Indoor Guide

IndoorGuide4Android
Bachelor Thesis

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2009

Studenten: Ossipova Tatjana, Reiter Michel
Betreuer: Prof. Stefan F. Keller, Institut für Software

IMPRESSUM

Tatjana Ossipova
Studierende Informatik
tossipov@hsr.ch

Michel Reiter
Studierender Informatik
mreiter@hsr.ch

Prof. Stefan F. Keller
Institut für Software
sfkeller@hsr.ch
www.ifs.hsr.ch

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

Sämtliche Daten dieses Projekts und die Projektwebseite sind verfügbar unter http://dev.ifs.hsr.ch/indoorguide4android/ .

DOKUMENTINFORMATIONEN

Datum	Version	Änderung	Autor
12.06.2009	1.0	Initialversion	tossipov, mreiter

ABSTRACT

Wir leben in einer Zeit, in der die technischen Möglichkeiten zur Ortsbestimmung und Navigation immer ausgefeilter und exakter werden, ein Beispiel dafür ist die Tatsache, dass eine Vielzahl der heutigen Automobile mit einem GPS Navigationsgerät ausgestattet sind. Die nächste logische Entwicklung sollte sein, diese Technologien in die Hosentasche zu bringen um das Zurechtfinden und Navigieren auch innerhalb von Gebäuden zu ermöglichen.

Im Rahmen dieser Bachelorarbeit wurde ein Indoor Informations- und Navigationssystem für die Android-Plattform entwickelt, welches seine Position mit Hilfe von Wireless LAN Signalen feststellt. Der IndoorGuide4Android stellt Informationen über sogenannte Points of Interest (POIs) in einer Augmented Reality Ansicht dar. Das bedeutet, dass das im Display angezeigte Kamerabild mit Computerenerierten Informationen überlagert und angereichert wird. Da die Anzeige von Informationen über POIs alleine noch nicht befriedigend genug ist, verfügt der Guide deshalb über eine Navigationsfunktion, welche es einem Benutzer erlaubt, den Weg zu einem bestimmten, von ihm gewünschten, POI zu finden.

MANAGEMENT SUMMARY

Ausgangslage

Grundsätzliches Ziel

Das Ziel war die Realisierung eines Mobile Guides für Gebäude und in überdeckten Anlagen wie zum Beispiel einem Hochschul-Campus auf Basis der Android-Plattform.

Dabei galt es, die folgenden funktionale Anforderungen zu erfüllen:

- Points of Interest (POIs) und Weganweisungen in geeigneter Weise im Kamerabild räumlich darstellen (augmented reality).
- Die POI- und Routenplan-Daten werden vom Benutzer kontrolliert entweder vor Ort oder vorgängig heruntergeladen.
- Die Positionierung innerhalb eines Gebäudes findet mittels Wireless LAN Signalen statt. Zu diesem Zweck sollte die „PointZero" Library eingesetzt und von diesem Projekt als Black-Box betrachtet werden.

Weiterhin galten folgende nicht-funktionalen Anforderungen:

- Gut ausgetestete Software.
- Automatische Unit Tests.
- Continuous Integration mit automatischen Builds auf einem Buildserver.
- Gute Dokumentation der Arbeit mit einem technischen Bericht über die gesammelten Erfahrungen mit dem Android Framework und der Entwicklung von Applikationen für die Android Plattform.

Ähnliche Arbeiten

Wikitude AR Travel Guide ist ein mobiler Reiseführer für die Android Plattform der auf standortbezogenen Wikipedia- und

Qype-Artikeln und Panoramio Photos beruht. Gefundene Artikel können unter Anderem in einer „Augmented Reality" Kameraansicht dargestellt werden. Dabei wird das Kamerabild des Geräts mit computergenerierten Daten überlagert.

AndNav2 ist eine Navigations-Lösung für Android, die auf die Daten des Open Source Projekts OpenStreetMap zugreift. Zusätzlich wird bei AndNav2 die Service-Infrastruktur von OpenRouteService.org genutzt.

Vorgehen

Das Projektmanagement sowie die Software-Entwicklung wurde anhand moderner und agiler Methoden durchgeführt. Für die Entwicklung der Software wurde ein inkrementeller Ansatz verwendet, so dass die Applikation aus kleineren Releases (Prototypen) besteht.

Es wurde festgelegt, dass die Applikation mittels fünf Prototypen inkrementell erstellt und erweitert werden soll, wobei die Prototypen aufeinander aufbauen.

Risiken

Wie jedes Projekt war auch dieses mit gewissen Risiken verbunden, wobei nachfolgend nur die grössten Risiken aufgeführt sind:

- Positionsbestimmung innerhalb eines Gebäudes nicht verlässlich
  - Position kann gar nicht bestimmt werden.
  - Positionsabweichung ist zu gross.
  - Koordinaten „springen" umher.
- Hardware Sensoren ungenau
  - Kompass ist zu empfindlich und neigt zu Abweichung.

Involvierte Personen

Name	Firma	Rolle
Hr. Stefan Keller	HSR	Betreuer der Bachelorarbeit
Hr. Reto Senn	bitforge AG	Externer Partner
Fr. Tatjana Ossipova	HSR	Studentin
Hr. Michel Reiter	HSR	Student
Hr. Sachin Patney	HSR	Praktikant & Entwickler der „PointZero" Library

Existierende Grundlagen

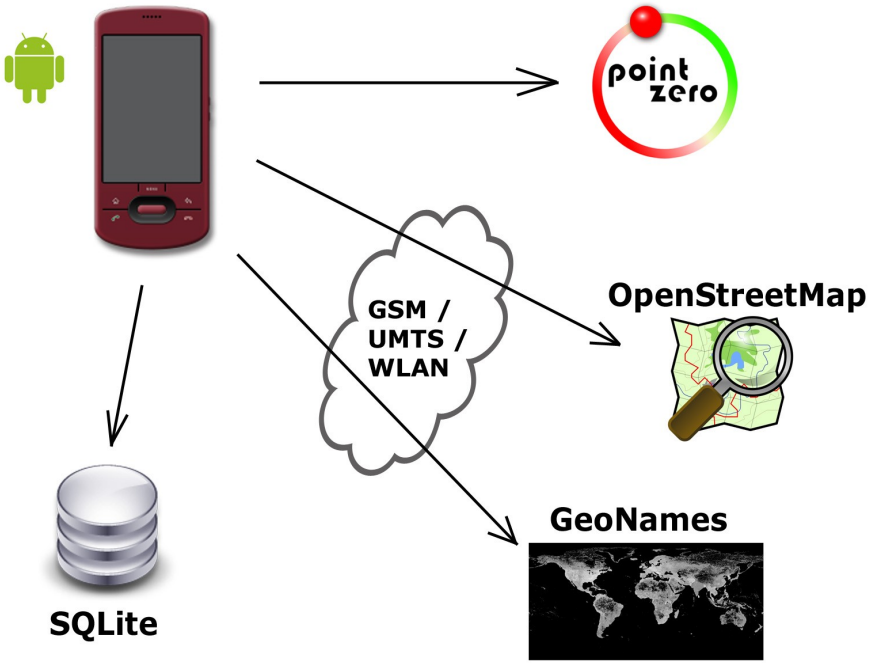
IndoorWPS stellt die grundlegende Infrastruktur zur Positionsbestimmung über Wireless LAN bereit und wurde an der HSR entwickelt.

OSMNavigation Library ist in eine in Java geschriebene Library, die diverse Algorithmen zur Routenberechnung sowie Funktionalität zur Navigation implementiert.

Ergebnisse

Resultat

Während der Arbeit wurde eine, gemäss unseren Recherchen, einzigartige Indoor Informations- und Navigations-Applikation erstellt, welche ihre Daten von frei verfügbaren Informationsquellen bezieht.



Die obige Grafik zeigt die Interaktion der Applikation mit diversen Datenquellen auf.

Bewertung

Die Applikation ist grundsätzlich universell einsetzbar und hat das Potential, die Grundlage für diverse Anwendungen zu bilden, da es grundsätzlich keine Rolle spielt, für welche Points of Interest sie verwendet wird.
Points of Interest können einzelne, spezielle Räume wie Sitzungszimmer oder aber auch einzelne Gegenstände in einem Raum sein – man denke an einen mobilen Museumsführer, welcher Informationen zum gerade durch die Kamera „angepeilten" Exponat liefert oder den Benutzer gezielt zu einem gewünschten Exponat führt.

Zielerreichung

Der IndoorGuide4Android erreicht 13 von 14 gesetzten Zielen was einer Zielerreichung von rund 93% entspricht.

Abweichungen

Das 14. Ziel, die verlässliche Positionsbestimmung via Wireless LAN ist zu zirka 80% erreicht. Grundsätzlich wird eine Position über WLAN festgestellt, diese weicht jedoch teilweise stark von der tatsächlichen Position ab. Dieses Verhalten erfordert weitergehende Abklärungen seitens „PointZero".

Ausblick

Gelerntes

- Neue Technologie
  - Android SDK
- Implementation von Mobile Applikationen
- Vertiefen des Wissens im Bereich Software Engineering
- Erfahrungen im Bereich des Projektmanagements (agiles Vorgehen)
- Funktionalität der Wireless LAN Positionierung
- Services wie
  - OpenStreetMap
  - Geonames.org
- Routing Algorithmen

Verbleibende Probleme

Das einzige verbleibende Problem ist die zuverlässige Positionsbestimmung über Wireless LAN Signaldaten, allerdings ist dies ein Problem von „PointZero" und nicht von diesem Projekt – allerdings hängt dieses Projekt von PointZero ab.

Was würden wir anders machen?

Eine weitere, ähnliche Arbeit, würden wir als Machbarkeitsstudie definieren und von Anfang an konkrete Ziele setzen, was genau erreicht werden sollte.
Weiterhin müsste die Teamgrösse für die Umsetzung der Arbeit aufgrund der agilen Projektmethode um eine Person erweitert werden, damit die Dokumentationen korrekt nachgeführt werden können.

INHALTSVERZEICHNIS

TEIL I: TECHNISCHER BERICHT

1. Einführung

1.1. Problemstellung

Wir leben in einer Zeit, in der die technischen Möglichkeiten zur Ortsbestimmung und Navigation immer ausgefeilter und exakter werden, ein Beispiel dafür ist die Tatsache, dass eine Vielzahl der heutigen Automobile mit einem GPS Navigationsgerät ausgestattet sind. Die nächste logische Entwicklung sollte sein, diese Technologien in die Hosentasche zu bringen um das Zurechtfinden und Navigieren auch innerhalb von Gebäuden zu ermöglichen.

1.2. Aufgabenstellung[1]

Einführung

Die Orientierung im Gebäude und in Anlagen - wie z.B. in einem Campus oder Einkaufszentrum - könnte kinderleicht werden, wenn man ein Mobile als Begleiter dabei hätte, das man vor sich hin hält: Das Kamerabild wird dabei überlagert mit Informationen über Points of Interest (POI), die über einen eigenen Server geholt werden. Eine Voraussetzung dazu ist jedoch, dass der eigene Standort und die Richtung bekannt sind.

Mit der Android-Plattform von Google wurden die Voraussetzungen geschaffen um „Augmented Reality Kamera-Ansichten" zu realisieren, wie dies Wikitude (www.mobilizy.com) zeigt. Zudem kann auf den Ergebnissen von Routenplanern wie AndNav 2 (www.andnav.org) aufgebaut werden.

Das Ziel dieser Arbeit ist die Realisierung eines Mobile Guides in Gebäuden (engl. indoor) und in überdeckten Anlagen (z.B. ein Hochschul-Campus) auf Basis der Android-Plattform, die werkseitig u.a. mit Sensoren wie Kamera, WLAN (ev. GPS), Kompass (Kamerarichtung) und Accelerometer (Kameraneigung) ausgerüstet ist. Damit sollen auch Erfahrungen mit Android Apps und 'echter' Hardware (z.Zt. G1 T-Mobile) dokumentiert werden.

Hier ein mögliches Szenario: „Ein externer Besucher kommt auf dem Campus der HSR an und sucht (zu Fuss) das Sitzungszimmer 1.227. Er gibt Angaben zum Sitzungszimmer (= POI) ein. Er richtet seine Kamera auf und es erscheinen POIs im Kamerabild. Zudem wird ein dreidimensionaler Richtungspfeil (geradeaus, vorne links, vorne rechts) angezeigt...". Weitere Details sollen innerhalb der Arbeit erarbeitet werden.

Aufgabenstellung

Folgende Software und Berichte sollen erstellt werden:

- Android-Applikation, die im Kamerabild POIs und Weganweisungen in geeigneter Weise räumlich darstellt.
- Die POI- und Routenplan-Daten werden vom Benutzer kontrolliert heruntergeladen (entweder gleich vor Ort oder vorgängig, z.B. mittels Angabe des Gebäude, bzw. Campus-Namens). Dies erfolgt über eine Gebäudesuche

1 Zitat der originalen Aufgabenstellung für die Bachelorarbeit von Stefan Keller

(vgl. Maps im IndoorWPS-Server-Projekt).

- Technischer Bericht über Erfahrungen mit dem Android-Framework.

Lieferdokumente (englisch wo angegeben, sonst deutsch):

- Installationsanleitung, falls sinnvoll (englisch).
- Dokumentierte Demo im Web (stichwortartig und bebildert).
- Gesamter compilier-bereiter Sourcecode (englisch) inkl. Programmdokumentation sowie als Download gekennzeichnetes Zip-File mit ausführbarem Bytecode.
- Technischer Bericht und SW-Engineering-Dokumentation, z.T. englisch.
- Allfällige Dokumente gemäss Vorgaben der Abt. I. (Plakat, ‚Abstract').

Hinweise

- Die Positionierungsdaten kommen einerseits vom WLAN-Netz (IndoorWPS3) und andererseits vom eingebauten GPS. Bei der WLAN-Positionierung ist mit ungenauen, „umherspringenden" Koordinaten zu rechnen.
- Die Arbeitsweise ist agil, wo sinnvoll mit Unit-Tests. Es wird Wert auf ausgetestete Software und einfache Installation gelegt.
- Eine Bedienungsanleitung ist nur in begründeten Fällen gefordert.
- 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. Administrativ

Wann	Was
05.06.2009	Abgabe der Kurzbeschreibung der Arbeit an Betreuer.
10.06.2009	Abgabe der genehmigten und korrigierten Kurzbeschreibung an Sekretariat der Abteilung I.
12.06.2009 12:00	Abgabe des Berichtes an Betreuer.
12.06.2009 17:00	Abgabe des A0 Plakats an Abteilungssekretariat.

Weitere Termine nach dem 12.06.2009 sind auf der Webseite der HSR [HSR] zu finden.

1.3.2. Technologisch

- Zielplattform: Android Plattform die auf dem Google G1 Smartphone ausgeführt wird.
- Google G1 Smartphone
  - HTC G1
  - Prozessor: 528 MHz
  - Bildschirm: 3.2" TFT Touchscreen, 320 x 480 Pixel, 65 000 Farben
  - Kamera: 3.2 Megapixel (2048 x 1536 Pixel), Autofokus
  - Speicher: 192 MB RAM, 256 MB ROM. Mit microSD erweiterbar
  - Sensoren: Hochempfindlicher GPS Empfänger, digitaler Kompass, Accelerometer
  - Schnittstellen: WiFi (IEEE 802.11 b/g), Bluetooth 2.0 (nur Handset), MiniUSB, GSM, UMTS (GPRS Klasse 10, EDGE Klasse 10, HSDPA 7.2 Mbps)
- Lokalisierung durch „PointZero" Library. Positionsungenauigkeit liegt bei zirka 4 Metern.

1.3.3. Entwicklungsumgebung

- Linux Betriebssystem auf Entwicklungsrechnern zwecks einfacherer Entwicklung mit dem Smartphone
- Eclipse 3.4.1 IDE
- Eclipse Plugins
  - Zum Android SDK gehörendes Eclipse Plugin
- Java 1.6
- Android SDK 1.5

1.4. Vorgehen, Aufbau der Arbeit

Aufgrund keiner Erfahrungen mit der Android-Plattform wurden zu Beginn der Bachelorarbeit einige kleine Testprogramme geschrieben um die grundlegenden Mechanismen der Softwareentwicklung für die Android-Plattform kennen zu lernen. Beispielsweise wurde getestet, wie die einzelnen Gerätesensoren mit einem Programm ausgelesen werden können.

Parallel zur Entwicklung der Testprogramme wurde iterativ und agil an der Erstellung des eigentlichen IndoorGuides gearbeitet: Anforderungen und Use Cases definieren, Gedanken zu Datenquellen und dem Erhalt von Daten aus diesen Quellen machen sowie die Entwicklungs- und Testumgebung aufsetzen (Buildserver, Subversion Repository).

Das grösste Risiko dieser Arbeit war die Bestimmung der eigenen Position mit Hilfe von Wireless LAN, jedoch lag die Verantwortung darüber nicht in den Händen des IndoorGuide4Android Projekts, sondern beim PointZero Projekt. IndoorGuide4Android war quasi abhängig von der Funktionsfähigkeit der PointZero Library, da sie einen, wenn nicht *den*, elementaren Bestandteil der Funktionalität darstellt.

Eine Übersicht der während dieser Arbeit involvierten Personen ist im Kapitel „1.3.1. Organisationsstruktur" auf Seite 21 zu finden, die Aufteilung des Projekts in Iterationen und Arbeitspakete befindet sich in den Kapiteln „1.4. Managementabläufe" und „1.5. Arbeitspakete" ab Seite 22.

Die nachfolgende Tabelle gibt eine Übersicht über den Aufbau dieser Dokumentation:

Kapitel	Beschreibung
TEIL I: Technischer Bericht (Seite 9)	Der erste Teil enthält den technischen Bericht, welcher eine Gesamtübersicht über die Arbeit gibt. Alle wichtigen Informationen sind darin enthalten, von der Aufgabenstellung bis zu den Resultaten der Arbeit. Weiterhin enthält dieser Teil einen Bericht über die Erfahrungen mit dem Android SDK sowie persönliche Berichte der Autoren.
TEIL II: Projektmanagement (Seite 20)	Der zweite Teil ist dem Projektmanagement der Arbeit gewidmet und gibt Aufschluss über Annahmen und Einschränkungen, die Projektorganisation, Qualitätsmassnahmen und enthält zum Schluss noch eine Zeitauswertung (Ist-Stunden).
TEIL III: Software-Projektdokumentation (Seite 31)	Dieser Teil richtet sich an Ingenieure und Softwareentwickler, welche Einblick in den Aufbau der Applikation gewinnen oder sie erweitern möchten. Er beschreibt die Applikation aus technischer Sicht.
TEIL IV: Anhang (Seite 83)	Der letzte Teil bildet den Anhang in welchem sich das Glossar und das Literaturverzeichnis befindet. Ebenfalls im Anhang befindet sich eine konzise Softwaredokumentation (Installationsanleitung und Kurzanleitung zur Benutzung der Applikation).

2. Stand der Technik

Es existieren für Android zwei Applikationen, welche als Inspiration dienen oder dienen könnten, „Wikitude AR Travel Guide" und „AndNav2". Diese werden nachfolgend kurz beschrieben.

Wikitude AR Travel Guide

Wikitude AR bezieht standortbezogene Informationen aus verschiedenen freien Quellen und stellt diese abhängig von der aktuellen Position, Höhe und Blickrichtung auf dem Display des Smartphones. Wikitude wurde auf dem Google G1 Android-Smartphone entwickelt und getestet. Es macht Gebrauch von der Kamera, Lage- und GPS Sensoren. Wikitude AR ist die erste Augmented Reality Anwendung für das G1 und ging mit dem US und UK Launch des G1 Handys im Oktober 2008 live.

Diese Applikation beweist sozusagen, dass es möglich ist, die Sensoren des Geräts zuverlässig auszulesen und anzusteuern.

AndNav2

Diese Applikation implementiert ein GPS Navigations- und Routenfindungssystem auf Basis von OpenStreetMap Karten- und Metadaten. Wie Wikitude auch, bezieht es Positionsdaten vom integrierten GPS Emfpänger. Diese Applikation beweist, dass es unter der Android-Plattform möglich ist, Daten von OpenStreetMap zu beziehen, zu verarbeiten und damit mittels Routenfindungsalgorithmen (zum Beispiel ShortestPath Algorithmen) eine

Route von einem annähernd beliebigen Startpunkt zu einem annähernd beliebigen Endpunkt zu berechnen – sofern ein Weg von Start zum Ziel führt.

OSMNavigation Library von „Traveling Salesman"

Traveling Salesman ist ebenfalls ein Open Source Routing- und Navigationsprogramm für OpenStreetMap, welches allerdings in Standard Java implementiert ist.

Die Grundlage der Traveling Salesman Applikation bildet die „OSMNavigation" Library, welche sich um Routensuche, Adresssuche und Navigation kümmert und diverse Algorithmen zur Routenberechnung implementiert. OSMNavigation selber ist abhängig von der LibOSM Library, welche die Arbeit mit OpenStreetMap Daten stark vereinfacht.

OSMNavigation ist insofern sehr interessant für das IndoorGuide4Android Projekt, da sie ausser LibOSM keinerlei Abhängigkeiten hat, weder zu weiteren Libraries, noch zu spezifischen Android Klassen, was sie universell einsetzbar und flexibel mach.

Grundsätzlich realisierbar

Ausgehend von obig genannten Applikationen und Libraries sollten sich sämtliche Ziele des Indoor Guides realisieren lassen. Einziger offener Punkt ist bis jetzt die Positionsbestimmung über Wireless LAN. Diese Lücke kann mit Hilfe der Ergebnisse von früheren Diplom- und Studienarbeiten gefüllt werden. Namentlich erwähnt sei hier das IndoorWPS Projekt, welches genau diese Funktionalität und Infrastruktur zur Verfügung stellt.

Als Schnittstelle zu IndoorWPS wird in diesem Projekt die Library „PointZero" verwendet werden, welche parallel zum IndoorGuide4Android Projekt entwickelt wird.

3. Ziele und Vision

Die Ziel des Projekt ist es, gemäss der Aufgabenstellung ein Indoor Informations- und Navigationssystem zu erstellen, das Benutzer über Points of Interest (POIs) in ihrer Umgebung informiert und Detailinformationen dazu darstellt sowie die Benutzer an ein gewähltes Ziel mittels von Auto-Navigationssystemen bekannten Richtungsanweisungen führen kann.

Eine Besonderheit an diesem System ist, dass es innerhalb von Gebäuden und überdachten Anlagen funktionieren soll, wo GPS Empfang nicht oder nur unbefriedigend verfügbar ist, wobei die Lokalisierung und Positionsbestimmung aufgrund von Wireless LAN Signalen stattfinden soll.

Dies stellt eine Ergänzung zu der bereits verfügbaren GPS Technologie dar und eröffnet Möglichkeiten für Anwendungen für die bisher keine oder nur verhältnismässig aufwändige und teure Lösungen verfügbar waren.

Die Einsatzmöglichkeiten für ein derartiges System sind vielfältig, wie nachfolgende, nicht abschliessende Auflistung aufzeigt:

- Persönlicher Museums- oder Messeführer. Einerseits können Informationen über die direkt sichtbaren Objekte bezogen werden, andererseits kann gezielt zu einem Objekt von erhöhtem Interesse navigiert werden.
- Orientierungshilfe bei Firmenbesuchen. Ein Kunde nimmt einen Termin an einem ihm unbekannten Firmensitz wahr und möchte seinen Weg zum vereinbahrten Sitzungszimmer finden. Das Indoor-Navigationssystem kann ihm dabei behilflich sein.
- Navigationshilfe für Studenten. Gerade auf einem grösseren Campus, beispielsweise den der ETH Zürich, stellt das Finden des benötigten Hörsaals eine Kunst für sich dar. Auch hier könnte mit dem zu erstellenden Guide den Studenten unter die

Arme gegriffen werden.

- Einkaufshilfe in grossen Selbstbedienungs-Möbelhäusern wie IKEA. Der Guide kann Aufschluss über Teile, welche in einem Regal gelagert sind geben. Falls ein Benutzer ein bestimmtes Warenregal nicht findet, kann er sich mit dem Navigationssystem dorthin führen lassen.
- Ladenführer in Einkaufszentren. Mit Hilfe des Guides kann ein Benutzer beispielsweise feststellen, welche Läden welche Artikel im Sonderangebot haben.

Die möglichen Anwendungen eines Indoor Informations- und Navigationssystem sind vielfältig, jedoch hängen sie von der bereitgestellten Datengrundlage ab.

4. Resultate

4.1. Bewertung

Es gibt bereits einige Implementationen von Indoor Positionierungssystemen, wobei nicht all zu viele von ihnen auf Wireless LAN basieren. Diejenigen, die auf Wireless LAN als Lokalisierungstechnologie setzen, sind oftmals akademischer Natur oder dann kommerzielle, sehr teure Lösungen. Allerdings existiert nach unserem Wissensstand bislang kein einziges Programm, welches als Indoor Informations- und Navigationssystem auf Basis von Wireless LAN einsetzbar ist.

Insofern stellt diese Kombination eine Neuheit dar und ebnet den Weg für viele neue Anwendungen. Siehe hierzu auch „3. Ziele und Vision“auf Seite 13. Der Vorteil an diesem System besteht darin, dass es bereits vielerorts bestehende Infrastruktur (Wireless LAN) verwendet, was keine zusätzlichen Installationskosten nach sich zieht, falls in einem neuen Gebäude die Indoor Information und Navigation eingeführt werden soll. Alles, was getan werden muss, ist das Erfassen von Wireless LAN Fingerprints und Daten in OpenStreetMap – alles Tätigkeiten, welche „lediglich“ Arbeit kosten oder sich zum Teil (Fingerprint-Erfassung) automatisieren lassen.

4.2. Zielerreichung

Um die Erreichung der festgelegten Ziele übersichtlich darzustellen, sind in nachfolgender Tabelle sämtliche gesetzten Ziele aus „2.2. Spezifische Anforderungen“ ab Seite 31, nochmals aufgeführt und mit einer Bewertung versehen.

ID	Ziel	B	Kommentar
F01	Positionierung in einem Raum wird zuverlässig bestimmt.		Die Positionierung erfolgt leider nur unzuverlässig. Dieses Problem ist jedoch weniger auf das IndoorGuide Projekt, sondern auf die PointZero Library zurückzuführen. Eine weitere Mögliche Erklärung könnte die Qualität der gemessenen Wireless LAN Fingerprints sein – Erfassen von mehreren Fingerprints an der selben Koordinate würde die Genauigkeit erhöhen.

