	Priorität
NF01	**User Interface** Das User Interface soll verständlich und effizient zu bedienen sein. Der Benutzer soll sich gut zurechtfinden und es sollen die üblichen Bedienelemente von Android eingesetzt werden. Um dies zu erreichen sollen folgende Punkte beachtet werden: 1. Buttons, Listeneinträge oder Objekte müssen generell gross genug sein, damit eine zufriedenstellende Benutzbarkeit entsteht. 2. Die diversen Abläufe sollen logisch aufgebaut sein und einen roten Faden bilden. Der Benutzer soll nichts suchen müssen. 3. Die Fehlermeldungen oder Informationen des *IndoorGuides* sollen immer klar verständlich und sinnvoll sein.	sehr hoch
NF02	**Dokumentation** Die Dokumentation soll klar und verständlich verfasst sein. Sie soll alle geforderten Inhalte beinhalten und soll logisch aufgebaut sein. Die Dokumentation im Code soll auf Englisch, der Rest auf Deutsch verfasst sein. Alle wesentlichen und notwendigen Entscheidungen sollen in der Dokumentation vorhanden sein.	sehr hoch
NF03	**Testing** JUnit-Tests sollen während dem Entwickeln des stetig aktuell gehalten und bei Änderungen laufen gelassen werden. Am Schluss soll ein Systemtest durchgeführt werden, der die ganze Funktionalität testet.	hoch

NF04	**Teamgeist** Es soll darauf geachtet werden, dass stets ein guter und motivierter Teamgeist vorhanden ist. Die Arbeiten sollen gerecht zwischen den Teammitgliedern aufgeteilt werden.	hoch
NF05	**Meetings** Es sollen regelmässig (wöchentlich) Meetings abgehalten werden, die zur Fortschritts- und Qualitätssicherung dienen sollen. Erarbeitetes und Neues soll besprochen und von allen Betroffenen abgenommen werden.	hoch
NF06	**Code** Der Code sollte nicht unnötig aufgeblasen werden. Es soll ein sauberer und effizienter Codierstil benutzt werden. Die JavaDoc soll vollständig und in Englisch verfasst werden.	sehr hoch

2.3 Use Cases

Bei den Use Cases gehen wir grundsätzlich davon aus, dass WLAN aktiviert ist und benutzt werden kann. Ebenfalls wird davon ausgegangen, dass *OSM*, *OSM-in-a-Box* und *IndoorWPS* ihre Dienste funktionstüchtig anbieten und die Software korrekt installiert wurde.

2.3.1 Use Case Diagram

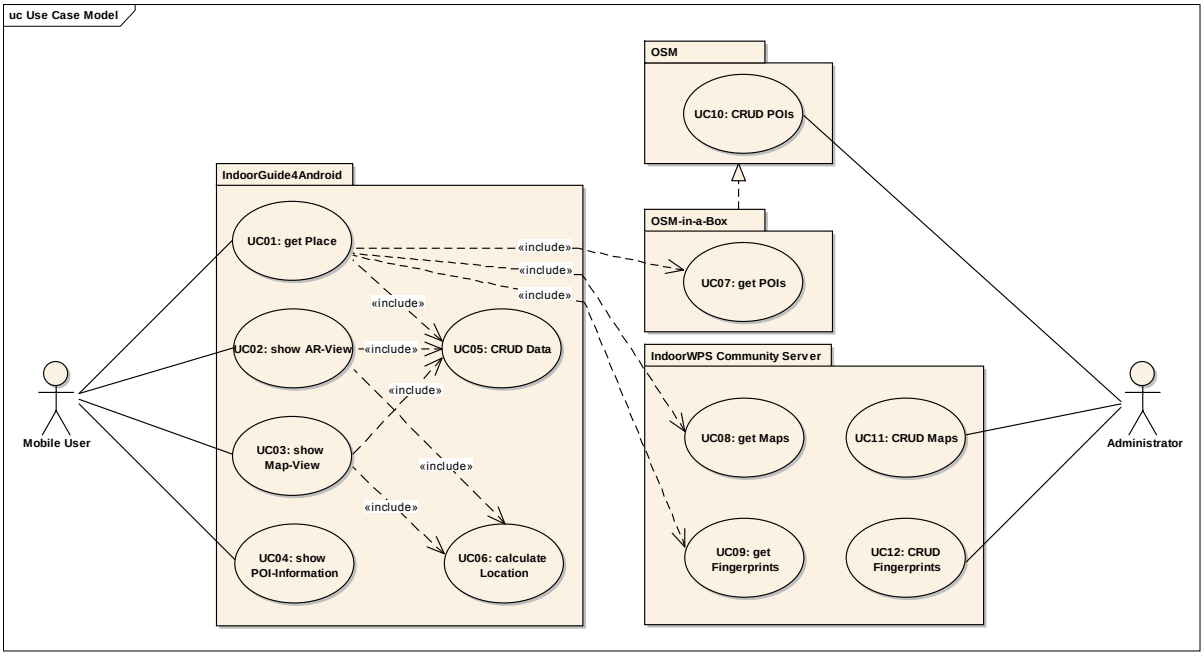


Bild 26: Use Case Diagram

2.3.2 Aktoren

Name	Beschreibung
Administrator	Verwaltet die Daten und sorgt für die Funktionalität der Libraries, welche der *IndoorGuide* benutzt.
Mobile User	Anwender des *IndoorGuides4Android*

2.3.3 Beschreibungen der Use Cases

UC01	get Place
Vorbedingung: Internetverbindung ist möglich.	
Der Benutzer möchte den *IndoorGuide* an einer Destination verwenden und muss dazu diesen Use Case ausführen. Der Use Case ist nötig um überhaupt die notwendigen Daten laden zu können. Er wird via Menü ausgeführt und bietet die Möglichkeit nach einer gewünschten Destination per Namen zu suchen. Ebenfalls ist es möglich, automatisch die richtige Destination zu finden (anhand der Koordinate). Hat der Benutzer seine gewünschte Destination gefunden, kann er diese anklicken, worauf der *IndoorGuide* den **UC07: get POIs**, **UC08: get Maps** und **UC09: get Fingerprints** startet. Um die Daten in der lokalen Datenbank zu speichern führt der *IndoorGuide* **UC05: CRUD Data** aus. Beim Anzeigen und öffnen von schon gespeicherten Places führt der *IndoorGuide* den **UC05: get Data** aus.	

UC02	show AR-View
Vorbedingung: -	
Die Augmented Reality Ansicht kann vom Benutzer jederzeit angezeigt werden, indem er das Handy senkrecht hält. Falls in seiner Umgebung geladene POIs vorhanden sind, werden diese angezeigt. Ansonsten sieht der Benutzer eine leere Liste und einen leeren Radar. Die POIs werden über den **UC05: CRUD Data** beschafft. Für die Funktionalität dieses Use Cases ist er auf den **UC06: calculate Location** angewiesen, welcher ihm permanent die aktuelle Position anhand der Messungen von *IndoorWPS* oder GPS liefert.	

UC03	show Map-View
Vorbedingung: -	
Die Kartenansicht kann vom Benutzer jederzeit angezeigt werden, indem er das Handy horizontal hält. Falls in seiner Umgebung POIs vorhanden sind, werden diese auf der Karte angezeigt. Ansonsten sieht der Benutzer eine leere Karte oder ein Standardbild, wenn keine Karte vorhanden ist. Die POIs werden über den **UC05: CRUD Data** beschafft. Für die Funktionalität dieses Use Cases ist er auf den **UC06: get Location** angewiesen, welcher ihm permanent die aktuelle Position anhand der Messungen von *IndoorWPS* oder GPS liefert.	

UC04	show POI-Information
Vorbedingung: UC01: get Place, UC02: show AR-View, UC03: show Map-View	
Der Benutzer kann diesen Use Case ausführen, wenn er sich zusätzliche Informationen zu einem POI ansehen möchte. Mittels antippen eines POIs auf dem Display, wird ein Fenster geöffnet, worin alle vorhandenen Informationen zum angewählten POI dargestellt werden.	

UC05	CRUD Data
Vorbedingung: -	
Dieser Use Case wird vom IndoorGuide benutzt und verwaltet die Daten in der lokalen Datenbank. Entweder werden die Daten erstellt, gelesen, aktualisiert oder gelöscht.	

UC06	calculate Location
Vorbedingung: *IndoorWPS* oder GPS funktioniert	
Dieser Use Case wird vom *IndoorGuide* benutzt und liefert kontinuierlich den aktuellen Standort des Benutzers in Form von Koordinaten. Diese Koordinate wird entweder vom *IndoorWPS* oder GPS empfangen. Die Auswahl zwischen diesen zwei Diensten hängt von der Konfiguration und dem Empfang ab.	

UC07	get POIs
Vorbedingung: UC01: get Place	
Dieser Use Case wird vom *IndoorGuide* gestartet und liefert aufgrund der eingegebenen Destination alle passenden POIs vom *OSM-in-a-Box Server* zurück.	

UC08	get Maps
Vorbedingung: UC01: get Place	
Dieser Use Case wird vom *IndoorGuide* gestartet und liefert aufgrund der eingegebenen Destination alle passenden Maps vom *IndoorWPS Community Server* zurück.	

UC09	get Fingerprints
Vorbedingung: UC01: get Place	
Dieser Use Case wird vom *IndoorGuide* gestartet und liefert aufgrund der eingegebenen Destination alle passenden Fingerprints vom *IndoorWPS Community Server* zurück.	

UC10	CRUD POIs
Vorbedingung: *OSM Server* ist in Betrieb	
Der Administrator kann mittels dieses Use Cases die POIs in *OSM* direkt bearbeiten und entsprechend für den *IndoorGuide* verfügbar machen.	

UC11	CRUD Maps
Vorbedingung: *IndoorWPS Community Server* ist in Betrieb	
Der Administrator kann mittels dieses Use Cases die Maps auf dem *IndoorWPS Community Server* direkt bearbeiten und entsprechend für den *IndoorGuide* verfügbar machen.	

UC12	CRUD Fingerprints
Vorbedingung: *IndoorWPS Community Server* ist in Betrieb.	
Der Administrator kann mittels dieses Use Cases die Fingerprints auf dem *IndoorWPS Community Server* direkt bearbeiten und entsprechend für den *IndoorGuide* verfügbar machen.	

3. Analyse

3.1 OSM POI-Modell

Damit die auf OSM definierten Indoor-POIs beim Transfer auf den OSM-in-a-Box Server in die richtige Tabelle gespeichert werden, müssen sie nach folgendem Muster erfasst werden.

	Tag	Beschreibung	Beispiel
Standard-Tags	name = <String>	Obligatorischer Tag für die Benennung des POIs.	„1.258"
	description = <String>	Empfohlener Tag für die Beschreibung des POIs.	„Classroom 1.258" oder Mensa-Menüplan
Indoor-Tags	indoor = yes	Obligatorischer Tag, der den POI als Indoor-POI bezeichnet. Ohne diesen Tag würde der POI den Weg in den IndoorGuide nicht finden.	„yes"
	level = <String>	Obligatorischer Tag, der das Stockwerk beschreibt, in welcher der POI liegt.	„1" für 1. Stockwerk
Info-Tags	website = <URL>	Optionaler Tag für das Speichern eines Links zu einer Website.	„http://www.hsr.ch"
	wikipedia = <URL>	Optionaler Tag für das Speichern eines Links zu einem Wikipedia-Artikel.	„http://www.hsr.ch"
	image_url = <URL>	Optionaler Tag für das Speichern eines Links zu einem Bild (oder mehreren).	„http://www.hsr.ch"
	video_url = <URL>	Optionaler Tag für das Speichern eines Links zu einem Video (z.B. auf YouTube).	„http://www.hsr.ch"
	audio_url = <URL>	Optionaler Tag für das Speichern eines Links zu einer Audiodatei.	„http://www.hsr.ch"

Um der POI einem Gebäude zuordnen zu können, wird eine Relation gebraucht. Diese muss nach folgendem Muster erfasst werden.

Tag	Beschreibung	Beispiel
name = <String>	Obligatorischer Tag für die Benennung der Relation. Es soll der Gebäudename dafür verwendet werden.	„HSR Gebäude 1"
type = building	Obligatorischer Tag, welcher bei allen Relationen zwingend gesetzt werden muss.	„building"

Beispiel eines solchen POIs:

```
<node id='411729141' timestamp='2010-04-01T08:43:36Z' uid='254788'
  user='ageiter' visible='true' version='4'
  lat='47.2233969' lon='8.8172783'>
     <tag k='name' v='1.258' />
     <tag k='description' v='Classroom 1.258' />
     <tag k='indoor' v='yes' />
     <tag k='level' v='2' />
     <tag k='website'

v='http://wiki.hsr.ch/StefanKeller/wiki.cgi?IndoorGuide4Android2' />
     <tag k='image_url'
       v='http://upload.wikimedia.org/wikipedia/commons/8/86/
       HSR_See_Geb6.jpg' />
     <tag k='video_url'
       v='http://www.youtube.com/watch?v=OSQlsLzK4qs&amp;NR=1' />
     <tag k='audio_url'
       v='http://www.mp3.ch/no_cache/music/download/sid/2063/
       ?tx_musicgate_pi1%5Bgo%5D=download&amp;plus=&amp;next=' />
</node>
```

Beispiel einer solchen Relation:

```
<relation id='659706' timestamp='2010-04-27T08:56:27Z' visible='true'
  version='1'>
     <tag k='name' v='HSR GebÃ¤ude 1' />
     <tag k='type' v='building' />
     <member type='node' ref='411729126' role='' />
     <member type='node' ref='411729127' role='' />
     <member type='node' ref='411729129' role='' />
     <member type='node' ref='411729131' role='' />
     <member type='node' ref='411729132' role='' />
     <member type='node' ref='411729134' role='' />
     <member type='node' ref='411729141' role='' />
</relation>
```

3.2 Datenbank-Modell

Hier wird das Datenbankschema (welches sehr einfach strukturiert ist) aufgezeigt.

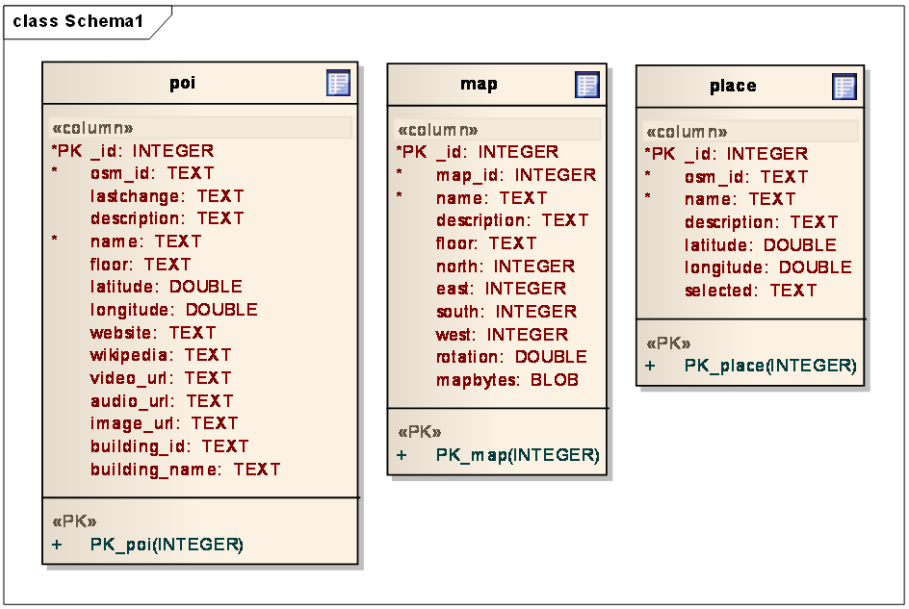


Bild 27: Datenbank-Modell

3.3 Android Lifecycle-Model

Bei Android ist etwas Wichtiges und grundlegendes der Activity Lifecycle[15]. Er wird in folgendem Bild illustriert.

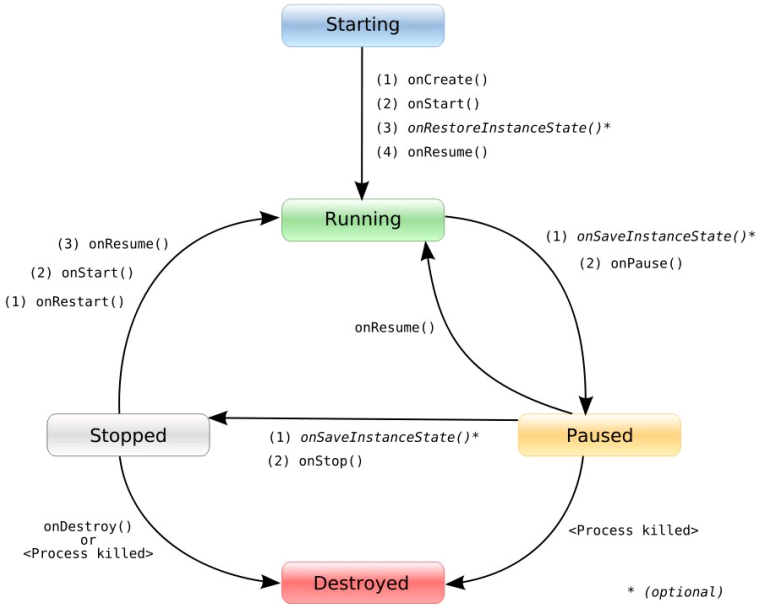


Bild 28: Activity Lifecycle

[15] Quelle Activity Lifecycle: Hello, Android 2.1, Seite 24

4. Design

4.1 Architecture

The IndoorGuide4Android is designed as a 3 tier architecture. The application doesn't need any special design in its architecture. A persistence layer with common POJOs is accessed from the higher level, the problem domain layer. The problem domain layer itself is used by the gui/view layer on top. The top layer is able to access the persistence layer directly, the so called transparent layer system. That's why sometimes it doesn't make sense to make a problem domain call just to receive a POJO. But the accessing from layer to layer is strict from the top to the bottom and never inverse. This would lead into cycles between classes which results in a bad design that is hard to use for future development or enhancements.

4.2 Logical Architecture

4.2.1 Introduction

In this chapter all the packages and sub packages are shown. It describes every class contained in the package and the relations between them. The classes are already reduced to the not self-explanatory methods and variables due to readability and clarity. Following the core functionality of the classes and their key methods are explained. For fully commented java source code please have a look at the generated JavaDoc.

4.2.2 Package business

4.2.2.1 *Description*

This package shows the business classes. They are the classes which are used from higher level classes and deal with lower level classes. They manage the providing of the data needed by the application.

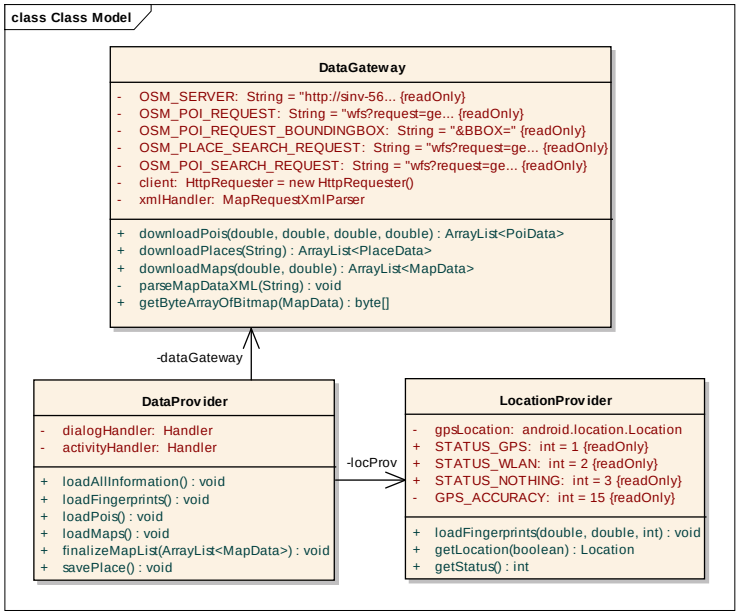


Bild 29: Package business

4.2.2.2 Classes and methods

DataGateway	
This Class implements the methods for all kind of data downloading used in the IndoorGuide4Android Project.	
Methods:	
downloadMaps(double, double)	Downloads all the `MapData` objects from the IndoorWPS project server which can be found based on the given parameters. The IndoorWPS server calculates a bounding box with 1 kilometer radius based on the given latitude longitude coordinates. The used data-interchange format is XML. This method just gets the additional information about the map file ("name", "id", "description", "upper right" (`GeoPoint`), "lower left" (`GeoPoint`), "rotation") and not yet the graphic data because the XML just contains a link to the graphics path on the IndoorWPS server.
downloadPlaces(String)	Downloads all `PlaceData` objects from the OSM-in-a-Box server which can be found based on the given parameter. The used data-interchange format is JSON. The method does two search requests on the OSM-in-a-Box server one for the WFS Feature place_lookup and a second for the WFS Feature poi_lookup.
downloadPois(double, double, double, double)	Downloads all `PoiData` objects from the OSM-in-a-Box server contained in the provided bounding box. The bounding box is calculated on the server based on the given parameters. The used data-interchange format is JSON.
getByteArrayOfBitmap(MapData)	Downloads the `byte[]` (Graphic-Data) of the given `MapData` from the IndoorWPS server.
parseMapDataXML(String)	Executes the `SAXParser` and parses the given map XML string.

