HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil
Herbstsemester 2011

Autor:	Lüchinger Dominik
Betreuer:	Prof. Stefan F. Keller, HSR, Institut für Software
Projektpartner:	OCAD Zug

Impressum

Lüchinger Dominik
Informatikstudent

Prof. Stefan F. Keller
Institut für Software

HSR Hochschule für Technik Rapperswil
Institut für Software
Oberseestrasse 10
8640 Rapperswil
Institutsleiter Peter Sommerlad
peter.sommerlad@hsr.ch
+41 55 222 46 30 Telefon
+41 55 222 46 29 Fax

Developer Wiki & Repository: http://dev.ifs.hsr.ch/redmine/projects/neomap

Erklärung der Eigenständigkeit

Ich erkläre hiermit,

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

Ort, Datum: Rapperswil, 23.11.2011

Name, Unterschrift: Lüchinger Dominik

Änderungsgeschichte

Datum	Version	Änderung
05.10.2011	0.0	Erstellung Dokument
10.10.2011	0.1	Ausarbeitung Projektplan, Anforderungspezifikation
13.10.2011	0.2	Use Cases
17.10.2011	0.3	Überarbeitung Design und Doku Erweiterung
24.10.2011	0.4	Tools
28.10.2011	0.5	Heutige Technik & Szenarios
03.11.2011	0.6	Additional Features hinzugefügt
18.12.2011	0.7	Abstract und Teil I
20.12.2011	0.8	Teil II
22.12.2011	0.9	Anhang & Überarbeitungen
23.12.2011	1.0	Abschluss

Freigabeliste

Datum	Version	Änderung
19.10.2011	0.2	Use Cases (Diagramm)
02.11.2011	0.5	Heutige Technik & Szenarios
09.11.2011	0.6	Additional Features & Wishlist
23.12.2011	1.0	Schlussabgabe

Abstract

Kartenapplikationen auf Smartphones (sog. Apps) gibt es schon seit einiger Zeit. Dabei ist Google Maps Wegbereiter und Marktführer. Die meisten dieser Apps sind von einer ständigen Datenverbindung abhängig. In verschiedenen Situationen möchte man aber unabhängig von der Internetverbindung sein (sog. Offline Apps), denn was macht man im Ausland, auf Geschäftsreise oder im Urlaub? Internet-Roaming ist immer noch teuer. Zudem gibt es bei Karten und Pläne solche, die nicht frei ins Internet gelangen sollen, wie z.B. private Parkanlagen (Privacy) oder OL-Karten (Copyright). Wegen des beschränkten Speicherplatzes auf mobilen Geräten bleibt nichts anderes übrig, als sich auf ausgewählte Kartenmaterialien zu beschränken. Etwas was bei Navigations-geräten selbstverständlich ist. NeoMap soll nun diese Anforderungen erfüllen.

Ziel ist die Entwicklung und Publikation einer App für Android Smartphones, mit dem man Karten und Pläne uneingeschränkt offline anzeigen kann unter Berücksichtigung der Privacy und des Copyrights.

Diese Probleme sollte man so lösen können, dass man eigene Offline-Karten erstellt, welche nur einen Bruchteil des Speichers benötigen.

Mit der Verwendung von OpenGL werden die Offline-Karten dargestellt. Dabei wird mit GPS Lokalisierung gearbeitet um die Position auf der Karte zu bestimmen.

Durch eine Kombination des Open Source Projektes OpenStreetMap und der Offline- Variante, ist es möglich individuell mit Karten zu navigieren. Die Offline-Karten können von einem Server bezogen oder hochgeladen werden.

Weitere Infos auf: www.gis.hsr.ch/wiki/NeoMap

Management Summary

Ausgangslage

Grundidee

Die Idee dieser Arbeit besteht darin, eine offline Applikation für das Android Mobile-Betriebssystem zu entwickeln. Dem Benutzer soll es möglich sein, eigene georeferenziere Karten, egal ob Satellit, Hybrid oder sonst eine Darstellung, in einer App zu benutzen. Dabei kann er seine eigene Karte der Öffentlichkeit zur Verfügung stellen und auch andere sogenannte NeoMaps von anderen Personen beziehen.

Ziele

Ziel ist die Entwicklung und Publikation einer App für Android Smartphones, die Karten und Pläne Offline anzeigen kann unter Berücksichtigung der Privacy und des Copyrights.
Näheres ist unter dem Kapitel 1.2 Aufgabenstellung zu finden.

Vorgehen

Für den Softwareentwicklingsprozess wurde RUP gewählt. Durch Einteilen in Iterationen konnte frühzeitig auf allfällige Probleme reagiert werden. Ausserdem wurde jede Woche ein Meeting durchgeführt.

Risiken

Das Risiko bestand vor allem darin, dass es sich bei der Android Programmierung um Neuland handelt. Somit ist es viel schwerer einzuschätzen wie viel Aufwand das Projekt mit sich bringen wird. Dazu kommt noch ein völlig neues Themengebiet mit Geodaten.

Involvierte Personen

Name	Rolle
Prof. Keller Stefan	Auftraggeber und Betreuer
Binna Tobias	Mitarbeiter des IFS, Entwickler der Serverumgebung
Lüchinger Dominik	Student, Entwickler der NeoMap App
Recher Patrick	Student, Mitentwickler der NeoMap App

Ergebnisse

Resultat

Das Ergebnis dieser Projektarbeit ist eine stabile Version der NeoMap App. Es ist momentan erst der Erste Schritt, jedoch sind die Grundfunktionalitäten vorhanden. Es ist möglich mit der OSM Karte zu navigieren und damit in Verbindung einige zusätzliche Funktionen auszuführen. Zudem können Offlien-Karten angezeigt werden.

Bewertung

Durch die Kombination von On-und Offline-Karten ist es gelungen eine originelle App für Jedermann zu entwickeln.

Zielerreichung

Die Grundanforderungen wurden erfüllt. Doch es gibt noch sehr viel Potenzial. Leider ist aus Mangel an Zeit nicht gelungen noch weitere Interessante Features zu implementieren.

Abweichungen

Einige der Features konnten noch nicht implementiert werden. Werden jedoch noch hinzugefügt. Die Testabdeckung fiel etwas gering aus. Dies konnte leider aus zeitlichen Gründen nicht mehr realisiert werden. Jedoch werden auf sie später noch fertig implementiert.

Kosten

Die App basiert nur auf Open Source Libraries. Darum entstanden während der Entwicklung keine Kosten in Form von Zahlungsmittel. Auch für die Benutzung der App muss mit keinen zusätzlichen Kosten gerechnet werden. Einzig und allein ein kompatibles Android Gerät wird benötigt.

Ausblick

Erfahrung

Mit der Durchführung dieses Projektes gewann man auch an Erfahrung:

- Neue Technologien im Programmieren (Android, JSON, Serveranbindung, SQLite)
- Einblick in die Welt der Geoinformatik
- Neue Mobiltelefonumgebung (Androidgeräte)
- Einblick in die Funktionsweise des Android Betriebssystems
- Services wie OpenStreetMap, Google Earth
- Einblick in OpenGL
- Vertiefung des Software Engineerings

- Anwendung von Git

Verbleibende Probleme

Siehe Kapitel „6.1 Codeinformationen, Bugs & Incomplete Features".

Was sollte nächstes Mal anders gemacht werden?

Es gibt eine Menge, welches ich nächstes Mal anders machen würde. Vor allem, da man nun Erfahrung in der Android Programmierung gesammelt hat, kann man den Aufwand besser einschätzen. Sicherlich versuchen die Anforderungen und den Zeitlichen Aufwand besser abschätzen. Auch das Projekt am Anfang etwas vorantreibt, damit man schnell mit der Implementierung beginnen kann und früh genug fertig ist, um noch Allfälliges zu verbessern.

Inhaltsverzeichnis

Impressum 2
Änderungsgeschichte 3
Freigabeliste 3
Abstract 4
Management Summary 5
Ausgangslage 5
Ergebnisse 6
Ausblick 6
1. Einführung 13
1.1 Problemstellung 13
1.2 Aufgabenstellung[1] 13
1.3 Rahmenbedingungen 16
1.3.1 Administratives 16
1.3.2 Bereitgestellte Gerätschaften 17
1.3.3 Entwicklungsumgebung & Tools 17
1.4 Vorgehen, Aufbau der Arbeit 19
2. Vision und Ziele 19
3. Stand der Technik 21
3.1 Bestehende Lösungsansätze 21
3.1.1 Maps (-) (v. 2.6.1) 21
3.1.2 Google Maps (v. 6.0.3) 22
3.1.3 MapDroyd (v. 1.1.0) 23
3.2 Realisierbarkeit und Herausforderungen 23
3.2.1 GPS Lokalisierung 23
3.2.2 Alternativen zur GPS Lokalisierung[3] 24
3.2.3 Externe Datenquelle 25
3.2.4 Karten Layer 25
4. Evaluation 26
4.1 Datenformat Webübertragung 26
4.1.1 Kriterienkatalog und Gewichtung 26
4.1.2 XML 26
4.1.3 JSON 27