F02	POI- und Routenplan-Daten für Areas of Interest werden von Benutzern kontrolliert heruntergeladen und gespeichert.		Ein Benutzer hat die Möglichkeit, eine Area durch expliziten Knopfdruck anzulegen respektive zu aktualisieren. Bei diesem Prozess werden POI- und Routenplan-Daten ebenfalls heruntergeladen und gespeichert.
F03	Drei nahe liegende POIs als Liste im Kamerabild mit der richtigen Distanz und nach Distanz sortiert anzeigen.		Es wird permanent eine Liste über das Kamerabild gelegt, welche die drei am nächsten liegenden POIs anzeigt. Der am nächsten liegende POI erscheint zuoberst auf der Liste.
F04	Der gewünschte Zielort muss via Eingabefeld eingegeben oder in einer Liste ausgewählt werden.		Ein Benutzer hat die Möglichkeit in einer speziellen Ansicht das Ziel für eine Routenplanung direkt in der Liste anzutippen oder bei einer umfangreichen Liste diese über eine Suchmaske soweit einzuschränken, bis er das gewünschte Ziel direkt antippen kann.
F05	Einen sinnvollen Weg bis zum gewünschten Zielort berechnen und mittels Richtungspfeil darstellen.		Die Route wird korrekt berechnet. Es ist allerdings wichtig, dass die OpenStreetMap Daten richtig erfasst sind und an jeder Weggabelung ein neuer Weg beginnt. Ansonsten können Richtungsänderungen nicht korrekt ermittelt werden, da das Routingbackend davon ausgeht, dass man einfach „dem Weg folgt" - was bei Strassen natürlich Sinn macht, bei Gebäudegängen jedoch eher weniger, wenn man präzise Richtungsanweisungen haben möchte.
F06	Nahtloser Übergang zwischen Positionierung mit „PointZero" und GPS.		Die Positionsbestimmung wechselt automatisch auf GPS, sobald es verfügbar ist.
F07	Der Benutzer kann in der Nähe liegende Areas of Interest ansehen und diese mit allen Daten herunterladen.		Es wurde eine Umgebungssuche anhand der aktuellen Position implementiert.
NF01	Die Dokumentation ist klar und verständlich. Es sollte dokumentiert werden, was sinnvoll und notwendig ist, damit die gemachte Arbeit nachvollziehbar ist („weniger ist mehr").		Die Dokumentation wurde anhand der Dokumentations-Richtlinien für Bachelorarbeiten erstellt und mit nötigen zusätzlichen Abschnitten versehen um den Gesamtüberblick über das Projekt zu wahren.
NF02	Software-Architekturdokument muss in Englisch geschrieben werden.		Die Softwaredokumentation (Software-Architekturbeschreibung) liegt in englischer Sprache vor. Gemäss Absprache mit dem Betreuer ist dies das einzige Dokument, das in Englisch verfasst sein soll.

NF03	Der Code muss sauber und in englischer Sprache dokumentiert werden.		Es wurde Wert auf eine ausführliche JavaDoc Kommentierung des Codes gelegt, welche in Englisch verfasst ist.
NF04	Das User Interface wird in englischer Sprache angeboten.		Sämtliche Elemente des User Interfaces sind in englischer Sprache beschriftet.
G01	Die Anklickbaren Objekte müssen gross genug sein, damit man diese auch ohne Problem treffen kann.		Sämtliche User Interface Elemente sind in einer Grösse, welche die Bedienung mittels Fingern ermöglicht. Es kommt jedoch in seltenen Fällen vor, dass die Berührungskoordinate des Bildschirms von der Android-Plattform nicht ganz korrekt ermittelt wird.
G02	Screen Orientation wird immer im Landscape-Modus angeboten, damit der Benutzer nicht immer wieder das Smartphone drehen muss.		Die Applikation ist durchgehend im Landscape Layout gehalten und das Gerät muss nie gedreht werden.
G03	Der Benutzer bekommt immer einen klaren Feedback oder eine verständliche (Fehler-) Meldung angezeigt.		Es wurde Wert darauf gelegt, dass der Benutzer immer weiss, dass gerade eine Operation abläuft und falls diese schief läuft, warum sie fehlgeschlagen ist.

Bewertungslegende

 Ziel erreicht.

 Auf dem Weg zum Ziel, jedoch ist noch weitere Arbeit nötig.

 Ziel nicht erreicht.

4.3. Schlussfolgerungen und Ausblick

Die grundlegenden Technologien für Indoor-Navigation existieren bereits und funktionieren unter bestimmten Voraussetzungen.

Ein Vergleich des IndoorGuide4Android mit bestehenden Lösungen gestaltet sich schwierig, da alle diese Lösungen einerseits auf GPS zur Positionsbestimmung setzen und sich andererseits entweder auf die Funktion als POI-Guide oder Navigationssystem konzentrieren. Während des Projekts ist jedoch eine Art simplifizierte Verschmelzung von Wikitude, AndNav2 und einem kleinen Teil der populären TomTom- beziehungsweise Garmin-Navigationssysteme entstanden, welche auf Wireless LAN als Positionierungssystem setzt.

Die entstandene Applikation ist als ausgereifter „Proof of Concept" und keinesfalls als eine bis ins letzte Detail ausgereifte Applikation zu verstehen .

Obschon Wert auf gute Funktionalität gelegt wurde, besteht durchaus noch Verbesserungspotential bei der Navigation sowie dem User Interface. Details zu Verbesserungsmöglichkeiten sind im Kapitel „5.2.Mögliche Weiterentwicklungen" auf Seite 80 zu finden.

5. Erfahrungen mit dem Android SDK

Installation und Einarbeitung

Die Installation sowie die Einarbeitung in das Android SDK verlief ohne Schwierigkeiten. Auf der Android Developer Webseite [android developers] werden alle nötigen Schritte für die Installation des SDKs und nötiger Plugins für Eclipse sehr ausführlich und verständlich beschrieben. Nach ein paar Stunden Einarbeitung und Studium der Tutorial-Seiten konnten wir schon ein wenig Text und Buttons auf dem Display darstellen.

Die Android Developer Webseite beinhaltet eine ausführliche und exzellent dokumentierte Framework-Referenz (API Beschreibung), sowie viele nützliche Beispiele, die fast alle Funktionen des Android SDKs abdecken.

Entwicklung

Die Entwicklung für GUI Komponenten oder alles, was nicht Zugriff auf Hardwaresensoren erfordert, muss nicht unbedingt auf einem Phone erfolgen. Android liefert mit dem SDK diverse nützliche Developer Tools aus, darunter befindet sich auch ein qualitativ hochwertiger Emulator, mit welchem sich fast alle Sachen emulieren lassen.

Zur GUI Struktur von Android ist zu erwähnen, dass sie speziell ist und für Entwickler, die sich traditionelle GUI Entwicklung unter Java gewohnt sind, etwas Einarbeitungszeit eingeplant werden muss. Beispielsweise gibt es grundsätzlich zwei Arten zur Erstellung von Screen-Layouts:

- XML-Layout. Man kann einen mitgelieferten graphischen Layout-Editor benutzen, welcher ein XML-Layout erstellt. Dieser Ansatz hat den Vorteil, dass das Aussehen des erstellten Layouts direkt ersichtlich ist.
- Programmatisches Layout. Die Layouts können, wie von anderen GUI Toolkits wie AWT oder Swing gewohnt, im Code kreiert werden. Diese Methode ist aber teilweise etwas umständlicher, da man das Resultat nicht sofort sehen kann, sondern die Applikation erst wieder auf einem Gerät oder dem Emulator installieren und starten muss.

Wir hatten uns für die Arbeit mit XML-Layouts entschieden und konnten somit schnell verschiedene Layouts erstellen.

Android bietet viele vordefinierte Layouts an, zum Beispiel Listen mit verschiedenen Darstellungsarten (Tabellenlayout, Linienlayout) oder Dialoge. Will man diese aber umgestalten oder vielleicht ein kleines Detail ändern, so muss man das ganze selber erstellen und Details beachten, damit es richtig funktioniert.

Positive Erfahrungen

Generell ist zu sagen, dass die Entwickler des Frameworks hervorragende Arbeit geleistet haben: das Framework und das angebotene API sind aus unserer Sicht sehr durchdacht und ausgezeichnet dokumentiert. Bei Problemen helfen die Mailinglisten oftmals weiter.

Positiv überrascht waren wir bei der Interaktion mit den Hardware-Sensoren. Das Ansprechen und Auslesen gestaltet sich als sehr einfach und pragmatisch, die gelieferten Werte sind normalisiert und in sinnvollen Einheiten (beim Kompass zum Beispiel in Grad, das Accelerometer liefert die Beschleunigung als Wert in m/s^2).

Obwohl Android noch ziemlich jung ist, existieren schon zahlreiche gute Entwickler-Com-

munities, Foren und Blogs, wo Erfahrungen anderer Entwickler beschrieben werden und viele nützliche Austauschmöglichkeiten existieren. Eine unserer Favoriten-Ressource, neben der offiziellen Android Developers Webseite, ist die Seite von AndDev [anddev], wo zahlreiche ausführliche Tutorials zur Android Programmierung verfügbar sind.

Probleme

Obschon die API relativ stabil ist, gab es beim Update von Release 1.0 auf Release 1.5 einige kleinere Änderungen. Einige Methoden deklarieren nun checked Exceptions, welche im Programmcode entsprechend behandelt werden müssen. Zudem stellten wir im oben erwähnten Emulator neu eingeführte Limitierungen fest: neuerdings werden die Kamera und einige Sensoren nicht mehr emuliert und wir mussten einen Workaround im Code einfügen, damit trotzdem auf dem Emulator entwickelt werden konnte.

Der IndoorGuide4Android hat als Default-Einstellung eine Landscape Screen Orientation, damit der Benutzer das Gerät nicht immer hin- und her drehen muss. Da unsere Entwicklungshardware, das T-Mobile G1, über eine ausklappbare QWERTZ-Tastatur verfügt, beobachteten wir teilweise ein merkwürdiges Verhalten der Applikation, welches auf die Events vom Auf- respektive Zuklappen der Tastatur zurückzuführen waren. Android ist so konfiguriert, dass beim Auf- oder Zuklappen der Tastatur die aktuell am Bildschirm gezeigte View, neu gezeichnet wird. Normalerweise stellt dies kein Problem dar, jedoch führt es zu Problemen, wenn die View beim Beginn der Anzeige einen neuen Thread für lange dauernde Operationen startet; sobald die View dann neu gezeichnet wird, wird versucht, die (schon gestarteten) Threads erneut zu starten, was selbstverständlich von Java mit einer Exception quittiert wird. Glücklicherweise bietet Android eine Konfigurationseinstellung für einzelne Activities an um dieses Verhalten zu unterdrücken.

Das wohl unangenehmste Problem wurde durch die integrierte Kamera erzeugt. Es kann vorkommen, dass dem Kamera-Thread, welcher seit dem Start des Geräts kontinuierlich im Hintergrund läuft, der Speicher ausgeht. Jeder weitere Zugriff auf die Kamera endet mit einer Fehlermeldung. Die einzige Lösung für dieses Problem bildet jeweils ein Reboot des Geräts. Dieses Problem ist den Entwicklern des Frameworks bekannt und weit verbreitet. Man ging davon aus, dass das Problem mit dem Release 1.5 behoben sei, was nicht der Fall ist. Es ist jedoch anzumerken, dass die Häufigkeit der Kamera-Abstürze seit dem Upgrade auf Release 1.5 drastisch zurückgegangen ist.

Fazit

Die Entwicklung mit und für Android macht Spass und es lassen sich, wenn die Konzepte des Frameworks erstmals verstanden sind, in vergleichsweise kurzer Zeit ansehnliche graphische Anwendungen entwickeln, welche auf einfache Art mit Daten der Hardware-Sensoren umgehen können. Dabei kann sich ein Entwickler auf die eigentliche Business-Logik konzentrieren, statt sich mit dem langwierigen Layouten und Zeichnen von UI Elementen aufzuhalten.

Durch die Wahl von Java als Programmiersprache wird die Entwicklung von Anwendungen für mobile Geräte stark vereinfacht und spricht so eine grössere Entwicklergemeinde an. Aus unserer Sicht hat Android noch sehr viel Potenzial und das, was wir bisher davon gesehen haben, ist erst der Anfang.

6. Persönliche Berichte

6.1. Tatjana Ossipova

Die Bachelor Arbeit verlief ohne irgendwelche Konflikte. Da es schon die vierte Arbeit gewesen ist, die Michel und ich zusammen machten, wussten wir, was auf uns zu kam. Wir haben als Team gut harmoniert.

Da Michel und ich ein Thema wählten, welches aus einem uns nicht vertrauten Bereich kam, war es für mich sehr interessant etwas neues kennen zu lernen und es auf dem echten Gerät auszuprobieren. Ich war von Anfang an sehr motiviert und habe viele Stunden mit Freude beim kennen lernen des Android-Frameworks verbracht.

Die Kommunikation mit dem Betreuer verlief teilweise nicht so wie wir es erwartet haben. Wir konnten aber die entstanden Kommunikations-Probleme in einem Gespräch klären, in welchem jeder von den Beteiligten seine Meinung anbringen konnte.

Im Grossen und Ganzen bin ich mit unserer Arbeit sehr zufrieden. Wir haben eine neue Technologie kennen gelernt und konnten unser Wissen vertiefen.

6.2. Michel Reiter

Die Zusammenarbeit im Team mit Tanja funktionierte einwandfrei. Wir haben schon diverse Arbeiten und Projekte miteinander durchgeführt und man könnte uns als eingespieltes Team bezeichnen, das die fachlichen Stärken und Schwächen des jeweils anderen Teammitgliedes kennt und auch auf sozialer Ebene gut miteinander umgehen kann. Ich bin auf alle Fälle auch für zukünftige Arbeiten sehr zuversichtlich, dass sie in dieser Teamkonstellation erfolgreich gelingen werden.

Während der Arbeit gab es ab und zu Kommunikations- und Verständnisprobleme zwischen unserem Team und dem Betreuer, welche über die Zeit ein angespanntes Verhältnis zwischen Betreuer und Studenten verursachten. In zirka der Hälfte der Arbeit musste diese Situation eskaliert werden und ein klärendes, offenes Gespräch unter sechs Augen wurde gesucht.

Ich möchte mich an dieser Stelle bei Herrn Stefan Keller für die gemachten und erlebten Erfahrungen bedanken und kann für zukünftige Arbeiten und das Berufsleben einen erweiterten Erfahrungsschatz mitnehmen, welcher mir in vielen Situationen Hilfreich sein wird.

TEIL II: PROJEKTMANAGEMENT

1. Projektplan

1.1. Einführung

1.1.1. Zweck

Der Projektplan soll eine Übersicht über das Projekt „IndoorGuide4Android", dessen Aufbau, Organisation und anderweitiger Bereiche betreffend des Projektmanagements liefern. Er basiert auf der Beschreibung in der Projektausschreibung sowie eigenen Nachforschungen.

1.1.2. Kapitel Übersicht

Die nächsten Kapitel sind wie folgt ausgelegt:

Kapitel	Beschreibung
1.2. Annahmen und Einschränkungen (Seite 20)	Behandelt die grundsätzlichen, Annahmen und Einschränkungen des Projekts.
1.3. Projektorganisation (Seite 21)	Gibt Aufschluss über die Zusammensetzung der beteiligten Personen sowie deren Rollen.
1.4. Managementabläufe (Seite 22)	In diesem Kapitel finden sich Angaben über Zeit- und Iterationsplanung sowie Projekttermine und Meilensteine.
1.5. Arbeitspakete (Seite 24)	Stellt die generelle Aufteilung des Projekts in Arbeitspakete dar und gibt eine Schätzung des Aufwandes für die einzelnen Pakete an.
1.6. Qualitätsmassnahmen (Seite 25)	Dieses Kapitel beschreibt die grundlegend getroffenen Massnahmen zur Qualitätssicherung.

1.2. Annahmen und Einschränkungen

Es gelten folgende Annahmen und Einschränkungen:

- Positionsbestimmung in Gebäuden aufgrund von WLAN Fingerprints. Diese Funktionalität wird von einem eigenständigen Projekt („PointZero") zur Verfügung gestellt. Das IndoorGuide4Android Projekt verwendet das Resultat von vorherig genanntem Projekt als Programmbibliothek.
- Anzeige von Points of Interest innerhalb des Gebäudes, in welchem sich der jeweilige Benutzer befindet. Ausserhalb des Gebäudes liegende POIs sind out-of-focus, können jedoch für eine allfällige Folgearbeit berücksichtigt werden.
- POI Daten werden jeweils im Voraus, vom Benutzer forciert, für ein ganzes Gebäude heruntergeladen. Nach dem Herunterladen kann die Software was die POIs betrifft, offline arbeiten.

1.3. Projektorganisation

1.3.1. Organisationsstruktur

Die nachfolgende Tabelle zeigt alle in das Projekt involvierte Personen und ihre Rollen.

Name	Kontaktdaten	Rolle
Herr Stefan Keller	sfkeller@hsr.ch	Betreuer der Bachelor Thesis
Herr Reto Senn	reto.senn@bitforge.ch	Externer Projektpartner
Frau Tatjana Ossipova	tossipov@hsr.ch	Studentin
Herr Michel Reiter	mreiter@hsr.ch	Student
Herr Sachin Patney	spatney@hsr.ch	Praktikant, Entwickler der „PointZero" Library

1.3.1.1. Rollenverteilung

Die Studierenden teilen die Arbeiten aller Disziplinen gleichmässig untereinander auf, soweit möglich. Dies forciert, dass immer beide Studierenden über alle Aspekte des Projektes up to date sind und sich im Notfall gegenseitig vertreten können.

1.3.2. Richtlinien

1.3.2.1. Dokumentationsrichtlinien

Grundsätzlich gelten die Dokumentationsrichtlinien der Abteilung I von der HSR. Weiterhin sollen zum Schluss mindestens folgende Dokumente erstellt werden:

- Von HSR geforderte Dokumente
  - Technischer Bericht
  - Abstract
  - Management Summary
  - Persönliche Berichte
- Von Betreuer geforderte Dokumente
  - Technischer Bericht (Schwerpunkt: Erfahrungen mit dem Android SDK)
  - Installationsanleitung, sofern sinnvoll (englisch)
  - Webdokumentation (stichwortartig und bebildert)
  - Programmdokumentation (JavaDoc)
  - Software Engineering Dokumentation (evtl. Englisch)

1.3.2.2. Coderichtlinien

- Um eine einheitliche Struktur und Dokumentation des Programmcodes zu garantieren, verwenden alle dieselben CodeFormatter Settings von Eclipse. Dies kann leicht durch aktivieren der projektspezifischen Einstellungen erzwungen werden.
- Alle Variablen und Methoden sind im Camel-Case zu benennen.

- Neue Klassen werden mit JavaDoc in ihrer Funktionalität kommentiert. Der Autor muss bekannt gegeben werden.
- Es wird darauf verzichtet, alle Methoden mit JavaDoc zu dokumentieren. Lediglich die public Methoden werden mit JavaDoc versehen (abgesehen von Gettern und Settern). Private Methoden sollen sprechende Namen haben. Im Zweifelsfall können sie zusätzlich mit einem kurzen Blockkommentar beschrieben werden.
  - Links zu andere Klassen setzten, wo nötig (mit dem Code {@link Class}) .
  - Methodenparameter sollen ebenfalls beschrieben werden (@param parameter-Name Beschreibung).
- Komplexere Membervariablen, deren Zweck sich nicht durch den Namen erschliessen lässt oder mehrere Interpretationsweisen zulassen, sollen mit einem line comment kurz beschrieben werden.
- Sprache für Kommentare und Code ist Englisch.
- TODOs, FIXMEs und XXX (Anmerkungen):
  - sehr triviale FIXMEs und XXX können auch in Deutsch geschrieben werden.
  - Ziel ist es, im Sourcecode am Projektende keine FIXMEs und Anmerkungen mehr zu haben. TODOs sollen auf ein Minimum reduziert und dokumentiert sein.

1.4. Managementabläufe

1.4.1. Projektplanung

Gemäss Modulbeschreibung der HSR für die Bachelor Thesis Informatik soll ein Aufwand von mindestens 360 Personenstunden innert 16 Arbeitswochen entstehen, welcher folgendermassen aufgeteilt wird:

- 14 Wochen semesterbegleitend zu jeweils 20 Personenstunden Aufwand.
- 2 Wochen Vollzeit zu jeweils 45 Personenstunden Aufwand.

Bei zwei Studenten ergibt dies einen Gesamtzeitaufwand von 720 bis 740 Stunden.

1.4.1.1. Phasenplan

Das Projekt wird anhand agiler Softwareentwicklungsmethoden durchgeführt und zusätzlich in folgende Phasen unterteilt:

Phase	Disziplinen
Inception	Requirements, Design
Elaboration	Requirements, Design, Implementation, Test
Construction	Design, Implementation, Test
Transition	Implementation, Test, Projektabschluss

Weitere Informationen sind im Kapitel „3. Projektplan „ im Anhang auf der Seite 85 zu entnehmen.

1.4.1.2. Iterationsplanung und Meilensteine

Iteration (Ziele)	Datumsbereich	Dauer
Inception 1 (Projektstart, Basis für das weitere Arbeiten einrichten, Requirements Engineering, Technische Analyse)	16.02.2009 – 01.03.2009	2 Wochen
Inception 2 (Requirements Engineering, Risikoanalyse, Technische Analyse, Technischer Bericht, GUI Design, Software Design, Testspezifikation, Prototyp 1)	02.03.2009 – 15.03.2009	2 Wochen
Elaboration 1 (Technische Analyse, Technischer Bericht, GUI Design, Software Design, Prototyp 1, Prototyp 2, Testspezifikation)	16.03.2009 – 29.03.2009	2 Wochen
Elaboration 2 (Technische Analyse, Technischer Bericht, Software Design, Prototyp 2)	30.03.2009 – 12.04.2009	1.5 Woche
Construction 1 (Technische Analyse, Technischer Bericht, Prototyp 2, Prototyp 3, Testspezifikation, Testdurchführung)	13.04.2009 – 26.04.2009	1.5 Woche
Construction 2 (Technischer Bericht, Prototyp 3, Prototyp 4, Testspezifikation, Testdurchführung)	27.04.2009 – 10.05.2009	2 Wochen
Construction 3 (Technischer Bericht, Prototyp 4, Testdurchführung, BA-Broschüre, A0-Poster)	11.05.2009 – 24.05.2009	2 Wochen
Transition 1 (Technischer Bericht, Prototyp 5, Testspezifikation, BA-Broschüre, A0-Poster)	25.05.2009 – 31.05.2009	1 Woche
Transition 2 (Technischer Bericht, Prototyp 5, Testdurchführung, BA-Broschüre, A0-Poster)	01.06.2009 – 07.06.2009	1 Woche
Transition 3 (Persönliche Berichte)	08.06.2009 – 12.06.2009	1 Woche

ID	Beschrieb	Datum
MS1	Prototyp 1 abgeschlossen	23.03.2009
MS2	Prototyp 2 abgeschlossen	20.04.2009
MS3	Prototyp 3 abgeschlossen	04.05.2009
MS4	Prototyp 4 abgeschlossen	25.05.2009
MS5	Abgabe der Kurzbeschreibung der Arbeit an Betreuer	29.05.2009
MS6	Prototyp 5 abgeschlossen	08.06.2009
MS7	Abgabe	12.06.09 12:00

1.4.1.3. Sitzungen

Es wird jede Woche eine Sitzung der Studenten mit ihrem Betreuer geben. Diese Sitzungen finden jeweils in den Räumlichkeiten der HSR statt. Zudem werden nach Bedarf Sitzungen mit Austauschstudenten des IFS, Sachin Patney, durchgeführt.

Das Sitzungsintervall mit dem betreuenden Dozenten kann bei Bedarf und in gegenseitiger Absprache angepasst werden.

1.4.1.4. Geplante Abwesenheiten

Stefan Keller

- keine

Tatjana Ossipova

- keine

Michel Reiter

- Militärdienst vom 19.05.2009 – 29.05.2009, die Zeit wird vor geholt.

Sachin Patney

- keine

1.5. Arbeitspakete

1	Projektmanagement
Beschreibung	In dieses Arbeitspaket fallen alle Arbeiten, welche sich mit dem Management des Projektes befassen. Darunter fallen die Erstellung von Dokumentvorlagen, Meetings, Erstellung und Anpassung des Projektplans oder Zeiterfassungen nachführen.
Abhängigkeit	-
Risiken / Probleme	Managementoverhead nimmt einen beträchtlichen Teil der gesamten Projektzeit für sich ein.

2	Analyse
Beschreibung	Dieses Arbeitspaket beinhaltet folgende Punkte: • Anforderungsspezifikation verfassen und Anforderungen erarbeiten • Personas und Kontext Szenarien • Domain Analyse • Analyse und Einarbeitung in diverse APIs
Abhängigkeit	-
Risiken / Probleme	Anforderungen ändern sehr häufig in massiver Art und Weise. Dies gefährdet eine speditive Projektdurchführung respektive lässt sich nur durch Mehraufwand wieder ausgleichen.

3	Design
Beschreibung	Software Architecture Dokument mit folgenden Hauptpunkten: • Architekturübersicht • Logische Architektur • Beschreibung einzelner Packages • Beschreibung von Designentscheiden
Abhängigkeit	Analyse
Risiken / Probleme	Es muss sich die begrenzte Leistungsfähigkeit der Zielplattform im Kopf behalten werden. Eventuell Wegoptimierung von eleganten Designs zu

	performanteren Designs.

4	Implementation
Beschreibung	Implementation der Prototypen
Abhängigkeit	Analyse, Design
Risiken / Probleme	Zeitaufwand für einzelne Prototypen höher als erwartet.

5	Test
Beschreibung	Dieses Arbeitspaket beinhaltet folgende Punkte: • Testspezifikation und Testplan • Testdurchführung mit Führung von Testprotokollen • Unit-Tests
Abhängigkeit	-
Risiken / Probleme	Gewisse Tests lassen sich unter Umständen aufgrund Abhängigkeiten zu externen Projekten noch nicht durchführen. Dies muss regelmässig überprüft werden.

6	Dokumentation
Beschreibung	Unter Dokumentation fällt das Verfassen von sämtlichen unter 1.3.2.1 Dokumentationsrichtlinien (Seite 21) erwähnten Dokumenten.
Abhängigkeit	-
Risiken / Probleme	

1.6. Qualitätsmassnahmen

1.6.1. Wöchentliche Meetings

Um auf den aktuellen Stand der Arbeit zu sein und früh auf eventuelle Probleme reagieren zu können, finden wöchentliche Meetings mit dem Betreuer statt, in welchen auf den Arbeitsstand, die weitere Planung und allfällige Probleme eingegangen wird.

Weiterhin finden täglich Teammeetings statt, um jederzeit über den Stand der Arbeiten des jeweils anderen Teammitgliedes informiert zu sein und allfällige Probleme oder Lösungen zu besprechen.

1.6.2. Funktionalitätstests

Am Ende jeder Prototyp-Entwicklungsphase werden Funktionalitätstests gemäss Testspezifikation durchgeführt und protokolliert. Testspezifikationen werden am Anfang jeder Entwicklungsphase definiert und beschrieben.

1.6.3. Test Driven Development und Continuous Integration

Während der Entwicklung wird konsequent die Methode des Test Driven Development („test a little, code a little") angewandt. Um die Qualität der Software ständig unter Kontrolle zu haben, wird ein Buildserver verwendet, welcher nach jedem Hochladen von Quellcode den gesamten Code vom Sourcecode Management System herunterlädt, kompiliert und Unit Tests ausführt.

2. Risikomanagement

Um frühzeitig Risiken zu erkennen und Reserven zu planen werden bei Bedarf Risiken identifiziert und bewertet. Die Form bleibt frei.

3. Verwendete Technologien

Android

Android ist eine Plattform für mobile Geräte wie Smartphones, Mobiletelefone und Netbooks, die von der Open Handset Alliance entwickelt wird. Basis hierfür bildet der Linux-Kernel 2.6. Er ist für Speicherverwaltung, Prozessverwaltung und die Netzwerkkommunikation zuständig. Ausserdem bildet er die Hardwareabstraktionsschicht für den Rest der Software und stellt die Gerätetreiber für das System.
Weitere wichtige Bausteine sind die auf der von Sun Microsystems entwickelten Java-Technik basierende virtuelle Maschine Dalvik und die dazugehörigen Android-Java-Klassenbibliotheken. [http://de.wikipedia.org/wiki/Android_(Plattform)]

ADT

Android Developer Tool ist ein Eclipse Plugin, welcher erlaubt schnelles und einfaches Erstellen und Debuggen von Android Applikationen. Es bietet verschiedene Tools und Features an, wie:

- Es bietet mit DDMS (Dalvik Debug Monitor Service) das Aufnehmen von Screen Shots von Devices, Thread und Heap Informationen, logcat, Prozesse und andere wichtige Device Informationen.
- „New Project Wizard" hilft beim erstellen und aufsetzen von Dateien, die für die Android Applikationen gebraucht werden.
- Android Code Editor zum erstellen von validen XML Dateien für die Applikation.
- Und viele andere Features, die das Entwickeln von Android Applikationen vereinfachen.