DataProvider	
This class is responsible for delegating the whole download and saving process.	
Methods:	
loadAllInformation()	Executes a thread which is created in this method. Without starting a thread the application would block. The thread activates the loading/saving of the whole data available to the `PlaceData` object which was provided to the `DataProvider` in the constructor. The thread sends updates in progress to the handlers.
loadFingerprints()	Delegates the loading of the `Fingerprint` objects.
loadPois()	Delegates the loading/saving of the `PoiData` objects.
loadMaps()	Delegates the loading/saving of the `MapData` objects.

finalizeMapList(ArrayList <MapData>)	Sets the `BoundingBoxE6` depending on the parsed upper right and lower left `GeoPoint` coordinate. Finally delegates the download of the graphics data.
savePlace()	Delegates the saving of the current loaded `PlaceData` object.

LocationProvider	
This class manages the GPS functionality and the functionality of the IndoorWPS library. Based on the GPS and WLAN (IndoorWPS library) positioning systems this class provides a `Location` or `GPSLocation` object.	
Methods:	
loadFingerprints(double, double, int)	Delegates the loading of the `Fingerprint` objects to the IndoorWPS library. The `FingerprintWsClient` loads the `Fingerprint` objects contained in the bounding box (calculated from params) and saves them to the database.
getLocation(boolean)	This method provides a GPS or a WLAN location object. If GPS is enabled the returned location will be an GPS location, thats because GPS is more accurate then WLAN positioning.

getStatus()	Returns the actual status of the `LocationProvider`
	`STATUS_GPS`
	`STATUS_WLAN`
	`STATUS_NOTHING`

4.2.3 Package business.data

4.2.3.1 Description

This package shows the DTOs and their manager classes. The DTOs are valid through the whole application and can be accessed from everywhere via the manager classes.

The DTOs are POJOs and won't be described nearer in this chapter, just one thing `PlaceData` and `PoiData` implements `Serializable` because they need to be passed between `Activities`.

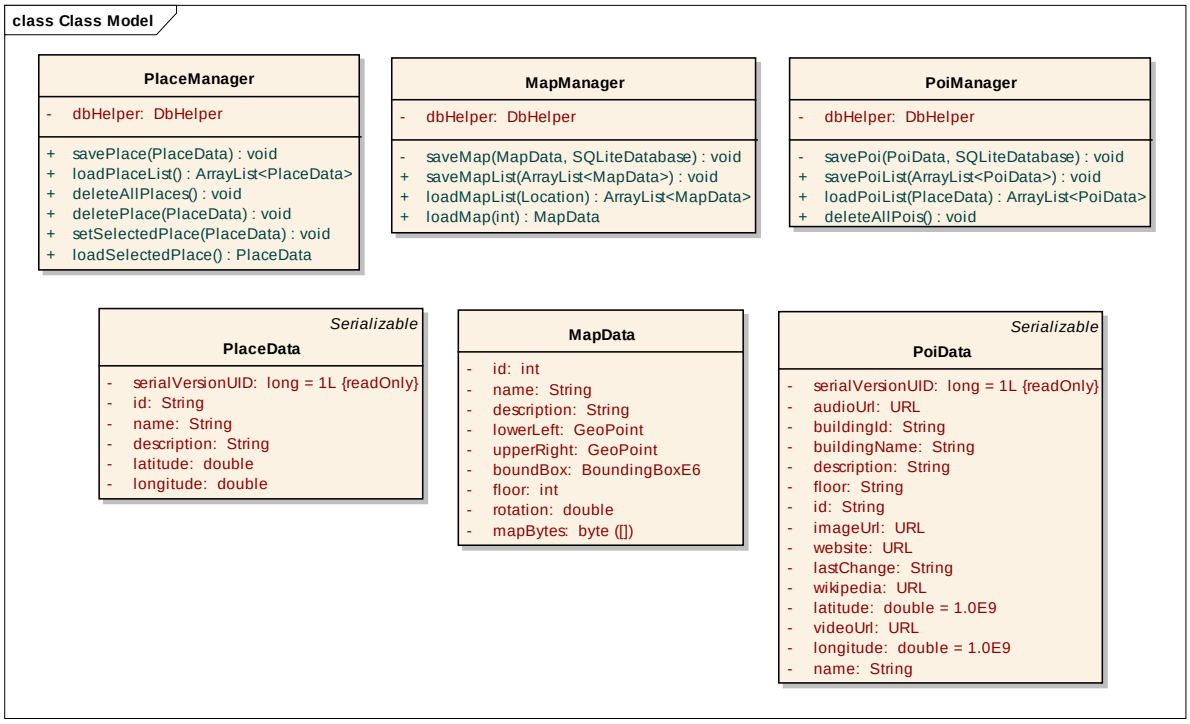


Bild 30: Package business.data

4.2.3.2 *Classes and methods*

PlaceManager	
This is the manager class of the `PlaceData` objects. The class is responsible for loading and saving the `PlaceData` objects to the database. This class also takes care of the currently selected `PlaceData` object, so it will be already selected on the next startup of the IndoorGuide4Android application.	
Methods:	
loadPlaceList()	Loads all available `PlaceData` objects from the database of the application.
loadSelectedPlace()	Loads the last selected `PlaceData` object on startup.

MapManager	
This is the manager class of the `MapData` objects. The class is responsible for loading and saving the `MapData` objects to the database.	
Methods:	
saveMap(MapData, SQLiteDatabase)	Saves the given `MapData` object to the given `SQLiteDatabase`. This method saves the coordinates latitude longitude span as Integer to the database, because the `BoundingBoxE6` object can't be saved like that.
loadMap(int)	Loads a complete `MapData` object from the database.

PoiManager	
This is the manager class of the `PoiData` objects. The class is responsible for loading and saving the `PoiData` objects to the database.	
Methods:	
savePoi(PoiData, SQLiteDatabase)	Saves the provided `PoiData` object to the `SQLiteDatabase`.
loadPoiList(PlaceData)	Loads all `PoiData` objects whose coordinates are in the circle around the given `PlaceData` object coordinates.

4.2.4 Package business.inet

4.2.4.1 *Description*

This package simply contains one class it's the `HttpRequester` which serves the business classes as a client that establishes connections to the internet.

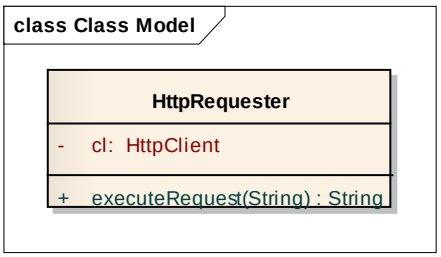


Bild 31: Package business.inet

4.2.4.2 *Classes and methods*

HttpRequester	
This class is a simple `HttpRequester` which is used in the IndoorGuide4Android project for executing HTTP queries.	
Methods:	
executeRequest(java.lang.String uri)	Executes the provided URL and returns the result string to the caller.

4.2.5 Package business.inet.util

4.2.5.1 *Description*

This package contains the utility classes for the business layer. It provides services for parsing JSON and XML strings.

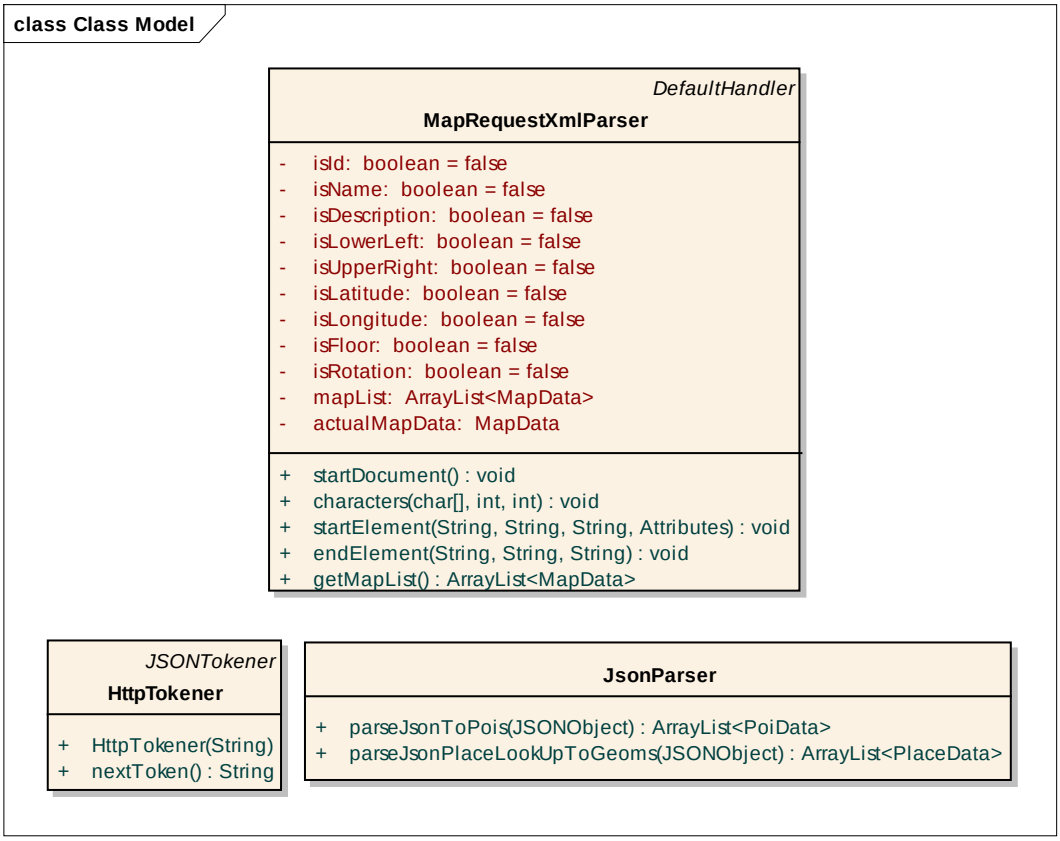


Bild 32: Package business.inet.util

4.2.5.2 Classes and methods

MapRequestXmlParser	
This class is the XML parser for the `MapData` objects provided by the IndoorWPS server.	
Methods:	
getMapList()	Returns an arraylist containing all parsed `MapData` objects.

HttpTokener	
This class extends the `JSONTokener` to provide additional methods for the parsing of HTTP headers.	
Methods:	
HttpTokener(String)	Construct an `HTTPTokener` from a string.
nextToken()	Get the next token or string. This is used in parsing HTTP headers.

JsonParser	
This class parses all kind of `JSONObject` objects which are used in the IndoorGuide4Android. `JSONObject` is the data-interchange format between the OSM-in-a-Box server and the IndoorGuide4Android.	
Methods:	
parseJsonTo-Pois(JSONObject)	Parses the *osm:place_lookup* `JSONObject` objects.
parseJsonPlaceLookUpToGeoms(JSONObject)	Parses the *osm:poi* and *osm:indoor_view* `JSONObject` objects.

4.2.6 Package db

4.2.6.1 Description

This package contains the helper class for accessing the SQLite database. For a proper design all the needed constants in the `DbHelper` class are provided from the `DbConstants` interface.

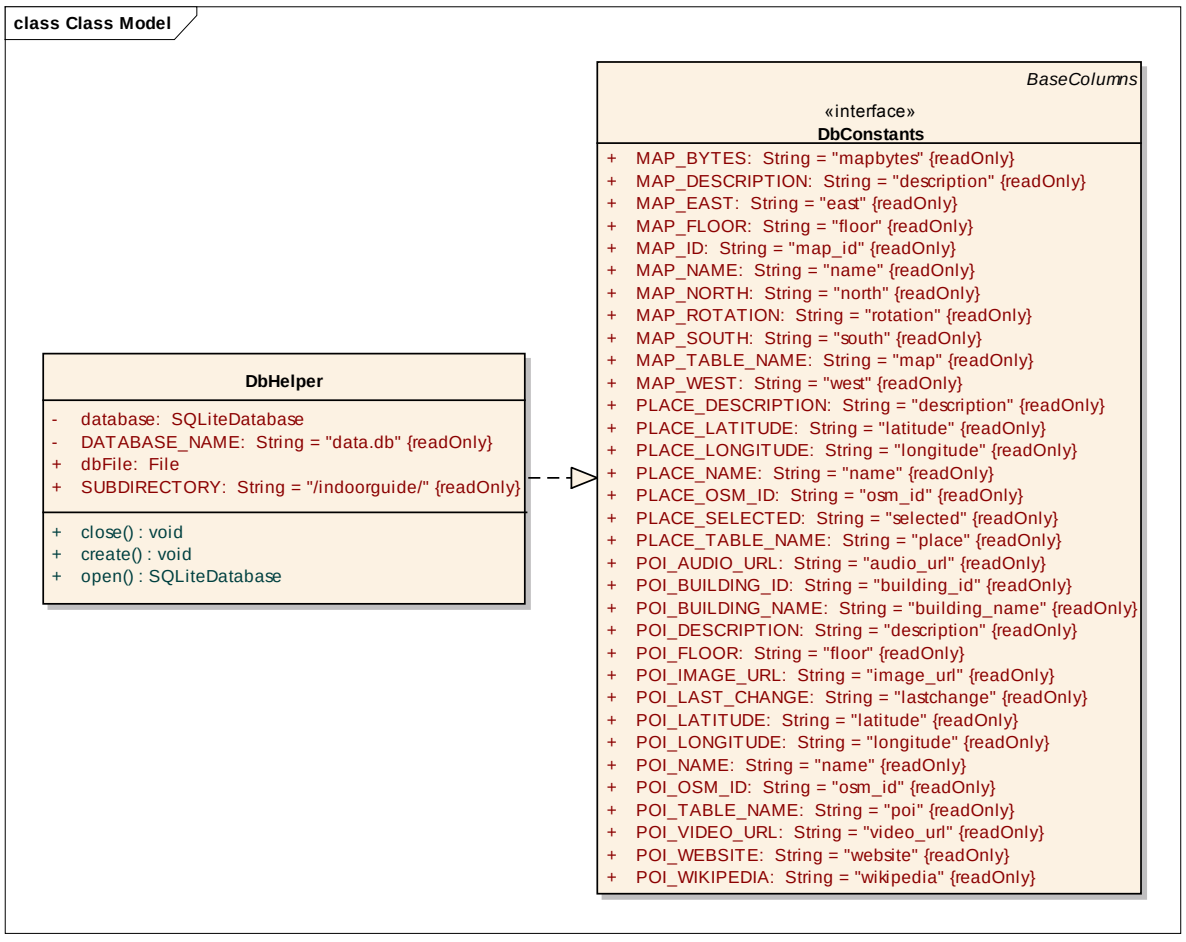


Bild 33: Package db

4.2.6.2 *Classes, interfaces and methods*

DbHelper	
This class is responsible for creating and opening the database used in the IndoorGuide4Android project. Also it takes care of closing the database. It creates all needed tables in the database if they don't already exist. The database is stored on the SD-Card of the smartphone, thats why this class doesn't extend from the existing Android `DbHelper` class. It's not now possible to create the database on SD-Card when using the Android `DbHelper` class.	
Methods:	
close()	Closes the connection to the database.
create()	Creates all the necessary tables.
open()	Checks if the SD-Card is available and creates the db file on the right path if it not already exists.

DbConstants
This class provides all constants used for the IndorGuide4Android database.

4.2.7 Package gui.activity

4.2.7.1 *Description*

This package contains all the activities. The activities provide the context in which everything runs. Activities control the views and provide them with the information they need to work properly.

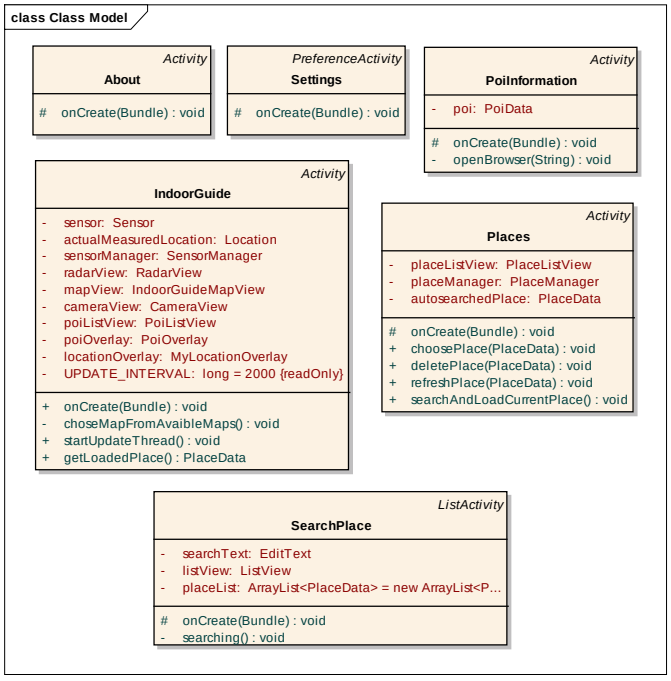


Bild 34: Package gui.activity

4.2.7.2 Classes and methods

IndoorGuide	
This is the main class of the IndoorGuide4Android project. It starts and handles the `CameraView`, `RadarView`, `PoiListView` and the `MapView`. When you run the application this activity is started and leads you to all the functionality of the application.	
Methods:	
choseMapFromAvailableMaps()	Starts the `MapListView` activity which allows you to choose between available `MapData` objects. Available means if the `BoundingBoxE6` of the `MapData` object contains the actual measured location.
startUpdateThread()	Starts the `Updater` thread which provides the IndoorGuide4Android application with the actual measured `Location` object. All views connected to the `IndoorGuide` are updated with the new location.
getLoadedPlace()	Provides the currently loaded `PlaceData` object.

SearchPlace	
This class is for searching a new place. It initializes a search of places in the `DataGateway`. After that, it shows the result in a list. A click on the list will start the download of all data for this place.	
Methods:	
searching()	This method runs in a separate thread and searches for a place (chosen by the user). It uses the `DataGateway` for downloading the list of possible places. Then it shows the result as a list.

Places	
This class shows all saved `PlaceData`. The user can manage his saved places by updating, deleting or choosing them. This class has also e method for automatically searching a location by using the current coordinates.	
Methods:	
choosePlace(PlaceData)	This method chooses a place and returns to the `IndoorGuide`.
deletePlace(PlaceData)	This method deletes a place and will show an empty text, when the place list is empty.
refreshPlace(PlaceData)	This method refreshes a place and loads all information about it.
searchAndLoadCurrentPlace()	This method searches data with the current coordinates. It runs in a separate thread and sends messages to the activity handler of the `Places` activity. If no position could be found in a period of time, it shows a message and returns. If a position could be found, it shows a prompt to label this place and then download all data from this place.

Settings
This class takes care of the settings for the whole IndoorGuide4Android. The settings are defined in the Android "R.xml.settings" XML file. Every setting defined in this XML file is displayed automatically through Android and can be accessed from everywhere in the project to get the status of the settings.

PoiInformation	
This class is responsible to show a clicked `PoiData` object. It shows the whole information and displays the browser links as buttons.	
Methods:	
openBrowser(java.lang.String uri)	Starts the Android browser with the provided URL.