4.1.4 Fazit 27
4.2 Datensicherung 27
4.2.1 Kriterienkatalog und Gewichtung 27
4.2.2 SQLite 28
4.2.3 Properties 28
4.2.4 Fazit 28
5. Umsetzungskonzept 29
5.1 Datenquellen/Services 29
5.1.1 NeoMap Server 29
5.1.2 OpenStreetMap (OSM) 29
5.2 Lokalisierung 29
5.3 Datenhaltung 29
6. Resultate und Ausblick 30
6.1 Codeinformationen, Bugs & Incomplete Features 30
6.2 Zielerreichung 31
6.3 Ausblick: Weiterentwicklung 32
6.4 Persönlicher Bericht 32
6.5 Dank 33
1. Vision 35
2. Anfoderungsspezifikation 35
2.1 Funktionale Anforderungen 35
2.1.1 Übersicht 35
2.1.2 Detailiert 35
2.2 Use Cases 38
2.2.1 Use Case Diagram 38
2.2.2 Use Case Beschreibung 39
2.3 Nicht funktionale Anfoderungen 40
2.4 Detailspezifikation 41
2.4.1 Zielgruppe 41
2.4.2 Erweiterungsmöglichkeiten (Whishlist) 42
2.4.3 Lizenzanforderungen 43
2.4.4 Verwendete Standards 43
3. Analyse 44

3.1 JSON Objekte 44
3.2 Datenbank Modell 46
3.3 Android Lifecycle-Modell[7] 46
4. Design 48
4.1 Domain Modell 48
4.2 Architektur 49
4.2.1 Klassenstruktur 49
4.2.2 Dependency Graph 51
4.3 Sequenz Diagramme 53
5. Implementation & Tests 53
5.1 Implementationsdetails 53
6. Resultate 53
6.1 Resultate 53
6.2 Codestatistik 53
7. Projektmanagement 53
7.1 Projektorganisation 53
7.2 Besprechungen 54
7.3 Projektvorgehen 54
7.3.1 RUP 54
7.3.2 Iterationsplanung 54
7.3.3 Meilensteine 55
7.4 Auswertung der Arbeitszeiten 56
7.4.1 Arbeitsumfang 56
7.4.2 Übersicht Kategorien und Arbeitspakete 58
7.4.3 Übersicht Kategorie 58
7.4.4 Aufwandsverlauf 59
7.4.5 Risikomanagement 59
8. Qualitätsmassnahmen 60
8.1 Sitzungsprotokolle 60
8.2 Iteratives Vorgehen 60
8.3 Einsatz eines Versionierungssystem 60
8.4 Testing 60
8.5 Coderichtlinien 60

9. Anleitungen und Tutorials 60
9.1 Einführung in die Applikation 60
9.2 Map View 62
9.3 Georeferenzierte Karten 63
1. ANHANG A: Inhalt der CD 65
1.1 Applikation 65
1.2 Dokumentation 65
2. ANHANG A: Glossar und Abkürzungsverzeichnis 65
3. ANHANG B: Literatur- und Quellenverzeichnis 67
3.1 Bücher und Artikel 67
3.2 Links und Informationen 67
3.3 Weitere Quellen 67

Abbildungsverzeichnis

Bild 1: Nexus S 17
Bild 2: Maps (-) 21
Bild 3: Google Maps 22
Bild 4: MapDroyd 23
Bild 5: Use Case Diagram 38
Bild 6: Datenbank Modell 46
Bild 7: Activity Lifecycle 47
Bild 8: Domain Modell 48
Bild 9: Paketstruktur 49
Bild 10: activiy Packet 49
Bild 11: view Packet 49
Bild 12: application Packet 50
Bild 13: map Packet 50
Bild 14: layers Packet 51
Bild 15: osm Packet 51
Bild 16: neomap Packet 51
Bild 17: network Packet 51
Bild 18: domain Packet 51
Bild 19: Class Dependency 51
Bild 20: onCreate() Method 52
Bild 21: NeoMap GUI-Elemente 61
Bild 22: Map View 62
Bild 23: Georeferenzierte Karte 63

Teil I
Technischer Bericht

1. Einführung

1.1 Problemstellung

Heutzutage finden sich über fünf Milliarden benutzte Mobiltelefone auf der Welt. Die ganze Welt hat sich vernetzt. Betrachtet man diesen Mark, so eröffnet sich einem geradezu die ganze Welt. Nachdem 2008 die ersten Android Applikationen sind es heute bereits schon über 600'000.

Versucht man ein möglichst breites Kundenfeld anzusprechen, so ist es aus Erfahrung das Beste, etwas unentgeltlich auf den Markt zu bringen.

Bei den Karten-Apps ist es meistens so, dass diese eine Internetverbindung benötigen um die entsprechenden Kartenteile zu laden. Dies kann bei schlechtem Empfang oder im Ausland schnell zu roten Köpfen und einer hohen Roaminggebühr führen.

Herkömmliche Navigationssoftware benötigt zwar kein Internet, benötigt jedoch wiederum Unmengen an Speicher und Ressourcen. Dazu benötigt man meistens nur ein Bruchteil der vorhandenen Daten.

Also muss eine App her, welche sowohl online als auch offline agieren kann. Dazu eine einfache Bedienung bietet und mit individuellen Kartengrössen umgehen kann.

1.2 Aufgabenstellung[1]

Ausgangslage

Kartenapplikationen auf Smartphones (sog. Apps) gibt es schon seit einiger Zeit und es werden immer mehr. Dabei ist Google Maps und „Google Maps for Mobile" Wegbereiter und immer noch Marktführer. Die meisten dieser Apps sind von einer ständigen Datenverbindung abhängig. In verschiedenen Situationen möchte man aber unabhängig von der Internetverbindung sein (sog. Offline Apps). Zudem gibt es bei bestimmten Karten und Pläne gute Gründe, dass diese nicht frei ins Internet gelangen, wie z.B. private Parkanlagen (Privacy) oder OL-Karten (Copyright). NeoMap soll nun diese beiden Hauptanforderungen erfüllen.

Aufgaben

Ziel ist die Entwicklung und Publikation einer App für Android Smartphones, die Karten und Pläne Offline anzeigen kann unter Berücksichtigung der Privacy und des Copyrights.

Funktionen:

- Als Basiskarte (Base Map) werden Kacheln von OpenStreetMap (OSM) angezeigt. Die SW-Komponente ist soweit sinnvoll so zu vorzubereiten, dass später weitere Basiskarten-Provider, wie namentlich „Google Satellite" integriert werden können.
- Datenpakete (sog. „NeoMaps") können durchsucht und heruntergeladen werden.
- Die NeoMaps können als Overlay angezeigt werden.
- Die Karten/Pläne unterstützen typische Kartenfunktionen, wie Pan, Zoom in/out, Zoom to full extend, Zoom to Neomap(s) und ggf. Recenter zu My Location, Karte nach Norden ausrichten (rotieren), Suche nach Ort, Eingabe Koordinaten.
- Anzeige der Koordinaten, der Orientierung (Kompass)

Nicht-funktionale Anforderungen:

- Intuitiv gestaltetes GUI gemäss Android „Look and Feel".
- App, die zum „Android Market" konform (und dort wenn möglich publiziert) ist.

Vorgaben

- Benutzerschnittstelle ist Englisch und Deutsch. Die Sprache ist Englisch in Code, Installationsanleitung, allfälliger Benutzerdokumentation, Grafiken. Sonst deutsch.
- Es wird auf modernes Software-Engineering geachtet. Die Softwarequalität soll hoch sein (inkl. Tests).
- Allfällige Webservices werden zur Verfügung gestellt. Ansonsten sind sinnvolle Defaults oder Mocks zu verwenden.

Beteiligte

Student(en)

Siehe oben. Da dies eine Einzelarbeit ist, macht es Sinn, die Zusammenarbeit mit Kommilitonen zu suchen. Dabei muss besonders darauf geachtet werden, dass im Code ersichtlich ist, wer der Autor ist.

Industriepartner/externer Projektpartner

Voraussichtlich OCAD Zug und/oder Bitforge Zürich.

Betreuung HSR

Prof. Stefan Keller, IFS-HSR

Projektabwicklung

Termine

Gemäss https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html (intern).

Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 8 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von mind. 240 Stunden.

Lieferdokumente

- Lauffähige Anwendung (App), welche die oben erwähnten Aufgaben erfüllt.
- Projektdokumentation und Software auf CD
- Website (ev. Wiki)
- Video

Inhalt der Dokumentation

- Die fertige Arbeit muss folgende Inhalte haben:
  1. Abstract, Management Summary, Aufgabenstellung, Eigenständigkeits-erklärung
  2. Technischer Bericht
  3. Dokumente des Projektdokumentation
  4. Anhänge (Literaturverzeichnis, CD-Inhalt)
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung für Broschüre, Poster, Video) gemäss www.hsr.ch und gemäss Absprache mit dem Betreuer.

Form der Dokumentation

- Bericht (Struktur gemäss Beschreibung) gebunden (2 Exemplare) und in Ordner (1 Exemplar „kopierfähig" in losen, gelochten Blättern).
- Alle Dokumente und Quellen der erstellten Software auf CD; CD's sauber angeschrieben (3 Ex.).

Bewertungsschema

Es gelten die üblichen Regelungen zum Ablauf und zur Bewertung der Arbeit des Studiengangs Informatik der HSR.