JUnit 4

JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units (Klassen oder Methoden) geeignet ist.

SQLite

SQLite ist eine Programmbibliothek, die ein relationales Datenbanksystem enthält. SQLite hat einige Besonderheiten gegenüber anderen Datenbanken: die Bibliothek ist nur wenige hundert Kilobyte gross. Die Datenbanken können verteilt auf meh-

reren Dateien, bei Bedarf auch in eine einzelne Datei, gespeichert werden, was das Austausch zwischen verschiedenen Systemen, sogar zwischen Systemen mit unterschiedlichen Byte-Reihenfolgen, vereinfacht.

EclEmma — EclEmma ist ein Eclipse Plugin, das zeigt welche Teile des Codes durch die Testfälle, die man programmiert hat, abgedeckt werden und welche nicht.

Ant — Apache Ant ist ein in Java geschriebenes Werkzeug zum automatischen Erzeugen von Programmen aus Quelltext.

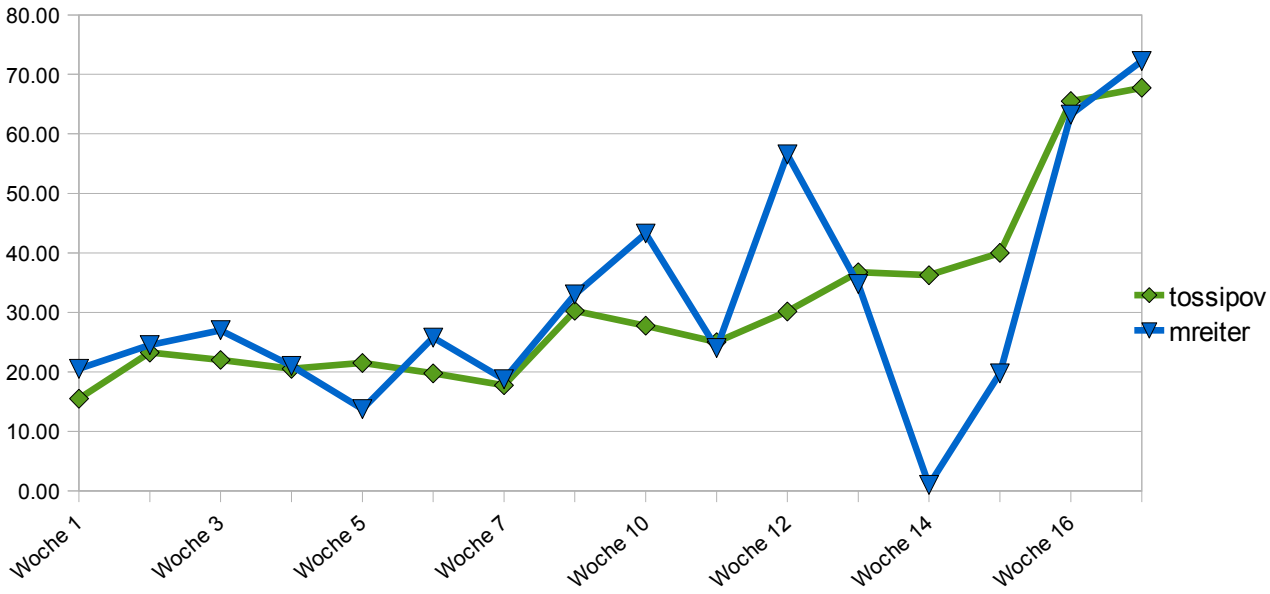
Subversion — Subversion ist ein Sourcecode Management System welches den im Repository verwalteten Code speichert und versioniert. Somit gehen keine Änderungen verloren und sind jederzeit abrufbar oder können rückgängig gemacht werden.

4. Zeitauswertung

4.1. Wochenauswertung

Woche	Tatjana Ossipova	Michel Reiter
Woche 1	15.50 Std.	20.50 Std.
Woche 2	23.25 Std.	24.50 Std.
Woche 3	22.00 Std.	27.00 Std.
Woche 4	20.50 Std.	21.00 Std.
Woche 5	21.50 Std.	13.75 Std.
Woche 6	19.75 Std.	25.75 Std.
Woche 7	17.75 Std.	18.75 Std.
Woche 8+9	30.25 Std.	33.00 Std.
Woche 10	27.75 Std.	43.25 Std.
Woche 11	25.00 Std.	24.00 Std.
Woche 12	30.14 Std.	56.55 Std.
Woche 13	36.75 Std.	34.75 Std.
Woche 14	36.25 Std.	1.00 Std.*
Woche 15	40.00 Std.	19.75 Std.*
Woche 16	65.50 Std.	63.25 Std.
Woche 17	67.75 Std.	72.25 Std.
Durchschnitt	**31.23 Std.**	**31.19 Std.**

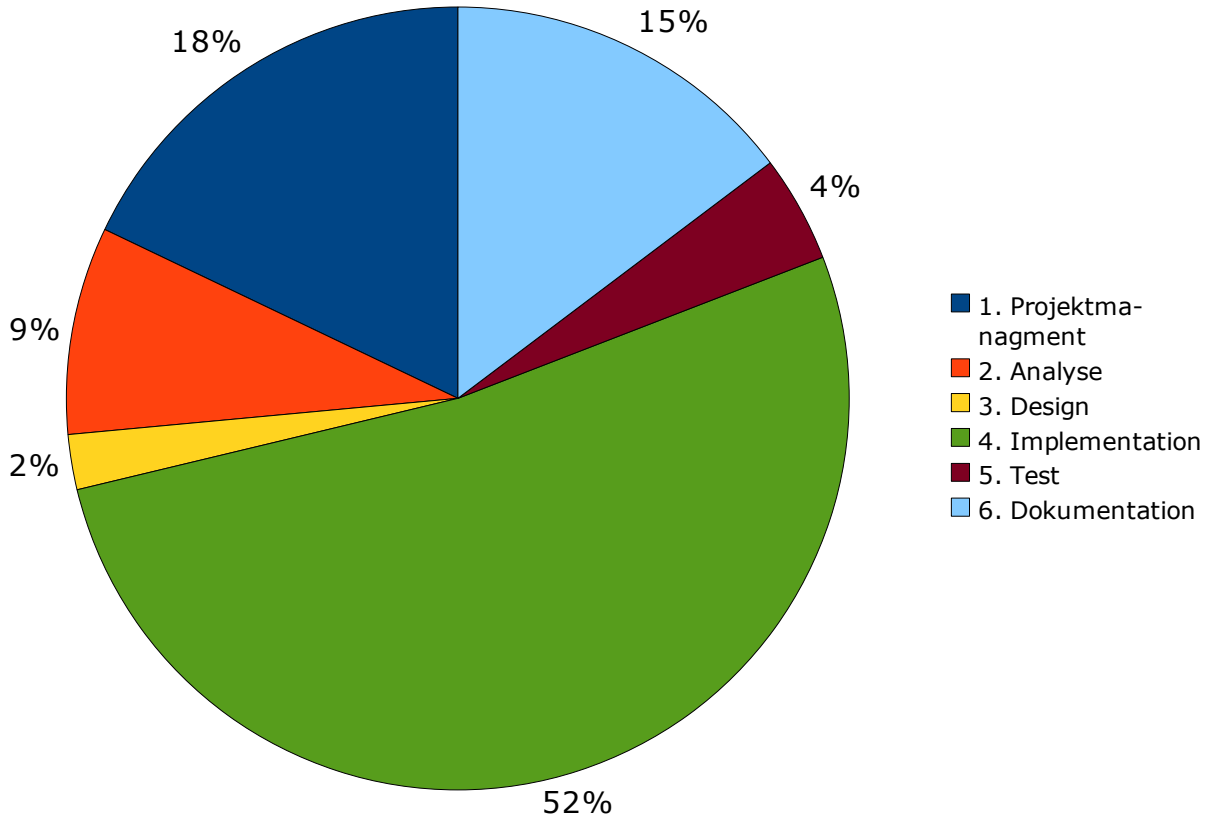
Summe	**499.65 Std.**	**499.05 Std.**



* Die geringe Stundenanzahl resultiert durch Abwesenheit wegen Militärdienst.

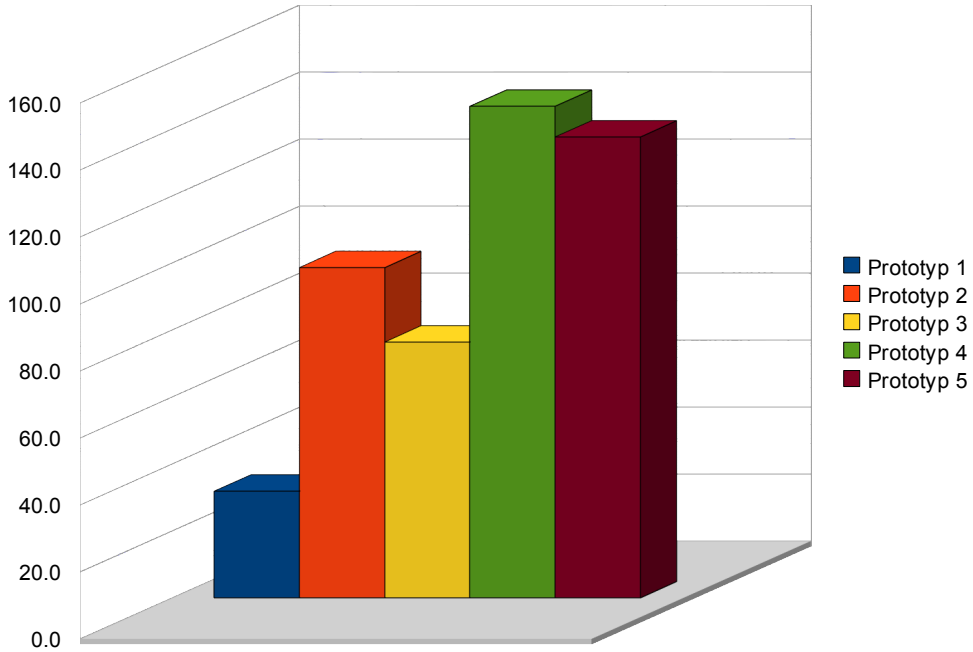
4.2. Zeit pro Arbeitspaket

Arbeitspaketbezeichnung	Anzahl Stunden
Projektmanagement	178.45 Std.
Analyse	85.50 Std.
Design	22.75 Std.
Implementation	519.80 Std.
Test	44.00 Std.
Dokumentation	146.75 Std.
Summe	**997.25 Std.**



4.3. Zeit pro Prototyp

Prototyp	Zeitspanne	Zeitaufwand
Prototyp 1	09.03.2009-22.03.2009	32.00 Std.
Prototyp 2	23.03.2009-19.04.2009	98.80 Std.
Prototyp 3	20.03.2009-03.05.2009	76.50 Std.
Prototyp 4	04.05.2009-24.05.2009	146.80 Std.
Prototyp 5	25.05.2009-11.06.2009	137.50 Std.



TEIL III: SOFTWARE-PROJEKTDOKUMENTATION

1. Vision

Details zur Vision sind im Kapitel „3. Ziele und Vision“ auf Seite 13 zu finden.

2. Anforderungsspezifikation

2.1. Einführung

2.1.1. Zweck

Diese Anforderungsspezifikation beschreibt:

- alle funktionalen und nicht-funktionalen Anforderungen an die zu erstellende Applikation.
- Personas, die Benutzergruppen festlegen und typische Benutzer aus den Gruppen beschreiben.
- Kontext Szenarien die einen möglichen Ablauf bei der Benutzung der Applikation beschreiben.

2.1.2. Kapitel Übersicht

Die nächsten Kapitel sind wie folgt ausgelegt:

Kapitel	Beschreibung
2.2. Spezifische Anforderungen (Seite 31)	Dieses Kapitel beschreibt die spezifische Anforderungen an die Applikation.
2.3. Personas Beschreibung (Seite 34)	In diesem Kapitel werden die Personas beschrieben, die die Benutzergruppen mit ihren Wünschen und Zielen an das Produkt repräsentieren.
2.4. Kontext-Szenarien (Seite 35)	Diese Szenarien beschreiben, wie die unter „2.3. Personas Beschreibung (Seite 34)„ beschriebenen Personas die Software benutzen können.
2.5. Use Cases (Seite 35)	Dieses Kapitel listet die Aktoren und die Use Cases auf und beschreibt sie.

2.2. Spezifische Anforderungen

2.2.1. Funktionale Anforderungen

2.2.1.1. Übersicht

ID	Ziel	Priorität
F01	Positionierung in einem Raum wird zuverlässig bestimmt.	sehr hoch

F02	POI- und Routenplan-Daten für Areas of Interest werden von Benutzern kontrolliert heruntergeladen und gespeichert.	hoch
F03	Drei nahe liegende POIs als Liste im Kamerabild mit der richtigen Distanz und nach Distanz sortiert anzeigen.	sehr hoch
F04	Der gewünschte Zielort muss via Eingabefeld eingegeben oder in einer Liste ausgewählt werden.	hoch
F05	Einen sinnvollen Weg bis zum gewünschten Zielort berechnen und mittels Richtungspfeil darstellen.	sehr hoch
F06	Nahtloser Übergang zwischen Positionierung mit „PointZero" und GPS.	mittel
F07	Der Benutzer kann in der Nähe liegende Areas of Interest ansehen und diese mit allen Daten herunterladen.	hoch

2.2.1.2. Detailliert

ID	Detailbeschrieb
F01	Die Positionierung wird mit Hilfe der „PointZero" Library über WLAN Fingerprints bestimmt. Die Library liefert die Koordinaten der aktuellen Position im standardisierten WGS-84 GPS Format.
F02	Es wird eine Verbindung zum POI- und Routenplan-Server aufgebaut und die Daten werden geholt. Hierbei muss beachtet werden, dass die Anzahl in Frage kommender POIs und Routenpläne in zwei Punkten begrenzt werden muss: • Das Gebiet (Area of Interest) mit den POIs in seiner räumlichen Ausdehnung • Die maximale Anzahl herunterzuladender POIs und dazugehörenden Routenplänen Der Download der Daten muss durch den Benutzer gezielt ausgelöst werden.
F03	Die Koordinaten der POI-Daten werden analysiert und laufend mit den Daten von der Position verglichen. Die POIs werden in geeigneter Weise im Kamerabild dargestellt. Dazu muss folgendes beachtet werden: • Die Distanz muss mit der Positionierungsweite laufend angepasst werden. So dass drei näher liegenden POIs in der Liste aufsteigend aufgelistet werden.
F04	Es muss möglich sein, den gesuchten Ort eingeben zu können. Dafür sollte eine passende Eingabemaske erstellt werden.
F05	Ein sinnvoller Weg muss anhand Routenplan berechnet und im Kamerabild mittels Weganweisungen im Form von Richtungspfeilen dargestellt werden, so wie sich die Benutzer dies von herkömmlichen GPS Navigationsgeräten gewohnt sind.
F06	Beim Verlassen eines Gebäudes soll die Positionierung gemäss dem Location Provider mit der höchsten Genauigkeit (mit hoher Wahrscheinlichkeit GPS) erfolgen. Ein Wechsel des Location Providers muss im Hintergrund geschehen, damit die eigene Position zu jeder Zeit bekannt ist.
F07	Nach dem der Benutzer die Anwendung gestartet hat, kann er eine Übersicht

	über die in der Nähe liegenden Areas of Interest ansehen. Er hat die Möglichkeit diese herunterzuladen oder die bereits gespeicherte Daten zu nutzen.

2.2.2. Nichtfunktionale Anforderungen

2.2.2.1. Übersicht

ID	Ziel	Kategorie
NF01	Die Dokumentation ist klar und verständlich. Es sollte dokumentiert werden, was sinnvoll und notwendig ist, damit die gemachte Arbeit nachvollziehbar ist („weniger ist mehr").	Dokumentation
NF02	Software-Architekturdokument muss in Englisch geschrieben werden.	Dokumentation
NF03	Der Code muss sauber und in englischer Sprache dokumentiert werden.	Entwicklung, Dokumentation
NF04	Das User Interface wird in englischer Sprache angeboten.	Entwicklung

2.2.2.2. Detailliert

ID	Detailbeschrieb
NF01	Es sollen die Anforderungen der HSR und des betreuenden Dozenten bezüglich Dokumentation beachtet werden, welche sich aus der Aufgabenstellung der Bachelorarbeit und ggf. späteren Abmachungen erschliessen.
NF02	Da es wahrscheinlich ist, dass Nachfolgearbeiten zu diesem Projekt getätigt werden, muss das Software-Architekturdokument in Englisch verfasst sein.
NF03	Die Dokumentation des Codes und Codekommentare selber müssen in Englisch geschrieben werden.
NF04	Die Benutzeroberfläche muss in englischer Sprache angeboten werden.

2.2.3. GUI Anforderungen

2.2.3.1. Übersicht

ID	Ziel	Priorität
G01	Die anklickbaren Objekte müssen gross genug sein, damit man diese auch ohne Problem treffen kann.	sehr hoch
G02	Screen Orientation wird immer im Landscape-Modus angeboten, damit der Benutzer nicht immer wieder das Smartphone drehen muss.	hoch
G03	Der Benutzer bekommt immer einen klaren Feedback oder eine verständliche (Fehler-) Meldung angezeigt.	hoch

2.2.3.2. Detailliert

ID	Detailbeschrieb
G01	Alle mit einem Finger anwählbare Elemente müssen eine passende Grösse haben, damit sie ohne Schwierigkeiten getroffen werden können.
G02	Die Screen Orientation wird auf den Landscape-Modus festgelegt. So wird verhindert, dass der Benutzer das Smartphone immer wieder drehen muss, wenn er zum Beispiel etwas via Tastatur etwas eingeben muss.
G03	Die Applikation informiert der Benutzer bei allen Aktivitäten, wenn zum Beispiel: • Benutzer warten muss (Fortschrittsanzeige) • Etwas nicht in Ordnung ist (Fehlermeldung mit verständlichem Inhalt) • Etwas abgeschlossen ist.

2.3. Personas Beschreibung

2.3.1. Persona „Peter Müller"

Peter Müller

Der externe Professor

Persönliches Profil:

Peter Müller ist Professor für Informatik an ETH. Er interessiert sich sehr für Smartphones, deren Entwicklung und neue Funktionen. An der ETH hat er schon mehrere Diplom-Arbeiten betreut, die sich mit Mobile-Technologie auseinander gesetzt haben. Herr Müller probiert gerne neue Erweiterungen aus, die nicht zu den Standard-Anwendungen von Smartphones gehören. Als Professor für Informatik hat er die Möglichkeit, nicht bekannte Anwendungen kennen zu lernen und zu testen.

Ausser an der ETH hält Herr Müller noch an anderen Hochschulen Vorlesungen und Vorträge. Meistens werden ihm nur Hochschule, Zimmer und Zeitpunkt seiner Veranstaltungen mitgeteilt. Um an den richtigen Ort zu gelangen, benutzt Herr Müller die Google Maps Anwendung und ist grundsätzlich davon begeistert. Leider kann er mit dieser Anwendung das richtige Gebäude und das richtige Stockwerk nicht finden. Im Moment fragt er jeweils Personen vor Ort, wo das bestimmte Zimmer zu finden ist. Ihm wäre es jedoch lieber, eine passende Anwendung zu haben, die ihm den Weg zu den Zimmern liefern kann, sowie er es sich von seinem Auto gewohnt ist.

Zur Person:

Alter:
40 Jahre

Familienstand:
Verheiratet, zwei Kinder

Bildungsabschluss:
Diplom-Informatiker ETH

Beruf:
Professor für Informatik an der ETH

Hobbys:
Computer, Musik, Sport

2.4. Kontext-Szenarien

2.4.1. Szenario „Vorbereitung zu Hause"

Herr Müller wird morgen einen technischen Vortrag an der HSR halten und will sich informieren, wohin er muss und wie er dorthin kommt.

Er startet der Indoor Guide und gibt in der Suchmaske der Areas of Interest den Suchbegriff „HSR" ein. Der Resultat wird mit verschiedenen Informationen, wie Land und Kanton, aufgelistet. Herr Müller wählt „HSR Rapperswil" aus der Liste aus und beginnt mit den Herunterladen von Daten. Die Applikation lädt die Daten erfolgreich herunter und Zeigt die gespeicherte Area of Interest. Herr Müller kann jetzt alle vorhandenen POIs ansehen und schauen, ob das Zimmer, wo er seinen Vortrag halten wird auch dabei ist. Er findet das Zimmer mit Hilfe von eingebauter Suche und kann die Details des Zimmers ansehen.

Nun ist Herr Müller für den morgigen Vortrag bereit und kann mit Hilfe des Indoor Guide vor Ort der Weg zum richtigen Zimmer finden.

2.4.2. Szenario „Sitzungszimmer Suche vor Ort"

Ein externer Besucher kommt auf dem Campus der HSR an und sucht (zu Fuss) das Sitzungszimmer 1.227. Er gibt den Namen des Sitzungszimmers (=POI) im Guide ein und richtet seine Kamera auf. Am Bildrand erscheinen die verfügbare POIs und im mittleren Kamerabild-Bereich wird zudem ein dreidimensionaler Richtungspfeil mit den Weganweisungen angezeigt.

Während er den Weganweisungen folgt, ändern sich die Distanzen der POIs. Der Benutzer sieht wie die POIs in der Liste mit den neuen POIs ersetzt werden.

Das Erreichen des Ziels wird ihm mit einem entsprechenden Symbol dargestellt.

2.5. Use Cases

Bei allen Use Cases wird davon ausgegangen, dass der Benutzer der Applikation Wireless LAN- oder GPS-Empfang zur Positionsbestimmung hat.

2.5.1. Aktoren

Name	Beschreibung
Benutzer	Endanwender der Applikation.

2.5.2. Use Case Definitionen

UC01	Wegführung zu einem gewünschten Ort (POI)
Vorbedingung: „*UC05 – Herunterladen von Daten*" ist erfolgreich abgelaufen. Der Benutzer möchte den Weg zu einem bestimmten Ort finden. Dazu gibt er in der dazu bestimmten Eingabemaske seinen gewünschten Zielort ein und bestätigt die Eingabe. Der IndoorGuide4Android berechnet anschliessend aufgrund der aktuellen Position und dem gewünschten Zielort eine Route. Der Benutzer folgt den angezeigten Anweisungen und wandert die Route ab. Wenn er	

am Ziel angekommen ist, wird ihm das durch ein entsprechendes Symbol auf dem Display angezeigt.
Parallel zu der Wegführung läuft „*UC02 – Drei nah liegende POIs der Umgebung anzeigen*" ab.

UC02	Drei nah liegende POIs der Umgebung anzeigen

Vorbedingung: „*UC05 – Herunterladen von Daten*" ist erfolgreich abgelaufen.

Dies ist der Standard Use Case des Systems, der grundsätzlich während der ganzen Laufzeit des IndoorGuide4Android läuft.

Der Benutzer richtet seine Kamera auf und im linken Teil der Kamerabild erscheint eine Auflistung der drei näheren POIs der Umgebung. Die Entfernung wird im Meter von aktueller Position bis zum POI berechnet und angezeigt.

Die dargestellte Distanz ändert sich mit der Änderung der aktueller Position. Bei Bedarf kann jederzeit durch den Benutzer „*UC03 – Detailinformationen eines POI ansehen*"ausgelöst werden.

Wenn keine POI-Daten vorhanden sind, werden auch keine POIs angezeigt. Der Benutzer muss in diesem Fall die Vorbedingung zu diesem Use Case selbständig erfüllen. Das System gibt einen Hinweis zu fehlenden Daten aus.

UC03	Detailinformationen eines POI ansehen

Im Display werden POIs der Umgebung gemäss „*UC02 – Drei nah liegende POIs der Umgebung anzeigen*" dargestellt. Durch Berührung eines POIs auf dem Display werden Detailinformationen des Punktes in geeigneter Art und Weise anzeigen.

UC04	Wegführung abbrechen

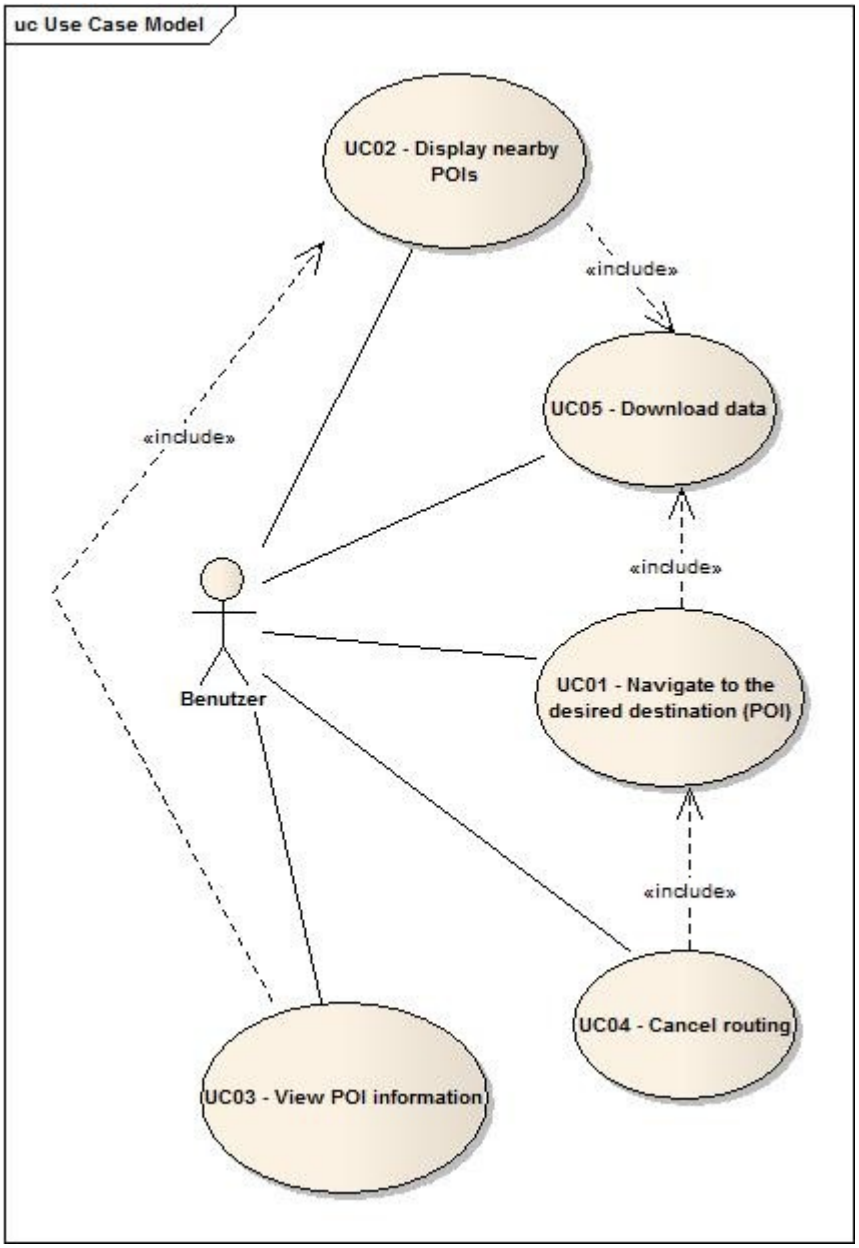
Vorbedingung: „*UC01 – Wegführung zu einem gewünschten Ort (POI)*" wurde erfolgreich gestartet.