4.2.8 Package gui.util

4.2.8.1 Description

This package contains all the util classes which are use all over the application. They provide functionality like sensor events, geo utilities, constants, dialog handling, read more in the following detail description.

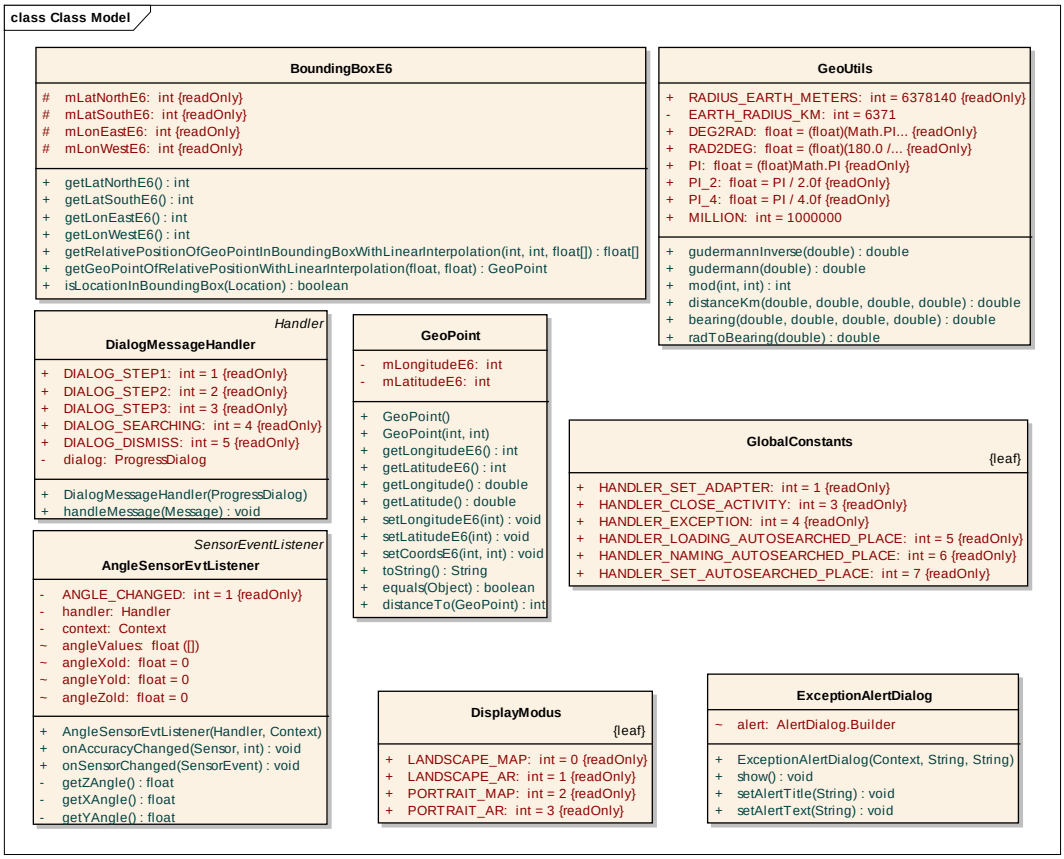


Bild 35: Package gui.util

4.2.8.2 Classes and methods

BoundingBoxE6	
This class represents a bounding box based on geo coordinates. There are methods to calculate from `GeoPoint` to `Point`(x & y on a screen) and inverse. Also some helper methods to deal with geo staff concerning to bounding boxes.	
Methods:	
getRelativePositionOf-GeoPointInBounding-BoxWithLinearInterpola-tion(int, in, float[])	Calculates the relative x and y values for a given latitude longitude (`GeoPoint`). The calculated x, y values represent the relative coordi-nates on the screen proportionally to the size of the bounding box.
getGeoPointOfRelative-PositionWithLinearInter-polation(float, float)	Calculates a `GeoPoint` object based on the parameters.
isLocationInBounding-Box(Location)	Calculates if a provided `Location` object appears in the bounding box.

AngleSensorEvtListener	
This class is responsible for providing the `IndoorGuide` with actual measured sensor events. An event contains the actual measured x, y and z angle value depending on how the user holds the phone. These values are necessary for the `IndoorGuide` to decide which view must be displayed.	
Methods:	
onAccuracy-Changed(Sensor, int)	Implements a compass TILT detector. Will show a message to the user if accuracy is low or no data can be measured at all.
onSensorChanged(Senso rEvent)	`ANGLE_CHANGED` message will be sent if the angle-change is bigger than 30 degrees. Otherwise there would be a bunch of messages sent to the handler which leads into performance lack.

GeoPoint	
This class represents a `GeoPoint` object which is a point in reality with its latitude and longitude.	
Methods:	
distanceTo(GeoPoint)	Calculates the distance between this and the provided `GeoPoint` object.

GeoUtils	
This class contains constants and geo methods used in the IndoorGuide4Android project.	
Methods:	
distanceKm(double, double, double, double)	Calculates the distance in kilometers between two points on earth.
Bearing(double, double, double, double)	Computes the bearing in degrees between two points on earth.

DialogMessageHandler
This class wraps a provided `ProgressDialog` object for taking control over it and be able to set dialog text and also dismiss the dialog.

ExceptionAlertDialog
This class wraps an exception message and stores it in an alert object which then can be shown in a dialog.

4.2.9 Package gui.view

4.2.9.1 Description

This package contains every view used in the application. The views run in the context of an activity.

There are several types of views:

- **View:** This is a simple view that can be instanced by an activity.
- **ListView:** This is a view that must have a list view defined, a list can be set through an adapter. The view will then display the objects of the list.
- **SurfaceView:** This type of view gives you the possibility to draw other views on top of it.
- **ListActivity:** In fact it's an activity that is able to show a list of objects just like the list view but with the additional functionality of an activity.

`MapListView` and `PlaceListView` won't be described because they're really simple list views.

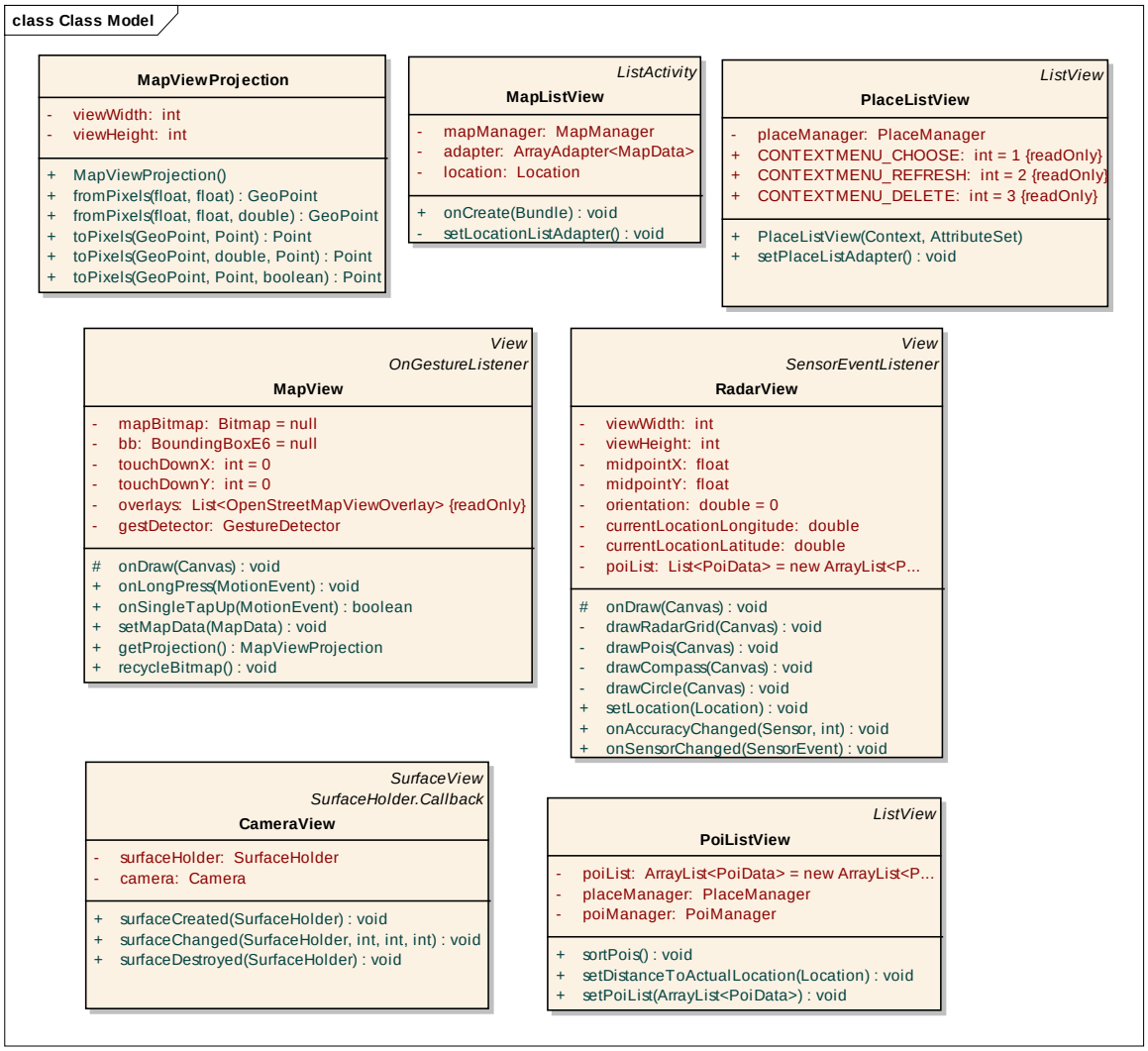


Bild 36: Package gui.view

4.2.9.2 *Classes and methods*

MapView	
This class is the main view for maps. The view displays the provided `MapData` object and manages the drawing of all `MapViewOverlay` objects. Its possible to move the map in every direction.	
Methods:	
onDraw(Canvas)	Draws the view at the correct position. Forces the drawing of all existing overlays.
onSingleTapUp(MotionEvent)	Executes the `onSingleTapUp()` method for all existing overlays. For example this is used to open a POI.
setMapData(MapData)	Decodes the `Bitmap` object from the provided `MapData` object which is used and manipulated by the `MapView`. Sets the `BoundingBoxE6` used for the geo calculating.

MapViewProjection	
This code was copied from OSM Andnav application and manipulated for the needs of the IndoorGuide4Android. This class wraps the whole calculating like from `GeoPoint` to `Point` and inverse. It's an inner class of the `MapView` because this class is just used in combination with the `MapView`.	
Methods:	
fromPixels(float, float)	Calculates a `GeoPoint` object based on the provided x, y values.
fromPixels(float, float, double)	Calculates a `GeoPoint` object with bearing based on the provided x, y values.
toPixels(GeoPoint, Point)	Calculates a `Point` object based on the provided `GeoPoint` object.
toPixels(GeoPoint, double, Point)	Calculates a `Point` object with bearing based on the provided `GeoPoint` object.

CameraView	
This class represents a view which shows what the camera sees. Used for the augmented reality. Provides a dedicated drawing surface embedded inside of a view hierarchy. You can control the format of this surface and, if you like, its size; the `SurfaceView` takes care of placing the surface at the correct location on the screen The surface is Z ordered so that it is behind the window holding its `SurfaceView`; the `SurfaceView` punches a hole in its window to allow its surface to be displayed. The view hierarchy will take care of correctly compositing with the surface any siblings of the `SurfaceView` that would normally appear on top of it. This can be used to place overlays such as buttons on top of the surface	
Methods:	
surfaceCreated(SurfaceHolder)	This is called immediately after the surface is first created. Implementations of this should start up whatever rendering code they desire.
surfaceChanged(SurfaceHolder, int, int, int)	This is called immediately after any structural changes (format or size) have been made to the surface. You should at this point update the imagery in the surface. This method is always called at least once, after `surfaceCreated(SurfaceHolder)`.
surfaceDestroyed(SurfaceHolder)	This is called immediately before a surface is being destroyed. After returning from this call, you should no longer try to access this surface.

RadarView	
This class represents a radar which is displayed on top of the `CameraView`. It contains all the `PoiData` objects which are visible in the given radius, a compass sensor is used to measure north direction, so the compass with its `PoiData` objects always point to north.	
Methods:	
drawRadarGrid(Canvas)	Draws all grids obtained in this class.
drawPois(Canvas)	Calculates the position of each `PoiData` object in the radar. Then draws the object on the radar.
drawCompass(Canvas)	Draws the compass with correct rotation.
onAccuracyChanged(Sensor, int)	Implements a compass TILT detector. Will show a message to the user if accuracy is low or no data can be measured at all.
onSensorChanged(SensorEvent)	Called when sensor values have changed.

PoiListView	
This class represents a list view which is displayed on top of the `CameraView`. All the `PoiData` objects are shown in a scrollable list.	
Methods:	
setDistanceToActualLocation(Location)	Calculates the actual distance from the measured location to each `PoiData` object provided to this class.
setPoiListAdapter()	Sets the adapter for this list view. This will show the `PoiData` objects provided to the adapter.

4.2.10 Package gui.view.overlay

4.2.10.1 Description

This package contains the classes for overlays on the `MapView`. The abstract `MapViewOverlay` can be subclassed to create any overlay you want. The IndoorGuide4Android uses two kinds of overlays one for the `PoiData` and a second for your position (`Location`).

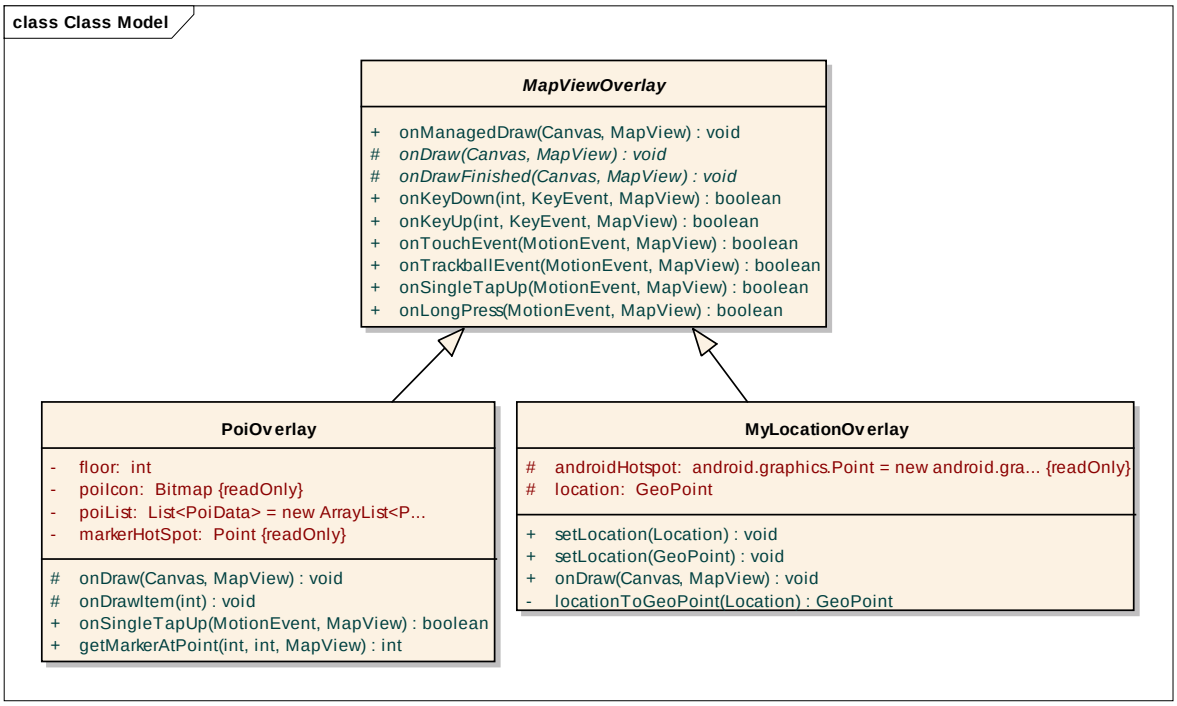


Bild 37: Package gui.view.overlay

4.2.10.2 Classes and methods

MapViewOverlay
Base class representing an overlay which may be displayed on top of a `MapView`. To add an overlay, subclass this class, create an instance, and add it to the list obtained from getOverlays() of `MapView`.

PoiOverlay	
This class draws all the `PoiData` objects onto a `MapView`. That the `PoiData` objects are drawn correctly this overlay needs a floor value provided by the `MapView` class, otherwise you would see `PoiData` objects from floor 1 even if you are on floor 2.	
Methods:	
onDrawItem(int)	Starts the `PoiInformation` activity which displays a clicked `PoiData` object.
getMarkerAtPoint(int, int, MapView)	Builds a hit box around the provided touchdown point and checks in the stored list of `PoiData` if one of the objects is contained in this hit box. Is executed from the `onSingleTapUp()` method.

MyLocationOverlay
This class draws the measured position on a `MapView`. The position is shown by a little Android figure with an accuracy circle around it.
Methods:
➔ This class simply draws its android figure to the provided `Location` onto the `MapView`.

4.2.11 Package gui.view.util

4.2.11.1 Description

This package contains the adapter classes which are use for example by list views. The standard adapter is the array adapter which is extended by the `PlaceAdapter` and the `PoiAdapter`. The reason to do that is because if you extend the adapter you are able to overwrite how the objects provided to the adapter are displayed in the list view or list activity. If the normal array adapter is used the `toString()` method of the objects will be called to display the item.

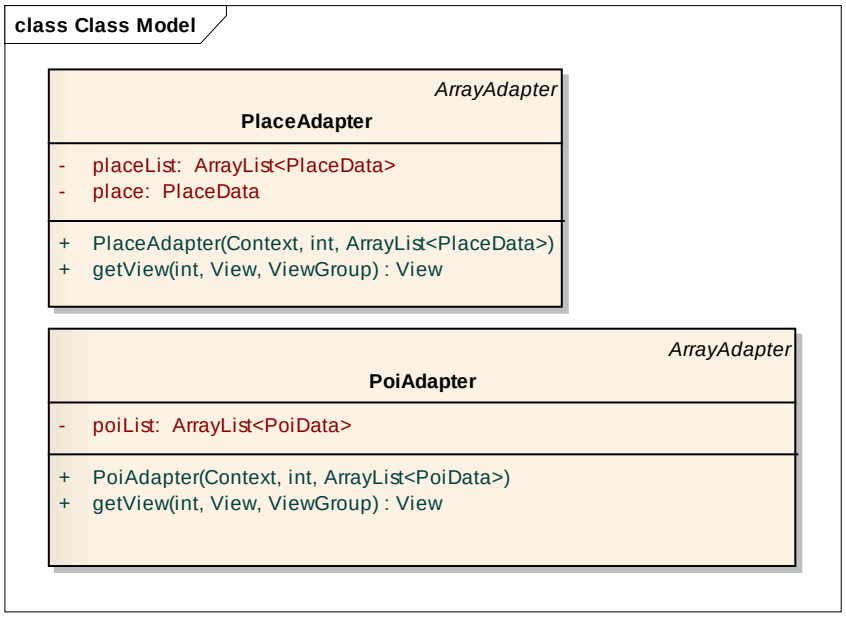


Bild 38: Package gui.view.util

4.3 Package and class diagrams

4.3.1 Package diagram

4.3.1.1 Description

This diagram shows the dependencies between all top level packages of the IndoorGuide4Android.