Das in den Regelungen erwähnte Bewertungsschema wird wie folgt konkretisiert:

- Projektorganisation (Gewichtung ca. 1/5)
- Bericht, Gliederung, Sprache (Gewichtung ca. 1/5)
- Inhalt inkl. Code (Gewichtung ca. 2/5)
- Gesamteindruck inkl. Kommunikation mit Industriepartner (Gewichtung ca. 1/5)

Rapperswil, Anfang Oktober 2011, S. Keller

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

1.3 Rahmenbedingungen

1.3.1 Administratives

Datum	Termin
19.09.2011	Einarbeitung Android
03.10.2011	Beginn der Studienarbeit
19.12.2011	Abgabe Abstract und A0 Poster an Betreuer
23.12.2011	Kurzfassung an das Studiengangsekretariat
23.12.2011	Abgabe des Berichtes an den Betreuer bis 17 Uhr

1.3.2 Bereitgestellte Gerätschaften

Arbeitsrechner	
Prozessor	Intel Xeon Quad Core X3450, 2.66 GHz
Arbeitsspeicher	8.00 GB
Betriebssystem	Windows 7 Enterprise 64bit, SP1

Google Nexus S		
Prozessor	1GHz Hummingbird-Prozessor	
Netzwerk	GSM, UMTS (GPRS Klasse 10, EDGE Klasse 10, HSDPA 7.2 Mbps, HSUPA 5.76 Mbps)	
Bildschirm	4,0 Zoll (480 x 800 Pixel), Super-LCD, kapazitiver Touchscreen	
Kamera	5 Megapixel (2560 x 1920 Pixel), Zweitkamera für Videotelefonie	
Betriebssystem	Android 2.3.6	

Bild 1: Nexus S

1.3.3 Entwicklungsumgebung & Tools

Logo	Software/ Version	Einsatzzweck	Beschreibung
	Android ADT 12.0.0	Mobile App Entwicklung	Android Development Tools als Eclipse Plugin. Quelle: developer.android.com
	Android SDK 2.2	Entwicklung	Software Development Kit. Quelle: developer.android.com
	Eclipse Indigo 3.7.0	Entwicklungsum-gebung	IDE für die Entwicklung in Java 7 Quelle: www.eclispe.org

	Redmine 1.2.1	Zeiterfassung & Projektverwaltung	Webbasiertes Projekt-management Tool. Quelle: www.redmine.org
	Git 1.7.4	Versionierung	Versionsverwaltungs Software für das Projekt. Quelle: git-scm.com
	Microsoft Office 2010	Dokumentation	Textverarbeitungspro-gramm und Co. Quelle: office.microsoft.com
	Balsamiq Mockups 2.1.5	Planung	Tool zum Erstellen von Paperprototypen Quelle: Balsamiq.com
	SQLite 3.7.7.1	Datenbank	Von Android unterstützte Programmbibliothek Quelle: www.sqlite.org
	Paint.NET 3.5.8	Bildbearbeitung	Bildbearbeitungssoftware. Quelle: www.getpaint.net
	Structure 101 2.5	Architektur Dokumentation	Programm für die Darstellung von der Software-Architektur. Quelle: www.headwaysoftware.com
	JUnit 4.10	Code-Testing	Framework zum Testen von Java-Programmen. Quelle: www.junit.org
	Metrics 1.3.6	Architektur Dokumentation	Eclipse Plugin zur Code und Klassenanalyse Quelle: metrics.sourceforge.net

1.4 Vorgehen, Aufbau der Arbeit

Da zu Beginn dieses Projektes der Umgang mit Android noch Neuland darstellte musste man sich erst einmal mit einem Emulator zufrieden geben.

Bald wurde einem jedoch klar, dass die Entwicklung mit solch einem Emulator sehr langsam und mühsam ist. Darum wurde von der HSR ein Google Nexus S zur Verfügung gestellt.

Die Einarbeitung in die Android Programmierung erfolgte mit Hilfe des Buchs *Android 3*[2]. Dieses Buch beinhaltet grundlegende und einfache praktische Beispiele zu der Androidprogrammierung. Nachdem klar war, dass Patrick Recher mich bei der Arbeit etwas unterstützen wird und er schon Map-Apps auf Android und dem iPhone programmiert hat, konnte er mir bei den ersten Schritten ein paar Tipps geben. Zum einen haben wir uns entschlossen, dass wir die App ohne XML Layouts machen werden. Dies ist etwas unüblich für Android und ist vor allem für Beginner viel komplexer. Es bietet jedoch auch gewisse Vorteile, So können zum Beispiel Klassen für mehrere Views verwendet werden, ohne viel „Duplicated Code" zu erzeugen. Jedoch kann die GUI auch nicht per Drag and Drop zusammengesetzt werden und zum Teil müssen zusätzliche Handler hinzugefügt werden.

Während des Entwickelns wurden zum Teil neue Features und Verbesserungen in die Applikation aufgenommen. Gewisse Details erkennt man erst während der Entwicklung.

Zum Abschluss der Arbeit wurden auf der Wiki-Seite noch einige Informationen beigefügt. Eine kurze Erklärung zu der App und schliesslich wo man die App überhaupt downloaden kann.

2. Vision und Ziele

Das Ziel dieser Arbeit ist es, eine Android Applikation zu programmieren, welche sowohl offline als auch online agieren kann. Mit offline ist gemeint, dass die App ohne Internet auskommt. Jedoch wird in beiden Fällen GPS für die Lokalisierung benötigt.

Die Offline-Karten oder auch NeoMaps können öffentlich oder privat als .kmz Datei auf das Handy via Speicherkarte geladen oder vom IFS-Server ge-downloadet werden.

Momentan gibt es viele Online-MapApps, jedoch fehlt im Markt eine Offline-Variante. Doch was machen, wenn man keinen online Datenverkehr hat oder gebrauchen will. Vor allem Roaming im Ausland kann ganz schön teuer werden. Also warum für etwas zahlen wenn man's auch mit gratis GPS und selbst erstellten Karten zum Ziel schafft.

Hier einige Einsatzszenarios:

London
Am nächsten Wochenende hat Maria eine Reise nach London geplant. Darum sucht sie kurzerhand den Busplan Plan von London aus dem Internet. Zudem will sie einige Sehenswürdigkeiten wie die London Bridge und den Big Ben besuchen. Auf der im Internet gefundenen Karte markiert sie also noch zusätzlich ihre POIs bevor sie sich eine NeoMap erstellt und auf ihr Handy lädt.

In London angekommen stellt sich jetzt erst mal die Frage wo ihr Hotel ist. Ein Griff zum Handy und schon ist die NeoMap App gestartet. Da sie keine zusätzlichen Roaming Kosten auf sich nehmen will, benutzt sie nur das GPS. Durch die GPS Ortung weiss sie sofort wo sie sich befindet. Sie erkennt ausserdem auf ihrer NeoMap sofort wo die nächste Busstation ist und mit welcher Linie sie bis wohin fahren muss.

Im Bus hat sie leider keine gute GPS Verbindung. Durch ihre NeoMap weiss sie jedoch an welche Haltestelle sie aussteigen muss und kommt so sicher und ohne Umweg an ihr Ziel.

Autosalon Genf
Marc ist ein grosser Autofan. Jedes Jahr findet der berühmte Autosalon in Genf statt, wo die neuesten Modelle und Prototypen vorgestellt werden.

Da Marc jedoch nur begrenzt Zeit hat und er sich nicht auskennt, lädt er sich den Messeplan der Austellung aus dem Internet und markiert darauf seine Teile, welche er sicher sehen will. Er lädt die Karte auf seine NeoMap App.

In Genf angekommen orientiert er sich mit dem Handy auf der NeoMap wo seine Markierten Ziele sich befinden.

Als er alle seine Ziele besucht hat, besitzt er durch die einfache und schnelle Navigation der NeoMap sogar noch Zeit sich ein Bierchen zu genehmigen, bevor es wieder nach Hause geht.

Orientierunglauf
Am nächsten Samstag wird Thomas seinem Hobby nachgehen und einen Orientierungslauf im Bündner Oberland absolvieren.
Am Event Tag wird jedem Teilnehmer ein Androidgerät ohne Simkarte verteilt. Darauf befindet sich die NeoMap App mit einer bereits vorinstallierter NeoMap der OL Umgebung.
Der Start für Thomas ist erfolgt. Er begibt sich zu den ersten Koordinaten auf der Karte. Nachdem er den Posten gefunden hat gibt er die am Posten befindenden neuen Koordinaten ein, welche ihn zum nächsten Posten führen.
Nach 3 Stunden kommt er am Ziel und und ist begeistert von der neuen App. Nachdem er das Gerät dem Veranstalter abgegeben hat, kann dieser die Daten

(Wegpunkte), welches die App während der Benutzung gespeichert hat abfragen und kann somit die Route jedes Teilnehmers im Nachhinein verfolgen.

3. Stand der Technik

3.1 Bestehende Lösungsansätze

Schaut man heute auf den heutigen App Markt, so findet man einige Applikationen, welche sich mit Karten, GPS und ähnlichem auseinander setzen. Jedoch bietet jede App andere Funktionalitäten. Eine App, welche offline voll funktionsfähig und individuell anpassbar ist fehlt in dieser Hinsicht jedoch.

Um einen Einblick in die Welt der Karten zu bieten sind hier einige Apps aufgelistet. So kann man sich ein Bild verschaffen über gute und auch weniger gute Ansätze der verschiedenen Angebote.

3.1.1 Maps (-) (v. 2.6.1)

<table>
<tr><th>Logo</th><th colspan="2">Inhalt</th></tr>
<tr><td>  </td><td colspan="2">Mehrere Karten Quellen zur Auswahl
Auto-Follow GPS
GPX-Dateien
Map Vergrösserung zur besseren Lesbarkeit
Recursive-Caching</td></tr>
<tr><th>Rating</th><th colspan="2">Preis</th></tr>
<tr><td>3.9</td><td>Webeunterstützt: gratis</td><td>Vollversion: Maps (+) 2.75 Fr.</td></tr>
<tr><th colspan="3">Beschreibung</th></tr>
<tr><td colspan="2">Bei dieser App ist es möglich die Karte online, also auch offline zu benutzen. Wenn man diese Offline benutzen will, so werden die Daten auf einer SD Karte gespeichert. Dabei können jedoch Probleme auftreten, dass der Speicher sich vom Real Cache Speicher beim Speichern auf die SD Karte verzehnfachen kann.