Die Wegführung kann zu jedem beliebigen Zeitpunkt vom Benutzer abgebrochen werden. Nach Abbruch der Wegführung führt das System automatisch „*UC02 – Drei nah liegende POIs der Umgebung anzeigen*" aus.

UC05	Herunterladen von Daten

Der Benutzer gibt in einer Eingabemaske den Namen des ihn interessierenden Standorts (Area of Interest) ein und bekommt eine Resultatliste angezeigt. Aus dieser Liste kann er den genauen Standort auswählen. Das System lädt anschliessend die zugehörigen Daten von einem Server herunter und speichert sie. Somit sind POI- und Routenplan-Daten auf dem Smartphone gespeichert und stehen zur Verfügung.

2.5.3. Use Case Diagramm



3. Design & Software Architecture

3.1. Introduction

3.1.1. Purpose

This chapter provides an architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

It is intended to be read by software developers who wish to get an understanding of the concepts of the system or want to extend and modify it.

3.1.2. Overview

The following table gives an overview of the chapter structure and their contents:

Chapter	Description
3.2. Architectural goals and constraints	This section describes the software requirements and objectives that have some significant impact on the architecture.
3.3. Use Case View	This section lists use cases or scenarios from the use-case model.
Fehler: Referenz nicht gefunden Fehler: Referenz nicht gefunden	This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages and for each significant package, its decomposition into classes and class utilities.
3.5. Process view	This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes).
3.6. Implementation view	This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.
3.7. Data view	This section gives a description of the persistent data storage perspective of the system.

3.2. Architectural goals and constraints

As the "IndoorGuide4Android" project is intended to run on hardware with limited resources, the main architectural constraint is to deal carefully with the resources. As the actual design is still in a state of a prototype and technical demonstration, there are some points where for example object pools would make sense.

Another constraint is, that much of the business and persistence code should be automatically testable with JUnit tests. During development, the architecture had to be changed and another layer of indirection introduced for interaction with the built-in SQLite database. The problem is, that the Android SDK delivers a library which contains only forward declarations of the framework API which all throw runtime exceptions. While for many parts, like the HttpClient class, the missing functionality could be added by including the correct libraries (in that example the HttpClient library from the official Apache.org project), this was not possible with the SQLite database due to an Android specific database API abstraction.

The problem is solved by using the Strategy design pattern and programming against the interfaces, instead of the concrete Android API. When running on the real Android platform the SQLite database is used whilst during JUnit tests, another database backend is used. To test even database specific behaviour, it was decided that also a SQLite interface via JDBC should be used.

3.3. Use Case View

For a Use Case model and a description of the Use Cases, please have a look at chapter

"2.5. Use Cases", beginning on page 35. The diagram is available with English text, but the textual description, however, is only available in German.

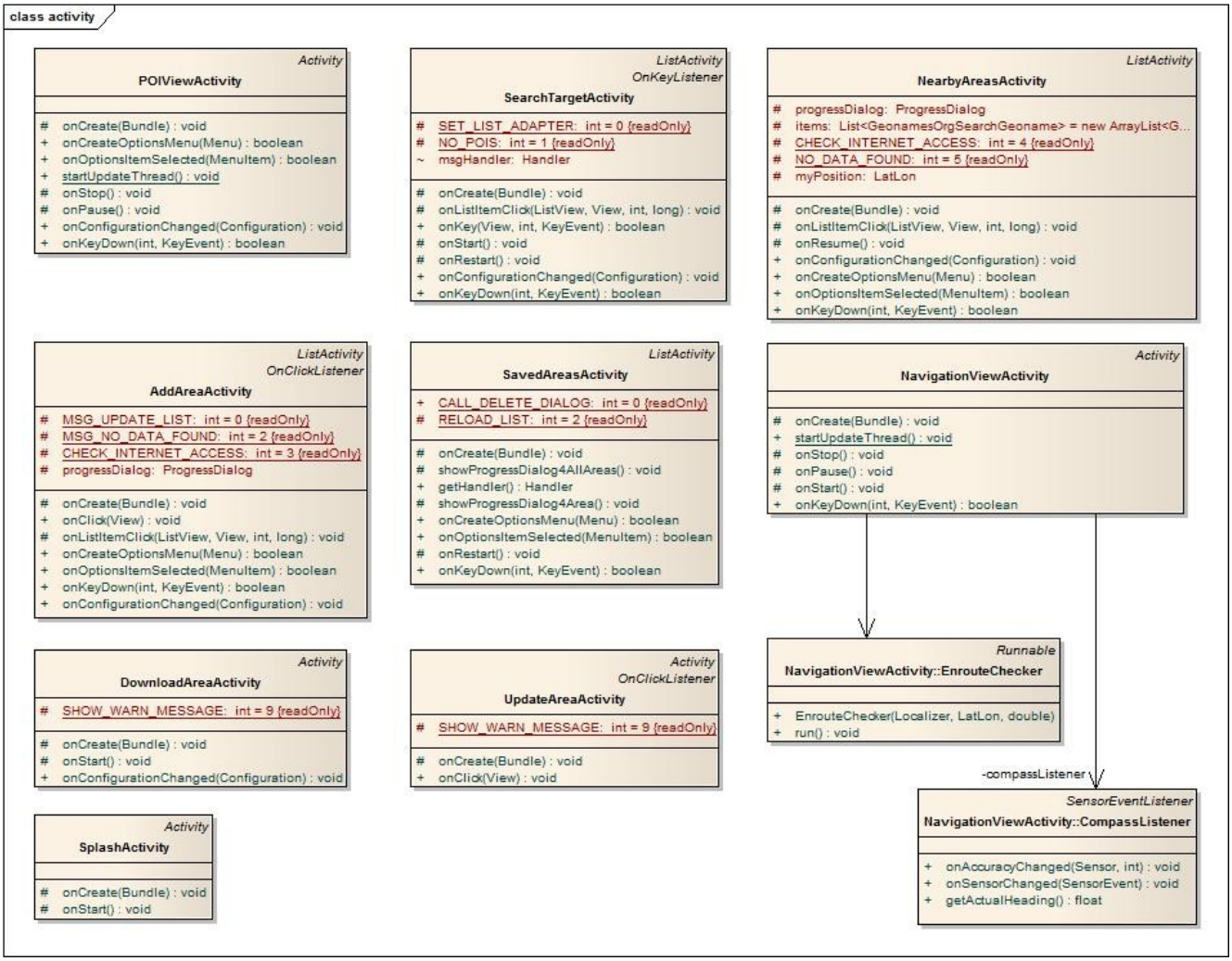
3.4. Logical View

3.4.1. Package gui.activity

3.4.1.1. Description

This package includes all `Activity` classes, which take care of creating windows in which we can place the graphical elements of the user interface. The `Activity` classes are an important part of an application's overall life cycle. The way activities are launched and put together is a fundamental part of the Android's platform application model.

3.4.1.2. Diagram



3.4.1.3. Class descriptions

1. SplashActivity	
This class is the first window the user sees on the start of the Indoor Guide. It is used to load the data of the last used Area of Interest from the database.	
Settings	
Content view	XML-Layout: layout_splash_screen.xml

2. POIViewActivity	
This activity is the main window, that shows the nearest POIs around the user and the compass in "augmented reality" view.	
Settings	
Content view 1	`CameraPreview`-class, to show the camera view.
Content view 2	`CompassView`-class, to show the compass in the right bottom corner.
Content view 3	`POIListView`-class, to show the nearest POIs.

3. SavedAreasActivity	
This activity is a `ListActivity` and lists all stored Areas of Interest form the database.	
Settings	
Content view	XML-Layout: layout_saved_areas.xml

4. NearbyAreasActivity	
This activity determines Areas of Interest that are in the users vicinity. First, the users actual location is determined and afterwards a database search done to list the nearest area of interest.	
Settings	
Content view	XML-Layout: layout_nearby_areas.xml

5. AddAreaActivity	
In this activity the user can search an Area of Interest, and afterwards choose one of the listed areas to download its data.	
Settings	
Content view	XML-Layout: layout_add_area.xml

6. SearchTargetActivity	
This activity lists all POIs of a current Area of Interest from the database.	
Settings	
Content view	XML-Layout: layout_search_target.xml

7. DownloadAreaActivity	
This activity shows the download progress of the chosen Area.	
Settings	
Content view	-

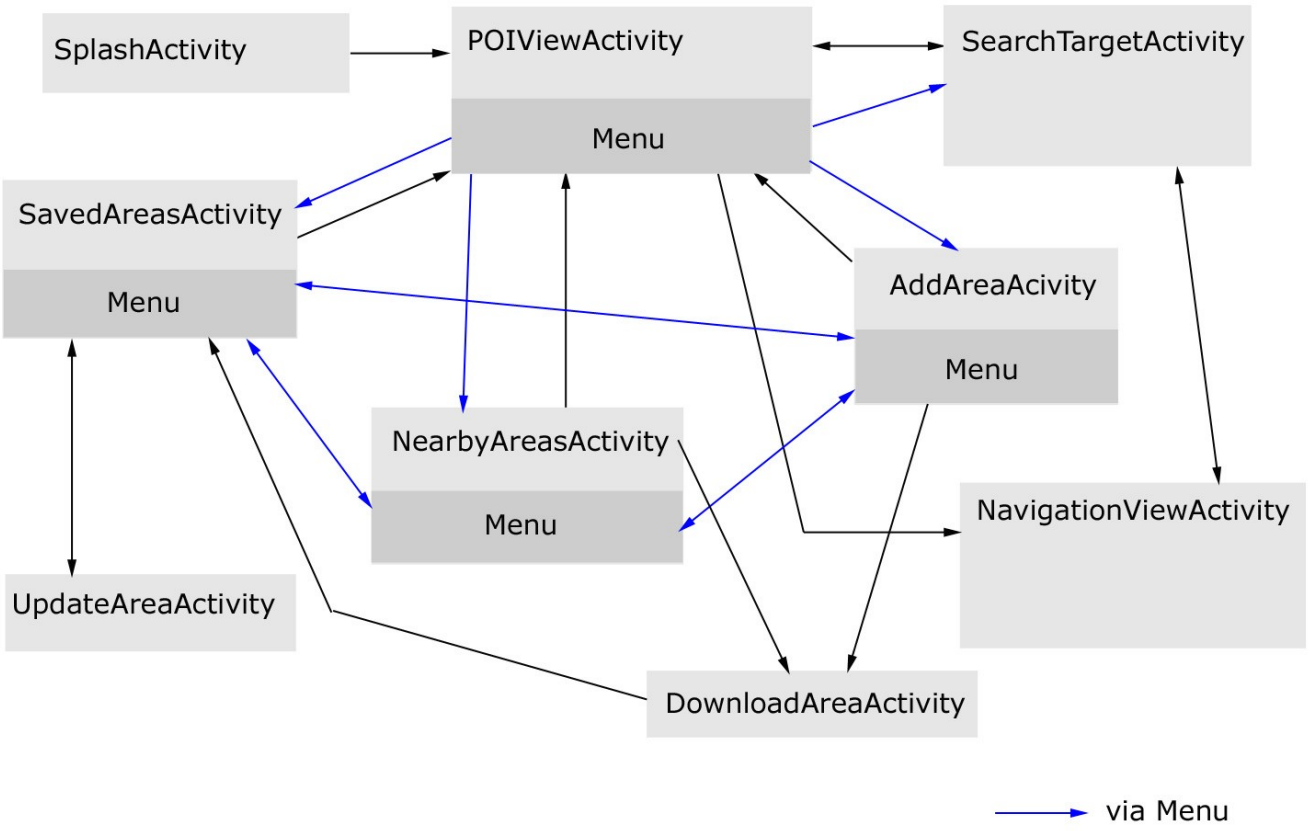
8. UpdateAreaActivity	
This activity shows the update progress of the chosen Area.	
Settings	
Content view	XML-Layout: layout_update_area.xml

9. NavigationViewActivity	
This activity shows the navigation route to the chosen POI.	
Settings	
Content view 1	`CameraPreview`-class, to show the camera view.
Content view 2	`CompassView`-class, to show the compass in the right bottom corner.
Content view 3	`POIListView`-class, to show the nearest POIs.
Content view 4	`NavigationArrowView`-class, to show the route to the chosen POI.

10. Inner class: EnrouteChecker	
This implementation of the `Runnable` interface is used within the `NavigationViewActivity` to periodically check the actual position of the device and decide in which direction an user should walk or turn to reach the next waypoint of a route.	
Operations	
`EnrouteChecker(Localizer, LatLon, double)`	The constructor. It takes as arguments an instance of the `Localizer`, a `LatLon` coordinate of the next waypoint and a proximity distance. The proximity distance indicates how close the user must get to the waypoint until the next routing step will be taken and thus a new navigation arrow will be displayed.

11. Inner class: CompassListener	
This listener receives direction information from the built-in magnetic compass sensor.	
Operations	
`getActualHeading()`	Returns the actual compass heading, the User is facing.

3.4.1.4. External design



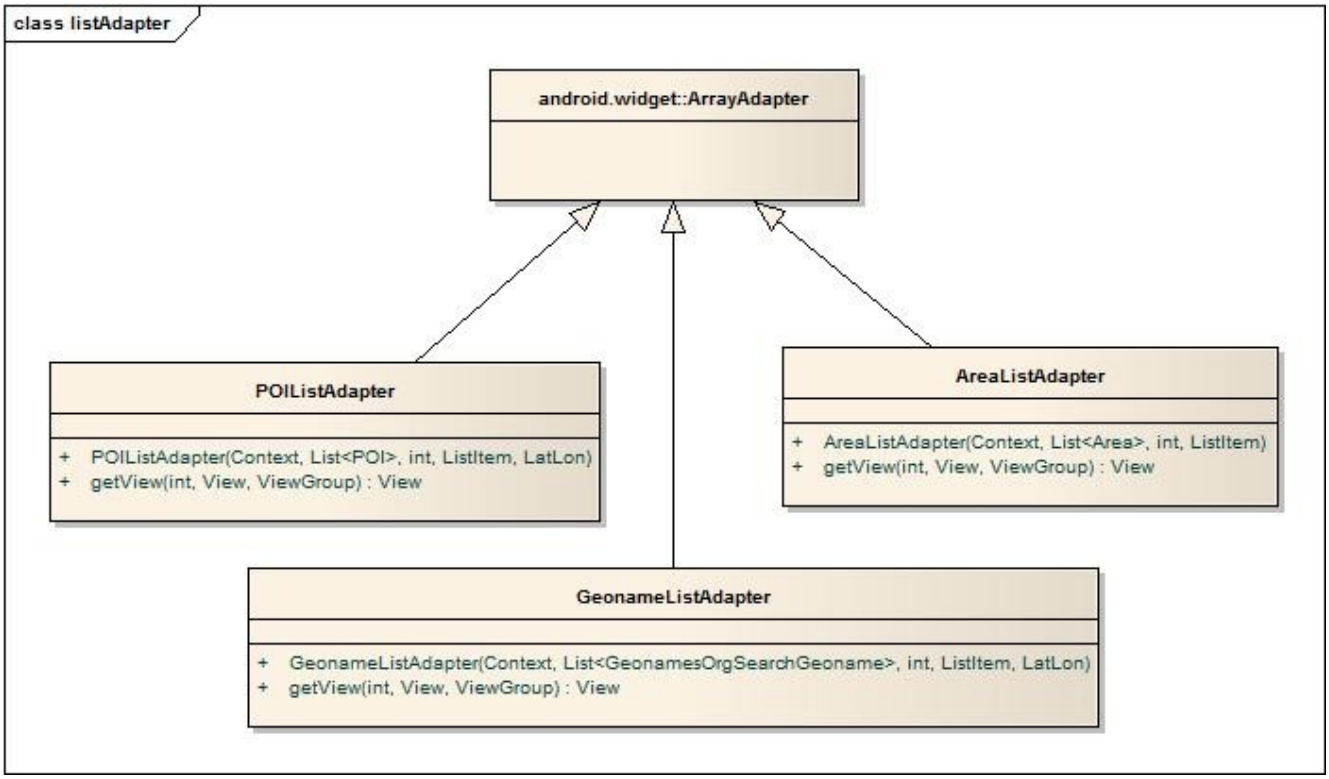
This diagram shows the interaction between all application activities. Blue arrows represent the interactions via Context Menu.

3.4.2. Package gui.listAdapter

3.4.2.1. Description

This package includes custom list adapters that manage different list views backed by an array of arbitrary objects. Each class overrides `getView(int, View, ViewGroup)` to return the type of view we want.

3.4.2.2. Diagram



3.4.2.3. Classes and operations description

1. POIListAdapter	
Custom list adapter to manage the POI list view.	
Operations	
`POIListAdapter( Context, List<POI>, int, ListItem, LatLon)`	• `ListItem` is the view with the `TextView` within the layout resource to be populated. • `LatLon` needs to calculate a distance from current position to the POI.

2. GeonameListAdapter	
Custom list adapter to manage the geoname list view.	
Operations	
`GeonameListAdapter( Context, List<GeonamesOrgSearchGeoname>, int, ListItem, LatLon)`	• `ListItem` is the view with the `TextView` within the layout resource to be populated. • `LatLon` needs to calculate a distance from current position to the geoname-area.

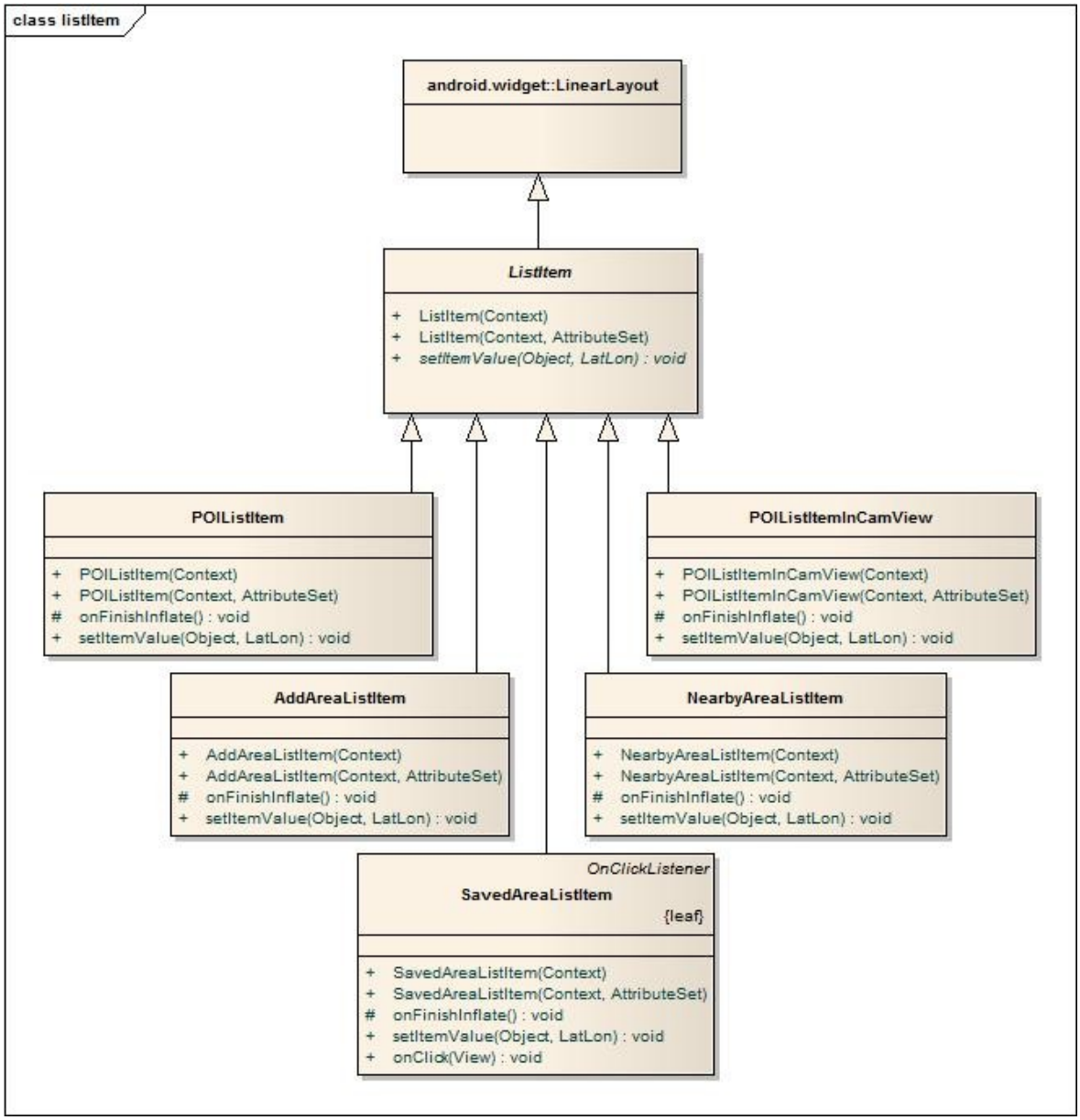
3. AreaListAdapter	
Custom list adapter to manage the Area list view.	
Operations	
`GeonameListAdapter( Context, List<Area>, int, ListItem)`	`ListItem` is the view with the `TextView` within the layout resource to be populated.

3.4.3. Package gui.listItem

3.4.3.1. Description

All `ListItem`-classes in this package represents a single list row and defines its layout.

3.4.3.2. Diagram



3.4.3.3. Classes and operations description

1. ListItem	
This abstract class declares which method should be implemented in subclasses. This class was created for better handling between different `ListAdapters`.	
Operations	
`setItemValue(Object, LatLon)`	This method sets an object to list. `LatLon` is to calculate a distance between current position and an object.

2. AddAreaListItem
This `ListItem` is used in `AddAreaActivity` to list the found areas.

3. NearbyAreaListItem
This `ListItem` is used in `NeabyAreasActivity`.

4. POIListItem
This `ListItem` is used in `SearchTargetActivity`.

5. SavedAreaListItem
This `ListItem` is used in `SavedAreaActivity` and includes three buttons near the text.

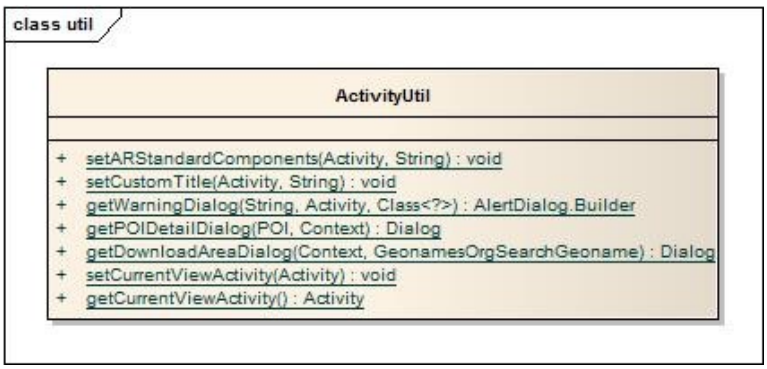
6. POIListItemInCamView
This `ListItem` is used in `POIViewActivity` and in `NavigationViewActivity` to list POI in camera view.

3.4.4. Package gui.util

3.4.4.1. Description

This package contains an utility class which provides methods that support the developer at tasks which have to be done repeatedly.

3.4.4.2. Diagram



3.4.4.3. Classes and operations description

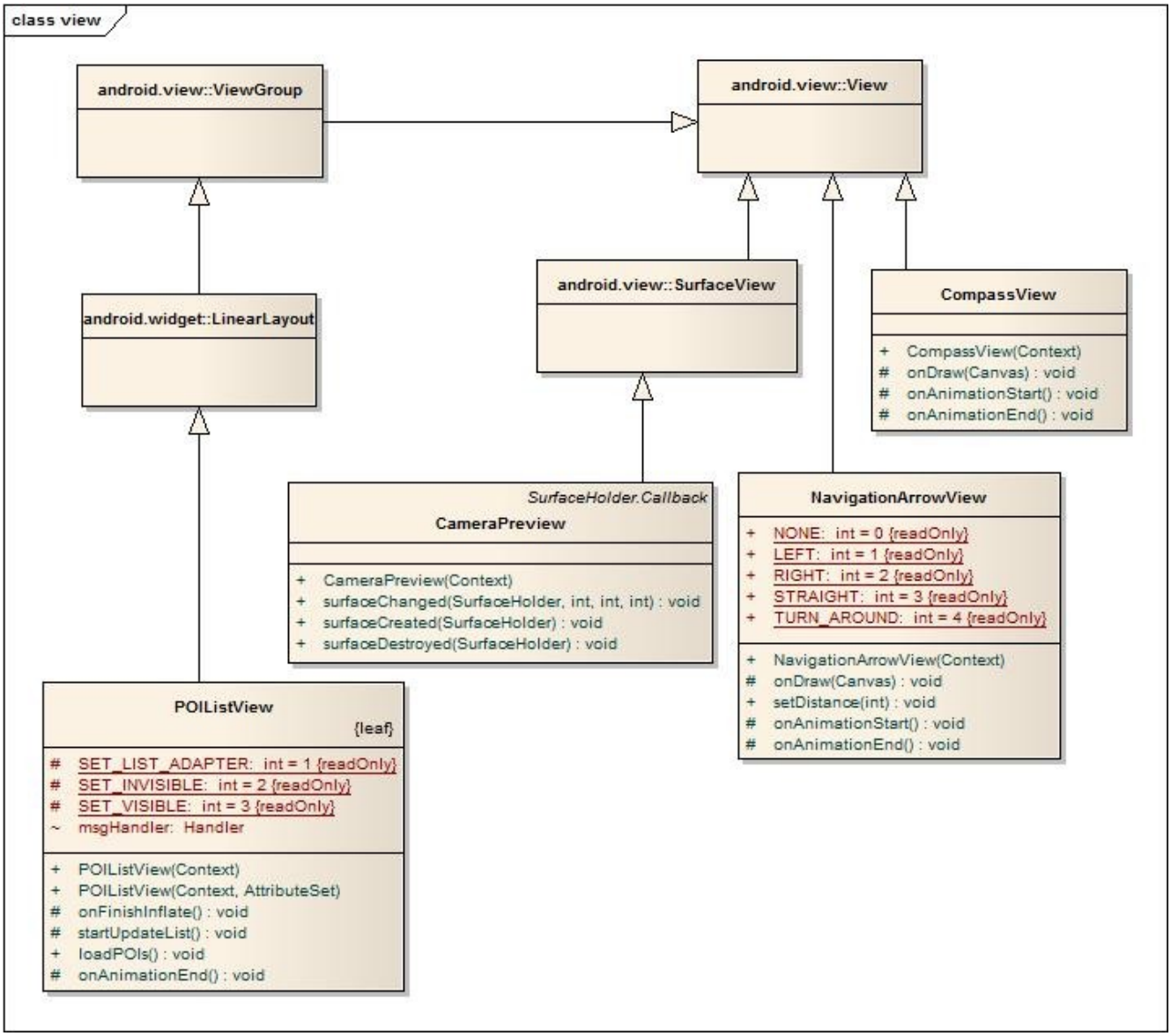
1. ActivityUtil	
Contains static methods to make the handling and setup of standard elements in certain Activitys easier.	
Operations	
`setARStandardComponents(Activity, String)`	This method sets the standard-used components in the augmented reality Activitys (CameraPreview and CompassView)
`setCustomTitle(Activity, String)`	Sets the custom title layout and fills it with the title text.
`Alert.Builder getWarningDialog(String, Activity, Class<?>)`	Return the Alert.Builder to show the warning.
`Dialog getPOIDetailDialog(POI, Context)`	Returns detail dialog for given POI.
`Dialog getDownloadAreaDialog(Context, GeonamesOrgSearchGeoname)`	Returns yes/no dialog to download an area for given geoname.
`setCurrentViewActivity(Activity)`	Sets which Activity currently is displayed. Currently, the only two Activitys that get registered with this method are either POIViewActivity or NavigationViewActivity.
`getCurrentViewActivity()`	Gets which Activity currently is displayed. Currently, the only two Activitys that could be returned are either POIViewActivity or NavigationViewActivity.