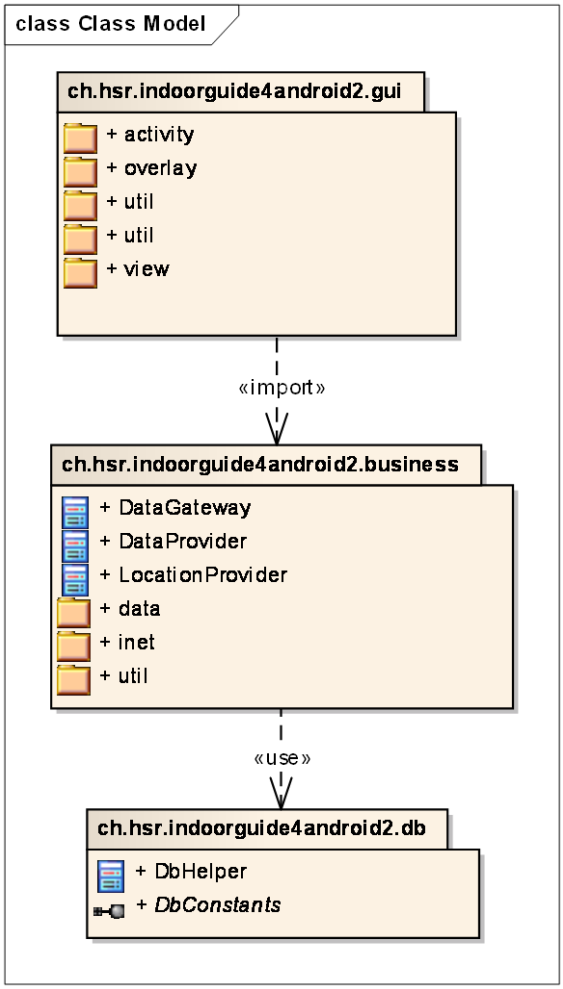


Bild 39: Package diagram

4.3.2 Layer class diagram

4.3.2.1 Description

This diagram shows an overview of the application. It's divided into 3 layers of abstraction. Also shown are the dependencies between the different classes and layers.

Persistence: Represents the lowest layer which stores the POJOs and provides the application with the needed data.

Problem Domain: Represents the manager layer with all the business classes which are responsible to make the 3 layers work together correctly.

View: Represents the gui layer on top of the application, this layer uses the lower layers to display the data to the user and interact with him.

Activity Context: The activities are logically placed on top of the system. They work close to the problem domain and build the environment for the views.

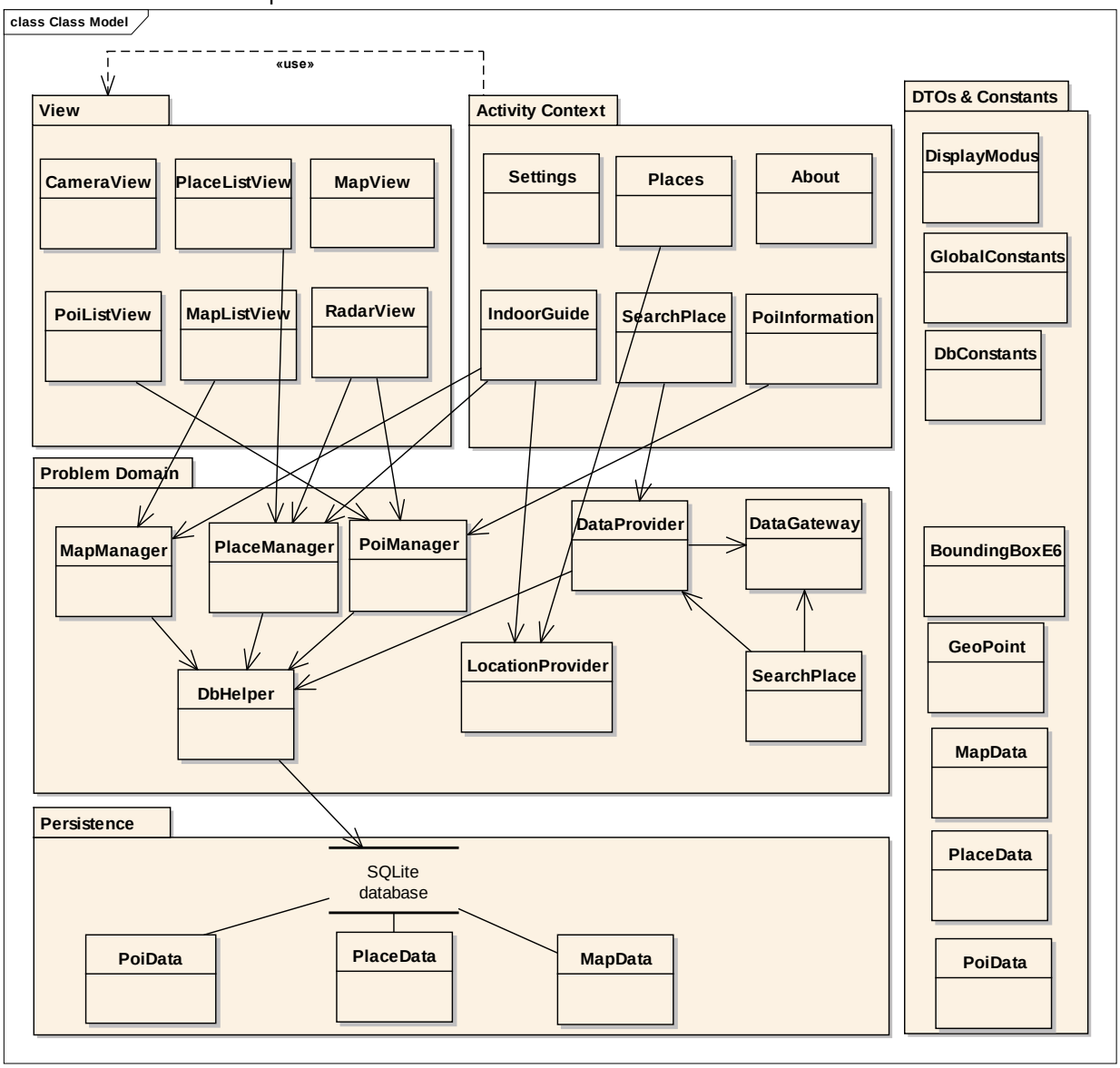


Bild 40: Layer class diagram

4.4 Sequence diagrams

4.4.1 SSD 01: walkthrough

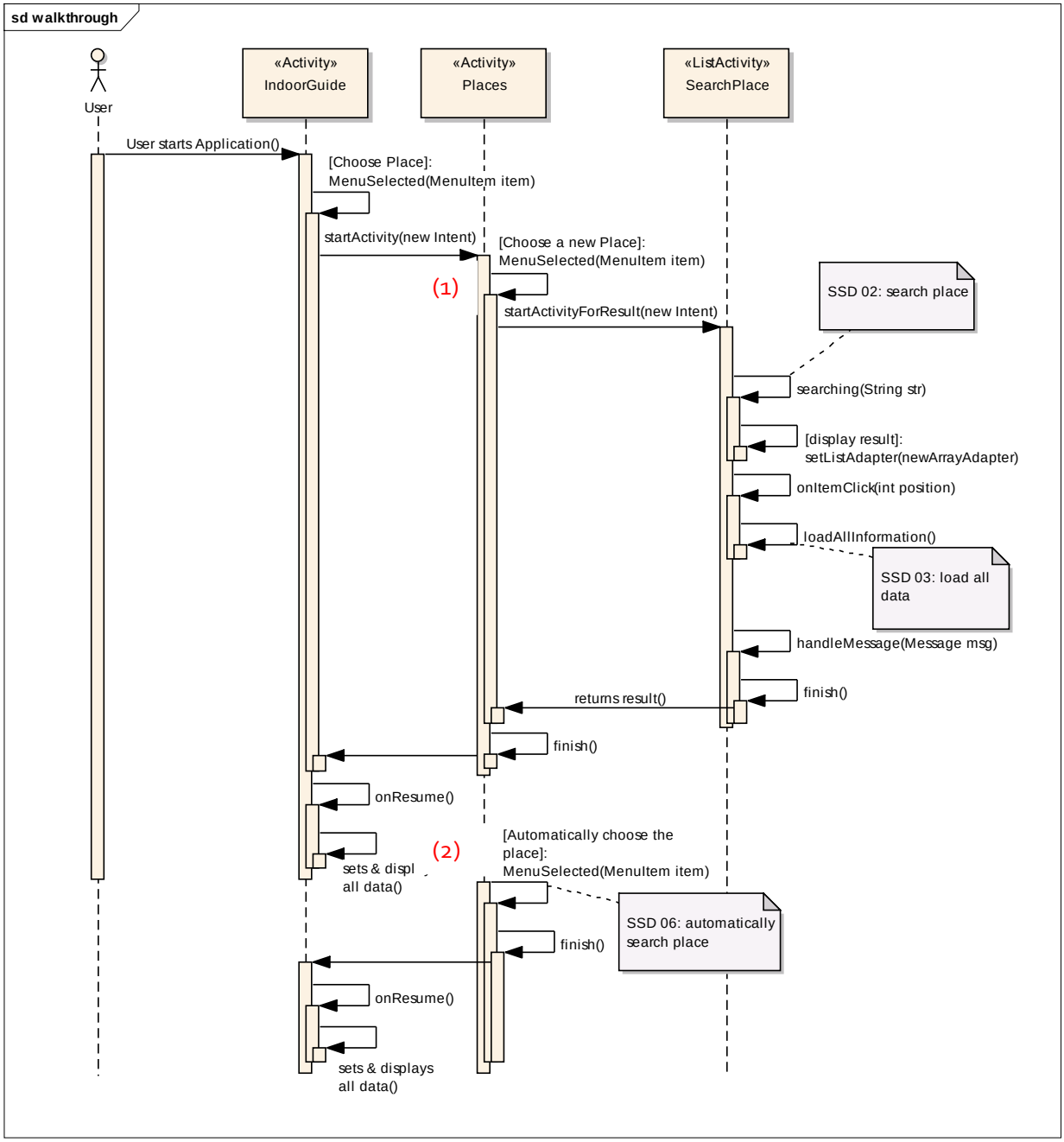


Bild 41: SSD01: walkthrough

4.4.1.1 *Description*

This sequence diagram shows a complete walkthrough of the application when you want to choose a new place (for example "Hochschule Rapperswil") and display its data. You have the opportunity to choose between "manual search" (1) or "automatically choose the place" (2) from the menu.

4.4.2 SSD 02: search place

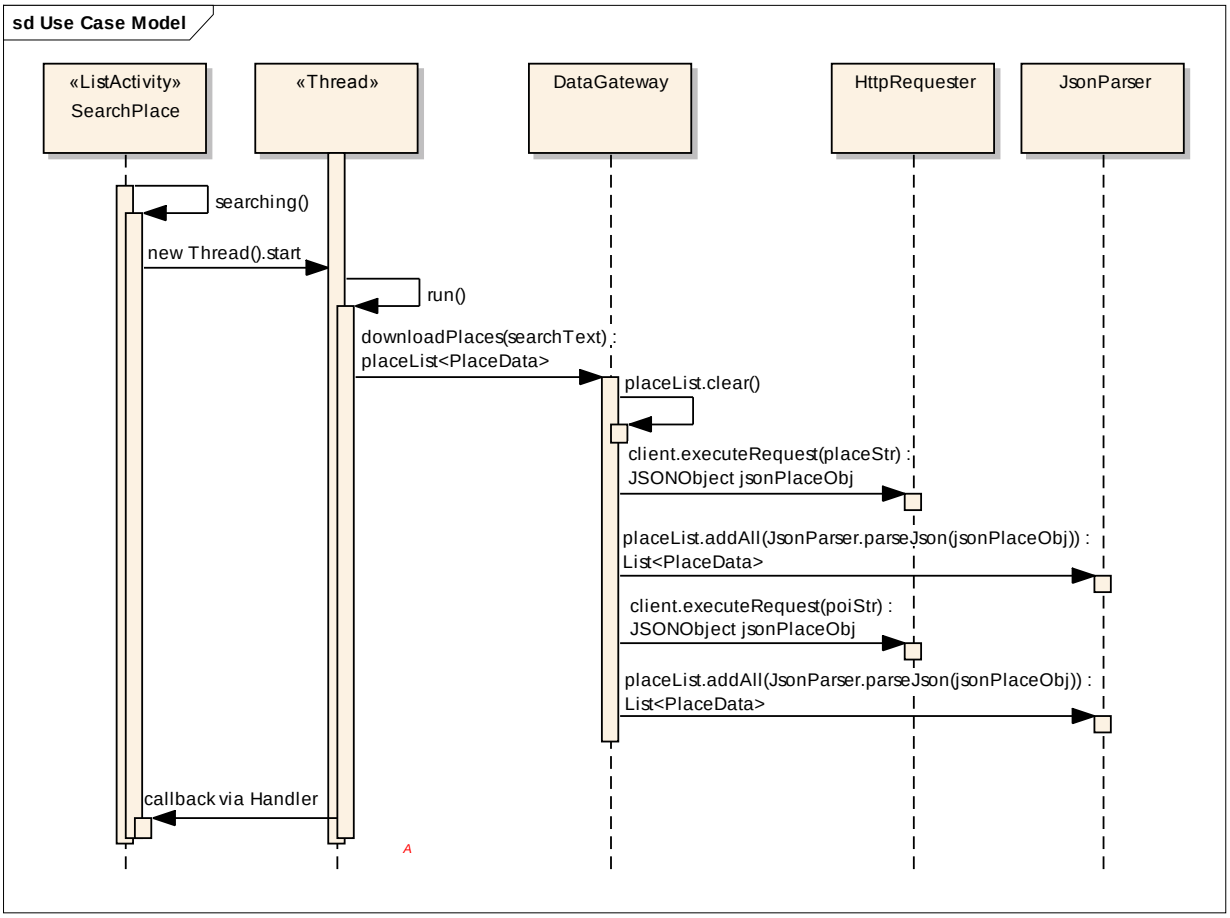


Bild 42: SSD02: search place

4.4.2.1 *Description*

This sequence diagram shows the program flow started if the user decided to manual search the place. The `SearchPlace` activity starts a thread and provides it with a handler whereby the thread will notify the activity when he has done his job or in worst case what went wrong.

4.4.3 SSD 03: load all data

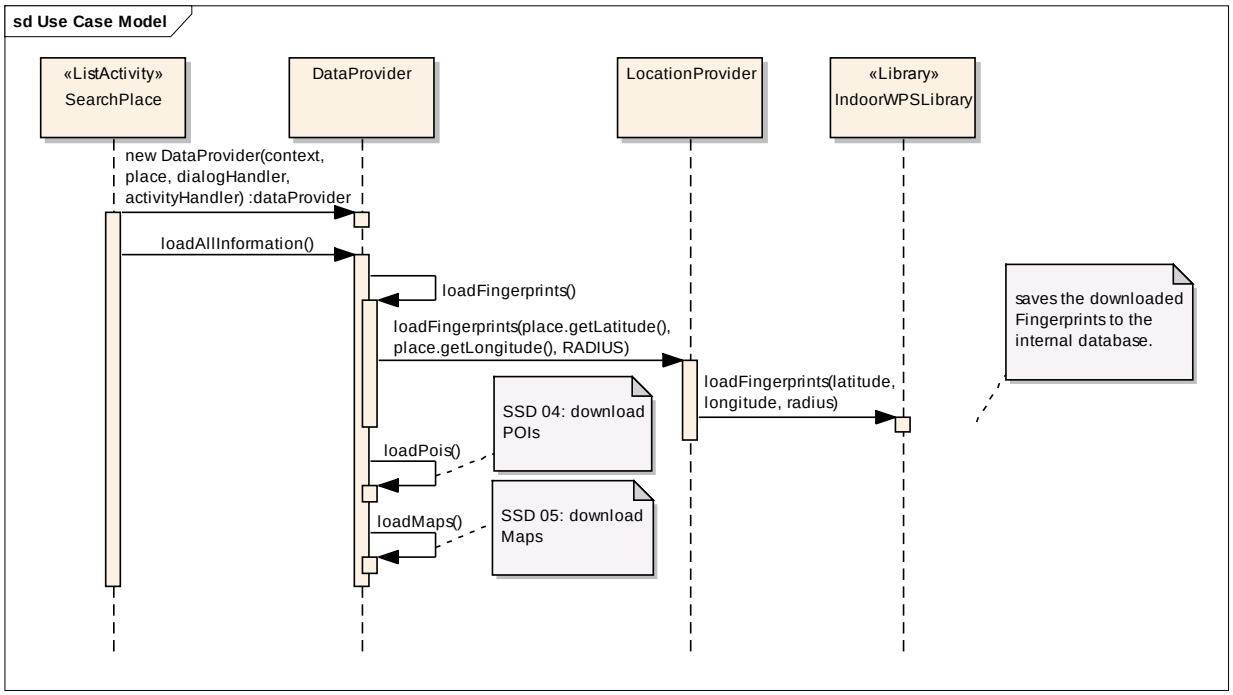


Bild 43: SSD03: load all data

4.4.3.1 *Description*

This sequence diagram shows the first part of the program flow after the user has finally chosen a place and the application is going to download all the related data. Here the fingerprints around your place are downloaded from the IndoorWPS Community Server.

4.4.4 SSD 04: download POIs

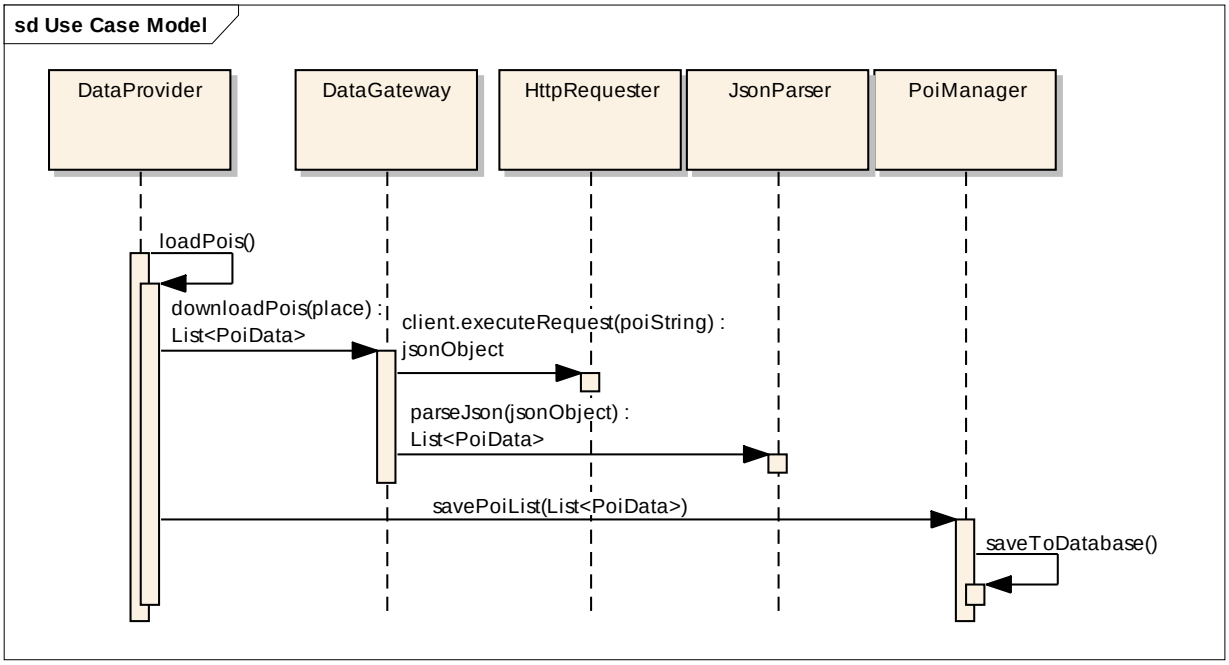


Bild 44: SSD04: download POIs

4.4.4.1 *Description*

This sequence diagram shows the program flow for downloading the POIs from the IndoorWPS Community Server. A bounding box is created out of the provide place, based on this bounding box the POIs will be downloaded, meaning every POI that is contained in the bounding box.