Weiter kann man bei dieser App aus mehreren Kartenarten auswählen. Es gibt zum Beispiel eine Open Street Map, Google Maps oder Open Cycle Map. So kann man seinen eigenen Favoriten wählen.

Es besitzt zudem einen GPS-Tracker, mit welchem man die aktuelle Position abfragen kann.</td><td> 

Bild 2: Maps (-)</td></tr>
</table>

3.1.2 Google Maps (v. 6.0.3)

Logo	Inhalt
	• Navigation: Kostenlose GPS-Navigation mit Sprachführung • Places: Orte finden, Bewertungen, Empfehlungen • Latitude: Freunde auf der Karte sehen und bei Orten einchecken
Rating	**Preis**
4.4	Vollversion: gratis

Beschreibung

Eines der stabilsten Apps ist sicherlich die App von Google. Die App arbeitet mit GPS und dem Mobilfunknetz. Es stickt vor allem die einfache Handhabung heraus. Es lassen sich einfach POIs finden und sich da hin lotsen.

Die Karten werden vor zu vom Internet geladen. Das wird vor allem dann schwierig und etwas mühsam, wenn keine gute Verbindung vorhanden ist.

Weitere Funktionen wie Latitude ermöglichen es die Standorte von Freunden auf der Karte darzustellen.

Offline Karten sind auch möglich, sind jedoch auf 16 Kilometer eingeschränkt und taugen für eine Zielführung leider nicht.

Bild 3: Google Maps

3.1.3 MapDroyd (v. 1.1.0)

Logo	Inhalt
	• Downloaden von Karten im Onlinemodus • Neu auch Offline-Download mittels PC • Rotierbare Karten • Aktuelle Positionsbestimmung und Verfolgung • Drag / Rubberband Modus
Rating	**Preis**
4.0	Vollversion: Gratis

Beschreibung

MapDroyd bietet weltweite Karten auf Basis von OpenStreetMap. Es können Offline-Karten heruntergeladen werden, welche auch Offline verfügbar sind. Die Karten sind stark vereinfacht und werden mit zunehmender Zoomstufe etwas detaillierter.

Die Karte kann nach Norden ausgerichtet werden oder im 45° Winkel geschwenkt werden.

Die Gesamtübersicht lässt etwas zu wünschen übrig, da sie etwas unübersichtlich ist.

Bild 4: MapDroyd

3.2 Realisierbarkeit und Herausforderungen

Die für die NeoMap App geplanten Komponenten sind keine Neuentwicklungen. Jede Komponente wird zum Teil schon in anderen Applikationen eingesetzt (siehe oben). Damit sollte der Realisierbarkeit nichts im Weg stehen. In den folgenden Abschnitten werden die grundlegendsten Herausforderungen näher beschrieben.

3.2.1 GPS Lokalisierung

Eine reine GPS Lokalisierung kann je nach Umgebung ein Problem darstellen. Sei dies durch eine zu ungenaue Ortung oder durch keinen Empfang. Das heisst, dass zum Beispiel in Manhattan der Grossstadt New York die Mobilgeräte kein wegen der vielen Wolkenkratzer kein GPS empfangen können. Auch in dichtem Wald fällt die GPS Ortung schwer. Die Folge wäre, dass man zwar eine Karte hätte, man sich jedoch nicht lokalisieren kann.

Bei einer ungenauen Positionierung hat man den Vorteil, dass man es auf dem Bildschirm durch einen zusätzlichen Kreis darstellen kann. Der Benutzer weiss somit, dass er mit Ungenauigkeiten rechnen muss und kann versuchen seine Position selbst einzuschätzen.

3.2.2 Alternativen zur GPS Lokalisierung[3]

Da eine zeitliche Einschränkung für diese Arbeit gilt muss die GPS Lokalisierung vorerst genügen. Es gibt jedoch Alternativen oder Erweiterungen, welche Ausfallsrate noch deutlich verringern können.

WLAN: WLAN Ortungssysteme eignen sich sowohl für In- als auch für Outdoor Ortungen. Einige Beispiele sind MagicMap, Ekahau, Microsoft RADAR oder IndoorWPS[4]. Eine Kombination von GPS und WLAN Ortung könnte die Genauig- und Verfügbarkeit der App deutlich steigern. Dadurch wären sogar Indoor Ortungen möglich.

Das Problem mit der WLAN Ortung besteht vor allem in der Geringen Reichweite und der zusätzlich benötigten Infrastruktur. Es könnte seinen Einsatz am ehesten in grösseren öffentlichen Gebäuden finden.

Satellitensysteme: Parallel zu dem momentan am weitesten verbreiteten Satellitensystems, dem GPS, werden in unterschiedlichsten Ländern eigene Ortungssysteme entwickelt. Dazu zählen unteranderem GLONASS, Euteltracs, Galileo, MSAS und Compass. GLONASS, welches von der Russischen Föderation entwickelt wird und bis 2012 die komplette Erde abdecken soll ist zusammen mit der Europäischen Variante Galileo die wahrscheinlich beste Alternative zum GPS, da diese die ganze Welt abdecken und technisch auf dem neusten Stand sind.

Es bestände die Möglichkeit, dass man die drei Satellitensystem GPS, Galileo und GLONASS in Kombination miteinander nutzt. Damit würde man eine sehr genaue Position erhalten.

GSM /UMTS: Eine gute und einfache Möglichkeit ist die Ortung via den bereits vorhandenen Sendemasten der Region. Mindestens die Zelle des nächsten Sendemastes wird bestimmbar sein (abhängig von dem Mobilgerät). Mit den zusätzlichen Messdaten wie Laufzeitdifferenz oder Signalstärke kann die Genauigkeit der GSM-Ortung um ein vielfaches noch verbessert werden.

Kombiniert man diese Technik mit dem GPS, so bekommt man auch mit schlechtem GPS Empfang eine gute Standort Ortung.

RFID: Mit dem sogenannten Radio Frequency Identification könnte man fest installierte Mikrochips mit dem Mobilgerät identifizieren und damit seine Position bestimmen.

Diese Variante eignet sich für den Allgemeinen Gebrauch jedoch weniger, da es mit zu viel Aufwand und Kosten verbunden wäre.

[3] Quelle: knol.google.com/k/alternative-ortungsm%C3%B6glichkeiten-zu-gps-speziell-auch-in-outhouse-lokalisierung#

[4] Quelle: gis.hsr.ch/wiki/IndoorWPS

3.2.3 Externe Datenquelle

Weiter kommt sicher auch die Frage auf, wie man die NeoMap Daten handhabt. Es muss also einen zentralen Anlaufpunkt geben, wo man diese Daten ablegen und bearbeiten kann. Nur mit standardisierten Daten kann garantiert werden, dass die App auch korrekt läuft. Die Daten müssen also von einem Server aus gemanagt werden.

NeoMaps: Primär und die einfachste Form sieht also vor, dass NeoMaps auf einen Server geladen werden. Da werden sie mit allen nötigen Informationen versehen. Es muss darauf geachtet werden, dass Namenskonflikte entstehen können. Ausserdem muss mindestens eine Information ersichtlich sein, welche sich bei sehr ähnlichen NeoMaps unterscheidet.

Beim manuellen laden von NeoMaps von zum Beispiel externen Speichergeräten besteht das Problem, wie man vom Server geladene NeoMaps mit denen vom externen Speichermedium korrekt synchronisiert. Sonst müsste man die NeoMaps erst einmal auf den Server laden oder mindestens einen Id-Request starten. Auf dem Server müsste man diese Request dann verarbeiten und dem Benutzer die neuen Informationen zurückschicken. Jedoch macht es dann nicht wirklich Sinn es als offline Variante anzusehen, wenn man dennoch eine Verbindung zum Server benötigen würde.

Benutzer: Um eine NeoMap auf den Server laden oder private NeoMaps herunter laden zu können muss der Server den Benutzer erst einmal korrekt authentifizieren können. Der Server benötigt also alle Account-Informationen des Benutzers. Auch wenn keine erfolgreiche Authentifizierung stattfinden konnte hat so soll er öffentliche NeoMaps einsehen oder downloaden können. Die Kommunikation zwischen Server und Client muss demensprechen möglichst einfach, schnell und zuverlässig sein.

3.2.4 Karten Layer

In der Kartenansicht der App besteht die Möglichkeit, dass sich mehrere NeoMaps an gewissen Koordinaten überschneiden. Nun kann nur der Benutzer entscheiden, welche er schlussendlich vollends sehen möchte. Die NeoMaps müssen also separierbar sein. Also muss man die Layer in der Z-Achse übereinander anordnen. Aber wie sieht das Handling solcher Übereinander-lappungen aus wenn sich statt ein paar plötzlich sehr viele Karten übereinander häufen? Die gewünschte Karte müsste schnell und einfach ausgewählt werden können. Dazu wird eine elegante Lösung gesucht, welche dem Benutzer eine einfache Auswahlmöglichkeit bietet.