3.4.5. Package gui.view

3.4.5.1. Description

This package contains views, which are displayed as a half transparent layer in the `CameraPreview` (augmented reality).

3.4.5.2. Diagram



3.4.5.3. Classes and operations description

1. POIListView

This class represents a POI list with a half transparent background.

2. CameraPreview

This view is used as a background (main content) view for the `POI-` and `Navigation-ViewActivity` and is part of augmented reality.

3. CompassView

The compass is painted in this `View`. The compass direction is dependent from Android hardware sensor that delivers the sensor values.

4. NavigationArrowView

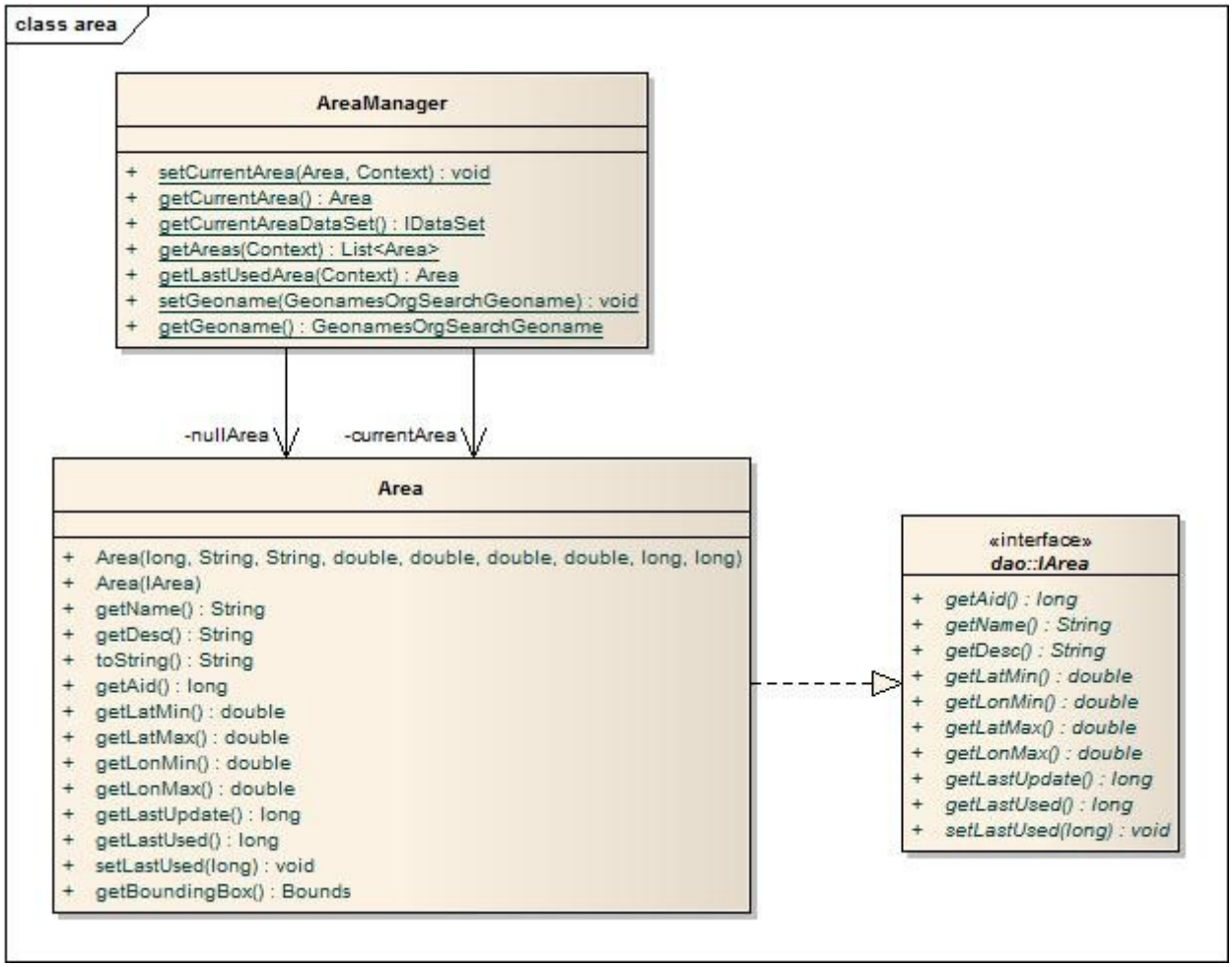
This view is responsible for displaying the direction-arrow for navigation.

3.4.6. Package business.area

3.4.6.1. Description

This package contains Area business object and its manager.

3.4.6.2. Diagram



3.4.6.3. Classes and operations description

1. Area	
This class represents an Area of Interest and includes all area properties.	
Operations	
`Area(IArea)`	Copy constructor from `IArea` to `Area`.

2. AreaManager	
This class manages all operations around the `Area`.	
Operations	
`IDataSet getCurrentAreaDataSet()`	Returns the `IDataSet` that is associated with the current `Area`.
`List getAreas(Context)`	Returns the list of all `Areas` from the database.
`Area getLastUsedArea(Context)`	Gets the last used area .

`setGeoname(GeonamesOrg-SearchGeoname)`
 	
`getGeoname()`
	These methods are used for the download of an area.

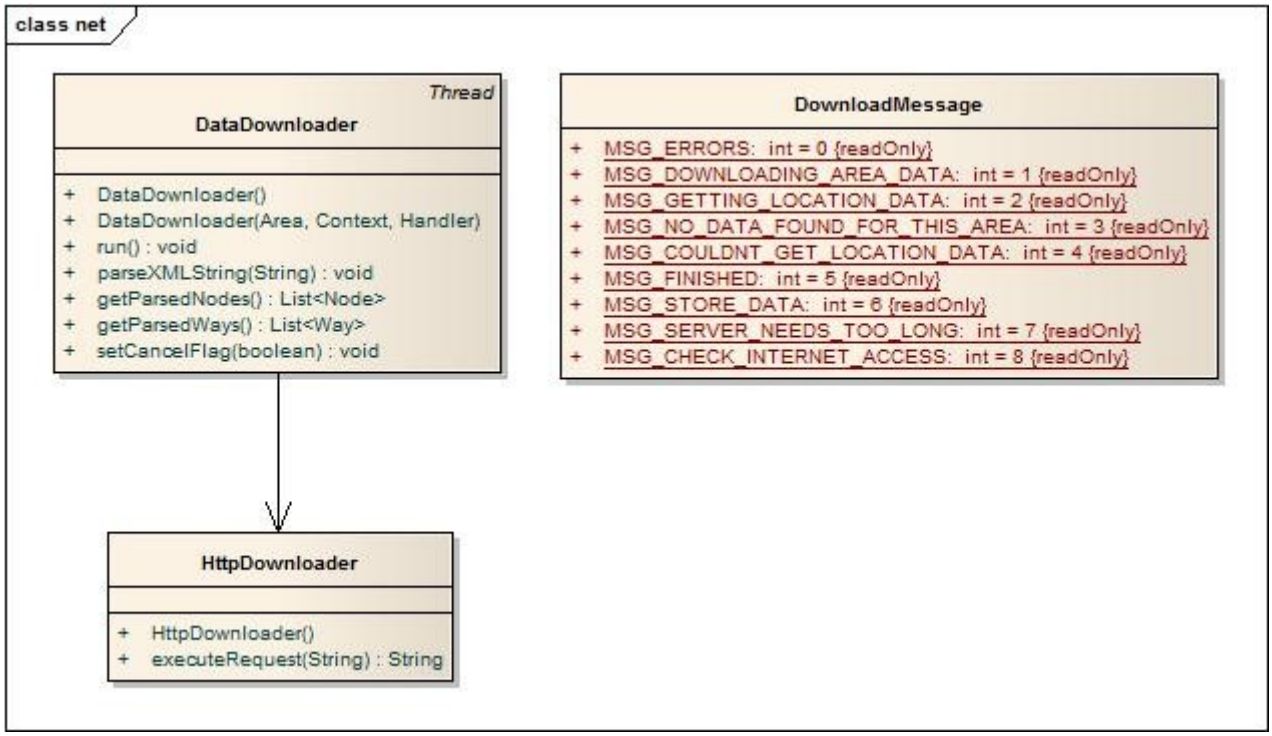
For a description of the `IArea` interface, please have a look at chapter "3.4.16. Package persistence.dao" on page 65.

3.4.7. Package business.net

3.4.7.1. Description

This package includes classes which executes a HTTP request and store its data into the database.

3.4.7.2. Diagram



3.4.7.3. Classes and operations description

1. HttpDownloader	
This is a simple convenience class that makes life easier when dealing with `HttpClient`.	
Operations	
`String executeRequest( String uri)`	Execute a HTTP GET request on the given URI and return a string containing the unmodified result of the request.

2. DataDownloader	
This class is used in `DownloadAreaActivity` and `UpdateAreaActivity` to download, update when necessary and store Area data. • It stores/updates an Area with the bounding box and a name. • Download the Open Street Map data for this bounding box and store/update this data to the database. • Gets the fingerprints for this Area from IndoorWPS server.	
Operations	
`run()`	Override the `Threads run` method and execute all above described tasks.
`parseXMLString(String)`	This method parses the xml string, which was returned from `HttpDownloader.executeRequest(String)`.

3. DownloadMessage

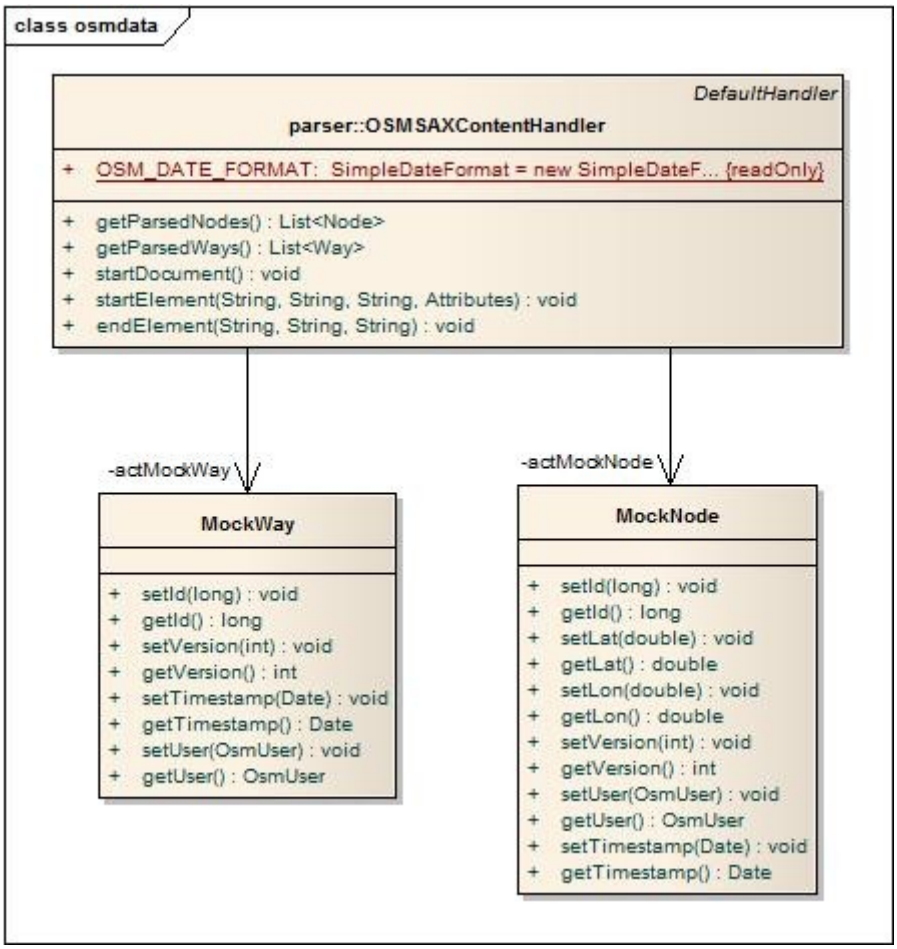
This class includes constant messages for better handling between `DataDownloader` and the `Activitys`, like `DownloadAreaActivity` (to download an area) and .`AddAreaActivity` (to download or update an area)

3.4.8. Package business.osmdata

3.4.8.1. Description

The osmdata package includes OpenStreetMap mock-classes and a `SAXContentHandler` to parse the OSM-data.

3.4.8.2. Diagram



3.4.8.3. Classes and operations description

1. MockWay

This class represents a mock object of one `Way` and is used for getting the data from the OSM server.

2. MockNode
This class represents a mock object of one Node and is used for getting the data from the OSM server.

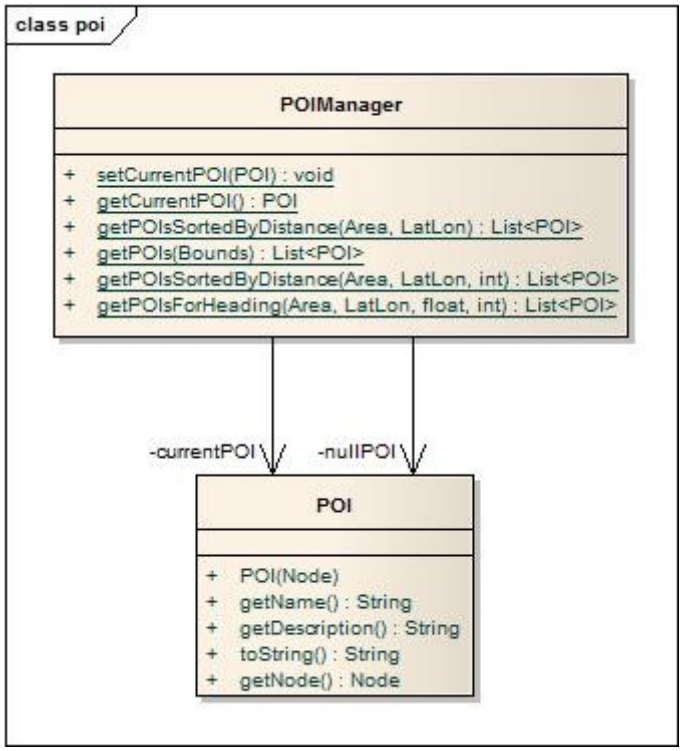
3. OSMSAXContentHandler
This content handler takes the result of a OSM data from api.openstreetmap.org and builds objects from the answer.

3.4.9. Package business.poi

3.4.9.1. Description

This package contains the class for POI business objects and the class for their manager.

3.4.9.2. Diagram



3.4.9.3. Classes and operations description

1. POI	
This class represents a Point of Interest and includes all of its properties.	
Operations	
`POI(Node)`	Wrap an OpenStreetMap `Node` into a `POI`.

2. POIManager	
This class manages all operations around the POI.	
Operations	
`List getPOIs(Bounds)`	This method returns a sorted by name list of POIs inside of the given bounding box.
`List getPOIsForHeading(Area, LatLon, float, int)`	This method returns a sorted by distance list with given number of nearest POIs and given compass heading inside of this Area.
`List getPOIsSortedByDistance(Area, LatLon)`	Returns a list of POIs inside a given Area, sorted by the calculated distance from a given position.
`List getPOIsSortedByDistance(Area, LatLon, int)`	This method returns the number of POIs inside a given Area, sorted by the calculated distance.

3.4.10. Package business.position

3.4.10.1. Description

This package provides classes that deal with everything that has to do with obtaining the actual position or dealing with data from geoinformation web-services.

3.4.10.2. DIAGRAM

3.4.10.3. Classes and operations description

1. Localizer	
This class is a helper to obtain the actual location fix of the device based on GPS and/or (if available/enabled) GSM cell locations. It is implemented as a Singleton to ensure, that only one instance exists application-wide and every caller has (read-)access to the same data.	
Operations	
`Localizer getInstance(Context)`	Creates and/or returns the instance of this `Localizer` service.
`Location getLocation()`	This method tries to determine the actual location based on Wireless LAN, GPS and / or GSM cell ID. If none of those three `LocationProviders` are enabled or the location could not be determined, this method will try to obtain the last known location by the GPS receiver or GSM cell. If all this fails, the method returns a default `Location` with the coordinates (1/1), but never `NULL`.

2. WlanLocationFinder	
This is our business access to the `PointZero` API. It implements the `Runnable` interface, so it can be run on its own in the background and automatically try to get the actual position of the phone, based on Wireless LAN fingerprints.	
Operations	
`setLocationUpdateReceiver(Handler)`	Sets the `Handler` that should receive a message, when the location was changed. It the has to query this class for the new `Location`.
`clearFingerprintCache()`	This clears the cache of stored Wireless LAN fingerprints, forcing the `PointZero` library to download new data for the last known position.
`Location getLocation()`	Returns the last determined `Location` by this class.
`String getLocationProvider()`	Returns the name of this provider (constant "IndoorWPS").

3. GazetteerQuery	
This is the business interface to the geonames.org gazetteer service. It is able to return a list of known geonames with given a name to search for (the list also contains the GPS coordinate of each found geoname) or to do a reverse geocoding, which means search for places in the vicinity of a given GPS coordinate (lat/lon combination). The term "gazetteer" references in the context of this class always to the ws.geonames.org gazetteer services.	
Operations	
`List getParsedGeonames()`	Returns the list with the parsed `GeonameOrgSearchGeoname` objects.
`String search()`	Execute the search of the search as a string.
`parseXMLString()`	Tries to parse a given XML string. The string should be the result of the search-method.

4. GeonamesOrgSearchSAXContentHandler
This `ContentHandler` implementation takes the result of a name-query of http://ws.geonames.org and builds objects from the answer.

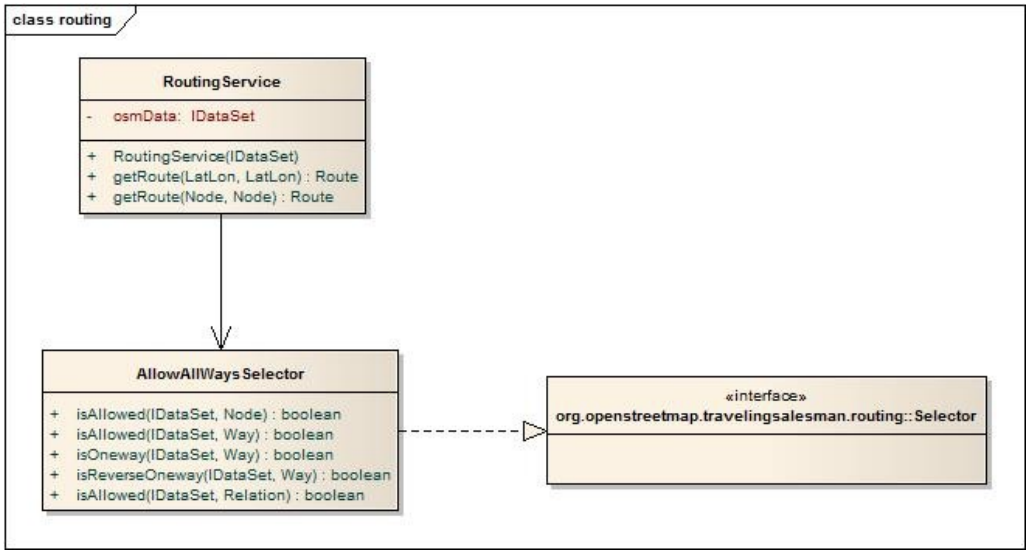
5. GeonameOrgSearchGeoname
This is a partial object representation of parsed data from a geonames.org geoname query.

3.4.11. Package business.routing

3.4.11.1. Description

This package builds the business interface to the `OSMNavigation` [OSMNav] routing core implementation.

3.4.11.2. Diagram



3.4.11.3. Classes and operations description

1. RoutingService	
Service that delivers an interface to the `OSMNavigation` routing-core. This service uses the `MultiTargetDijkstraRouter` as routing algorithm.	
Operations	
`Route getRoute(Node, Node)`	This method tries to calculate a route between the two Nodes. No restrictions are applied which means any possible way is used.
`Route getRoute(LatLon, LatLon)`	This is basically the same as `getRoute(Node, Node)` but before the route calculation, it tries to determine the closest Nodes to the given `LatLon` coordinates.

2. AllowAllWaySelector
Simple `Selector` implementation that allows all ways to be used for routing. Please refer to the documentation of `OSMNavigation` for method description.

3.4.11.4. Interfaces

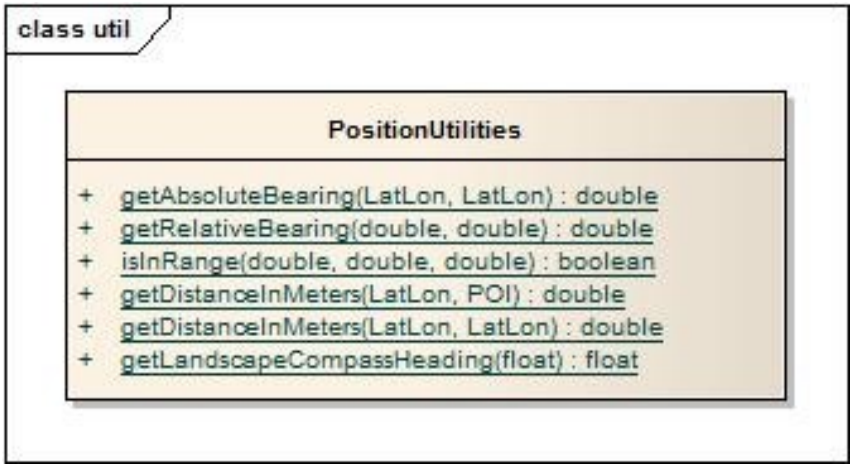
Interface	Description
`Selector`	Instances of this interface can determine if a given `Node`, `Relation` or `Way` is allowed to route through.

3.4.12. Package business.util

3.4.12.1. Description

This package provides utility classes and methods to simplify the work with position data like distance or angle calculations.

3.4.12.2. Diagram



3.4.12.3. Class and operations description

PositionUtilities	
Contains static methods for better handling of position and distances.	
Operations	
`double getAbsoluteBearing(LatLon, LatLon)`	This method calculates the absolute bearing between to given LatLon points relative to true North.
`double getRelativeBearing(double, double)`	This calculates the relative bearing respecting the actual heading.
`double getDistanceInMeters(LatLon, POI)`	This method returns the distance between the devices actual position and the POI position.
`double getDistanceInMeters(LatLon, LatLon)`	This method calculates the distance between two given coordinates.
`float getLandscapeCompassHeading (float)`	This method corrects the given compass value so that it shows the correct heading when the application is displayed in landscape mode (the compass sensor always returns the heading, the tip of the phone is turned to). It basically just adds 90 degrees to the compass value and then performs a compass-range check.
`boolean isInRange(double, double, double)`	Convenience method that checks whether a given value is in a specified range.

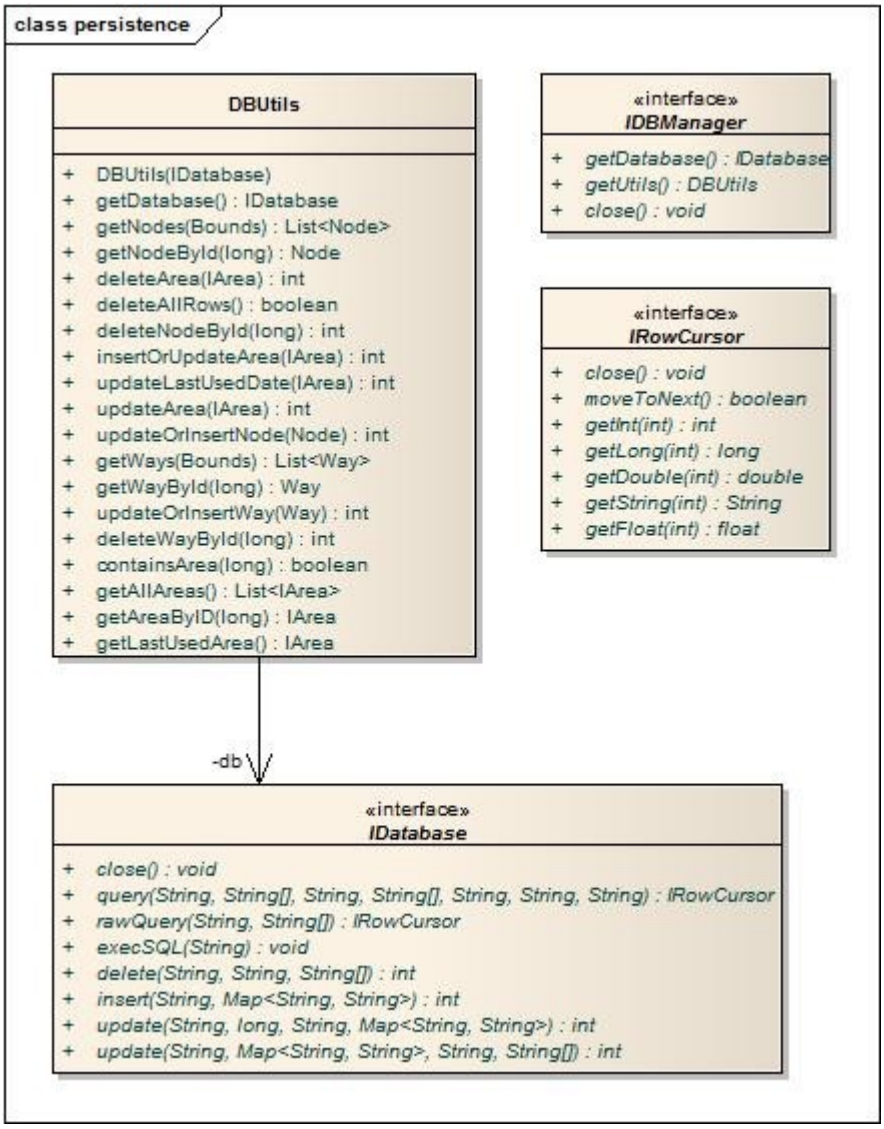
3.4.13. Package persistence

3.4.13.1. Description

The persistence package acts as the interface between the business layer and the database. It provides functionality to persist objects from a higher layer into the database and also to reload persisted objects, represented as database tuples, from the database back into active objects.

To provide the possibility to test this package and its sub-packages automatically, we do not interact directly with Android's SQLite classes (please refer to "3.2. Architectural goals and constraints" on page 38 to learn about the reasons for this approach). Instead, another thin abstraction layer is inserted by applying the well-known Strategy (GoF) design pattern.

3.4.13.2. Diagram



3.4.13.3. Classes and operations description

1. DBUtils	
Utility class, that does the O/R mapping between the OpenStreetMap library data types and our representation of them in the database.	
Operations	
`containsArea(long)`	This method checks whether a specific `Area` with a given ID exists in the database.
`deleteAllRows()`	This method deletes all data rows from the database.
`deleteArea(IArea)`	This method deletes the given `Area` from the database. This also deletes elements like `Nodes` with `Tags` that contains in this `Area` and `Ways` in the database that refers to the given `Nodes` in this `Area`.
`deleteNodeById(long)`	These methods delete a `Node` from the database by

	OSM ID.
deleteWayById(long)	Deletes the tuple with the given ID from the way storage table. This also deletes following: • any Tag in the database that refers to the given Way ID. • any WayNode in the database that refers to the given Way ID (Nodes that are referenced by a WayNode become explicitly NOT deleted)
getAreaByID(long)	Gets Area for given ID.
getLastUsedArea()	This method returns the last used Area from the database.
IDatabase getDatabase()	This method returns the implementation of IDatabase
insertOrUpdateArea(IArea)	Inserts a new Area into the database or updates an existing row.
List getAllAreas()	Returns all Areas from the database.
List getNodes(Bounds)	This method lists OSM-Nodes inside of given bounding box.
List getWays(Bounds)	Fetches all Ways from the database that are completely in or have at least one Node inside the given bounding box (way that intersect the Bounds).
Node getNodeById(long)	This method returns the Node that matches with the given OSM ID.
updateArea(IArea)	This method updates the data that belong to the given Area and sets the date of last update to *now*.
updateLastUsedDate(IArea)	This method updates the Last Used Date of the given Area and sets it to now.
updateOrInsertNode(Node)	This method updates an existing Node and its data or creates a new Node with new data.
updateOrInsertWay(Way)	This method updates an existing Way in the database or create a new one.
Way getWayById(long)	This method returns the Way that matches the given OSM ID.