4.4.5 SSD 05: download maps

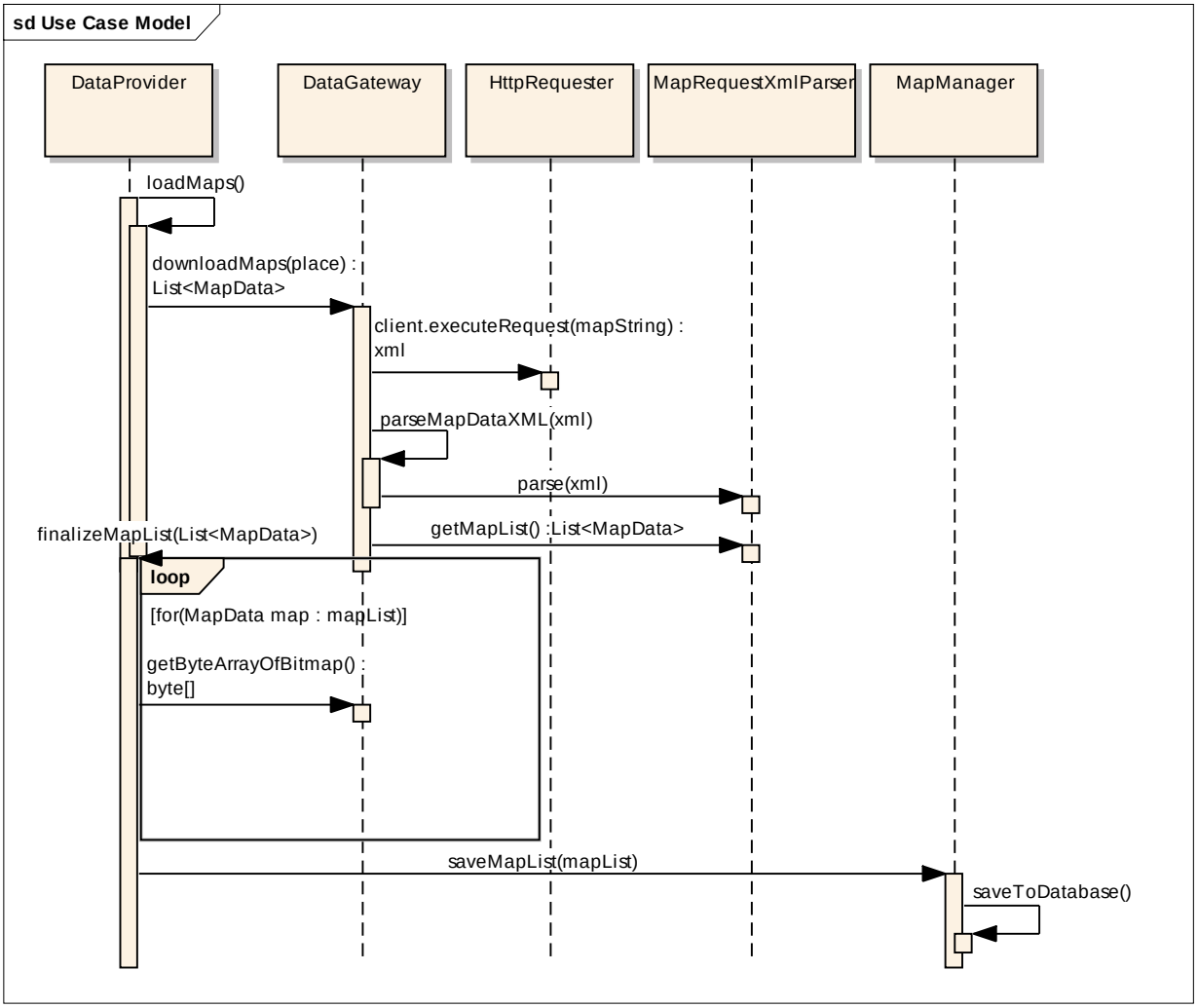


Bild 45: SSD05: download maps

4.4.5.1 *Description*

This sequence diagram shows the program flow for downloading the maps from the IndoorWPS Community Server. Based on the provided places all the maps near to it will be downloaded.

First just the XML with the available maps is parsed, it doesn't yet contain the effective map graphic but all the meta information. In the `finalizeMapList()` method finally the map graphic is downloaded and will be saved as BLOB to the database.

4.4.6 SSD 06: automatically search place

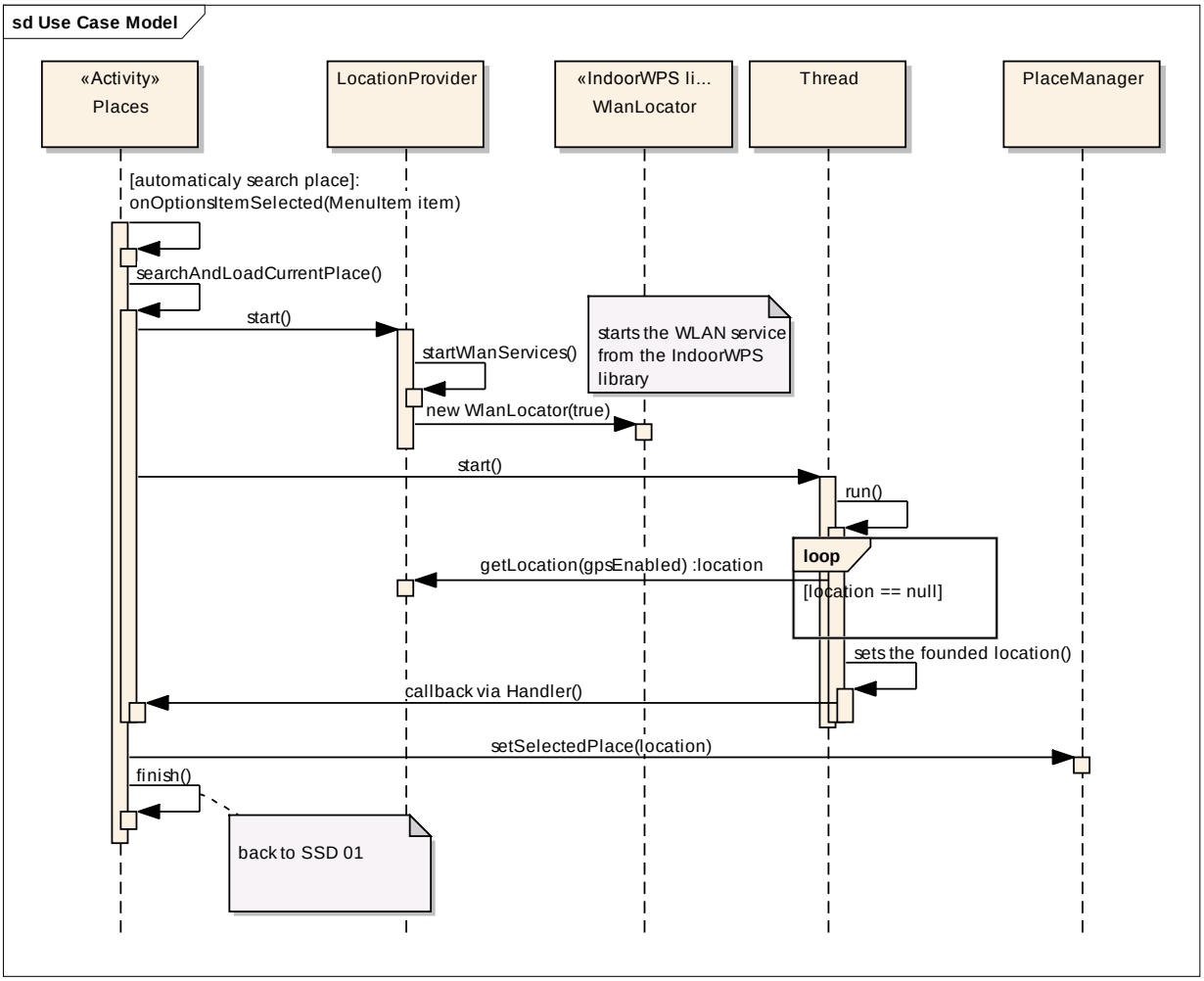


Bild 46: SSD06: automatically search place

4.4.6.1 *Description*

This sequence diagram shows the program flow if the user chose to let the place be searched automatically.

In this mode the application tries to determine its location itself. This means the `LocationProvider` is started to receive localization via GPS or WLAN. The application waits for 15 second if no location could be provided the thread notifies the activity via the handler otherwise the determined location is set and the activity is notified about the success. The application will now start downloading all related data as shown in SSD01.

5. Implementation und Test

5.1 Implementationdetails

5.1.1 IndoorWPS

Für die Lokalisierung mittels WLAN wird auf die Applikation *IndoorWPS* zurückgegriffen. Diese wurde vom Institut für Software (IFS) der HSR entwickelt. Folgend eine kurze Beschreibung:[16]

> Das Wireless Positioning System (WPS) des Instituts für Software/GISpunkt (HSR) basiert auf einem Positionierungsverfahren mit Hilfe von WLAN. Es funktioniert in dicht besiedelten Gebieten und in Gebäuden ('indoor'), also genau dort wo GPS seine Probleme hat oder gar nicht funktioniert und ist deshalb eine praktische Ergänzung zu GPS.
> Die eigene Position wird aufgrund von Signalstärkenlisten (sogenannten 'Fingerprints') rein lokal bestimmt (offline-Betrieb) und ist 'raumgenau', d.h. 4 bis 10 Meter.

Die Entscheidung für diese Indoor-Lokalisierungslösung fiel auf *IndoorWPS*, da diese von der HSR entwickelt wurde, für die auch der *IndoorGuide4Android* entwickelt wird. Dies bringt den Vorteil, dass die Infrastruktur schon vorhanden und der Entwickler (Michael Klenk) als Ansprechsperson bei Problemen und Erweiterungswünschen erreichbar ist. Zudem wurde im Vorgängerprojekt (erste Version des *IndoorGuide4Android*) zuerst die alternative Lokalisierungssoftware *PointZero*[17] benutzt, welche keine zufriedenstellende Resultate lieferte. Beim damaligen Projekt wurde zu einem späteren Zeitpunkt auf *IndoorWPS* gewechselt, was für den heutigen Technologiestand gute Resultate brachte.

5.1.1.1 *Funktionsweise*

Bei IndoorWPS wird die Lokalisierung anhand von Fingerprinting gemacht. Hierzu werden an Referenzpunkten Messwerte aufgenommen, welche alle empfangenen WLAN-Basisstationen und die dazugehörigen Informationen enthalten. Dieser Punkt wird dann mit der Koordinate versehen als Fingerprint auf dem IndoorWPS Community Server abgespeichert. Diese Erfassung der Fingerprints kann z.B. mit dem IndoorWPS-Client für Android gemacht werden.

Der IndoorWPS-Client kann nun die Fingerprints vom Server herunterladen und lokal in einer Datenbank abspeichern. Die Anzahl Fingerprints kann mit Angabe des Radius (geografische Umgebung, in der sich die Fingerprints befinden sollen) eingegrenzt werden. Danach kann man sich (offline) mit dem Client bewegen.

Für die Lokalisierung werden die aktuellen Messwerte von sichtbaren Access Points mit den Fingerprints in der Datenbank abgeglichen. Ausgewählt werden die gespeicherten Werte, welche möglichst viele gleiche Access Points, identifiziert durch die MAC-Adresse, enthalten. Zu jedem dieser Fingerprints wird ein Distanzwert berechnet, der anzeigt wie ähnlich sich der Messwert und der Fingerprint sind. Aus den Positionen der Fingerprints und den dazu berechneten Distanzwerten wird dann ein geometrisch gewichteter Mittelwert berechnet.

[16] Quelle: Auszug aus der IndoorWPS-Beschreibung aus dem GISpunkt-Wiki der HSR. (http://gis.hsr.ch/wiki/IndoorWPS)
[17] Quelle PointZero: http://wiki.hsr.ch/StefanKeller/wiki.cgi?PointZero

Da die Fingerprints nur alle paar Meter erfasst werden und die empfangenen Signale der Access-Points unterschiedlich ausfallen können (durch Reflexionen, Abschwächungen, unterschiedlichen Ausgangsstärken, ...), braucht es beim vergleichen mit den Werten in der Datenbank gewisse Toleranz, um überhaupt einen Treffer zu bekommen. Dadurch kann die aktuelle Position nur vermutet werden. Um ein „herumspringen" der aktuellen Position zu vermindern, wird die Position des Geräts durch eine Mittelung über mehrere Berechnungen noch weiter verfeinert. Als Resultat wird die Koordinate der errechneten Position zurückgegeben.

5.1.1.2 Integration in IndoorGuide4Android

Vom *IndoorWPS* Projekt wurden zwei Libraries verwendet. `IndoorWPSLib`[18] beinhaltet die Hauptfunktionalität vom *IndoorWPS*. `IndoorWPSAndroid` beinhaltet Funktionalität, die auf die Android-Plattform zugeschnitten ist (Datenbank, WLAN-Service usw.). Diese beiden Libraries wurden als Java Archive (JAR-File) ins *IndoorGuide4Android*-Projekt importiert.

Es wurde eine Klasse `LocationProvider` erzeugt, welche die Schnittstelle zum *IndoorWPS* darstellt. Dort werden die Fingerprint-Datenbank und der `WlanLocator` erstellt. Der `LocationProvider` greift dabei auf Methoden vom *IndoorWPS* zu. Dies wäre unter anderem `loadFingerprints(latitude, longitude, radiusInKm)` und `getAveragedLocation()`. Zudem wird die Methode `getStatus()` auf dem `WlanLocator` aufgerufen, um den Status der Resultate zu erfahren (ob veraltet oder neu) und dementsprechend zu reagieren.

5.1.2 Requests zum OSM-in-a-Box Server

Die Anfragen zum OSM-in-a-Box Server werden über WFS (Web Feature Service) gemacht. Dieser Service ist speziell für Geodaten gemacht und ermöglicht den Zugriff auf geographische Features in Datenbanken. Als Responstetype wird JSON verlangt, da dieses Format praktisch ohne Overhead auskommt.

Die Suche nach einem Place wird mit folgendem Request gemacht:

1. Teil der URL: Server
http://sinv-56029.edu.hsr.ch/geoserver/

2. Teil der URL: WFS-Request für die Suche nach Places
wfs?request=getfeature&typename=**osm:place_lookup**&outputformat=**json**
2. Teil der URL: WFS-Request für die Suche nach POIs
wfs?request=getfeature&typename=**osm:poi_lookup**&outputformat=**json**

3. Teil der URL: Filter (z.B. Suche nach „Rapperswil")
&filter=%3CPropertyIsLike%20wildCard%3D%22*%22%20singleChar%3D%22.%22%20escape%3D%22!%22%20matchCase%3D%22false%22%3E%3CPropertyName%3Ename%3C%2FPropertyName%3E%3CLiteral%3E*__Rapperswil__*%3C%2FLiteral%3E%3C%2FPropertyIsLike%3E

[18] Quelle IndoorWPS Libraries: http://dev.ifs.hsr.ch/indoorwps/

Die POIs werden über folgenden Request geholt:

1. Teil der URL: Server
http://sinv-56029.edu.hsr.ch/geoserver/

2. Teil der URL: WFS-Request
wfs?request=getfeature&typename=**osm:indoor_view**&outputformat=**json**

3. Teil der URL: Boundingbox (Latitude SW, Longitude SW, Latitude NE, Longitude NE)
&BBOX=**47.2135058,8.8072663,47.2335057,8.8272663**

5.1.3 Request zum IndoorWPS Community Server

Die Requests zum IndoorWPS Community Server sehen wie folgt aus:

Request für Fingerprints (Latitude / Longitude / Radius in km)
http://labs.geometa.info/indoorwps/rest/fingerprint/search/**47.222803/8.817025/0.1**

Request für Maps (Latitude / Longitude)
http://labs.geometa.info/indoorwps/rest/geomap/search/**47.222803/8.817025**

5.1.4 Trennung von Code und Layout

Das Layout wurde vom eigentlichen Code getrennt. Es ist in XML-Files definiert und wird jeweils beim Start eines Activitys als Ressource geholt und eingebunden. Die `ContentView` muss jeweils in der `onCreate()`-Methode gesetzt werden. Wichtig zu wissen ist, dass nur im Gültigkeitsbereich dieser Methode Ressourcen (z.B. Views) aus der `ContentView` geholt werden können. Danach ist ein Zugriff darauf nicht mehr möglich. Eine Ressource wird wie folgt geholt:

```
MapView mapView = (MapView) findViewById(R.id.map_view);
```

Alle Farben und Strings, welche in der Applikation verwendet werden, sind in ein externes XML ausgelagert (`colors.xml` und `strings.xml`). Die Bilder sind alle im Ordner `drawable-hdpi` zu finden.

5.2 Entscheidungen & Knackpunkte

5.2.1 Kartenauswahl

Eine wichtige Entscheidung war, wie wir das Auswählen der korrekten Karte dem Benutzer ermöglichen. Der offensichtliche Ansatz wäre hier anhand der errechneten Position die korrekte Karte zu laden. Diesem Ansatz stehen allerdings einige Probleme gegenüber, die das Vorhaben etwas komplizierter machen.

Wir kennen zwar die Position anhand der Berechnungen durch die Fingerprints, jedoch kann es sein, dass Fingerprints aus verschiedenen Gebäuden in die Berechnung einfliessen. Dies ist anhand des folgenden Bildes etwas übertrieben illustriert.

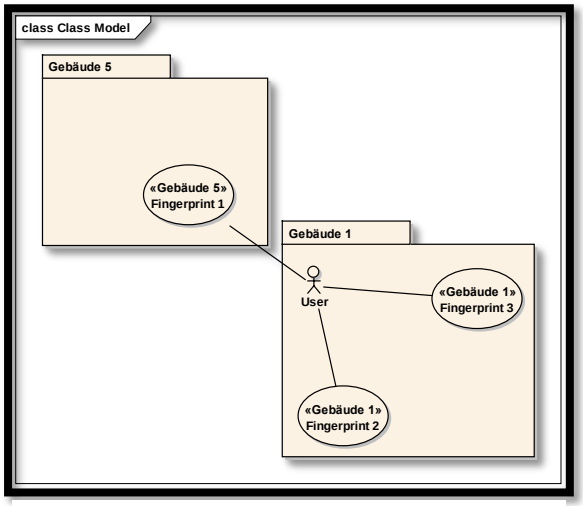


Bild 47: Position-Berechnungsproblem

Es ist zu sehen, dass die Position aus dem Dreieck von zwei Fingerprints aus dem Gebäude 1 und einen Fingerprint aus dem Gebäude 5 berechnet wird. Somit ist es offensichtlich, dass anhand der errechneten Position nicht mit 100 prozentiger Sicherheit festgestellt werden kann in welchem Gebäude sich ein Benutzer aufhält.

Dies zu ermöglichen bedingt, dass der Berechnungsalgorithmus des IndoorWPS erweitert werden müsste. Allerdings müssten dazu noch die bestehenden Fingerprints um ein Attribut „building" erweitert werden welches noch nicht existiert. Dann könnte der Service soweit gebracht werden, dass er mit einer sehr hohen Trefferrate das korrekte Gebäude angibt. Somit könnte die Karte aus dem „richtigen" Gebäude geliefert werden.

Eine weitere Überlegung war, dass die Bounding Boxen der geladenen Karten bekannt sind und anhand dieser Box überprüft werden könnte ob die errechnete Position darin vorkommt. Somit wäre das Gebäude in welchem man sich befindet eindeutig identifiziert. Leider funktioniert auch dieser Ansatz nicht, denn um die Karte verläuft nicht entlang des Grundrisses, sondern die Box umspannt die Karte in einem Rechteck. Dies folgt daraus, dass die Karten mit einer Rotation von 0 zu Norden ausgerichtet sind (georeferenziert). Somit können sich Punkte auf den Karten überlagern und es kann wieder nicht mit Sicherheit entschieden werden in welchem Gebäude sich der Benutzer befindet.

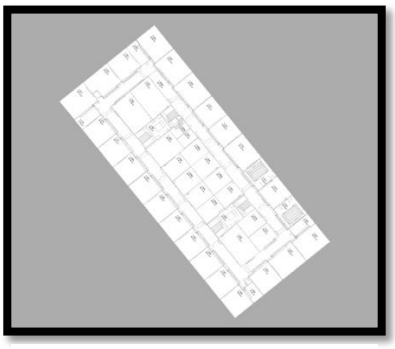


Bild 48: georeferenzierte Karte

Gelingt es, die Wahl auf ein Gebäude festzulegen, muss desweiteren entschieden werden, in welchem Stockwerk sich der Benutzer befindet. Die Fingerprints enthalten bereits jetzt ein Attribut „floor", welches jedoch noch nicht verwendet wird. Um automatisch das richtige Stockwerk zu wählen, stellt sich das gleiche Problem wie bei den Gebäuden, was im obigen Bild dar-

gestellt ist. Allerdings ist dies im Gebäude noch schwieriger. Der IndoorWPS-Algorithmus muss folglich noch intelligenter werden.