4. Evaluation

Die Lösungsentscheidung wir mittels eines Kriterienkatalogs evaluiert. Um die verschiedenen Kriterien in ihrer Wichtigkeit zu forcieren wird ihnen eine Gewichtung von 1 bis 3 zugeteilt.

Das Bewertungsschema wird in folgende Punkte unterteilt:

1	2	3	4	5
nicht geeignet	bedingt geeignet	geeignet	gut geeignet	bestens geeignet

Multipliziert mit der Gewichtung und zusammenaddiert ergibt es die Gesamtpunktzahl. Je höher die Punktzahl umso idealer die Lösung

4.1 Datenformat Webübertragung

In diesem Fall wird darüber gewertet, welches Datenformat für die Datenübertragung gewählt werden soll. Die Server – Client Kommunikation erfolgt über HTTP. Evaluationsobjekte sind **XML** und **JSON**.

4.1.1 Kriterienkatalog und Gewichtung

Kriterium	Gewichtung
Overhead Um den Datentransfer des Mobilgeräts möglichst gering zu halten, soll auf überflüssigen Datenverkehr verzichtet werden.	3
Komplexität Es wird nicht nach einer komplexen Technologie gesucht. Es muss keine speziellen Anforderungen erfüllen und es soll einfach bedienbar sein.	1
Funktionalität Die Funktionalität sollte mit REST ausführbar sein. Es muss die Funktionen beinhalten, welche für die Erfüllung der Aufgabe notwendig sind oder noch für spätere Erweiterungen sein werden.	2

4.1.2 XML

Kriterium	Beschreibung	Gew.	Punkte	Total
Overhead	XML produziert ziemlich viel Overhead.	3	2	6
Komplexität	XML ist ein gängiges Format und wird vielseitig eingesetzt.	1	4	4
Funktionalität	Es ist etwas unperfomant XML zu parsen.	2	3	6

Summe:	**16**

4.1.3 JSON

Kriterium	Beschreibung	Gew.	Punkte	Total
Overhead	Overhead entsteht bei JSON nicht direkt. Man kann selbst entscheiden, wie viel man Metadaten mitsenden will.	3	5	15
Komplexität	JSON wird als einfach bedienbar beschrieben. Es ist eigentlich nur ein Text (String).	1	4	4
Funktionalität	Ein JSON Objekt kann einfach geparst und gelesen werden.	2	4	8
		Summe:		**27**

4.1.4 Fazit

Das Resultat ist eindeutig. JSON besitzt eindeutig die besseren Eigenschaften für die geforderten Kriterien. Es ist zwar eine neue Technologie, welche man erst noch erlernen muss. Die App-Performance wird jedoch davon profitieren.

4.2 Datensicherung

Bei der Datensicherung gilt es zu evaluieren, welches Speicherverfahren zur Anwendung kommen soll, um die Speicherdaten der App zu speichern. Dabei stehen **SQLite** und das Speichern in **Properties** zur Auswahl.

4.2.1 Kriterienkatalog und Gewichtung

Kriterium	Gewichtung
Performance (Lese- und Schreibgeschwidigkeit) Die App soll performant sein, und Speicher und Ladeoperationen schnell durchführen können.	3
Komplexität Da es hier um Datenhandling geht, spielt die Komplexität nicht so eine grosse Rolle. Wichtig ist, dass die Daten gut gehandhabt werden können.	1
Funktionalität Ein wichtiger Punkt ist in diesem Fall die Funktionalität, welche eine Variante bietet. Daten müssen geladen, gespeichert, ersetzt, erweitert und anderweitig manipuliert werden.	3
Schutz gegen unbefugte Manipulationen Ein Benutzer soll nicht in der Lage sein aus Versehen Daten zu manipulieren. Da die App keine sensiblen Daten beinhaltet ist dies nur teilweise wichtig.	2

4.2.2 SQLite

Kriterium	Beschreibung	Gew.	Punkte	Total
Performance	SQLite braucht eine gewisse Zeit um Lese- und Schreiboperationen durchzuführen. Mit neueren Geräten jedoch sinkt die Zeit im Verhältnis zu den Properties jedoch	3	3	9
Komplexität	SQLite benötigt schon eine gewisse Einarbeitungszeit. Jedoch ist SQL noch von früher bekannt.	1	2	2
Funktionalität	SQLite bietet viele Funktionalitäten. Unteranderem können Daten gut geordnet abgelegt werden.	3	5	15
Schutz	Im Programm selber muss erst Schreib oder Leserecht angefordert werden, bevor Daten Manipuliert werden können. Gross geschützt ist die Datenbank jedoch nicht.	2	2	4
			Summe:	**30**

4.2.3 Properties

Kriterium	Beschreibung	Gew.	Punkte	Total
Performance	Lese- und Schreiboperationen sind sehr performant.	3	5	15
Komplexität	Die Handhabung ist einfach gehalten.	1	4	4
Funktionalität	Die Hauptfunktionen von Properties sind lesen und schreiben.	3	1	3
Schutz	Biete in diesem Sinne keinen Schutz.	2	1	2
			Summe:	**24**

4.2.4 Fazit

Für die App eignet sich besser eine SQLite Datenbank. Sie bietet vor allem gute Voraussetzungen für das Handling der NeoMap Daten. Sie bietet eine geordnete Speicherstruktur.

5. Umsetzungskonzept

In diesem Kapitel beinhaltet eine grobe Übersicht der anwandten Lösungskonzepte. Eine Detailliertere Fassung ist unter Teil II zu finden.

5.1 Datenquellen/Services

Die NeoMap App benutzt diverse Datenquellen. Eine kurze Erläuterung dazu finden in nachfolgenden Kapiteln.

5.1.1 NeoMap Server

Der NeoMap Server dient als Verwalter der NeoMaps. Um private NeoMaps auf den Server laden zu können muss man nur ein Account auf der Webseite erstellen und schon ist man startklar. Öffentliche NeoMaps können auch ohne Account geuploadet werden. Die NeoMaps werden ausserdem beim Uploaden mit allen erforderlichen Informationen versehen.

Die NeoMaps können später ganz einfach vom Server auf das Android Gerät geladen werden. Meldet man sich auf dem Mobilgerät an, so kann man auch seine privaten NeoMaps downloaden.

5.1.2 OpenStreetMap (OSM)

Von OSM werden die Kartentiles bezogen, welche für den Aufbau der online Karte benötigt werden. Ausserdem könnte über diesen Service nach Places und POIs gesucht werden. Die OSM und NeoMap daten werden bei der App dann für OpenGL in Mercator Koordianten umgerechnet.

Sollte im späteren Gebrauch OSM zu stark ausgelastet werden müssen die Daten auf den NeoMap Server gespiegelt oder OpenStreetMap-in-a-Box benutzt werden.

5.2 Lokalisierung

Momentan erfolgt die Lokalisierung erfolgt mittels GPS. Jedoch ist in einer späteren Version geplant die Lokalisierung zusätzlich über GSM/UMTS-Sendemasten zu ergänzen.

Eine GPS Lokalisierung wird nur auf Anfrage ausgelöst. Das hat den Vorteil, dass die App nur sehr wenig Akku verbraucht. Es gibt jedoch eine Funktion, der sogenannte Follower-Mode welcher in Regelmässigen Abständen die Position neu ermittelt. Aktuell eingestellt bei einem Intervall von drei Sekunden oder bei schnellerer Fortbewegung alle 10 Meter (= ab 36km/h).

5.3 Datenhaltung

Heruntergeladene NeoMaps werden direkt im Android Filesystem gespeichert. Da aber noch mehr Informationen zu jeder NeoMap vorhanden sind, als das KMZ File selbst, wird zusätzlich zu jedem Download ein SQLite Eintrag mit den zusätzlichen Informationen abgespeichert.

Beim Laden der NeoMapinformationen auf der SQLite Datenenbank wird parallel ein Thread ausgeführt, welcher die Namen der im Filesystem befindenden NeoMaps mit denen in der Datenbank vergleicht. Damit Namenskonflikte ausgeschlossen werden können, wird beim Abspeichern in das Filesystem dem Dateiname die uniq Id hinzugefügt.
Findet der Algorithmus Unregelmässigkeiten, z.B. ist keine ID im Dateiname vorhanden, so ist es sehr wahrscheinlich, dass kein Datenbank Eintrag dazu existiert. Nun ist es dem Benutzer überlassen, ob er diesen „File Mismatch" beheben will und das unbekannte File löschen will oder einfach zu ignorieren.

Später ist gedacht, dass so auch offline hinzugefügte NeoMaps verwaltet werden können. Der Algorithmus erkennt das neue File und beginnt es im Hintergrund mit allen nötigen Informationen zu versehen. Im Onlinemodus könnten die NeoMaps so sogar automatisch auf den Server geladen werden. Der Benutzer müsste dies nur noch bestätigen und alles wird vollautomatisch ausgeführt.

Die App besitzt ausser den NeoMap Einträgen auch noch zwei SQLite Tables in denen die App und Layer Settings abgespeichert sind.

Weiteres ist unter dem Kapitel „3.2 Datenbank Modell" im Teil II beschrieben.

6. Resultate und Ausblick

6.1 Codeinformationen, Bugs & Incomplete Features

Da Patrick Recher sich nicht mit Dokumentationen oder Abgabetermine halten muss, wird das Projekt parallel immer noch erweitert. Dies macht es extrem schwer die Dokumentation auf dem neusten Stand zu halten. Deshalb sind im Code auch noch Kommentare vorhanden, welche momentan noch am Entwickeln sind. Und um Merge-Konflikte zu vermeiden, sollte nur der Ersteller diese Kommentare diese später auch wieder löschen. Vor allem auch, da er diese Vermerke noch braucht. Das Refactoring war darum nur zu einem gewissen Teil durchführbar.