3.4.13.4. INTERFACES

1. IDATABASE

This Interface defines a basic set of methods that must be provided by all concrete implementations/database backends (eg. SQLite, generic JDBC).

This interface exists to implement the GoF Strategy design pattern which is needed for purpose of being able to test the persistence layer with JUnit on a development machine or a build server.

Operations

`IRowCursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)`	Query the given table, returning an `IRowCursor` over the result set.
`IRowCursor rawQuery(String sql, String[] selectionArgs)`	Runs the provided SQL and returns an `IRowCursor` over the result set.
`execSQL(String sql)`	Executes the given SQL statement, which may return multiple results. Currently it is primary used for CREATE statements, that need no check.
`delete(String table, String whereClause, String[] whereArgs)`	Convenience method for deleting rows in the database.
`insert(String table, Map<String, String> calsAndValues)`	Convenience method for inserting rows in the database.
`update(String table, long id, String idDesc, Map<String, String> colsAndValues)`	Convenience method for updating a single row in the database.
`update(String table, Map<String, String> colsAndValues, String whereClause, String[] whereArgs)`	Convenience method for updating a single row in the database.

2. IDBManager

This is an interface that defines methods for basic management of a database.

Operations	
`IDatabase getDatabase()`	This method tries to open the managed database with write access.
`DBUtils getUtils()`	Delivers the `DBUtils` instance to interact with the database that belongs to this manager.
`close()`	Closes the database.

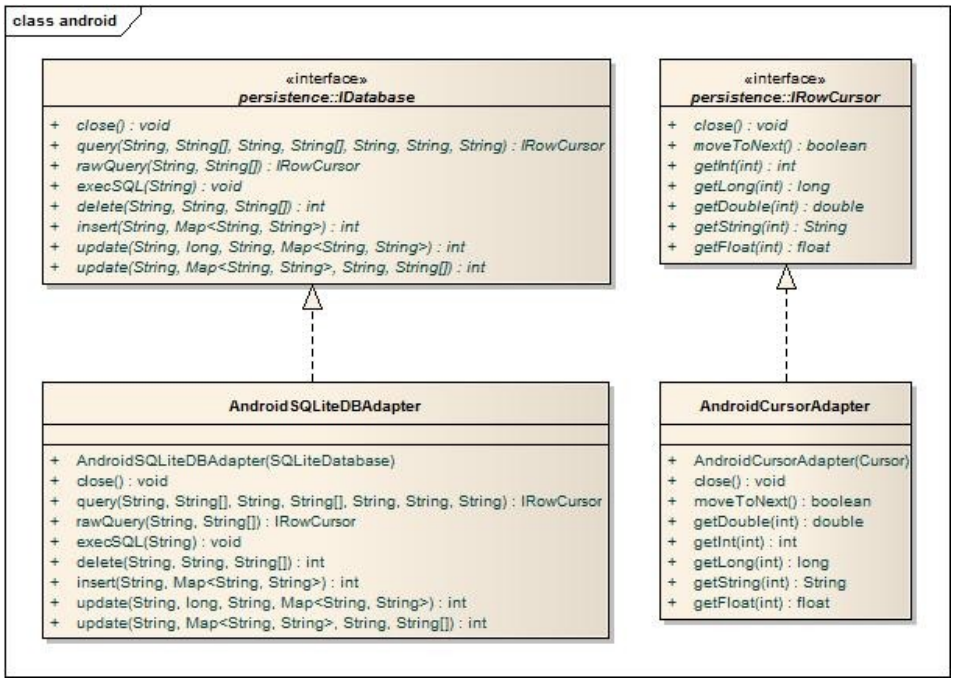
3. IRowCursor	
This interface defines all methods, that are essential for interaction with database query-results, i.e. getting data and closing the database cursor.	
Operations	
`close()`	Closes the cursor, releasing all of its resources and making it completely invalid.
`moveToNext()`	Move the cursor to the next row
`int getInt(columnIndex)`	Returns the value of the requested column as an `int`.
`long getLong(columnIndex)`	Returns the value of the requested column as a `long`.
`double getDouble(columnIndex)`	Returns the value of the requested column as a `double`.
`String getString (columnIndex)`	Returns the value of the requested column as a `String`.
`float getFloat(columnIndex)`	Returns the value of the requested column as a `float`.

3.4.14. Package persistence.adapter.android

3.4.14.1. Description

This packages hosts implementations of the interfaces defined in the persistence package as one concrete strategy implementation. The implementations are intended to run on the Android platform and interact with the `SQLite` database there.

3.4.14.2. Diagram



3.4.14.3. Classes description

1. AndroidCursorAdapter

This is an implementation of the `IRowCursor` interface. It is an object adapter which takes a `Cursor` (`android.database.Cursor`) implementation and lets with results the `SQLite` database engine of Android.

2. AndroidSQLiteDBAdapter

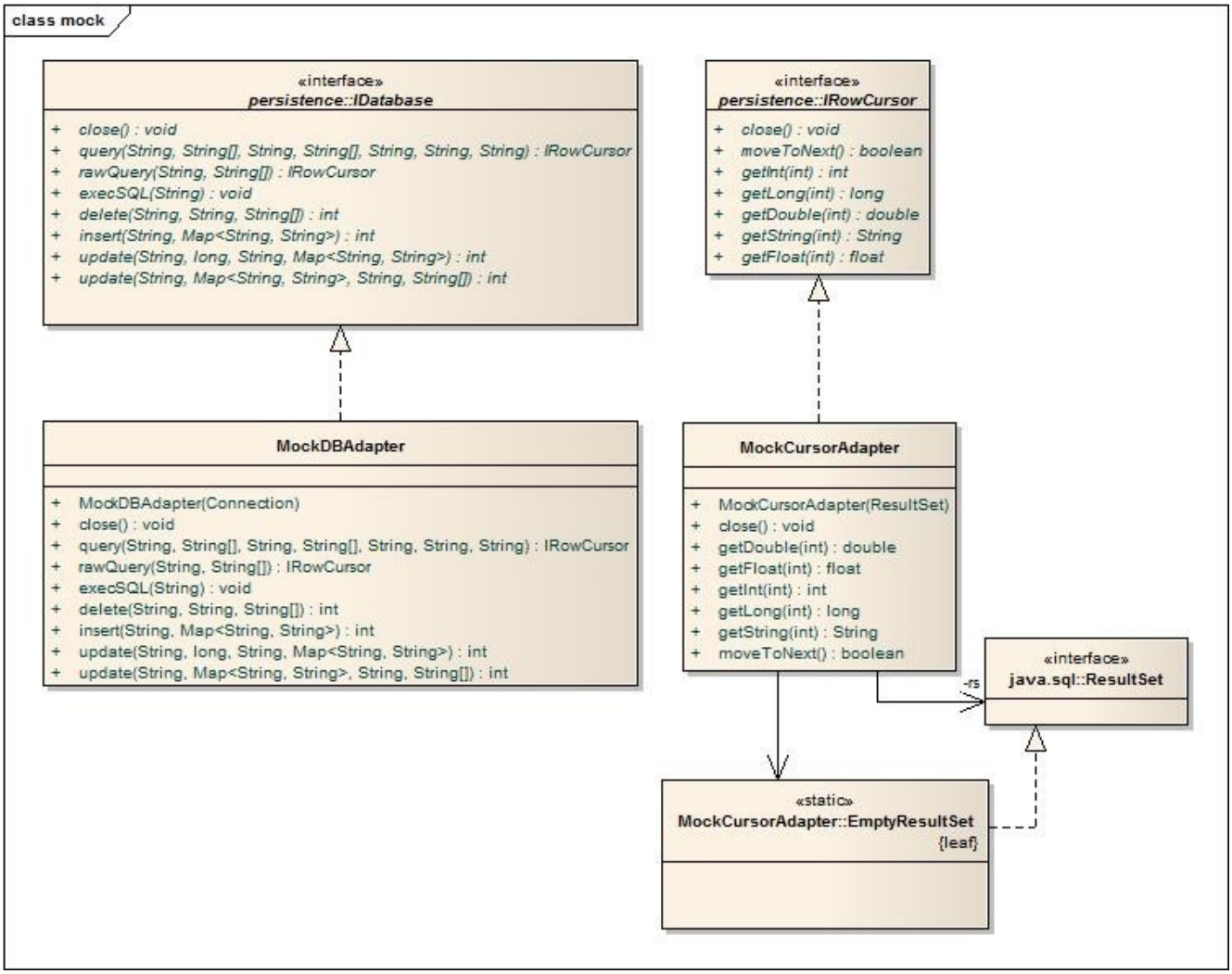
This is an implementation of the `IDatabase` interface. It is an object adapter which takes a `SQLiteDatabase` (`android.database.sqlite.SQLiteDatabase`) and thus lets interact (query, send SQL, etc.) with the `SQLite` database engine of Android.

3.4.15. Package persistence.adapter.mock

3.4.15.1. Description

This packages hosts implementations of the interfaces defined in the persistence package as another concrete strategy implementation. The implementations are intended to enable automated persistence layer testing with JUnit.

3.4.15.2. Diagram



3.4.15.3. Design decisions

Decision	Description
EmptyResultSet	Application of Null-Object pattern for a missing `ResultSet` ("provide something for nothing"). This is necessary due to a bug (mis concept?) of the `SQLite` JDBC implementation. When no results are found in a SELECT query, then an ugly `SQLException` with a nothing-meaning message is thrown.
Mock	In this project the sole use for this adapter is to enable automated persistence layer testing with JUnit.

3.4.15.4. Classes and operations description

1. MockDBAdapter

This is an implementation of the `IDatabase` interface. It is an object adapter which takes a `Connection` (`java.sql.Connection`) and thus lets interact (query, send SQL, etc.) with any kind of JDBC database connection.

2. MockCursorAdapter

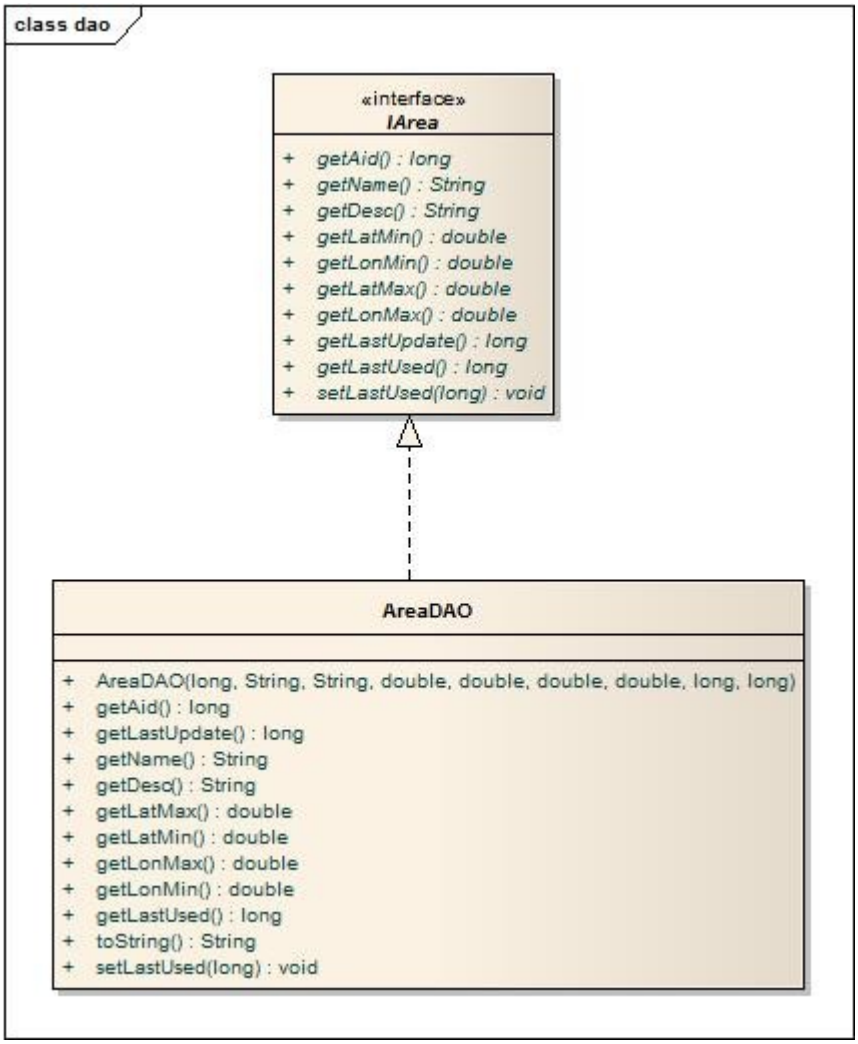
This is an implementation of the `IRowCursor` interface. It is an object adapter which takes a `ResultSet` (`java.sql.ResultSet`) and lets interact with any kind of result sets produces by basically any kind of JDBC database connection.

3.4.16. Package persistence.dao

3.4.16.1. Description

This package includes interfaces for business objects and it data access object implementation.

3.4.16.2. Diagram



3.4.16.3. Interfaces

1. IArea

Interface that defines all methods, that must be implemented by a class, if it wants to be an Area.

3.4.16.4. Classes and operations description

1. AreaDAO

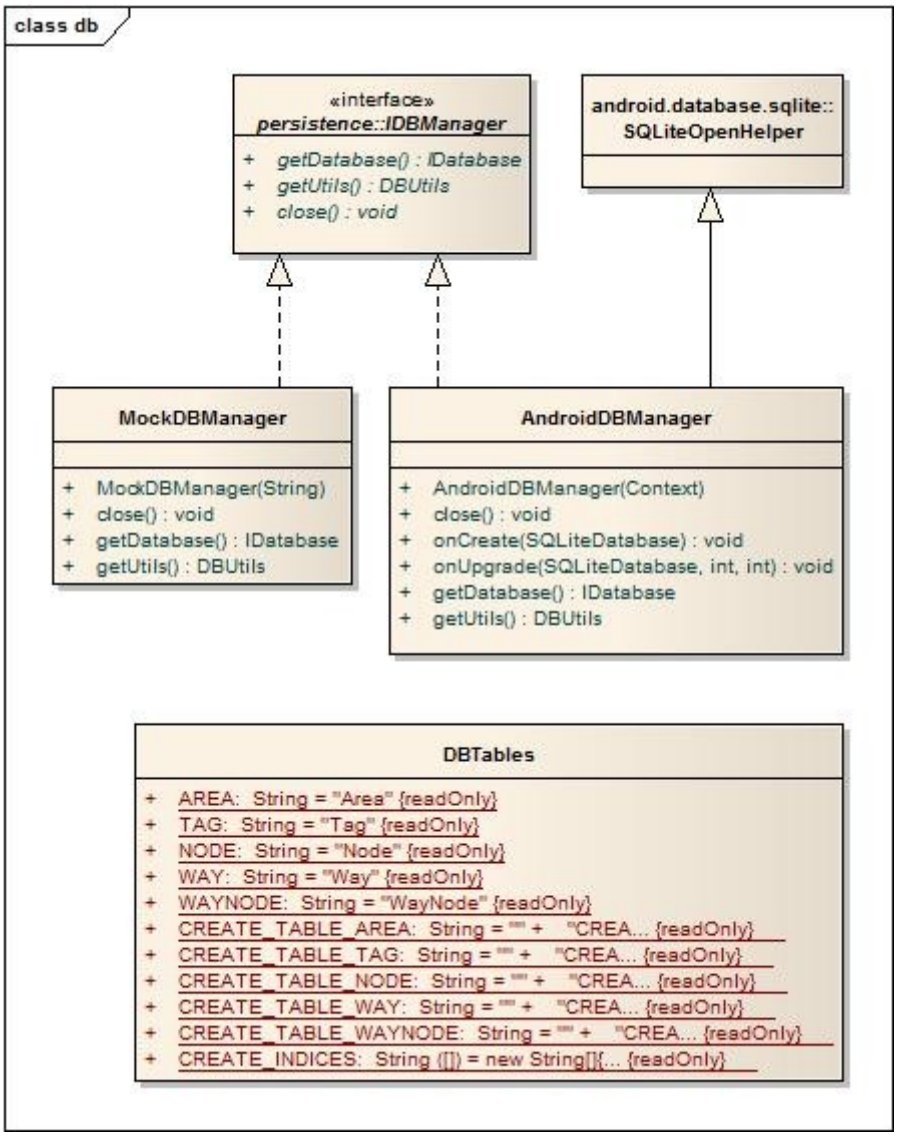
This class implements all methods from IArea-Interface.

3.4.17. Package persistence.db

3.4.17.1. Description

This package includes all classes that affect the database.

3.4.17.2. Diagram



3.4.17.3. Interfaces

1. IDataSet

IDataSet is the data behind the OSM application. It can consist of only a few points up to the whole OSM database.

3.4.17.4. Classes and operations description

1. MockDBManager

This is the database management class for basic interaction with the mock/testing database used in this project.

2. AndroidDBManager

This is the database management class for basic interaction with the SQLite database used in this project.
It is responsible for the opening (writable) of the database and (if not exist) its creation and the correct creation of all needed tables.

3. DBTables

Class of all database tables and their CREATE SQL scripts used in the IndoorGuides main database.

3.4.18. Package util

3.4.18.1. Description

This package includes utilities for this project.

3.4.18.2. Diagram

3.4.18.3. Design decisions

Because of emulator limitations on SDK 1.5 we had to integrate a workaround to test our application on the emulator. Following limitations affect us:

- No support for camera/video capture (input).
- No support for sensors and WiFi.

3.4.18.4. Classes and operations description

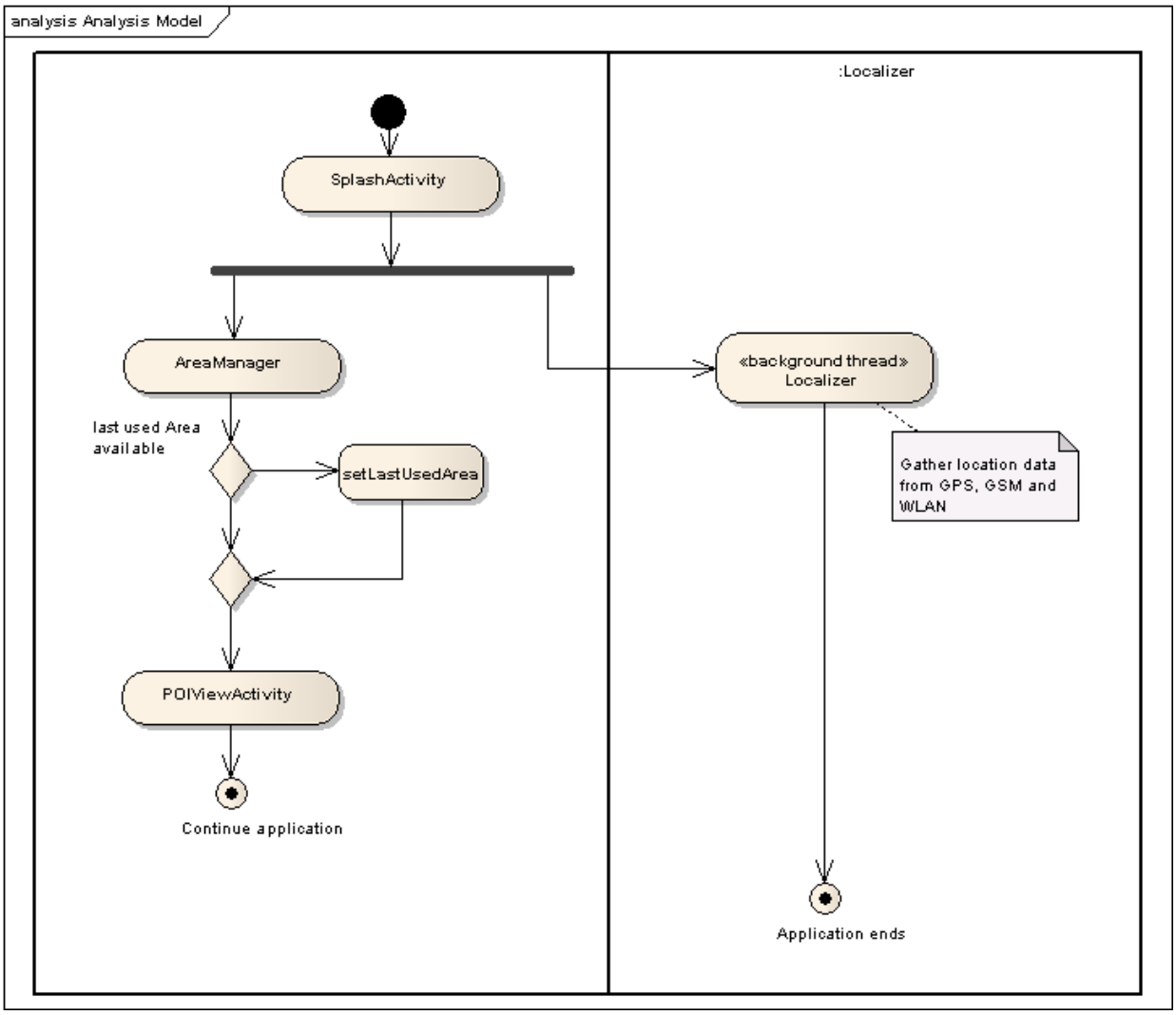
1. EmulatorUtilities	
This class is used for implementation on emulator.	
Operations	
`setIsEmulator(boolean)`	While developing in the emulator, this should always be TRUE.

3.5. Process view

According to the Android documentation, it runs each Activity in the main program process (which in turn is a separate instance of the Dalvik VM). As this is fine normally, there is sometimes the need for a background process or thread which does some permanent calculation.

The following activity diagram shows only the most interesting and noteworthy threads.

3.5.1. Diagram

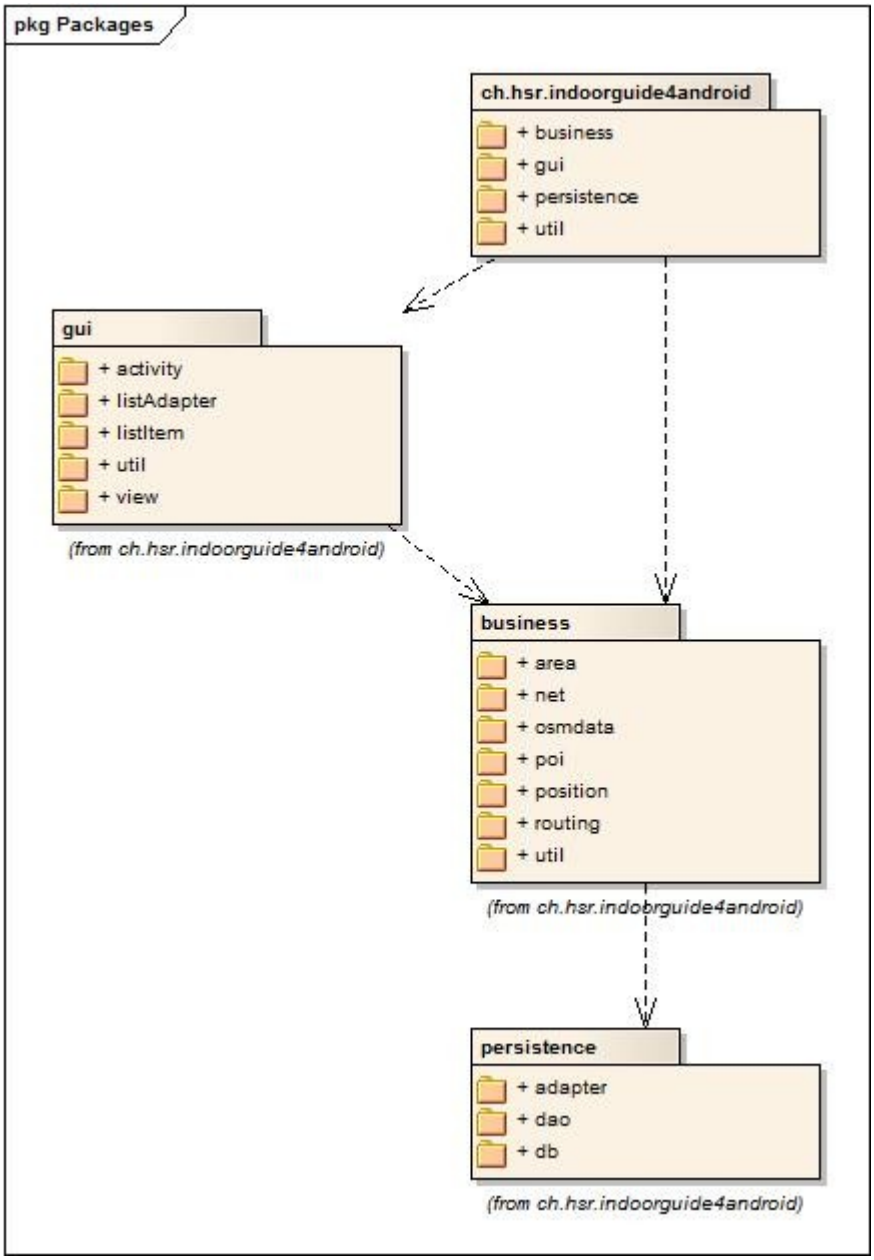


3.6. Implementation view

3.6.1. Overview

As visible in the following figure, the software consists of basically three layers. The role and responsibility of each layer is described in the next sub-chapter. It is intended, that each layer is independent from higher-level layers and could be exchanged during a possible port to another platform. For example, if a new way of persistence or another database is used, a developer just has to exchange the persistence layer.

The access from layer to layer is strictly from higher to lower layers. This means, that you can access the business- and persistence layer from the UI, but not the UI from the business- or persistence layer.



3.6.2. Layers

1. GUI

This layer is responsible for anything that is visible to an user of the application or has to do with it. It contains all Android-Activities, List-Adapters or Views. It does no calculations or long running tasks, but it can instruct the business layer to do so; during the long running tasks, the GUI layer display a progress dialog to keep the user informed about the overall progress.

2. Business

The business layer's responsibility is to implement the core logic of the whole application and also to interact with the persistence layer.
To bring some structure into the business layer, it has been divided into further sub-layers which each take care of special problems. One example of such a sub-layer is the "routing" layer which deals with anything that has to do with the routing and navigation (but of course not the display of it in the user interface).

3. Persistence

The persistence layer is responsible for interaction with the database (where theoretically other ways of persisting data are possible) and the mapping of business objects from the business layer to the persistence format and vice versa.