Aus all diesen Gründen haben wir uns beim IndoorGuide4Android für eine manuelle Auswahl der Karten entschieden. Die möglichen Karten werden aufgrund der errechneten Position und der Bounding Boxen der Karten erstellt, die sich wie erwähnt überlagern können.

5.2.2 Memory leaks

Dies war ein grosses Thema im späteren Verlauf unserer Bachelorarbeit. Als wir die Kartenansicht implementierten und die ersten Bitmaps generieren wollten ging die Dalvik VM bald einmal in die Knie „java.lang.OutOfMemoryError: bitmap size exceeds VM budget".

Dieses Problem ist in der Android Community allgemein bekannt und ist teilweise auf einen Bug in den bisherigen Android SDK Versionen zurückzuführen (dies soll in SDK 2.2 besser werden). Jedoch liegen viele Ursachen dieser Exception wohl beim Programmierer selbst. Es ist extrem wichtig, sauber und ressourcenschonend zu programmieren. Das heisst:

- Keine unnötigen Referenzen übergeben
- Referenzen nicht unnötig halten
- Bitmaps nur dann erstellen wenn sie gebraucht werden
- Bitmaps immer sauber aufräumen (`bitmap.recycle()`)
- Wiederverwendete Bitmaps „static" definieren

Der Code wurde nach all diesen von uns ermittelten Regeln angepasst. Zu beachten ist, dass Android, wenn man das Handy kippt, die komplette Activity beendet und sie neu startet. Dies bietet die grösste Angriffsfläche für OutOfMemoryExceptions. Es muss also akribisch genau aufgeräumt werden.

Eine weitere Handlung, die wir vorgenommen haben, war das Auslagern der Datenbank auf die SD-Karte. Dadurch hatten wir uns eine Verbesserung der Situation erhofft, da die Datenbank dann nicht im internen Speicher liegt und das System nicht zusätzlich belastet.

Nach diesen Anpassungen verschwanden praktisch alle Memory Leak Probleme. Sollten weitere auftreten, kann dies von vielen Faktoren abhängen. Zum Beispiel von anderen Applikationen die gestartet sind oder zu grossen Karten, die auf dem IndoorWPS Community Server abgelegt sind. Die Applikation IndoorGuide4Android ist in Sachen Speicherverwaltung auf dem neusten Stand[19].

5.2.3 Wechsel AR-/Kartenansicht

Wir wollten eine innovative Idee für den Wechsel zwischen der AR- und Kartenansicht realisieren. Das Ergebnis war, dass wir die Interaktion des Benutzers auf ein Minimum reduzierten und den Wechsel zwischen den Ansichten automatisierten.

[19] Quelle Tech Blog und Android-Dev:
- http://ttlnews.blogspot.com/2010/01/attacking-memory-problems-on-android.html
- http://android-developers.blogspot.com/2009/01/avoiding-memory-leaks.html

Hält der Benutzer das Handy senkrecht, erscheint die AR-Ansicht. Will er sich die Karte ansehen, hält er sein Handy waagerecht vor sich hin. Wie wir festgestellt haben ist dies das Normverhalten (Kamerahaltung und Haltung einer „echten" Papierkarte). Wir konnten also genau diesen Ablauf mit dem im Handy integrierten Winkelsensor erkennen und die Ansichten automatisch umschalten. Dies trägt existentiell zur angenehmen Benutzung des IndoorGuides bei.

Als das Feature implementiert war, bemerkten wir einen grösseren Performanceverlust unserer Applikation, der uns zu denken gab. Auch traten dann sporadisch Heap Exceptions auf, welche uns darauf aufmerksam machten, weshalb die Applikation sich so verlangsamte. Nach einer Analyse des Heaps (mittels des vom Android SDK mitgelieferten DDMS-Tools) zeigte sich, dass der Winkelsensor extrem viele Message-Objekte produziert.

Diese Objekte wurden allesamt via Handler an die `IndoorGuide`-Activity zurückgesendet und füllten den Heap stetig weiter auf. Aus diesem Grunde erweiterten wir die Implementation unseres `AngleSensorEvtListener`s, damit er nur noch markante Änderungen der Winkel weiterleitet. Somit waren die Probleme gelöst und das Feature funktionierte wie gewünscht. Es brachte dem Benutzer noch einen zusätzlichen Nutzen: Die Ansichten wechseln nicht schon bei kleinsten Bewegungen (Erschütterungen).

5.2.4 Location Provider

Die Auswahl, welcher Location Provider genommen werden soll, hängte von verschiedenen Kriterien ab. Zum einen ist GPS nicht immer eingeschaltet, zum anderen sind die Dienste nicht immer verfügbar. Wir haben uns dazu entschieden GPS zu bevorzugen, wenn der Dienst ein gutes Ergebnis liefert. Dies aus dem einfachen Grund, dass dort in der Regel genauere Werte zurückgegeben werden.

Die zentrale Methode des Location Providers ist `public Location getLocation(boolean gpsEnabled)` der Klasse `LocationProvider`. Hier wird kurz der Ablauf in Pseudocode beschrieben, der durchlaufen wird:

```
1.  if gpsEnabled            // Abfrage ob GPS in den Settings des IndoorGuides aktiviert ist
        if gpsServiceStarted == false
            startGpsServices()              // Services für GPS starten
        if isProviderEnabled()              // Abfrage ob GPS-Provider in den
                                              Android-Settings aktiviert ist
            gpsLocation = getLastKnownLocation()    // Location von GPS holen
            if gpsLocation != null
                if currentTimeMillis() - gpsLocation.getTime() <= 10000 &&
                gpsLocation.getAccuracy() <= 1      // diese zwei Abfragen
                                                      sind um veraltete und
                                                      ungenaue Locations
                                                    zu ignorieren
                    return location         // Location zurückgeben
```

```
2. if wlanLocator.getStatus() == active          // Abfrage ob IndoorWPS neue Locations liefert
       toleranceCounter = limit                  // Toleranzzähler aufs Limit setzten
       return wlanLocator.getAveragedLocation()  // Location zurückgeben

3. else if toleranceCounter > 0                  // Wenn Status NICHT „active" ist, jedoch der Zähler noch nicht 0 erreicht hat, wird nochmals die letzte Location zurückgegeben
       toleranceCounter – 1
       return wlanLocator.getAveragedLocation()

4. return new Location(0.0, 0.0, 0)              // Wenn die Toleranz unter 0 geschritten ist, wird eine leere Location zurückgegeben
```

Wir haben uns für den Toleranzwert entschieden, da es oft vorkommt, dass der Status vom `WlanLocator` für kurze Zeit nicht „active" ist. Dies würde bewirken, dass immer dann eine „leere" Location zurückgegeben würde. Das würde natürlich den aktuellen Standort auf der Karte verschwinden lassen und Unruhe in die POI-Liste und den Radar bringe. Erst wenn 15 Mal nacheinander ein anderer Status als „active" eingetroffen ist, gilt IndoorWPS wirklich nicht mehr als verfügbar.

5.3 User Interface Implementation

5.3.1 IndoorGuide – Activity

Beim Starten der Applikation wird das Activity `IndoorGuide` gleich mit dem zuletzt gewählten Place angezeigt. Der geladene Place wird zur Information kurz als Text eingeblendet. Der `IndoorGuide` kennt die zwei verschiedenen Ansichten AR- und Kartenansicht.

`CameraView`: Diese Ansicht wird angezeigt, wenn das Handy senkrecht (wie beim Fotografieren) gehalten wird. Dabei wird im Hintergrund das Kamerabild eingeblendet, welches dann mit Informationen überlagert wird. Im linken oberen Bereich des Bildschirms ist die Liste mit den POIs halbtransparent eingeblendet. Sie ist nach Distanz sortiert. Die POIs sind mit Name, Beschreibung und Entfernung beschriftet. Im rechten unteren Bereich ist ein Radar eingeblendet, welcher die POIs in nächster Nähe anzeigt. Der Kompass richtet die POIs korrekt aus und ist als „N" am Rand des Radars ersichtlich. Im rechten oberen Bereich werden Statusinformationen zum aktuellen Location-Dienst (WLAN / GPS / keine Location) mit einem Icon und beim Wechsel einem eingeblendeten Text angezeigt.

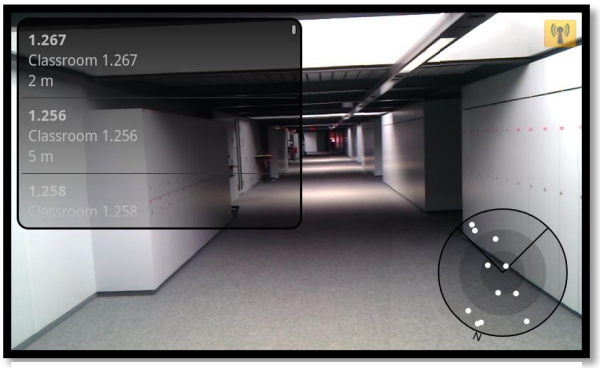


Bild 49: AR-Ansicht

MapView: Diese Ansicht wird angezeigt, wenn das Handy waagrecht gehalten wird. Dabei wird die Karte eingeblendet, auf welcher der aktuelle Standort und die POIs angezeigt werden. Die Karte kann mit dem Finger verschoben werden. Beim ersten Öffnen wird automatisch auf den aktuellen Standort zentriert. Ist keine Karte geladen, erscheinen das Logo der Applikation und ein Text mit dem Hinweis, dass mit längerem Drücken auf den Bildschirm die Karte gewählt werden kann. Es werden dann alle Karten angezeigt, die vom aktuellen Standort aus in Frage kommen würden (und geladen sind).

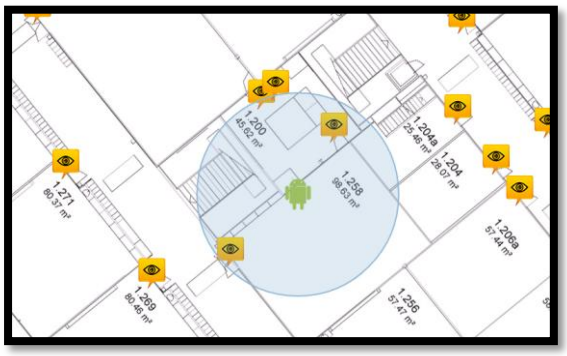


Bild 50: Kartenansicht

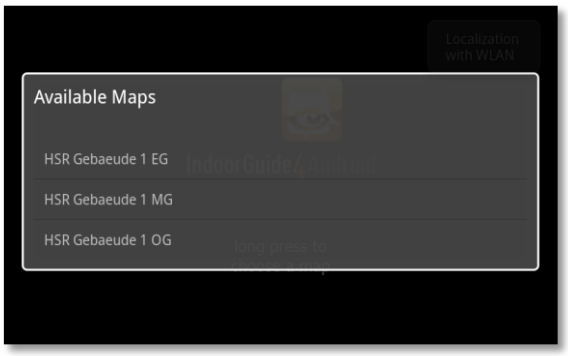


Bild 51: Kartenansicht

Mainmenu: Beim Drücken der Menütaste auf dem Handy wird ein Menü mit folgenden Menüpunkten eingeblendet:

- Choose place
- Settings
- About
- Exit

Bild 52: Menü

5.3.2 PoiInformation – Activity

Beim anwählen eines POIs (in der Liste oder auf der Karte) werden die Informationen über diesen POI in einem eigenen Activity angezeigt, welches das Aussehen einer Dialogbox hat. Als Informationen werden Name, Gebäude, Stockwerk, Distanz zum aktuellen Standort, Datum der letzten Änderung und Beschreibung angezeigt. Falls die Beschreibung mehr Text beinhaltet, als Platz ist, kommt eine Scrollbar zum Einsatz. Zusätzlich werden Links zu Multimediainhalten und Websites dargestellt. Diese sind gelb, wenn ein Link hinterlegt ist, ansonsten sind sie deaktiviert und grau. Diese Links werden im externen Browser geöffnet. Die Bedeutung der Buttons ist von links nach rechts gelesen: Website, Wikipedia, Bild, Film, Audio.

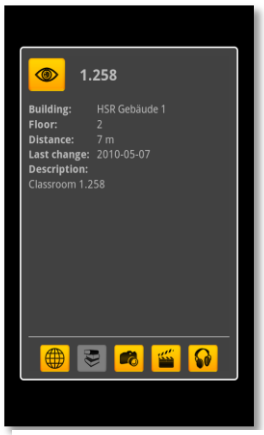


Bild 53: POI-Information

5.3.3 Places – Activity

In diesem Activity werden alle geladenen Places angezeigt. Sind noch keine Places geladen, wird dies mit einem Text angezeigt. Über den Menübutton gelangt man zur Auswahl um einen Place zu laden. Zum einen gibt es die Möglichkeit für eine Suche mit einem Suchbegriff, zum anderen das automatische Suchen anhand der aktuellen Koordinate. Bei der ersten Variante wird das `SearchPlace` – Activity gestartet. Bei der automatischen Suche wird eine gewisse Zeit lang versucht, eine Koordinate zu empfangen (GPS / WLAN). Falls diese Suche nicht erfolgreich ist, wird abgebrochen und dem Benutzer eine Information ausgegeben. Bei erfolgreicher Suche wird der Benutzer aufgefordert, dem Place einen eigenen Namen zu geben. wird dieser Place nun mit der Bemerkung, dass er automatisch gesucht wurde.

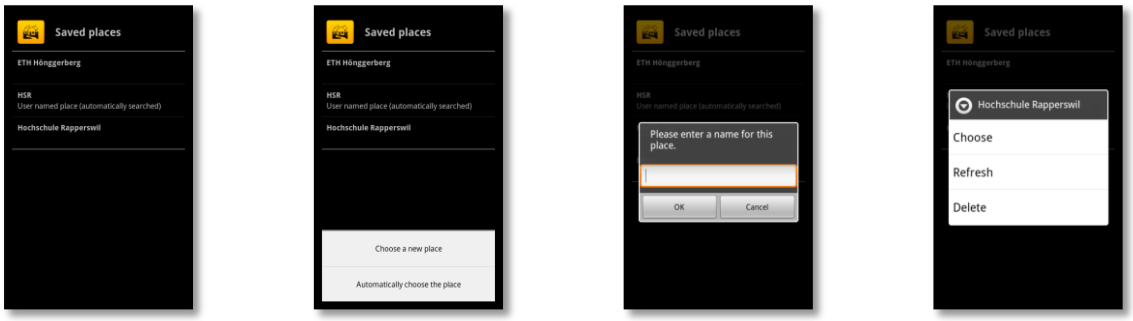


Bild 54: (von links nach rechts) gespeicherte Places, Menü, Namenvergabe bei Auto-Suche, Kontextmenü

Um ein Place auszuwählen kann darauf gedrückt werden. Zudem erscheint bei längerem Drücken ein Kontextmenü, welches die drei Menüpunkte Auswählen, Aktualisieren und Löschen beinhaltet. Beim Aktualisieren werden alle Daten dieses Places erneut heruntergeladen und gespeichert, falls eine neuere Version vorliegt (z.B. ein POI neue Daten aufweist). Beim Löschen wird noch eine Sicherheitsabfrage gestellt, danach wird der Place aus der Datenbank gelöscht.

5.3.4 SearchPlace – Activity

Sucht man nach einem neuen Place, gelangt man in dieses Activity. Hier kann man im Suchfeld einen Suchbegriff eingeben, wobei dies ein Ort, POI (z.B. Hochschule Rapperswil) oder ein Teil eines Wortes sein kann. Unter den angezeigten Resultaten kann man den gewünschten Place auswählen, worauf dessen Daten heruntergeladen werden. Beim Herunterladen wird ein Progress-Dialog eingeblendet, der anzeigt, bei welchem Schritt sich der Prozess befindet. Sind alle Daten geladen, wird dieser Place direkt ausgewählt und im `IndoorGuide` angezeigt.

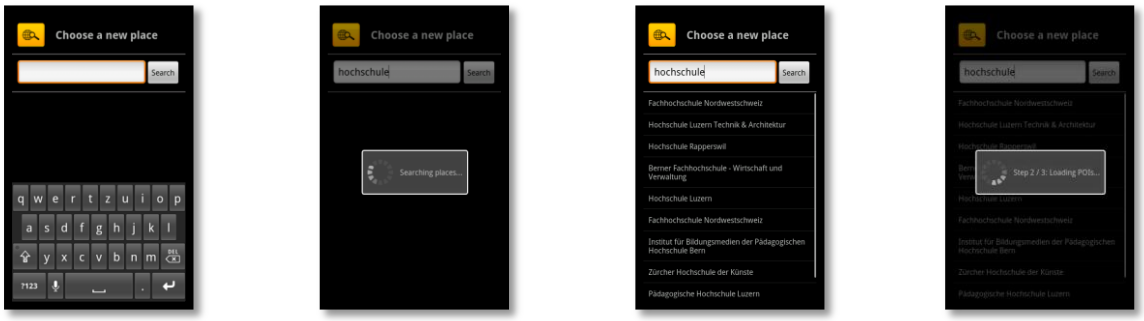


Bild 55: (von links nach rechts) Suchbegriff eingeben, Suche läuft, Resultate, Download der Daten

5.3.5 Settings – Activity

In diesem Activity können Einstellungen für die ganze Applikation gemacht werden. Diese werden dauerhaft gespeichert, damit bei erneutem Start die gleichen Einstellungen geladen werden. Folgende Einstellungen stehen zur Verfügung:

- GPS ein- / ausschalten (Default: ausgeschaltet)
- Radar ein- / ausblenden (Default: eingeblendet)
- Statussymbol der aktuellen Lokalisierungsart ein- / ausblenden (Default: eingeblendet)
- Statusnachrichten über den Wechsel der Lokalisierungsart ein- / ausblenden (Default: eingeblendet)

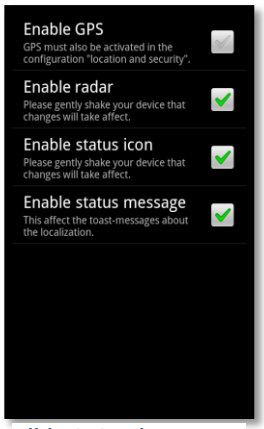


Bild 56: Settings

5.3.6 About – Activity

Dieses Activity hat das Aussehen eines Dialogfensters und zeigt Informationen über die Software, die Version und die Entwickler. Zudem verweist ein Link auf die Website des *IndoorGuide4Android.* Der Dialog wird über die aktuelle View geblendet.

5.3.7 Sprache

Das UI ist komplett in englischer Sprache verfasst. Durch die Verwendung eines XML-Files mit allen Texten ist es jedoch gut möglich, Internationalisierung einzuführen.

5.3.8 Layout

Das ganze Layout ist so gestaltet, dass man das Handy im Portrait und im Landscape Modus halten kann. Dazu mussten lediglich die Layouts für das `IndoorGuide`- sowie das `PoiInformation`-Activity in beiden Varianten separat erstellt werden. Alle anderen werden automatisch korrekt dargestellt.

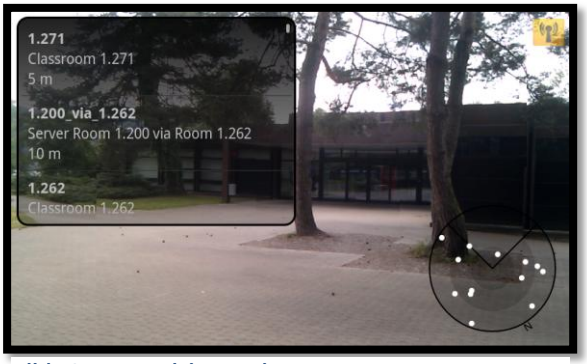


Bild 58: AR-Ansicht Landscape

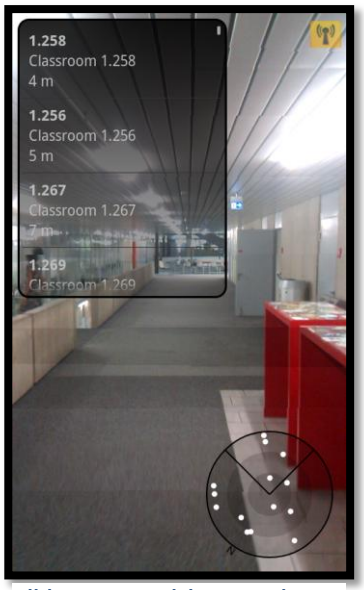


Bild 57: AR-Ansicht Portrait

5.3.9 Exceptions

Für die Ausgabe von Fehlermeldungen wurden Dialoge erstellt, die ein normaler Benutzer verstehen kann. Zudem stehen auch Details drin, die dem Entwickler helfen können, das Problem zu beheben.