Die Folgenden Todo's werden in der nächsten Version noch vervollständigt.

ID	Bug / Incomplete Feature (Stand 22.12.2011)	Typ
B01	Die Koordinaten Angaben werden momentan nur im erst geladenen Raster angezeigt. Danach werden die Zahlen nicht mehr aktualisiert.	Bug
B01	Bei der Überprüfung der Daten und der SQLite Datenbank wird zwar geprüft, ob die Daten stimmen, werden jedoch noch nicht gehandelt, respektive bei einem Fehler verarbeitet.	Incomplete Feature
B03	Die Layer Settings sind soweit bereit, sind jedoch noch nicht	Incomplete

	mit der MapView fertig verknüpft.	Feature
B04	Die NeoMaps auf der Kartenansicht sind noch nicht korrekt rotiert und beim Dragen können sie etwas verschieben.	Incomplete Feature

6.2 Zielerreichung

In der nächsten Tabelle findet sich eine Übersicht der funktionalen und nicht funktionalen Anforderungen. Zu den einzelnen Punkten ist jeweils eine Beschreibung dazu abgegeben, wie und warum es sich so entwickelt hat.

ID	Anforderung	Beschreibung	
F01	Die App soll im Offlinebetrieb (ohne Internet) benutzbar sein.	Heruntergeladene NeoMaps werden angezeigt. Ausserdem zeigt es im Cache befindliche OSM Tiles auch offline an.	✓
F02	Neue NeoMaps und OSM-Karten sollen über das Internet geladen werden können.	Dies kann über den NeoMap Server geschehen.	✓
F03	Die Sprache für die Applikationsoberfläche kann geändert werden.	Die App kann Deutsch und Englisch (für alle anderen Sprachen) anzeigen.	✓
F04	Mittels Eingabe von GPS Koordinaten kann die Karte zentriert werden.	Aus Zeitgründen konnte diese Funktion noch nicht implementiert werden, sie wird jedoch in nächster Zeit noch implementiert.	✗
F05	Bei den NeoMaps soll es eine Funktion geben, welche wenn möglich automatisch die Gesamtübersicht der Karte auf dem Bildschirm darstellt. Zoom to Layer Extents.	Wiederum aus Zeitgründen konnte man dieses Feature noch nicht realisieren.	✗
F06	Die eigene momentane Position soll bestimmt werden können.	Auf der Menüleiste in der MapView geschiet dies, wenn man auf das Pfeilsymbol klickt.	✓
F07	Die Koordinaten des Fadenkreuzes (Bildschirmmittelpunkt) können abgefragt werden.	Es wird die momentane Position des Cursors angezeigt.	✓
F08	Die Karteansicht soll sich bei Bedarf automatisch nach	Mit der Kompassfunktion richtet sich die Karte dynamisch nach Norden	✓

	Norden ausrichten können z.B. gemäss Kompass.	aus.	
F09	Es soll möglich sein, dass die momentane Position immer im Bildschirm Mittelpunkt angezeigt wird und sich die Karte Drumherum mit aufbaut (Follower Mode).	Aktiviert man den Follower-Modus, so wird die Position in regelmässigen Abständen aktualisiert.	✓
F10	OSM (Tiles) und NeoMaps (Tiles) sollen als deckende Ebenen (Layer) dargestellt werden.	Der unterste Layer wird mit der OSM dargestellt. Darüber sind dann die NeoMaps zu sehen.	✓

6.3 Ausblick: Weiterentwicklung

Das Projekt hat sich bisher gut entwickelt. Jedoch bedarf es noch einige Arbeitsstunden um die Applikation Marktreif zu machen. Die App wird noch weiterentwickelt und es werden die fehlenden Funktionen noch implementiert.

6.4 Persönlicher Bericht

Ich es sehr genossen diese Arbeit absolvierten zu dürfen. Vor allem hat es mich wegen der Android Programmierung interessiert. Ich konnte mir einen tiefen Einblick in diese neue Technologie gewähren, da einerseits Server-Client Kommunikation stattfinden musste. Andererseits konnte etwas über das Datenhandling von Android erfahren (SQLite und Filesystem). Auch der Einblick in OpenGL war ziemlich interessant. Leider hat die Zeit nicht gereicht sich vertieft mit diesem Thema zu befassen.

Die NeoMap App hat noch grosses Potential. Es ist erstaunlich was man alles für Smartphones machen kann. Für die App gäbe es noch sehr viele Features die leider nicht implementiert werden konnten, jedoch sehr interessant wären.

Etwas umständlich fand ich jedoch die unregelmässige Terminfindung mit Herrn Recher. Herr Recher ist ein sehr talentierter Programmierer. Jedoch ist es in einer Einzelarbeit schon ziemlich schwer alles alleine zu Planen und zu organisieren. Da Herr Recher auf einem anderen Niveau tätig ist, wo er nicht gross unter Druck steht ist es für mich sehr schwierig für ihn mit zu organisieren. Zudem ist dies mit Teilzeitstudenten nochmals etwas komplizierter.
Ich habe die Arbeit jedoch mit Herrn Recher als interessant und lehrreich empfunden. Jedoch kann ich diese Vorgehen nicht empfehlen. Sollte eine Zusammenarbeit stattfinden müssen die Personen auf demselben Niveau, respektive denselben Motivationsgrund besitzen (SA, BA, …).

Im Allgemeinen hat mir die Arbeit jedoch spass gemacht.

6.5 Dank

Folgender Abschnitt ist an die unterstützenden Personen gerichtet.

Prof. Stefan Keller (HSR)
Ich danke Ihnen für die Unterstützung während des Projekts. Die Zusammenarbeit mit Ihnen hat mein Wissen in der Geodatenverabeitung sehr bereichert.

Binna Tobias (IFS)
Vielen Dank für deine Unterstützung im Serverbereich. Die Kommunikation hat auch über den elektronischen Weg gut und schnell funktioniert.

OCAD AG
Mein Dank geht auch an die Firma OCAD, welche uns Testdaten zur Verfügung gestellt hat.

Recher Patrick
Ich danke Herr Recher für seine Mitarbeit und Unterstützung in diesem Projekt.

Teil II
SW Projektdokumentation

1. Vision

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

2. Anfoderungsspezifikation

2.1 Funktionale Anforderungen

2.1.1 Übersicht

ID	Anforderung	Priorität
F01	Die App soll im Offlinebetrieb (ohne Internet) benutzbar sein.	sehr hoch
F02	Neue NeoMaps und OSM-Karten sollen über das Internet geladen werden können.	hoch
F03	Die Sprache für die Applikationsoberfläche kann geändert werden.	niedrig
F04	Orte können mittels Eingabe von GPS Daten lokalisiert werden.	mittel
F05	Bei den NeoMaps soll es eine Funktion geben, welche wenn möglich automatisch die Gesamtübersicht der Karte auf dem Bildschirm darstellt.	hoch
F06	Die eigene momentane Position soll bestimmt werden können.	hoch
F07	Die Koordinaten des Fadenkreuzes (Bildschirmmittelpunkt) können abgefragt werden.	mittel
F08	Die Karteansicht soll sich bei Bedarf automatisch nach Norden ausrichten können z.B. gemäss Kompass.	mittel
F09	Es soll möglich sein, dass die momentane Position immer im Bildschirm Mittelpunkt angezeigt wird und sich die Karte Drumherum mit aufbaut (Follower Mode).	mittel
F10	OSM (Tiles) und NeoMaps (Tiles) sollen als deckende Ebenen (Layer) dargestellt werden.	sehr hoch

2.1.2 Detailiert

Offline Modus	
F01	Der Benutzer ist in der Lage dieses App offline zu benutzen. Wird eine verfügbare NeoMap im Programm ausgewählt, wird die Offline NeoMap aus dem NV-Storage geladen und über der OpenStreetMap dargestellt.

	Dabei kann auch ausgewählt werden, dass die Karte ohne die OpenStreetMap-Karte angezeigt werden soll.
Online Modus	
F02	Schaltet man den Online Modus ein, so ist der Benutzer in der Lage neue Kartensektoren direkt vom Internet zu laden. Dabei muss mindestens Wireless Lan oder eine Funkverbindung (3G, EDGE, o.ä.) vorhanden sein. Der Benutzer soll durch eine Anmeldung auch in der Lage sein, private NeoMaps von einem Server zu downloaden. Die neusten Kartenteile der geladenen OpenSteetMap sollen auch später noch offline verfügbar sein. Dies ist jedoch abhängig vom Speicher des Gerätes und der eingestellten Queue-Size.
Spracheinstellungen	
F03	Die Spracheinstellungen der App sollen von der im Betriebssystem eingestellten Sprache übernommen werden. Vorerst werden jedoch sollen jedoch nur die Sprachen Englisch und Deutsch unterstützt werden. Wird auf dem Gerät eine andere Sprache als die oben genannten verwendet, wird das Programm automatisch die englische Sprache beanspruchen.
Orte lokalisieren mittels GPS Daten	
F04	Gibt man manuell GPS Koordinaten in die App ein, so soll sich der Bildschirmmittelpunkt auf die eingegebenen Koordinaten ausrichten.
Sofortige Gesamtübersicht einer NeoMap	
F05	Eine Funktion soll helfen, schnell und einfach die Gesamtübersicht einer NeoMap darzustellen. Dies kann nützlich sein, wenn man eine NeoMap sucht. Auch wenn man in der NeoMap schnell den Gesamtüberblick haben möchte ist diese Funktion sehr nützlich.
My Location	
F06	Durch eine GPS Lokalisierung soll man in der Lage sein die momentane Position bestimmen zu können. Voraussetzung ist, dass eine GPS-Verbindung möglich ist. Je nach Aufenthaltsort kann die Genauigkeit der Ortung stark variieren. Dem Benutzer kann diese Ungenauigkeit angezeigt werden.
Koordinatenabfrage	
F07	Will man die Koordinaten eines gewissen Punktes auf der Karte wissen, so soll es eine Möglichkeit geben, die Position anzuvisieren und die Koordinaten abzufragen.
Ausrichtung nach Norden	
F08	Die Kartenansicht kann nach Norden ausgerichtet werden. Die Position kann sich automatisch neu anpassen, sollte der Benutzer seine Richtung verändern.