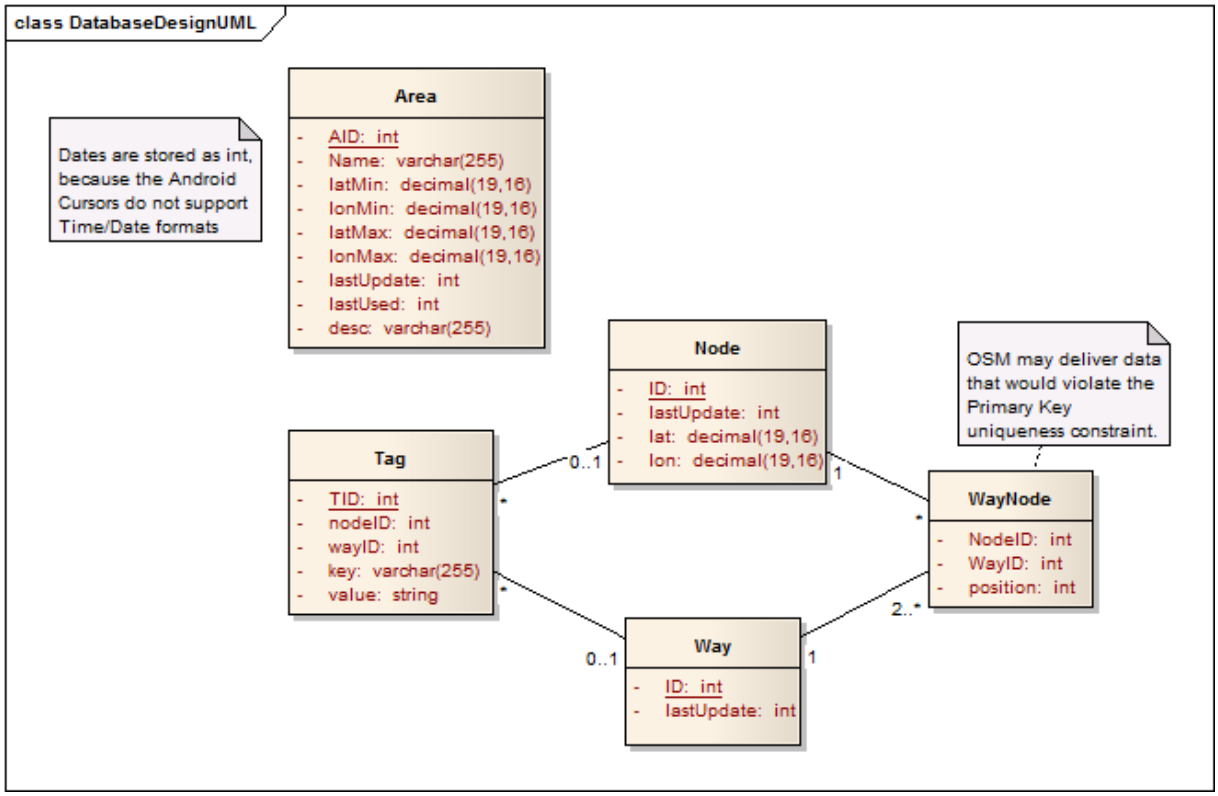
3.7. Data view

3.7.1. Description

The following UML diagram shows all database tables, that are used inside the application and also displays the references between tables. The ID column in the Node and Way tables stores the OpenStreetMap ID value. This must be considered when the data sources are expanded and data is received not only from OpenStreetMap.

To get further information about the OpenStreetMap data format visit www.openstreetmap.org.

3.7.2. Diagram



3.7.3. Table descriptions

1. Area	
This table holds all information that is necessary to represent an Area business object.	
Attributes	
AID (primary key)	The surrogate primary key. It is a consecutive number that is distributed automatically by the database.
lastUpdate (nullable)	This is the *long*-value of the date, when an area was updated the last time.
lastUsed (nullable)	This is the *long*-value of the date, when an area was used the last time.
latMax (not null)	The maximum latitude value an area covers.
latMin (not null)	The minimum latitude value an area covers.
lonMax (not null)	The maximum longitude value an area covers.
lonMin (not null)	The minimum longitude value an area covers.
Name (not null)	The name of an area as it is presented to the User.
desc(null)	The description of an area as it is presented to the User.
Relations	
none	

2. Node

This table holds all information that is necessary to represent a simplified OpenStreetMap Node business object. Some OpenStreetMap attributes like the UserID are not stored, because they are not relevant for this application.	
Attributes	
ID (primary key)	The OpenStreetMap ID as it is received by the OSM server.
lastUpdate (nullable)	This is the *long*-value of the date, when a node was updated the last time.
lat (not null)	The geographic latitude of a node as it is received from the OSM server.
lon (not null)	The geographic longitude of a node as it is received from the OSM server.
Relations	
Tag	A node may have an undefined number of tags associated.
WayNode	A node may belong to an undefined number of waynodes.

3. Way

This table holds all information that is necessary to represent a simplified OpenStreetMap Way business object. Some OpenStreetMap attributes like the UserID are not stored, because they are not relevant for this application.	
Attributes	
ID (primary key)	The OpenStreetMap ID as it is received by the OSM server.
lastUpdate (nullable)	This is the *long*-value of the date, when a way was updated the last time.
Relations	
WayNode	A way may consist of an undefined number of waynodes, but at least two.
Tag	A way may have an undefined number of tags associated.

4. WayNode

This table holds all information that is necessary to represent an OpenStreetMap WayNode business object. This table intentionally has no primary key, because in OpenStreetMap a way is also used to represent areas like lakes. In that case, a way is "closed" which means the first and last waynode refer to the same node.	
Attributes	
NodeID (foreign key, not nullable)	This is a reference to a tuple in the node-table to give a way node a geographic location.
position (not null)	This column stores the sequence number of a waynode.

	As a way is a sequence of waynodes, the sequence number directly influences the way itself.
WayID (foreign key, not nullable)	This is a reference to a tuple in the way-table.
Relations	
Node	A waynode references always one and only one node.
Way	A waynode belongs always to one and only one way.

5. Tag	
This table holds all information that is necessary to represent an OpenStreetMap Tag business object. A tag is essentially a key/value pair that is associated to a node or a way. For simplicity, this table holds tags for nodes and ways where either the nodeID or the wayID contains a value.	
Attributes	
key (nullable)	The key of the tag.
nodeID (foreign key, nullable)	If a tag belongs to a node, then this column contains the id of the referenced node.
TID (primary key)	The surrogate primary key. It is a consecutive number that is distributed automatically by the database.
value (nullable)	The value of the tag. As the value may be very long, i.e. a description of a Point of Interest, this column is defined as a database datatype that is able to hold such big values.
wayID (foreign key, nullable)	If a tag belongs to a way, then this column contains the id of the referenced way.
Relations	
Node	A tag may belong to zero or one node.
Way	A tag may belong to zero or one way.

4. Implementation und Test

4.1. Implementation

Informationen zur Implementation sind im Kapitel „3.6. Implementation view“ auf Seite 70 beschrieben.

4.2. Automatische Testverfahren

Während der Entwicklung wurde grosser Wert auf Test Driven Development („test a little, code a little“) und das automatisierte Unit Testing gelegt.

Das Werkzeug der Wahl war JUnit 4, welches einerseits mit der Eclipse Entwicklungsumgebung, als auch mit Apache's Ant verwendbar ist.

Durch die konsequente Anwendung der TDD Prinzipien sind der Persistenz- und der Business-Layer zu einem grossen Teil ausgetestet und können jederzeit automatisch getestet werde. Einige Klassen im Business-Layer können jedoch nicht ausgetestet werden, da sie von Hardware abhängig sind, beispielsweise der Code zur Positionsbestimmung mittels Wireless LAN.

4.3. Systemtests

Die Systemtests wurden am Anfang jedes Prototyps definiert und beim Abschluss mit den vorherigen Systemtests durchgeführt, wenn auch die Tests der vorherigen Prototypen nicht explizit nochmals erwähnt und aufgeführt sind. Somit ist gewährleistet, dass bisherige Funktionalität erhalten bleibt.

4.3.1. Prototyp 1

4.3.1.1. Voraussetzungen

Die unten aufgelisteten Funktionen können nur auf der echten Hardware getestet werden.

4.3.1.2. Vorbereitungen

Android Package für Prototyp 1 auf dem Smartphone installieren.

4.3.1.3. Systemtests

Nr.	Funktionalität	Implementiert	Status	Bemerkung
1	Kompassinformationen auslesen	Ja	OK	Kompass ist sehr empfindlich und reagiert auf die elektrischen Felder der Umgebung mit Ablenkung.
2	Kompassdaten updaten, wenn das Gerät bewegt wird.	Ja	OK	Siehe Bemerkung von 1.
3	Kamera Output auslesen und darstellen	Ja	OK	
4	Kompass-Anzeige im Kamerabild darstellen	Ja	OK	Die Anzeige der Richtungsänderung verzögert sich ein wenig. Die Ausgabe des Kompasses wurde als Kompassnadel dargestellt.
5	Auslesen der Neigung des Gerätes (liegt es horizontal auf dem Tisch oder wird es vertikal in der Hand gehalten)	Ja	OK	

4.3.1.4. Zusammenfassung

Die gewünschte Funktionalität für Prototyp 1 wurde erreicht.

4.3.1.4.1. Bekannte Einschränkungen

Der Kompass ist extrem empfindlich auf jegliche Art von elektromagnetischer Störung und lässt sich leicht ablenken, beispielsweise durch im Boden verlaufende Strom-/Netzwerkkabel.

Weiterhin muss man dem Kompass jeweils einige Sekunden Zeit geben, bis er sich neu ausgerichtet hat.

4.3.2. Prototyp 2

4.3.2.1. Voraussetzungen

Die unten aufgelisteten Tests wurden als JUnit Tests implementiert und die Tests werden mittels JUnit durchgeführt.

4.3.2.2. Vorbereitungen

Projektdatei mit den Daten für Prototyp 2 in Eclipse öffnen und sicherstellen, dass alles sauber kompiliert.

Anschliessend die JUnit Tests ausführen.

4.3.2.3. Systemtests

Nr.	Funktionalität	Implementiert	Status	Bemerkung
1	Test-OSM Datensatz erstellt.	Ja	OK	Testdaten wurden mit JOSM und IndoorWPS Gebäudeplänen erstellt.
2	Lokaler HTTP Server, der einen Test-OSM Datensatz liefert.	Ja	OK	Bisher nur durch JUnitTest verwendet.
3	Erste GUI Implementationen	Ja	OK	Noch nicht alle Masken sind vollständig implementiert. Funktionalität fehlt aufgrund fehlender Implementation in Business-Layers.
4	Navigationssoftware	Ja	OK	OSMNavigation Library wurde integriert. Routenfindung und Navigation mit den Testdaten funktioniert tadellos.

4.3.2.4. Zusammenfassung

Die gewünschte Funktionalität für Prototyp 2 wurde erreicht.

4.3.2.4.1. Bekannte Einschränkungen

Zur Zeit wird lediglich die Navigation auf einem Stockwerk des Gebäudes unterstützt. Eine Erweiterung auf Inter-Stockwerk-Navigation ist in einem späteren Prototyp vorgesehen.

4.3.3. Prototyp 3

4.3.3.1. Voraussetzungen

Die Software für Prototyp 3 sowie eine Datenbank mit Testdaten sind auf dem Smartphone installiert.

4.3.3.2. Vorbereitungen

Android Package für Prototyp 3 normal auf dem Smartphone installieren. Die Software erstellt eine Datenbank mit Testdaten beim ersten Aufruf.

4.3.3.3. Systemtests

Nr.	Funktionalität	Implementiert	Status	Bemerkung
1	Anforderungen an Software von Partnerprojekten (Point-Zero und IndoorWPS) definieren.	Ja	OK	Die Anforderungen wurden definiert und mit den jeweiligen verantwortlichen Personen besprochen.
2	Daten aus der lokalen SQLite Datenbank lesen.	Ja	OK	
3	Daten in die lokale SQLite Datenbank schreiben.	Ja	OK	
4	Persistenzlayer mittels JUnit Tests vollständig durchtesten.	Ja	OK	Testumgebung musste mittels Strategy Pattern (GoF) aufgebaut werden.
5	POIs in einer „POI View" anzeigen (augmented reality)	Ja	OK	Anzeige und Positionierung der POIs im richtigen Bildschirmsegment funktioniert. Details werden noch nicht angezeigt.

4.3.3.4. Zusammenfassung

Die gewünschte Funktionalität für Prototyp 3 wurde erreicht.

4.3.3.4.1. Bekannte Einschränkungen

Anforderungen an Partnerprojekte

Die Anforderungen wurden definiert. Die Umsetzung und Implementation ist für Prototyp 4 geplant.

Testen vom Persistenzlayer mittels JUnit

Um den Persistenzlayer mit Hilfe von JUnit testen zu können, musste eine weitere Abstraktion der Datenbankschicht und Mock-Objekte erstellt werden. Der Grund für dieses Vorgehen ist, dass die Entwicklungslibrary für die Android Plattform die gesamte API lediglich als Stub-Implementation enthält und jeder Aufruf einer Methode der API in einer RuntimeException endete.

Um die Funktionalität des Persistenzlayers so realitätsnah wie möglich testen zu können wurde in den Mock-Objekten eine lokale SQLite Datenbank mittels JDBC angesprochen. Dieses dieses Vorgehen birgt ein gewisses Mass an Restunsicherheit jedoch kann davon ausgegangen werden, dass die Persistenzschicht funktioniert, sofern die JUnit Tests nicht fehlschlagen.

Anzeigen der POIs in einer POI View

Zur Zeit werden die einzelnen POIs im korrekten Abschnitt des Bildschirms abhängig von der Blickrichtung dargestellt. Die Ausgangsposition (Position des Geräts in GPS Koordinaten) ist noch fix auf eine Position an der HSR einprogrammiert. Dies ändert, sobald die PointZero Library eingebunden ist und funktioniert.

Weiterhin werden noch keine erweiterte Informationen zu POIs angezeigt, lediglich der POI selber.

4.3.4. Prototyp 4

4.3.4.1. Voraussetzungen

Die Software für Prototyp 4 ist auf dem Smartphone installiert.

4.3.4.2. Vorbereitungen

Android Package für Prototyp 4 auf dem Smartphone installieren. Die Software erstellt eine leere Datenbank beim ersten Aufruf.

4.3.4.3. Systemtests

Nr.	Funktionalität	Implementiert	Status	Bemerkung
1	Gazetteer Interface implementieren	Ja	OK	
2	PointZero Mobile integrieren	Ja	OK	PointZero ist integriert, liefert aber nur sehr ungenaue Positionsdaten.
3	Geonames Suche für „Areas in der nähe“	Ja	OK	
4	Aktualisieren von Area-Daten	Ja	OK	

5	Areas of interest in my vicinity	Ja	OK	
6	Refine GUI	Ja	NOK	GUI wird noch verfeinert
7	Neue Area erstellen und dazugehörige Daten herunterladen	Ja	OK	
8	Routing integrieren	Ja	NOK	Kann nicht live getestet werden, da PointZero nicht 100% funktioniert
9	POI-Daten im OSM erfassen	Ja	OK	POI-Daten wurden für Gebäude 1 OG, Gebäude 3 EG und Gebäude 6 OG erfasst

4.3.4.4. Zusammenfassung

Die gewünschte Funktionalität für Prototyp 4 wurde noch nicht erreicht.

4.3.4.4.1. Bekannte Einschränkungen

PointZero Mobile integrieren

Das Resultat der Integration (ungenaue Positionsbestimmung) wurde an die zuständige Stelle weitergeleitet und Korrekturen für PointZero werden so schnell als möglich implementiert.

Refine GUI

Die Verfeinerung des GUIs wird laufend gemacht. Da im Prototyp 4 umfangreiche Änderungen am GUI getätigt wurden, geschieht diese Veränderung kontinuierlich bis zum Ende der Arbeit.

Routing integrieren

Routing wurde mit verschiedenen Richtungspfeilen implementiert, kann aber noch nicht getestet werden.

Um das Routing testen zu können muss man Mock Interfaces implementieren, die dauern GPS-Koordinaten liefern.

4.3.5. Prototyp 5

4.3.5.1. Voraussetzungen

Die Software für Prototyp 5 ist auf dem Smartphone installiert.

4.3.5.2. Vorbereitungen

Android Package für Prototyp 5 auf dem Smartphone installieren. Die Software erstellt eine leere Datenbank beim ersten Aufruf. Wichtig: zuvor muss eine allfällig vorhandene ältere Version der Software komplett entfernt werden.

4.3.5.3. Systemtests

Nr.	Funktionalität	Implementiert	Status	Bemerkung
1	GUI verfeinern	Ja	OK	
2	Fehler beheben und stabilisieren	Ja	OK	

4.3.5.4. Zusammenfassung

4.3.5.4.1. Bekannte Einschränkungen

Positionierung innerhalb des Gebäudes mittels PointZero

Die Position kann nicht exakt bestimmt werden. Die Genauigkeit beträgt im Moment ca. 20%, wenn nicht weniger. Mit den gelieferten Positionen ist das Navigieren nicht möglich und die „POIs in Blickrichtung" stimmen durch den falschen Referenzpunkt nicht mit den POIs überein, die tatsächlich in Blickrichtung liegen.

Es wird vermutlich eine Weiterentwicklung der PointZero-Bibliothek geben, um brauchbare Resultate zu erziehen.

Routing über Stockwerke

Aufgrund von Zeitmangel konnte dieser Punkt nicht mehr behandelt werden und wird als mögliche Weiterentwicklung aufgenommen.

5. Resultate und Weiterentwicklung

5.1. Resultate

Informationen über die erreichten Resultate sind auf Seite 14 im Kapitel „4. Resultate" zu finden.

5.2. Mögliche Weiterentwicklungen

5.2.1. Navigation über Stockwerke hinweg

Im Moment ist die Navigation über mehrere Stockwerke hinweg noch nicht implementiert. Die bereits erfasste POI Daten auf dem OSM Server beinhalten jedoch die nötigen Informationen zu den Stockwerken. Die Nodes und Ways wissen somit zu welchem Stockwerk sie gehören.

Da Fingerprints für jedes Stockwerk getrennt erfasst werden, sind die Stockwerkinformationen in diesem Fall auch erfasst und eindeutig.

Ein möglicher Ansatz zur Implementation dieser Funktionalität wäre die Einführung von so „floor change points" - spezielle Wegpunkte an welchen signalisiert wird, dass ein Stockwerkwechsel erfolgen kann.

Ein anderer Ansatz wäre es, die Ways von zwei unterschiedlichen Stockwerken etwas „versetzt" zueinander zu erfassen und dann diese versetzten Wegnetze an geeigneten Stellen zu verbinden. Unserer Meinung nach dürfte dies die am wenigsten Aufwändige Lösung darstellen und sollte ohne Änderungen an der bisherigen Implementation funktionieren.

5.2.2. Navigation zwischen Gebäuden

Eng mit „5.2.1. Navigation über Stockwerke hinweg" verknüpft ist die Erweiterung der Navigation um die Fähigkeit, zwischen verschiedenen Gebäuden zu navigieren, beispielsweise von einem Punkt im Gebäude A zu einem Punkt in Gebäude B.

Auch hier sollte eine intelligente Erfassung und Verknüpfung der OpenStreetMap Wegdaten das Problem mit der bestehenden Implementation lösen.

5.2.3. GUI Erweiterungen

5.2.3.1. Radar anstatt Kompass

Da der Kompass nicht immer sehr verlässlich ist und seine Werte durch elektromagnetische Felder stark irritiert werden können, wäre es besser eine Art „Radar" anstelle des Kompasses darzustellen.

Die Idee des Radars entstand durch die Wikitude Applikation, welche einen solchen Radar für die Points of Interest einsetzt.

5.2.3.2. Richtung in der POI Liste

Damit man weiss in welcher Richtung die aufgelisteten POIs liegen, könnte man noch einen Richtungspfeil pro POI in der Liste anzeigen. Da die Liste regelmässig aktualisiert wird, weiss man immer welche POIs vor dem Benutzer liegen und in der welcher Richtung sie sind.

5.2.4. Eigener Geoname Service

In der aktuellen Version des IndoorGuides werden die gazetteer Queries an den WebService von geonames.org übermittelt. Die Idee wäre es, einen eigenen Service zu erstellen, der nur Areas of Interest zurück liefert, welche auch sicher Daten (Nodes, Ways und Fingerprints) enthalten. Diese Service hat folgende Vorteile:

- Man kann die Menge der vorhandenen Daten Server-Seitig abfragen und in der Auflistung anzeigen.
- Es werden nur wirkliche Areas of Interest angezeigt und nicht alles was um den Benutzer rundum ist.
- Die Masken „Nearby Areas" und „Search and add Area" können zu einer zusammengefasst werden.

5.3. Codestatistik

5.3.1. Codestatistik Summarisch

Kriterium	Packages	Klassen	Methoden	SLoC	JavaDoc
Projekt	21.00	107.00	567.00	3'959.00	246.00
Package		5.10	27.00	188.52	11.71
Klasse			5.30	37.00	2.30
Methode				6.98	0.43

5.3.2. Projektstatistik pro Package

Das Root-Package für die Packagestatistik ist „ch.hsr.indoorguide4android".

Package	Klassen	Methoden	SLoC	JavaDoc
.business.area	4	23	127	12
.business.net	3	9	160	11
.business.osmdata	2	20	58	22
.business.osmdata.parser	1	6	98	3
.business.poi	4	14	105	12
.business.position	6	25	200	19
.business.position.parser	2	24	172	19
.business.routing	2	8	34	5
.business.util	1	6	30	7
.gui	1	3	20	3
.gui.activity	38	91	1137	15
.gui.listAdapter	3	6	87	6
.gui.listItem	6	24	236	19
.gui.util	8	14	120	8
.gui.view	11	35	361	12
.persistence	4	50	299	43
.persistence.adapter.android	2	17	67	4
.persistence.adapter.mock	3	158	495	5
.persistence.dao	2	22	55	13
.persistence.db	3	10	89	5
.util	1	2	9	3
Total	**107**	**567**	**3'959**	**246**

TEIL IV: ANHANG

1. Glossar

Begriff	Erklärung
Area of Interest	Geographisch begrenztes Gebiet, für welches der User die benötigten Daten wie POIs und Routen herunterladen kann. Ein Beispiel für eine Area of Interest ist der Campus der HSR Rapperswil.
Fingerprint	Ein Fingerprint ist das Ergebnis einer Signalmessung an einem bestimmten Referenzpunkt. Ein Fingerprint besteht aus einer Referenzkoordinate und den Messwerten aller dort empfangenen Wireless LAN Signalen. Die Signale wiederum beinhalten die MAC Adresse des Access-Points sowie die Signalstärke. Die damit erreichte Genauigkeit liegt im Bereich einiger Meter, abhängig von der Anzahl empfangener Signale. Die Genauigkeit liesse sich durch intelligentes Positionieren der Access-Points erhöhen.
GPS	GPS ist ein satellitengestütztes, weltweit verfügbares Positionierungssystem (Global Positioning System), welches aus einer Anzahl geostationärer Satelliten besteht. GPS-Empfänger können aufgrund der Satellitendaten ihre eigene Position, Geschwindigkeit und Bewegungsrichtung berechnen.
IndoorWPS	Das Wireless Positioning System (WPS) basiert auf einem Positionierungsverfahren mit Hilfe von Wireless LAN. 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 nur lokal bestimmt und ist „Raumgenau".
POI	Point of Interest (deutsch: „interessante Orte") ist ein Begriff aus dem Bereich der Navigationssysteme und Routenplaner. POIs sind Orte, die für den Nutzer einer Karte oder eines Navigationssystems von Interesse sein können.
SLoC	Source Lines of Code. Anzahl der effektiven Quellcodezeilen ohne Leerzeilen und Kommentare.

2. Literaturverzeichnis

[andADBusage] Anleitung zur Verwendung der Android Debug Bridge. http://developer.android.com/guide/developing/tools/adb.html#move

[anddev] Umfangreiche Webseite mit diversen Tutorials für viele Android-spezifische Probleme. http://www.anddev.org

[android developers] Offizielle Android Entwicklungsseite mit SDK Beschreibung, Dev Guide und Android References. http://developer.android.com

[androidforums ru] Russische Android Community. http://androidforums.ru

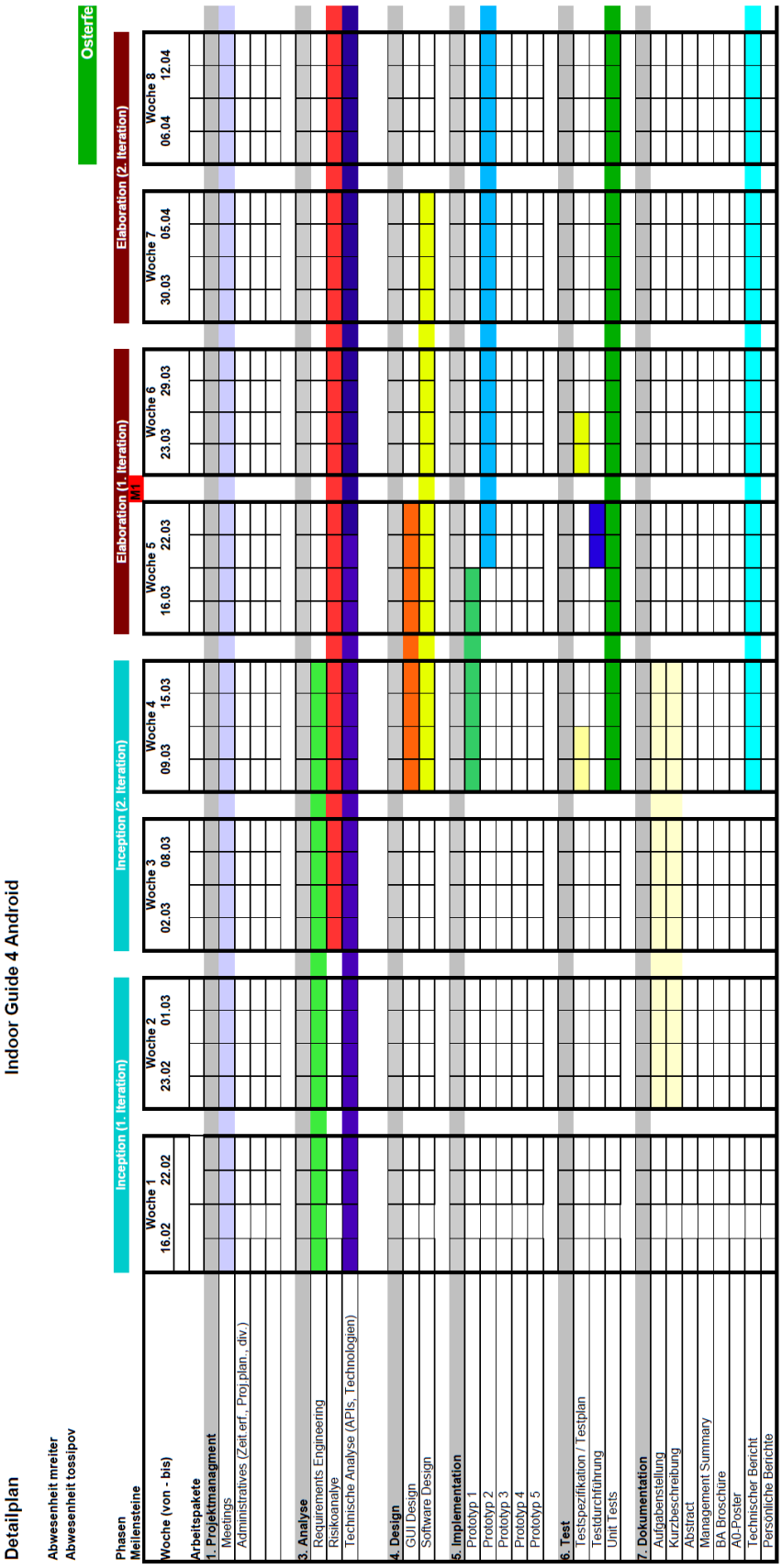
[andUSBDebug] Anleitung zur Aktivierung von „USB Debugging" auf einem Gerät. http://developer.android.com/guide/developing/device.html#setting-up