Bild 59: Exception

5.4 Automatische Testverfahren

Während der Entwicklung wurden automatisierte Unit Tests eingesetzt. Als Werkzeug wurde JUnit 4 eingesetzt, welches sich gut in die Eclipse Entwicklungsumgebung einbinden lässt.

Die Tests wurden hauptsächlich für den Business- und Persistence-Layer gemacht. Gewisse Komponenten konnten jedoch nicht getestet werden, da diese von der Hardware abhängig waren. Zudem können Activities bei Android nur beschränkt getestet werden und somit wurde mehr Wert auf ausführliche Systemtests gelegt.

Code welcher die Android API benutzt kann NUR im Emulator oder auf dem Gerät selbst getestet werden. Klassen aus dem android.jar können nicht ausserhalb einer Android-Umgebung laufen gelassen werden. „Android ist nicht Java – das Ziel ist ein Android-Gerät, keine Java Virtual Machine". Informationen dazu sind zu genüge in Internetforen[20] zu finden.

5.5 Systemtests

Die Systemtests wurden am Anfang eines Sprints definiert und beim Abschluss zusammen mit den vorherigen Systemtests durchgeführt.

5.5.1 Release 0.1

ID	Testfall	Bemerkung zum Testergebnis	✓
01	Hauptactivity wird gestartet	-	
02	Aboutscreen & Settingscreen können übers Menü angezeigt werden.	Aboutscreen wird als Dialog-Activity, Settingscreen als Setting-Activity geöffnet	
03	Updatethread wird gestartet und liefert regelmässig eine (leere) Koordinate.	Es wird regelmässig eine Koordinate mit Lat 0.0, Lon 0.0 geliefert.	

5.5.2 Release 0.2

ID	Testfall	Bemerkung zum Testergebnis	✓
04	Kamera wird korrekt im Landscape- & Portrait-Modus angezeigt.	Bei Nexus One ok. Beim Milestone wird die Kamera im Portrait-Modus gedreht angezeigt. Einträge in Internetforen und das Debugging bestätigte, dass die Kamera beim Milestone-Handy im Portrait-Modus nicht verwendet werden kann.	
05	Fingerprints werden in die Datenbank geladen.	Fingerprints werden heruntergeladen und in die Datenbank gespeichert.	

[20] Quelle zur Testing-Problematik aus dem Android Developers Forum:
- http://groups.google.com/group/android-developers/browse_thread/thread/a680b65f287e1b8c
- http://markmail.org/message/yasuuw3lh5ccoiwl#query:android%20junit%20RuntimeException%20stub!+page:1+mid:or77qx4bz73xzbve+state:results

		Dabei werden sie bei erneutem Herunterladen noch nicht überschrieben sondern zusätzlich angelegt. Dieses Problem liegt bei IndoorWPS und wird den Entwicklern weiter kommuniziert.	
06	Standort-Koordinate wird korrekt angezeigt.	Die Koordinate wird angezeigt, ist jedoch teilweise noch etwas ungenau. Dies soll mit mehr Fingerprints und einem Durchschnitt mit mehr Messungen verbessert werden.	
07	Die Suche eines Places funktioniert mit folgenden Eingaben: „Rapperswil", „Hochschule Rapperswil", „HSR"	Die Suche war mit all diesen Begriffen, ausser „HSR", erfolgreich. „HSR" wurde nicht gefunden, da die Schule nicht mit dieser Abkürzung auf OSM getagt wurde. Suche dauert noch etwas lang.	

5.5.3 Release 0.3

ID	Testfall	Bemerkung zum Testergebnis	✓
08	POIs werden über WFS heruntergeladen und in einem JSON-File mit den korrekten Daten zurückgegeben.	Die Daten werden korrekt im JSON-File geschickt.	
09	Radar zeigt die POIs korrekt an. Es sollen für den Test vier Eckpunkt-Koordinaten des Gebäude 1 verwendet werden, um die Platzierung und Rotation zu testen.	-	
10	Moduswechsel zwischen AR- & Kartenansicht funktioniert.	Der richtige Winkel musste noch herausgefunden und justiert werden.	

5.5.4 Release 0.4

ID	Testfall	Bemerkung zum Testergebnis	✓
11	POIs werden nach Distanz sortiert in der POI-Liste angezeigt. Da die POIs noch nicht in der Datenbank sind, sollen Mock-Daten verwendet werden.	*Anmerkung nach einem späteren Test:* funktioniert nun auch mit den echten Daten.	
12	IndoorWPS funktioniert nach Umstellung immer noch zuverlässig.	Standortbestimmung wurde sogar noch verbessert.	
13	Suche funktioniert immer noch korrekt. Suchbegriffe sind: „Hochschule", „Rapperswil"	Suchdauer wurde gegenüber früher sogar noch verkürzt. Resultate werden korrekt angezeigt.	

14	POI-Informationen werden korrekt angezeigt.	Das Layout muss noch verbessert werden, bei längeren Bemerkungstexten verschiebt sich das Layout unschön. *Anmerkung nach einem späteren Test:* funktioniert nun auch mit längeren Bemerkungen (-> Scrollleiste)	
15	Daten (Places, POIs und Maps) werden korrekt in der Datenbank abgespeichert.	Daten werden korrekt abgespeichert, wobei noch keine Kartendaten vorhanden sind. Das einzige Problem war der Update, welcher die Daten nicht wirklich ersetzte. Beides wird zu einem späteren Zeitpunkt getestet. *Anmerkung nach einem späteren Test:* auch die Maps werden korrekt abgespeichert und der Update funktioniert auch ordnungsgemäss.	

5.5.5 Release 0.5

ID	Testfall	Bemerkung zum Testergebnis	✓
16	Multimedia-Links werden im externen Browser geöffnet.	-	
17	Nach der Auswahl eines Places werden dessen Daten heruntergeladen und abgespeichert.	Funktioniert eigentlich, wobei noch keine Progress-Dialoge angezeigt werden. *Anmerkung nach späteren Test:* Nun funktioniert auch das Anzeigen eines Progress-Dialogs mit 3 Schritten.	
18	Kompass zeigt nach Norden.	-	
19	Mock-Karte wird in korrekter Grösse angezeigt und kann verschoben werden.	Das Verschieben ruckelt noch, muss noch verbessert werden. *Anmerkung nach späteren Test:* Verschieben funktioniert nun auch gut.	

5.5.6 Release 0.6

ID	Testfall	Bemerkung zum Testergebnis	✓
20	POIs werden am korrekten Ort auf der Karte platziert.	-	
21	POIs können auf der Karte angewählt werden und die Informationen werden angezeigt.	Das Auswählen geht noch nicht so gut, dies muss noch verbessert werden. *Anmerkung nach späteren Test:* Anwählen funktioniert nun gut.	
22	Standort wird korrekt angezeigt und verschiebt sich, wenn man sich bewegt.	-	
23	GPS kann genutzt werden.	-	
24	GPS und IndoorWPS wechseln automatisch zum besseren Dienst.	Dem Benutzer muss noch kommuniziert werden, welcher Dienst gerade aktiv ist. *Anmerkung nach späteren Test:* Statussymbol wird nun angezeigt.	
25	Places werden aufgelistet und können gewählt, aktualisiert und gelöscht werden.	-	
26	Bei Programmstart wird der zuletzt aktive Place geladen.	Wird ebenfalls kurz angezeigt.	
27	Aktueller Place wird mit GPS automatisch gesucht und unter einem eigenen Namen gespeichert.	Noch Probleme, wenn kein Signal gefunden wird. *Anmerkung nach späteren Test:* Problem gelöst, indem nach 15 Sekunden Suchzeit abgebrochen wird.	
28	Nur die POIs des geladenen Places werden in der Liste angezeigt.	-	
29	Progress-Dialoge werden bei der Suche und beim Herunterladen angezeigt.	Wenn bei der Suche mit der Enter-Taste auf der Softtastatur gestartet wird, bricht der Dialog danach nicht ab. *Anmerkung nach späteren Test:* Problem ist behoben.	

5.5.7 Release 0.7

ID	Testfall	Bemerkung zum Testergebnis	✓
30	Karten herunterladen funktioniert (auch bei mehreren Karten).	Bei zu vielen Karten wird OutOfMemoryException geworfen. Dies konnte noch so gut wie möglich eingeschränkt werden. Zudem wurden die Karten verkleinert.	
31	Es werden nur die aktuell möglichen Karten angezeigt (je nach Standort).	-	
32	Die Karte kann ausgewählt werden.	-	
33	Datenbank auf der SD-Card funktioniert (auch mit grösseren Datenmengen).	-	
34	Settings können gesetzt werden, wirken sich entsprechend aus und werden persistiert.	Beim Neustart der Applikation sind immer noch die gleichen Einstellungen aktiv.	
35	Compass-Tilt-Detector gibt die Meldung aus, falls der Kompass kalibriert werden muss.	Konnte leider nicht getestet werden, da wahrscheinlich die Hardware diese Funktionalität (über Sensorstatus) noch nicht unterstützt. Wird evt. bei neueren Handys irgendwann mal funktionieren.	
36	Exceptions werden in einer Dialogbox angezeigt.	-	
37	Statussymbol wird entsprechend dem gewählten Lokalisierungsdienst angezeigt. Beim Wechsel wird eine Meldung ausgegeben.	-	

6. Resultate und Weiterentwicklung

6.1 Resultate

Siehe Kapitel „6. Resultate und Ausblick" im Teil I.

6.2 Mögliche Weiterentwicklungen

6.2.1 Genauigkeitsangabe

Momentan wird auf der Karte nur ein statischer blauer Kreis um den aktuellen Standort gezeigt. GPS liefert zwar einen Genauigkeitswert, jedoch fehlt dieser bei IndoorWPS. Sobald dort auch ein brauchbarer Wert geliefert wird, kann auf der Karte ein dynamischer Kreis angezeigt werden.

6.2.2 Fingerprints anzeigen

Eine kleinere Weiterentwicklung wäre das Anzeigen der Fingerprints auf der Karte. Dies könnte dem Anbieter eine Hilfe bei der Fingerprinterfassung sein, da damit die Verteilung der erstellten Fingerprints im Gebäude übersichtlich dargestellt würde. Da die Anzeige ähnlich wie bei den POIs gemacht werden könnte, ist der Implementationsaufwand relativ gering. Natürlich müsste dieses Feature standardmässig ausgeschaltet sein.

6.2.3 POIs filter

Da momentan alle POIs eines Places in der Liste angezeigt werden, wäre eine Filtermöglichkeit sinnvoll. Durch die vom IndoorWPS gelieferten Informationen ist zum jetzigen Zeitpunkt noch nicht bekannt, in welchem Gebäude oder Stockwerk sich der Benutzer befindet. Wenn diese Information zu einem späteren Zeitpunkt vorhanden ist, könnte man nach diesen Kriterien filtern und nur die POIs des aktuellen Stockwerks und Gebäude anzeigen.

6.2.4 Automatisierte Kartenwahl

Da die automatisierte Kartenwahl noch nicht realisiert werden konnte, könnte dies ebenfalls implementiert werden, wenn die zusätzlichen Informationen über Stockwerk und Gebäude vom IndoorWPS geliefert werden. Es ist jedoch noch herauszufinden, ob beispielsweise die Stockwerkinformation genügend genau ist, um diese auf die Karte anwenden zu können. Wenn es dabei zu ungewollt vielen Kartenwechseln und unzuverlässigen Kartenwahlen kommen würde, müssten wohl andere Wege eingeschlagen werden.

6.2.5 Wegdistanzen berechnen

Die momentan angegebene Luftdistanz zu den POIs kann natürlich sehr irritierend sein, wenn der POI hinter einer Wand oder sogar in einem anderen Stockwerk liegt. Der Weg zu diesem wäre dann viel länger als erwartet und auch die Sortierung in der Liste hilft nur beschränkt. Um dies zu optimieren wären Routeninformationen (theoretisch über OSM beziehbar) und entsprechende Routenalgorithmen nötig. Dies wäre eine umfangreiche Weiterentwicklung, die gut geplant sein sollte.

6.2.6 POIs vom Handy aus erfassen

Damit der Anbieter die POIs nicht über einen PC erfassen muss, wo er die Koordinate mühsam herausfinden muss, könnte eine Weiterentwicklung Richtung POI-Erfassung auf dem Handy laufen. Dies könnte unter Umständen mit der Fingerprinterfassung verbunden werden. Für dieses Feature ist ein Upload auf den OSM Server resp. IndoorWPS Community Server nötig.

6.2.7 POIs mit Notizen versehen

Um die Applikation noch interaktiver zu gestalten, könnte man dem Benutzer anbieten, Kommentare und Notizen zu hinterlassen. Beispielsweise wäre dies in einem Museum attraktiv, wenn der Besucher seine Meinung bei den Ausstellungsstücken anheften und die Meinung anderer lesen könnte. Dies würde natürlich ebenfalls einen Upload auf den OSM Server erfordern. Zudem müsste die POI-Struktur erweitert werden. Es ist auch fraglich, ob solche Daten überhaupt von OSM zugelassen wären. Evt. muss für diese Weiterentwicklung eine andere Lösung zur Datenhaltung gefunden werden.

6.3 Voraussetzungen für gewisse Weiterentwicklungen

Einige der oben genannten Weiterentwicklungen setzen die Weiterentwicklung des IndoorWPS voraus. Es ist also darauf zu achten, dass folgende Punkte zuerst beim IndoorWPS implementiert wurden:

- IndoorWPS gibt der Location die Genauigkeit mit.
- Die Location enthält Informationen über das Stockwerk und das Gebäude.

Mit diesen zwei Voraussetzungen können die Features „Genauigkeitsangabe" und „POIs filtern" relativ einfach umgesetzt werden. Zudem kann damit die „automatisierte Kartenauswahl" umgesetzt werden.

6.4 Codestatistik

- Anzahl Packages: 10
- Anzahl Klassen: 41
- Anzahl Methoden: 277
- Anzahl Lines of Code: 3680

7. Anleitungen und Tutorials

7.1 Installation

Die Applikation IndoorGuide4Android kann über die Website http://dev.ifs.hsr.ch/indoorguide4android als .apk heruntergeladen werden. Speichern Sie das .apk-File auf der SD-Card und installieren Sie die App mit einem File Explorer (z.B. ES File Explorer).

Sie sollten die App auch über den Android-Market finden und installieren können.

7.2 Benutzeranleitung für den IndoorGuide4Android

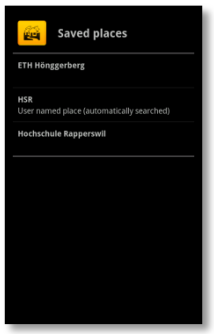
Hier folgt eine Anleitung, wie der IndoorGuide4Android zu bedienen ist.

Starten

Starten Sie den IndoorGuide4Android nach der Installation über das Icon im Android-Applikationsmenü. Beim Start wird direkt der zuletzt verwendete Place geladen.

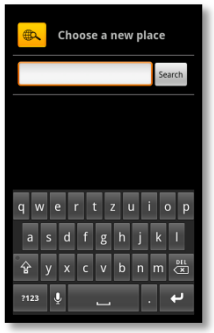
Place wählen

Um den IndoorGuide4Android an einem bestimmten Ort zu verwenden, müssen Sie den entsprechenden Place (Ort) laden. Bitte beachten Sie, dass der IndoorGuide4Android nur an Orten verwendet werden kann, an denen er vorgesehen und vom Anbieter angeboten wird.
Drücken Sie die Menütaste und wählen Sie „Choose place" aus. Nun haben Sie drei Möglichkeiten:



Neuen Place suchen

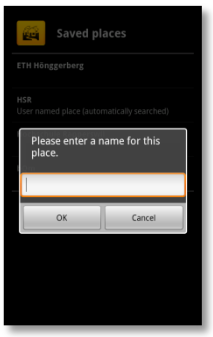
Drücken Sie erneut die Menütaste und wählen Sie „Choose a new place" aus. Nun erscheint ein Fenster mit einem Suchfeld. Tippen Sie den gewünschten Ort ins Suchfeld ein. Nachdem Sie die Suche gestartet haben, erscheint eine Liste mit möglichen Places. Wählen Sie den gewünschten Place oder starten Sie eine neue Suche, falls der Begriff nicht die gewünschten Treffer brachte.



Gut zu wissen: *Für das Herunterladen der Daten ist natürlich eine Internetverbindung nötig. Der Vorgang kann je nach Datenmenge länger dauern. Brechen Sie den Vorgang bitte nicht ab. Falls die Daten nicht heruntergeladen werden könnten, würde eine Fehlermeldung erscheinen.*

Automatisch den Place suchen

Wenn Sie sich an einem Ort befinden, an dem Sie wissen, dass Daten für den IndoorGuide erfasst sind, können Sie automatisch nach dem Place suchen. Drücken Sie dafür erneut die Menütaste und wählen Sie „Automatically choose the place" aus. Nun beginnt die Suche nach dem Place. Wenn ein Place gefunden wurde, kommt die Aufforderung, einen Namen einzugeben. Benennen Sie den Place so, dass Sie ihn diesem Ort, an dem Sie sich befinden, später wieder zuordnen können. Nach Bestätigung mit OK beginnt das Herunterladen der Daten.



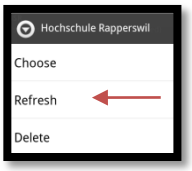
Gut zu wissen: *Um den Place automatisch wählen zu können muss eine GPS-Verbindung bestehen oder die Lokalisierung über WLAN funktionieren.*
Für das Herunterladen der Daten ist eine Internetverbindung nötig. Der Vorgang kann je nach Datenmenge länger dauern. Brechen Sie den Vorgang bitte nicht ab. Falls die Daten nicht heruntergeladen werden könnten, würde eine Fehlermeldung erscheinen.

Einen gespeicherten Place laden

Um einen schon früher gespeicherten Place zu laden drücken Sie auf den betreffenden Place.

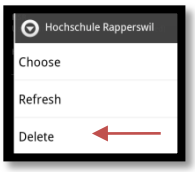
Place aktualisieren

Um einen schon gespeicherten Place zu aktualisieren, drücken Sie die Menütaste und wählen Sie „Choose place" aus. Drücken Sie nun lange auf den zu aktualisierenden Place und wählen Sie dann aus dem Kontextmenü den Punkt „Refresh".



Place löschen

Gehen Sie wie beim Aktualisieren vor, nur wählen Sie hier „Delete" aus dem Kontextmenü. Bestätigen Sie die Sicherheitsabfrage mit „Yes".



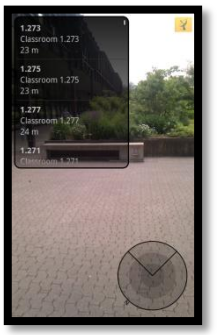
IndoorGuide benutzen

Der IndoorGuide besitzt zwei verschiedene Hauptansichten. Wenn Sie das Handy senkrecht halten, erscheint die AR-Ansicht (Kamera). Wenn Sie das Handy horizontal halten, erscheint die Kartenansicht. Bei beiden Ansichten finden Sie die Points-of-Interest (POIs) aus der näheren Umgebung.

AR-Ansicht

POI-Liste: Im linken oberen Bereich finden Sie die POI-Liste, welche nach Distanz zu Ihrem aktuellen Standort sortiert ist. Drücken Sie auf ein POI, um dessen Informationen anzuzeigen.

Radar: Im rechten unteren Bereich finden Sie den Radar, welcher die POIs aus unmittelbarer Umgebung anzeigt. Dabei sind die POIs in Blickrichtung oben angezeigt.