Follower Mode	
F09	Mit dem Follower Mode soll es dem Benutzer einfach gemacht werden die momentan aktuelle Position im Mittelpunkt des Bildschirmes zu verfolgen. Dies ist vor allem dann nützlich, wenn sich der Benutzer in regelmässigen Abständen bewegt. Das erspart ihm das ständige Zentrieren seiner Position.
Mehrere Layer	
F10	Die OSM und NeoMap sollen so dargestellt werden, dass der Benutzer nicht mitbekommt, dass sich die verschiedenen Karten-Layer in der Z-Achse unterscheiden, resp. auf verschiedenen Ebenen befinden. Der OSM befindet sich dabei auf dem untersten Layer.

2.2 Use Cases

Bei den Use Cases wird davon ausgegangen, dass alle externen Dienste der Applikation funktionstüchtig sind und die App richtig installiert wurde. Sprache ist auf Deutsch eingestellt.

2.2.1 Use Case Diagram

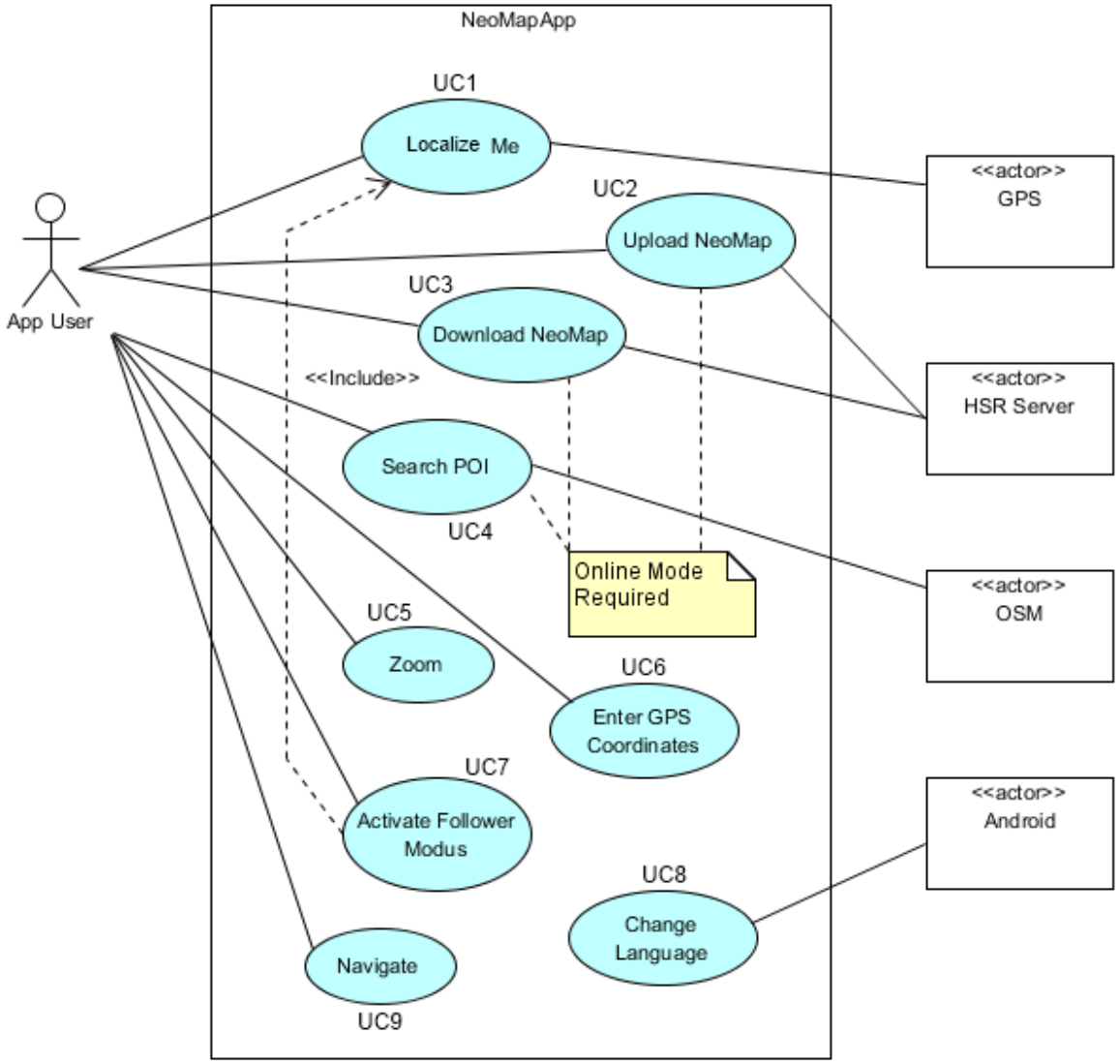


Bild 5: Use Case Diagram

2.2.2 Use Case Beschreibung

Localize Me	
	Precondtition: GPS Lokalisierung möglich
UC1	Der Benutzer möchte sich via GPS orten. Dazu drückt er in der Kartenansicht der App auf den lokalisieren Button. Darauf startet die App eine GPS. Sobald die GPS Daten empfangen wurden positioniert sich der Mittelpunkt der App am Zielort des Benutzers. Dabei wird auch die Genauigkeit (Abweichung) der Ortung angezeigt.
Upload NeoMap	
	Precondition: PC/Notebook mit Internetzugang
UC2	Will man seine eigenen Karten verwenden muss man sie auf dem Computer in ein kompatibles Format bringen. Nachdem der Benutzer dies erledigt hat, kann er seine Karte auf den Server laden. Die hochgeladenen NeoMaps sind für jedermann zugänglich und können auch heruntergeladen werden. Will man die Karte nicht öffentlich zugänglich machen, kann man sich natürlich auch ganz einfach auf dem Server anmelden und die NeoMaps als privat deklarieren.
Download NeoMap	
	Precondtition: Internetzugang (Online Modus) für 1. Möglichkeit
UC3	Will ein Benutzer neue NeoMaps herunterladen, kann er in der Karten-ansicht auf den „Menü"-Button drücken und im Menü „NeoMaps" den Button „NeoMaps durchsuchen & hinzufügen" drücken. Nun bieten ihm sich zwei Möglichkeiten: 1. Möglichkeit Klickt man auf den Button wird eine Liste mit allen verfügbaren NeoMaps vom Server herunter geladen und in einer scrollbaren Liste dargestellt. Nun besteht die Möglichkeit in der Suche nach gewünschten Namen zu suchen. Dabei hilft ein REGEX Algorithmus, der alle verfügbaren NeoMaps auflistet, welche die Buchstabenfolge beinhalten.
Search POI	
	Precondtition: Internetzugang (Online Modus)
UC4	Falls man auf die NeoMaps verzichten will kann man auch mit den OSM Orten suchen. Die entsprechende Suche findet man ganz oben auf dem Bildschirm der Kartenansicht. Hier kann man nach allem Suchen, was bereits in der OSM-Library enthalten ist. Mittels Internet wird die Anfrage an den OSM Service gesendet. Dieser sendet eine entsprechende Antwort mit den angefragten Daten, falls erfolgreich. Die geladenen Kartenfragmente sind bis zu einer gewissen Speichergrösse auch noch nach dem Neustart der App offline Verfügbar, da sie im NV-Storage abgelegt werden.

	Wird die Speicherkapazität überschritten werden alte Kartenfragmente wieder freigegeben und durch neue ersetzt. (Variante ist vorerst nicht vorgesehen)
Zoom	
	Precondtition: Benutzer befindet sich in der Kartenansicht
UC5	Ein Benutzer kann durch ausführen der sogenannten Zwei-Finger-Pinch-Geste die Zoomstufe der Karte verändern, solange er nicht das Minimum oder das Maximum erreicht hat. Bewegt man den Daumen und Zeigefinger auseinander, so wird die Karte präzisier dargestellt. Werden Daumen und Zeigefinger zusammengeführt, so bekommt man eine grössere Übersicht der Karte.
Enter GPS	
	Precondtition: Benutzer befindet sich in der Kartenansicht
UC6	Im Menü der Kartenansicht kann der Benutzer manuell GPS Koordinaten eingeben. Die App sucht berechnet dann wo sich der Punkt befindet und stellt in im Bildschirmmittelpunkt dar.
Activate Follower Mode	
	Precondtition: Benutzer befindet sich in der Kartenansicht
UC7	Aktiviert der Benutzer den Follower Mode im Menü, so wird seine Position, bis er diesen wieder deaktiviert, im Bildschirmmittelpunkt gehalten und aktualisiert.
Change Language (CRUD)	
	Precondtition: -
UC8	Die Sprache selbst wird bestimmt vom Betriebsystem. Der Benutzer kann jedoch in den Systemeinstellungen von Android die Sprache einstellen. Die eingestellte Sprache wird vom System in die App übernommen.
Navigate	
	Precondtition: Benutzer befindet sich in der Kartenansicht
UC9	Mit einer Drag-Geste kann der Benutzer sich auf der Karte bewegen. Wird mit einem Finger zum Beispiel von Oben nach unten gezogen, so bewegt sich mit dem Finger. Dabei kann in jede Richtung navigiert werden.