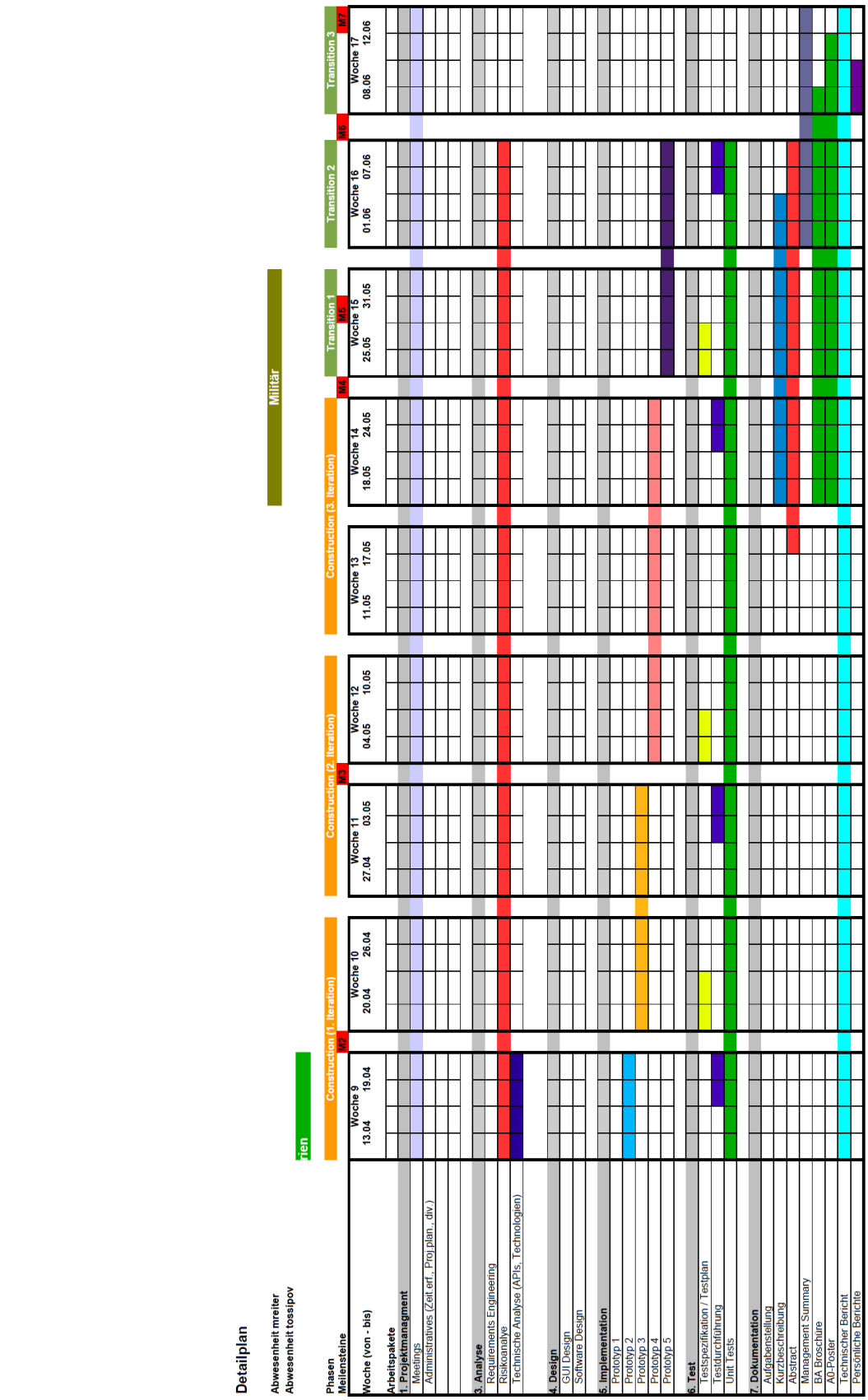
[HSR] Offizielle Webseite der Hochschule für Technik in Rapperswil. http://www.hsr.ch

[OSMNav] OSMNavigation ist eine Library, welche die Entwicklung von Navigations- und Routenplanungs-Applikationen auf Basis von OpenStreetMap Daten vereinfacht. Sie enthält alle Komponenten, welche dafür nötig sind, mit Ausnahme der graphischen Benutzeroberfläche. http://apps.sourceforge.net/mediawiki/travelingsales/index.php?title=OSMNavigation

[wikitude] Webseite für „Wikitude AR Travel Guide" unter http://www.mobilizy.com

3. PROJEKTPLAN




Detailplan
Abwesenheit mreiter
Abwesenheit tossipov
Militär
Ferien
Phasen
Meilensteine
Woche (von - bis)
Arbeitspakete
1. Projektmanagment
Meetings
Administratives (Zeit.erf., Proj.plan., div.)
3. Analyse
Requirements Engineering
Risikoanalye
Technische Analyse (APIs, Technologien)
4. Design
GUI Design
Software Design
5. Implementation
Prototyp 1
Prototyp 2
Prototyp 3
Prototyp 4
Prototyp 5
6. Test
Testspezifikation / Testplan
Testdurchführung
Unit Tests
7. Dokumentation
Aufgabenstellung
Kurzbeschreibung
Abstract
Management Summary
BA Broschüre
A0-Poster
Technischer Bericht
Persönliche Berichte
Construction (1. Iteration)
Construction (2. Iteration)
Construction (3. Iteration)
Transition 1
Transition 2
Transition 3
M2
M3
M4
M5
M6
M7
Woche 9 13.04 19.04
Woche 10 20.04 26.04
Woche 11 27.04 03.05
Woche 12 04.05 10.05
Woche 13 11.05 17.05
Woche 14 18.05 24.05
Woche 15 25.05 31.05
Woche 16 01.06 07.06
Woche 17 08.06 12.06

4. Softwaredokumentation

4.1. Installationsanleitung

Zur Installation der Applikation muss zuerst die APK Datei von der Bezugsquelle (siehe Impressum) heruntergeladen werden.

Sofern nicht schon erfolgt, muss auf dem Android-Gerät „USB Debugging" aktiviert werden. Bitte folgen Sie hierzu der Anleitung unter [andUSBDebug].

Um die APK Datei auf dem Gerät installieren zu können, muss der Befehl *„adb push <APK Datei>"* ausgeführt werden. Eine Anleitung zur Verwendung der Android Debug Bridge (ADB) ist unter [andADBusage] zu finden.

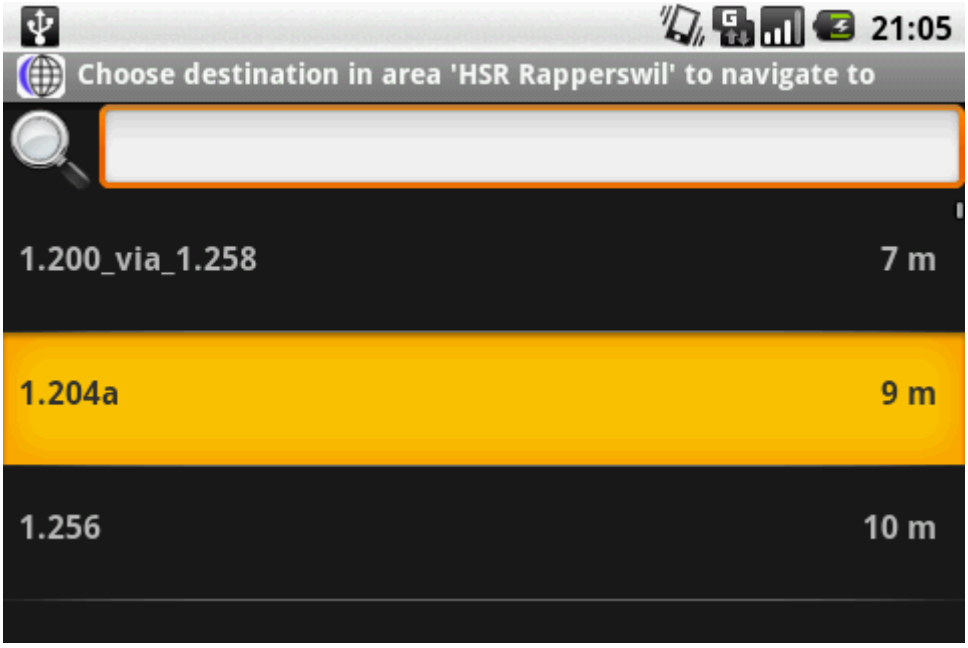
4.2. Kurzanleitung

Die Applikation startet mit folgendem Screen:

Hier werden die Daten der zuletzt aktiven Area of Interest geladen. Sofern keine Daten verfügbar sind oder die Anwendung das erste Mal gestartet wird, wird auf dem nachfolgenden Screen eine entsprechende Hinweismeldung ausgegeben.

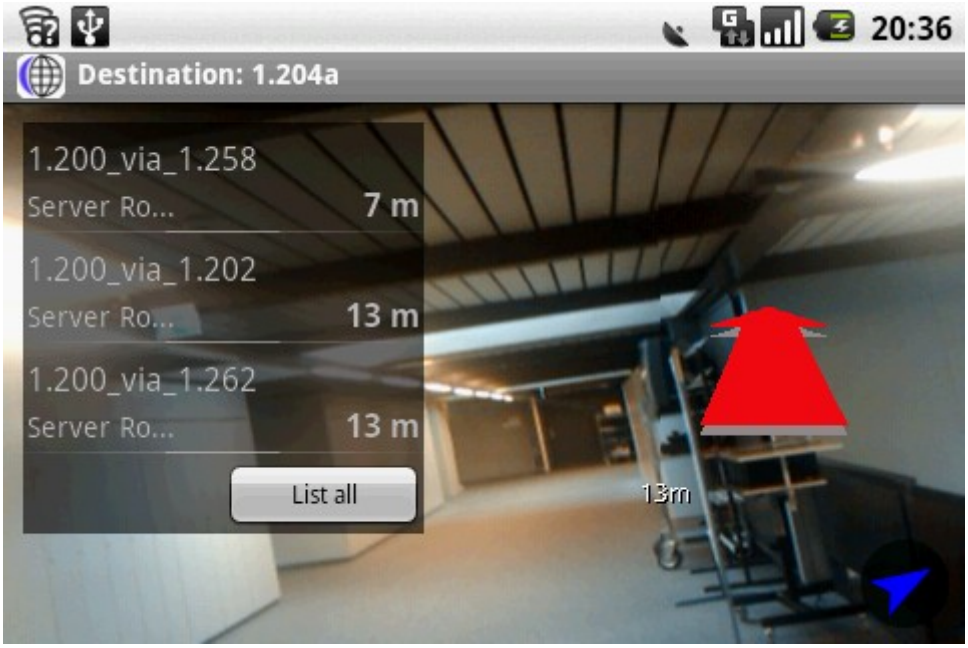
Dieser Screen zeigt die geladenen Daten an. Links werden die näheren POIs in dieser Blickrichtung aufgelistet.

Mit Hilfe vom Kontext Menü bekommt man ein Übersicht über weitere Funktionen des Guide's.



Wählt man „Navigate to...“ im Menü, so werden die vorhandene POIs dieser Area nach Distanz sortiert aufgelistet. Mit dem Klick auf einen dieser Einträge wird ein Dialog mit den Detailinformationen angezeigt:

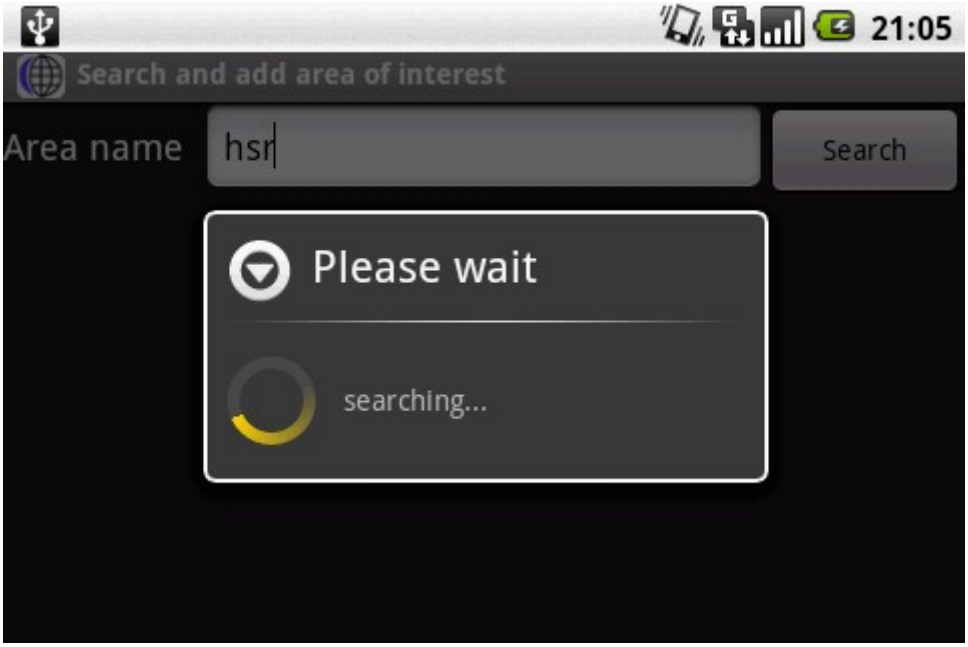
In diesem Dialog erscheinen detaillierte Informationen zum ausgewählten POI. Aus dieser Sicht kann in den Navigationsmodus gewechselt werden.

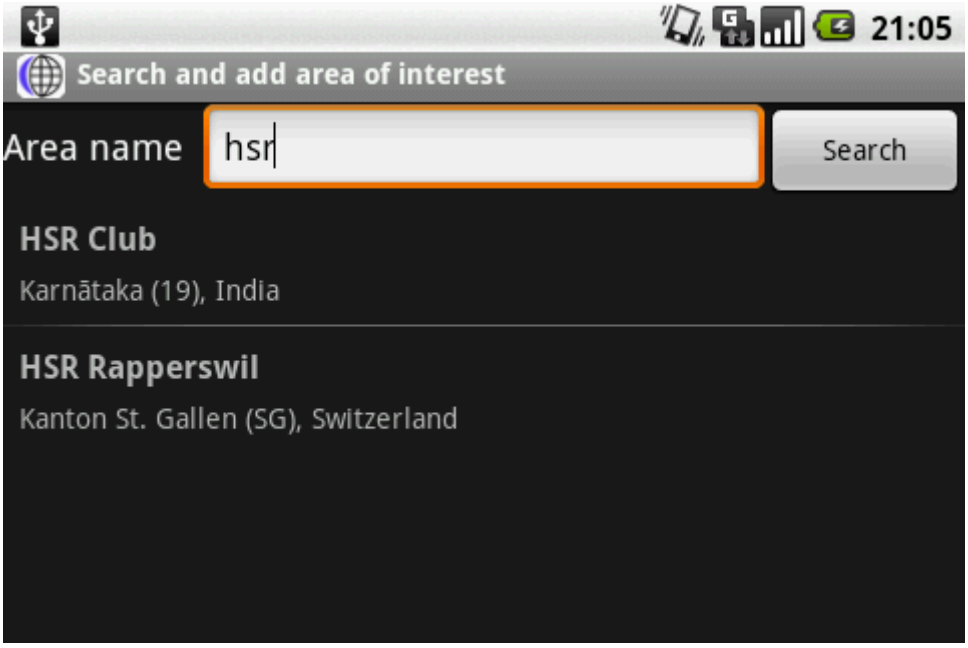


Die Navigationsansicht führt den Benutzer zum ausgewählten Ort hin.

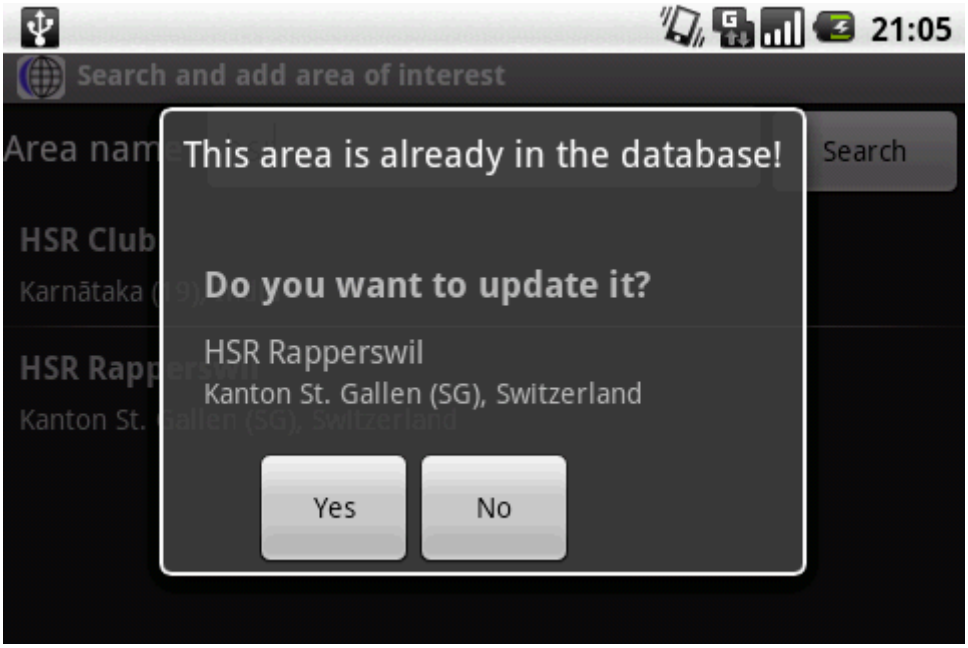
Um eine neue Area of Interest hinzuzufügen gibt es zwei Möglichkeiten: nach einer bestimmter Area suchen oder Areas in der Umgebung auflisten lassen.

Folgender Screen zeigt wie man nach einer Area sucht:



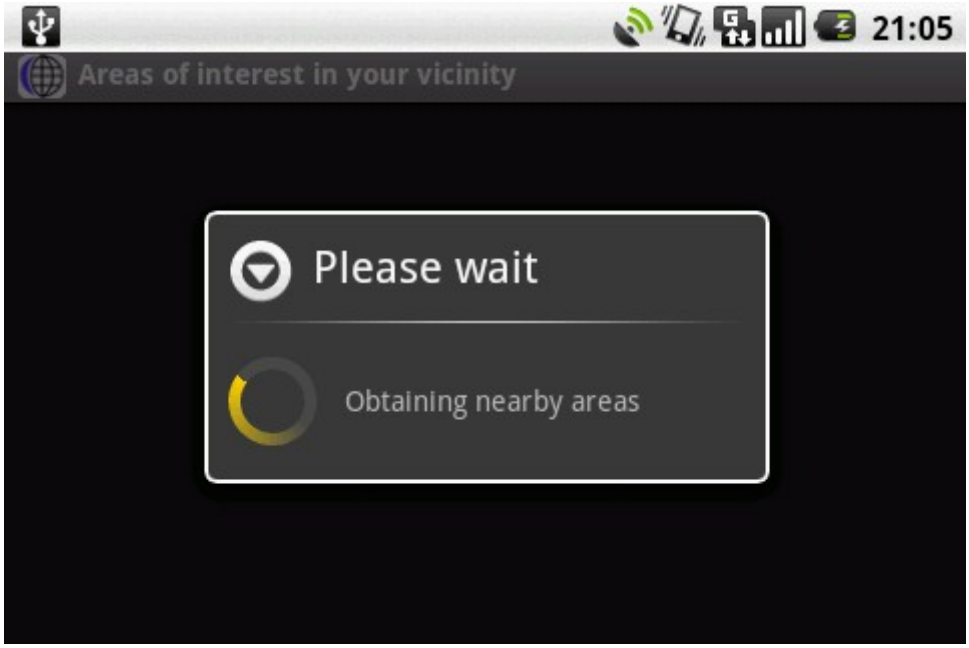


Es werden zwei Areas für den Begriff „hsr" gefunden. Mit den angezeigten Detailinformationen weiss man genau, in welchem Land die gefundene Area ist. Mit dem Klick auf den gewünschten Listeneintrag erscheint folgender Dialog:

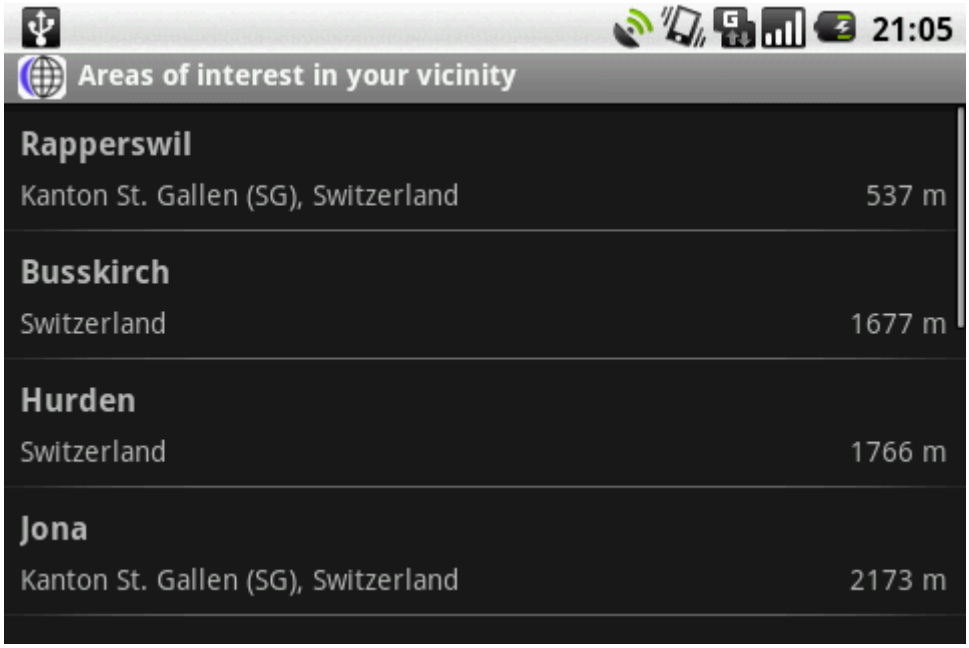


Da diese Area in der Datenbank schon vorhanden ist, wird angeboten die Daten für sie zu Aktualisieren. Sollte die Area noch nicht in der Datenbank gespeichert sein, wird sie neu angelegt und die dazugehörigen Daten automatisch heruntergeladen.

Folgender Screen zeigt an, wie man die in der näheren Umgebung liegende Areas findet:

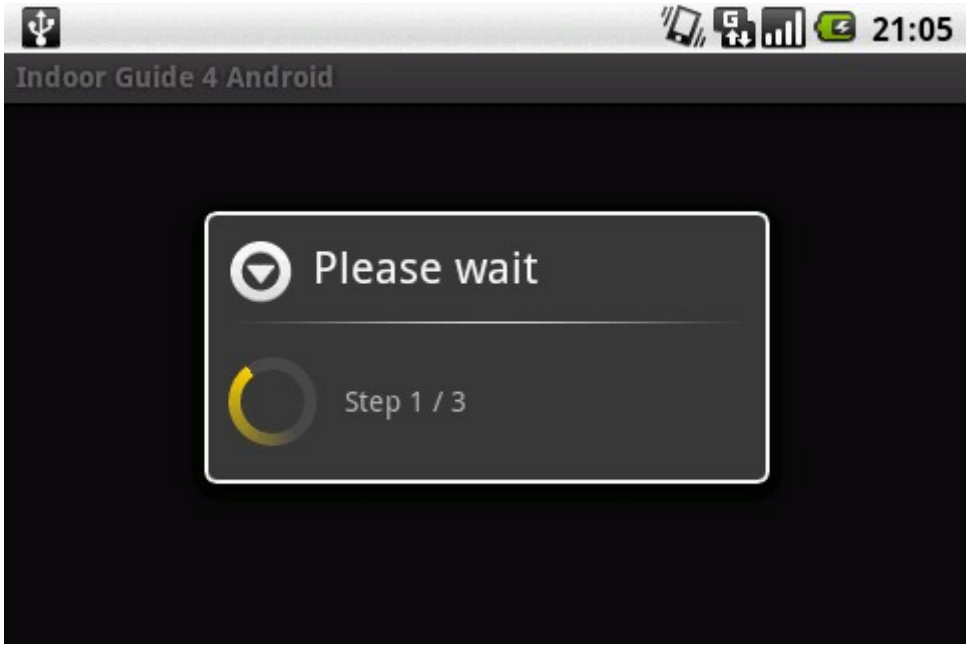


In folgendem Screen werden die gefundenen Areas um den HSR Campus herum aufgeführt:



Die Areas werden nach Entfernung sortiert und die Listeneinträge ebenfalls mit Detailinformationen versetzt.

Mit dem Klick auf einen dieser Einträge wird dem Benutzer ein Download-Dialog angezeigt.



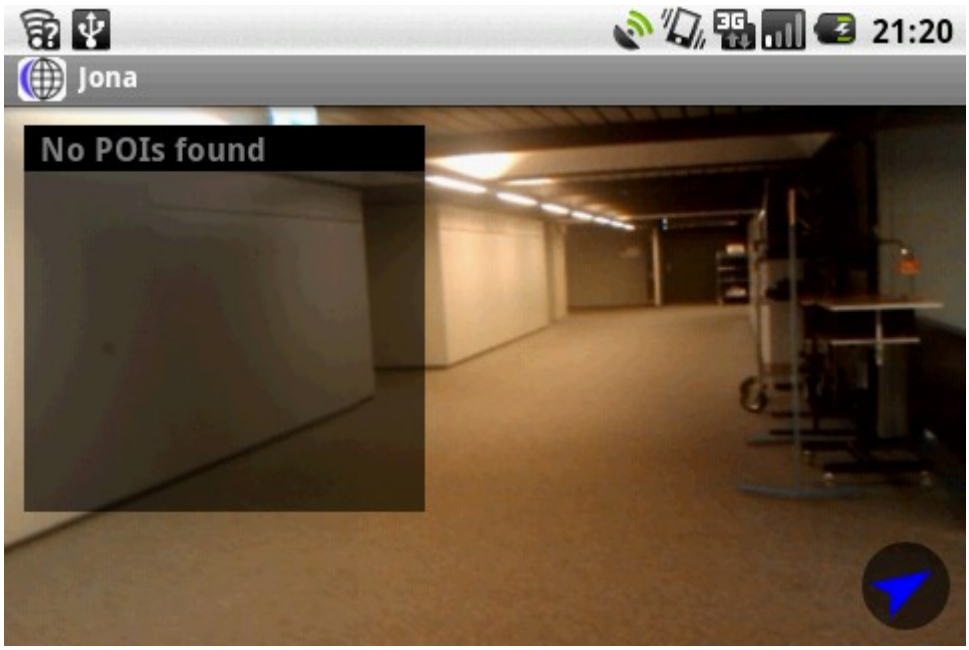
Die Daten werden Step by Step heruntergeladen und gespeichert. Dem Benutzer wird jeweils der Schritt angezeigt, in welchem sich der Download-Prozess befindet. Je nach der verfügbaren Internetverbindung und der Datenmenge kann dieser Prozess einige Minuten dauern.

Nach dem die Daten erfolgreich heruntergeladen worden, wird der „Saved Areas" Screen angezeigt:

Die drei Buttons neben dem Namen lösen nach einem Klick jeweils eine andere Aktion aus

- „GO!": Die entsprechende Area wird aktiviert.
- „Refresh" (zwei grüne Pfeile): Die Daten der jeweiligen Area werden aktualisiert.
- „Delete" (rotes Kreuz): Die Area und alle zu ihr gehörenden Daten werden aus der Datenbank gelöscht.

Nach einem Klick „GO!" werden die Daten der gewählten Area geladen und man bekommt den „POI View" Screen angezeigt:



Diese Area verfügt über keine Daten und darum wird die POI List leer dargestellt.