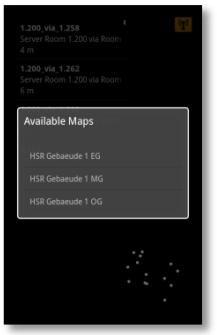
Status: In der rechten oberen Ecke befindet sich das Statussymbol. Dieses kann drei Zustände annehmen:

-  „No localization possible!“: Ihre Position kann nicht bestimmt werden. Entweder kann die WLAN-Lokalisierung an diesem Ort nicht gemacht werden, das WLAN ist ausgeschaltet oder es besteht keine GPS-Verbindung.
-  „Localization with WLAN“: Die Lokalisierung wird über WLAN gemacht.
-  „Localization with GPS“: Die Lokalisierung wird über GPS gemacht.

Kartenansicht

Kartenauswahl: Beim ersten Wechseln in die Kartenansicht wird angezeigt, dass eine Karte geladen werden soll. Drücken Sie dazu lange auf den Bildschirm und wählen Sie eine Karte aus, falls eine an Ihrem aktuellen Standort verfügbar ist.

Karte: Die Karte wird an Ihren aktuellen Standort zentriert. Ihr Standort wird mit dem grünen Android  gekennzeichnet. Der blaue Kreis um Ihren Standort zeigt, in welchem Rahmen sich die Ungenauigkeit befinden kann. Die gelben Symbole  kennzeichnen die POIs.

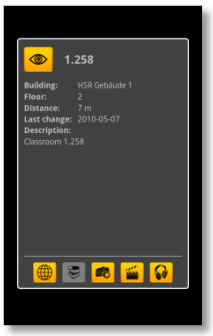


POI-Auswahl: Drücken Sie auf einen POI, um dessen Informationen anzuzeigen.

Kartenwechsel: Wenn Sie sich z.B. in ein anderes Stockwerk begeben und die Karte nicht mehr stimmt, können Sie mit langem Drücken auf den Display die richtige Karte laden.

POI-Informationen anzeigen

Wenn Sie ein POI angewählt haben, öffnet sich ein neues Fenster, in dem Sie die Informationen zu diesem POI finden. Wenn längere Beschreibungen keinen Platz haben, können Sie diese scrollen.
Die Buttons im unteren Bereich des Fensters sind grau, wenn kein Link hinterlegt ist und gelb, wenn sie nutzbar sind. Wenn Sie darauf drücken, öffnet sich das Browserfenster mit dem entsprechenden Link zum POI. Hier noch die Beschreibung der verschiedenen Links:

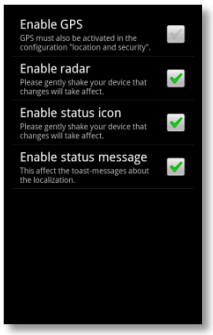


- Website
- Wikipedia
- Bild
- Film
- Audio

Einstellungen machen

Drücken Sie die Menütaste und wählen Sie „Settings" aus. Hier können Sie nun gewisse Einstellungen vornehmen:

- „Enable GPS": diese Einstellung ist standardmässig ausgeschaltet. Sie muss aktiviert werden, wenn die Lokalisierung (auch) über GPS gemacht werden soll. Denken Sie auch daran, in den Android-Einstellungen GPS zu erlauben.
- „Enable radar": diese Einstellung können Sie deaktivieren, wenn Sie den Radar ausblenden wollen.
- „Enable status icon": diese Einstellung können Sie deaktivieren, wenn Sie das Statussymbol (in der AR-Ansicht rechts oben) ausblenden wollen.
- „Enable status message": diese Einstellung können Sie deaktivieren, wenn Sie die Meldungen über den Wechsel des Lokalisierungsstatus ausblenden wollen.



Um die letzten drei Einstellungen aktivieren zu können, müssen Sie das Handy vorsichtig schütteln. Sie können auch einen kurzen Wechsel zwischen AR- & Kartenansicht machen.

7.3 Tutorial zur POI-Erfassung

Hier folgt ein kurzes Tutorial fürs Erfassen von POIs. Dazu wird der Java-Editor „JOSM" (Java OpenStreetMap) eingesetzt. Natürlich kann die Erfassung auch über die OSM Website mit dem „Potlatch"-Editor gemacht werden, was jedoch umständlicher und weniger intiutiv zu bedienen ist.

JOSM herunterladen und installieren

JOSM kann über http://josm.openstreetmap.de heruntergeladen werden. Installieren und starten Sie danach das Tool.

Karte herunterladen

Klicken Sie nun auf „File – Download from OSM..." und suchen Sie sich auf der angezeigten Karte den Bereich, in dem Sie POIs erfassen wollen. Spannen Sie über diesen Bereich ein Rechteck auf und wählen Sie „Download".

Navigation

Mit dem Mausrad können Sie zoomen und mit gedrückter rechter Maustaste die Karte verschieben.
Unten links sehen Sie die Koordinate, an welcher sich der Mauszeiger befindet.

Neuer POI erstellen

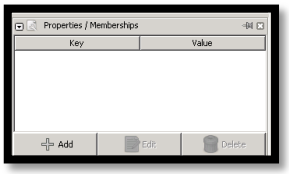
Wählen Sie das Werkzeug „Draw nodes"  . Klicken Sie nun (doppelt) an die Stelle, an der Sie den POI platzieren wollen. Der Node ist nun ausgewählt (rot) und kann auf der rechten Seite mit Informationen versehen werden.

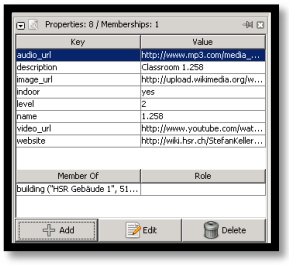
„Tags" hinzufügen

Um einen neuen „Tag" hinzuzufügen, müssen Sie auf „Add"  klicken. Fügen Sie folgende „Tags" für ein Indoor-POI hinzu:

(key -> value-Beispiel)

- name -> 1.258
- description -> Classroom 1.258
- indoor -> yes
- level -> 2
- website -> http://www.hsr.ch
- wikipedia -> http://www.hsr.ch
- image_url -> http://www.hsr.ch
- video_url -> http://www.hsr.ch
- audio_url -> http://www.hsr.ch



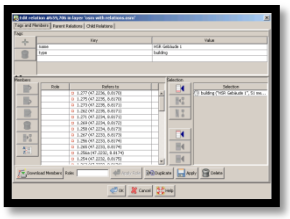


Relation erstellen

Um dem POI ein Gebäude zuzuweisen, muss eine Relation fürs Gebäude erstellt werden. Klicken Sie dazu unten auf „Create a new relation“  . Fügen Sie folgende Tags hinzu:

(key -> value-Beispiel)

- name -> HSR Gebäude 1
- type -> building

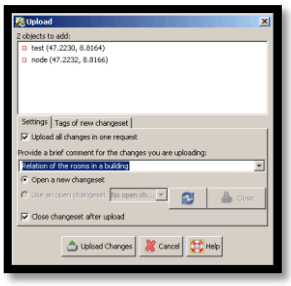


POIs einer Relation hinzufügen

Markieren Sie alle POIs, die sich in einem Gebäude befinden. Diese Nodes sollten nun alle rot markiert sein (achten Sie darauf, dass NUR die POIs ausgewählt wurden). Suchen und wählen Sie nun die passende Relation, die sich in der Liste mit Relations befinden sollte. Klicken Sie auf „Open an editor for selected relation“  . Sie sehen die ausgewählten POIs auf der rechten Seite unter „Selection”. Diese können Sie nun auf die linke Seite transferieren. Somit sind die POIs dieser Relation hinzugefügt.

Daten auf OSM speichern

Um nun die erstellten Daten auf OSM verfügbar zu machen, müssen Sie hochgeladen werden. Klicken Sie dazu auf „File – Upload data“. Nun sollten Sie alle Änderungen sehen, die übermittelt werden. Kontrollieren Sie diese bitte sorgfältig. Geben Sie noch einen kurzen Kommentar über die Änderung ab und klicken Sie dann auf „Upload Changes“.



Kontrolle auf OSM

Wenn Sie kontrollieren wollen, ob die Daten richtig übermittelt wurden, können Sie diese auf www.openstreetmap.org anschauen.

7.4 Tutorial zur Fingerprint- & Map-Erfassung

Die Tutorials zur Erfassung von neuen Fingerprints und Maps finden Sie auf der IndoorWPS-Projektwebsite: http://dev.ifs.hsr.ch/indoorwps/

ANHANG

1. ANHANG: Inhalt der CD

1.1 Applikation

- IndoorGuide4Android.apk
- *Sourcefolder*
  - Workspace
  - JavaDoc
- *Bilder*

1.2 Dokumentation

- *Sitzungsprotokolle*
- IG4A2 Bericht.docx
- IG4A2 Bericht.pdf
- IG4A2 Poster.pptx

2. ANHANG: Glossar und Abkürzungsverzeichnis

Begriff	Erklärung
Android	Ist ein Betriebssystem für mobile Geräte wie Smartphones. Wird von der Open Handset Alliance entwickelt.
Augmented Reality (AR)	Augmented Reality = Erweiterte Realität, die computergestützte Erweiterung der Realitätswahrnehmung. Häufig in Form von visueller Darstellung von Informationen in Bildern, Videos oder Kamerabild.
Backlog	Ist ein Teil von Scrum und enthält die Features der zu entwickelnden Applikation.
BLOB	Binary Large Object, ist eine Variante um Objekte in Datenbanken zu schreiben. Es wird dann der Bytewert eines Objekts wie er ist in die Datenbank geschrieben.
Bounding Box	Ist ein Rechteck bestehend aus vier Geographischen Punkten als Ecken.
Dalvik VM	Ist die bei Android eingesetzte Virtual Machine.
DTO	Data Transfer Object, sind einfache Java Objekte die innerhalb einer Applikation benutzt und manipuliert werden können.
Features	Features werden als Funktionen einer Software bezeichnet. Als Feature wird in einem Geoinformationssystem (GIS) auch die kleinste, fachlich sinnvolle Einheit innerhalb eines raumbezogenen Datenbestandes bezeichnet.
Fingerprint	Ist die Grundlage zur Positionsberechnung via WLAN. An bestimmten Punkten misst man WLAN-Signale und packt diese Daten inklusive der Position zu einem Fingerprint zusammen.

Geonames.org	Ist eine freie geographische Datenbank, auf die über diverse Webservices zugegriffen werden kann.
GIS	Geoinformationssystem
GPS	Ist ein sehr exaktes Positionierungssystem, welches mittels Satelliten die Position trackt. Es funktioniert auf der ganzen Welt.
HSR	Hochschule der Technik in Rapperswil
Hysterese	
IFS	Ist ein Institut an der HSR (Institut für Software)
IndoorWPS Community Server	Dieser Service liefert Fingerprintdaten und Karten zu Gebäuden.
JavaDoc	Kommentare im Java Source Code.
JSON	JavaScript Object Notation, ist ein kompaktes Datenformat in für Mensch und Maschine einfach lesbarer Textform zum Zweck des Datenaustauschs zwischen Anwendungen.
KML	Keyhole Markup Language, ist eine Auszeichnungssprache zur Beschreibung von Geodaten für die Client-Komponenten der Programme Google Earth und Google Maps.
Latitude	Bezeichnet die geographische Breite einer Position auf der Erde.
Location	Stellt Ihre aktuelle Position dar.
Longitude	Bezeichnet die geographische Länge einer Position auf der Erde.
Memory Leaks	Smartphones oder allgemein Computer haben einen beschränkten Adressraum (sogenanntes Memory) wird dieser überbeansprucht kommt es zu Abstürzen, sprich Memory Leaks.
OpenStreetMap	Ist ein Open Source Dienst der eigene Standards entwickelt und eine immer grössere Community erreicht. Der Dienst georeferenziert die ganze Welt.
OpenStreetMap-in-a-Box	Ist ein Projekt der HSR, es spiegelt sämtliche Daten von OSM und beschäftigt sich damit diese Daten in ein fein strukturiertes Datenbankmodel zu mappen.
Overhead	Bezeichnet die zusätzlichen Daten, die nebst den Nutzdaten im selben Stream, ausgetauscht werden müssen.
Parsen	Ist das einlesen von Datenaustausch Formaten wie XML. Dabei werden aus den Textstrukturen wieder Objekte generiert.
Place	Bezeichnet einen Ort, zum Beispiel Rapperswil.
Points-of-Interest (POIs)	Dabei handelt es sich um interessante Punkte die sich an einem geografischen Ort befinden.
POJO	Plain Old Java Object, bezeichnet ein simples Java Objekt, das mehrheitlich nur Getter- & Setter-Methoden besitzt.

Product Owner	Der Auftraggeber eines Scrumprojekts.
Progress-Dialog	Ist eine Anzeige auf dem Smartphone, welche einem über den Fortschritte der Applikation informiert.
QR-Code	Der QR-Code (quick response) ist ein zweidimensionaler Code, der von der japanischen Firma Denso Wave im Jahr 1994 entwickelt wurde.
RFID	Ermöglicht die automatische Identifizierung und Lokalisierung von Gegenständen und Lebewesen und erleichtert damit erheblich die Erfassung und Speicherung von Daten.
Scrum	Scrum ist eine agile Projektmanagement Methode.
SD-Card	Ist eine Speicherkarte eines Smartphones.
Sprint	Ist ein Teilabschnitt eines Scrum-Projektes. Bei Abschluss jedes Sprints entsteht ein lauffähiger Prototyp der zu entwickelnden Applikation.
SQLite	Ist eine Programmbibliothek, die ein relationales Datenbanksystem enthält.
Story	Bezeichnet eine kleine Teilaufgabe eines Sprints bei Scrum.
Use Case	Ist ein Anwendungsschritt, der mittels der dazugehörigen Software durchführbar sein soll.
User Interface	Bezeichnet den Teil einer Software, die der Benutzer sieht und bedient.
WFS	Unter einem Web Feature Service versteht man den internetgestützten Zugriff auf Geodaten innerhalb eines verteilten GIS.
XML	Extensible Markup Language, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten.

3. ANHANG: Literatur- und Quellenverzeichnis

3.1 Bücher und Artikel

[1] Prof. Stefan Keller, „Zitat der originalen Aufgabenstellung für die Bachelorarbeit", 2010
[2] T. Ossipova / M. Reiter, „Bachelor Thesis – IndoorGuide4Android", 2009
[3] Ed Burnette, „Hello, Android 2.1", 2010, ISBN: 978-1934356494

3.2 Links und Informationen

[4] **Layar:** http://www.layar.com
[5] **Wikitude:** http://www.wikitude.org
[6] **RMaps:** http://code.google.com/p/robertprojects
[7] **Peak.AR:** http://peakar.salzburgresearch.at
[8] **IndoorWPS:** http://wiki.hsr.ch/StefanKeller/wiki.cgi?IndoorWPS
[9] **IndoorWPS Community Server:** http://labs.geometa.info/indoorwps
[10] **VisiVis:** http://www.ifs.hsr.ch/VisiVis.6057.0.html
[11] **Computer-Vision:** http://www.technologyreview.com/computing/22218/page1
[12] **WFS:** http://www.opengeospatial.org/standards/wfs
[13] **Agiles Manifest:** http://www.agilemanifesto.org
[14] **ScrumDesk:** http://www.scrumdesk.com
[17] **PointZero:** http://wiki.hsr.ch/StefanKeller/wiki.cgi?PointZero
[18] **OSM-in-a-Box:** http://dev.ifs.hsr.ch/osminabox
[19] **OSM:** http://www.openstreetmap.org
[20] **Android Developers:** http://developer.android.com

4. ANHANG D: Originale, unterschriebene Aufgabenstellung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

HSR Hochschule für Technik Rapperswil (FH Ostschweiz), Abteilung Informatik

Bachelorarbeit Frühlingssemester 2010

IndoorGuide4Android2

Mobile Guide indoors and in campusses or museums based on Android

Autoren/Studenten:	Adrian Geiter, HSR, Abteilung Informatik Christoph Egger, HSR, Abteilung Informatik
Verantwortlicher/ Betreuer:	Prof. Stefan Keller, HSR, Abt. Informatik
Partner (Firma oder Verwaltung), externer Betreuer:	Reto Senn, bitforge AG Ltd., Rapperswil

Einführung

"Augmented Reality" (AR) ist eine neue Entwicklung, die auf Mobiles Phones (bzw. Smartphones) besonders zur Geltung kommt. Dabei wird das Kamerabild mit kontextabhängigen Informationen überlagert, beispielsweise mit Points of Interest (POIs), die als dreidimensionale Icons (Stecknadeln) an ihrem Ort schweben. Beispiele dafür sind Wikitude oder Layar. Innerhalb der AR gibt es verschiedenste Ansätze, darunter die Orientierung im Gebäude und in Anlagen ('indoor') - wie z.B. in einem Campus, Einkaufszentrum oder Museum.

Aufgabenstellung

Für diesen Ansatz soll auf Basis eines Android-Smartphones, das u.a. mit Kamera, Wifi, GPS und mit Kompass ausgestattet ist, ein Prototyp neu entworfen und realisiert werden. Es handelt sich dabei um ein mobiles Besucher-Informationssystem (Mobile Guide) innerhalb der HSR und auf dem HSR-Campus als AR-Applikation.

Zwei Anwendungsszenarien stehen im Vordergrund: Einerseits soll sich ein ortsunkundiger Besucher eines Campus' über die Räume in der Nähe informieren können. Er soll sich auch Hypertext und Multimedia-Inhalte zu naheliegenden Objekten anzeigen lassen können, dies mittels AR-Kamerabild und Karte.

Hier der Beginn der zwei Szenarien: "Ein externer Besucher kommt auf dem Campus der HSR an und sucht das Sitzungszimmer 1.227. Er richtet seine Kamera im Hauptgebäude 1 auf den Kiosk, es erscheint eine Infobox mit Öffnungszeiten und den in der Nähe befindlichen Räumen. Wenn er sein Handy waagerecht hält, erscheint ein Gebäudeplan (...)".

Im zweiten Szenario gibt es eine Job-Ausstellung im Hauptgebäude 1 mit Herstellern-Ständen. Der Ausstellungsbesucher richtet sein Handy auf einen Stand und erhält

Seite 1 von 2

Informationen dazu. Weitere Details sollen innerhalb der Arbeit erarbeitet werden. Wichtiger Bestandteil der Arbeit ist auch der Entwurf von stimmigen Bildschirmdialogen.

Folgende Lieferdokumente sollen erstellt werden (deutsch wo nichts angegeben):

- Android-Applikation, User Interface englisch (Bedienungsanleitung ist nicht nötig).
- Vollständiger compilier-bereiter Sourcecode (englisch) sowie als Download gekennzeichnetes Zip-File mit ausführbarem Bytecode.
- Installationsanleitung, falls sinnvoll (englisch).
- Technischer Bericht (deutsch) und SW-Engineering-Dokumentation, z.T. englisch.
- Allfällige Dokumente gemäss Vorgaben der Abt. I. (Plakat, ‚Abstract'), deutsch.
- Dokumentierte Demo im Web (stichwortartig, bebildert, ggf. Video).

Hinweise

- Es wird in diesem Prototyp ein Not-Always-Online-Betrieb angenommen, was die Realisierung eines Cachings bedingt.
- Der Positionierungsdienst für 'indoor' basiert auf IndoorWPS, einem anderen HSR-Forschungsprojekt, das auf Basis von Wifi-Signalen arbeitet. Es ist abzuklären, inwiefern sich ohne grossen Aufwand auch das eingebaute GPS einbinden lässt. Bei der Wifi-Positionierung ist mit ungenauen, "umherspringenden" Koordinaten zu rechnen.
- Die POIs und Gebäudepläne werden vom Benutzer wenn „Online" (ggf. vorgängig) heruntergeladen (z.B. mittels Angabe des Gebäude- oder Ortsnamen-Suche).
- Die Arbeitsweise ist agil, wo sinnvoll mit Unit-Tests. Es wird Wert auf ausgetestete Software und einfache Installation gelegt.
- Für die erfolgreich abgeschlossene Arbeit werden 12 ECTS angerechnet, d.h. es wird eine Arbeitsleistung von mind. 360 Stunden pro Person erwartet.

Randbedingungen, Infrastruktur, Termine und Beurteilung

- Randbedingungen Hardware/OS:
  - Android-Hardware
  - OS: Android/Java
- Software:
  - Java SE gem. Android
  - ANT, Eclipse IDE
- Termine und Beurteilung: gemäss Angaben auf www.i.hsr.ch.

Rapperswil, 10. März 2010

Der Betreuer

Prof. Stefan Keller

Die Studierenden