2.3 Nicht funktionale Anfoderungen

Benutzbarkeitsanforderungen	
NF1	Es wird darauf geachtet, dass die App in möglichst vielen Situationen Alternativen bietet, sodass man auch bei Verbindungsausfällen eine möglichst hohe Verfüg- und Benutzbarkeit garantieren kann. Damit man die App benutzen kann wird mindestens Android 1.6 vorausgesetzt. Eine

	hohe Hardwareleistung wird nicht vorausgesetzt.
Performance	
NF2	Bei der Performance wird darauf geachtet, dass der Akku möglichst wenig ausgelastet wird. Das heisst nur Daten verarbeitet und versendet wenn nötig. Die Kartensegmente (Tiles) werden möglichst effizient im Speicher gehalten, sodass die bestehenden Ressourcen wiederverwendet werden können.
Bedientbarkeit	
NF3	Die Benutzerfreundlichkeit der App soll möglichst einfach gehandhabt werden können. Die Menus sind gut sichtbar dargestellt und werden durch Bilder zusätzlich unterstützt. Die App versucht sich hauptsächlich an die Standards und Richtlinien von Android zu halten.
Dokumentation	
NF4	In der Dokumentation werden die wesentlichen Entscheidungen und wichtigsten Kernpunkte der Arbeit festgehalten.
Testing	
NF5	Die App soll mit GUI- und Logik-Tests kontrolliert werden. Dadurch wird auch die Codequalität erhöht. Ausserdem ist es für eine Weiterentwicklung notwendig.
User Interface	
NF6	Die Benutzeroberfläche soll verständlich und einfach bedienbar sein. Der Benutzer soll sich wohl und sicher in der Bedienung fühlen. • GUI-Elemente sollen gross genug sein, und gut erkennbar sein. • Nur so viele Informationen wie nötig darstellen • Die GUI soll erwartungstreu und strukturiert auftreten
Code	
NF7	Der Code soll von Anfang an sauber gehalten werden. Die Codestruktur sollte von Anfang bis Schluss gleich gehalten werden.

2.4 Detailspezifikation

2.4.1 Zielgruppe

Der Fokus dieser App fällt zum einen auf Organisationen, welche Anlässe veranstalten und einfache Möglichkeit benötigen Standorte spezifisch zu beschreiben. Zum anderen sind da auch noch die Personen, denen örtliche Kenntnisse einer Region fehlen. Die Zielgruppe fällt also grundsätzlich auf Personen, die ein Android Mobilgerät besitzen und eine günstige und einfache Orientierungshilfe benötigen.

2.4.2 Erweiterungsmöglichkeiten (Whishlist)

Hier sind noch ein paar mögliche Erweiterungen für die App aufgelistet. Dies sind nur ein paar von vielen Erweiterbarkeiten.

Integriertes Zeichnungstool

Grundfunktion:

Mit dem Zeichnungstool kann man auf dem mobilen Gerät individuell auf dem NeoMap oder dem OSM Layer zeichnen. Dazu wird ein transparenter Layer über dem gewünschten Objekt hinzugefügt. Ausserdem kann man auf dem Layer auch verschiedene vordefinierte (geometrische) Figuren setzen.

Zusätzliche Funktionen:

Durch ein zusätzliches Zeichnungsmenü können Strichstärke, Farbe geändert werden. Es können auch zusätzliche Layer hinzugefügt werden und benutzer-definiert ein- oder ausgeblendet werden.

Manuelle Routeneingabe

Grundfunktion:

Durch setzen von manuellen POIs kann man eine Funktion aktivieren, welche die verschiedenen Punkte miteinander verbindet.

Zusätzliche Funktionen:

Erweitert man dieses Szenario mit mehreren möglichen Routen, so kann man die POIs einer Farbe (oder Form) zuweisen. Dadurch lassen sich mehrere alternative Routen darstellen. Zudem könnte man je nach Belieben alle oder nur gewisse Routen anzeigen lassen. Die Länge der Route kann angezeigt werden.

Friends

Grundfunktion:

Es gibt die Möglichkeit seinem Account ein Bild zu zuweisen. Aktiviert man in seinem Account die Funktion „Freunde anzeigen" und sind diese in einer Friendlist, so werden diese Freunde auf der OSM oder NeoMap mit ihrem Bild dargestellt.

Zusätzliche Funktionen:

Klickt man auf das Bild des Freundes, so bekommt man seinen Namen und eine Statusmeldung. Die Statusmeldung lässt sich beliebig setzen. Will ein Freund die Aufmerksamkeit auf sich lenken so kann er einen „Ping" senden. Sein Bild wird dazu mit einem Ausrufezeichen versehen.

Wetter

Grundfunktion:

Das Wetter und weitere Aussichten einer Region kann auf der Karte angezeigt werden.

Tracklog

Grundfunktion:

Nachträglich kann man seine Route auf der Karte abfragen. Dazu werden die GPS Daten während einer gewissen Zeit dokumentiert und auf Anfrage

ausgewertet und dargestellt.

Zusätzliche Funktionen:
Es können noch zusätzliche Informationen abgerufen werden:

- Timestamp
- Absolvierte Kilometer/Höhenmeter
- Geschwindigkeiten

2.4.3 Lizenzanforderungen

Für die App wird die MIT-Lizenz[5] verwendet. Hier ein Auszug von der Wikipedia Seite:

2.4.4 Verwendete Standards

Die Android App richtet sich so weit wie möglich an die Android Guidelines[6]. Sie können auf der Android Webseite gefunden werden.

[5] Quelle: de.wikipedia.org/wiki/MIT-Lizenz

[6] Quelle: developer.android.com/guide/practices/ui_guidelines/index.html

3. Analyse

3.1 JSON Objekte

Damit die JSON Objekte vom Server verarbeitet werden können müssen sie folgende Konventionen (API) befolgen. Die Übertragung findet über HTTP statt.

Folgend werden die Benutzerinformationen an den Server geschickt. Dieser authentifiziert diese Informationen und gibt ein Feedback.

Login		
URL	labs.geometa.info/neomapserver/api/[version]/user/login/	
Parameters	-	
Method	POST	
Returns	200 OK	All ok. Logged in.
	401 Unauthorized	Either the user provided credentials are wrong or there were no credentials at all. Also in case the user account is not active.

Description	Login the user given in POST data. data should be in JSON format as shown below.
	`{`
	`"username":"user",`
	`"password":"user"`
	`}`	

Hier werden alle auf dem Server befindenden NeoMap Informationen abgerufen. Hat sich der Benutzer zuvor noch erfolgreich eingeloggt, werden zusätzlich auch seine privaten NeoMap Informationen mitgeliefert.

Get NeoMap List	
URL	labs.geometa.info/neomapserver/api/[version]/neomap/
Parameters	-
Method	GET
Returns	200 OK

Description	Returns a list of NeoMaps. This should always return successfully. In case there are no maps the objects array will be empty.
	`{`
	`"meta":{`
	`"limit":20,`
	`"next":null,`
	`"offset":0,`
	`"previous":null,`
	`"total_count":3`
	`},`

```
    "objects":[
       {
          "boundingBox":"[8.81…, 47.21…, 8.93…, 47.23…]",
          "fileSize":1120938,
          "id":"1",
          "isPublic":true,
          "mapDescription":"Die Karte liegt etwas schief
                            in der Gegend.",
          "mapKMLFile":"http://localhost:8000/media/maps/
                        Chlosterwald_2003_8.kmz",
          "mapName":"Chlosterwald",
          "resource_uri":"/api/v1/neomap/1/",
          "uploadDate":"2011-12-09T09:51:00.173598",
          "user":{
             "resource_uri":"/api/v1/user/3/",
             "username":"Testuser"
          }
       },
       {
          … //more Data
          }
       }
    ]
}
```


Die nächste Anfrage liefert das eigentliche NeoMap Objekt.

Get NeoMap		
URL	labs.geometa.info/neomapserver/api/[version]/neomap/[id]/	
Parameters	-	
Method	GET	
Returns	200 OK	All ok.
	401 Unauthorized	In case the map with [id] is private and the requesting user is not the map owner.
	400 Bad Request	In case no map with [id] exists.
Description	Returns a specific NeoMaps identfied by [id]. { "boundingBox":"[8.818…, 47.217…, 8.902…, 47.226…]", "fileSize":715585, "id":"2", "isPublic":true, "mapDescription":"\u00f6ffentliche Karte von Oberwald", "mapKMLFile":"http://localhost:8000/media/maps/ Oberwald_19.kmz", "mapName":"Oberwald Karte Testuser", "resource_uri":"/api/v1/neomap/2/",	

```
    "uploadDate":"2011-12-09T09:52:07.330575",
    "user":{
       "resource_uri":"/api/v1/user/3/",
       "username":"Testuser"
    }
}
```

3.2 Datenbank Modell

Das Datenbankschema der App ist einfach aufgebaut. Folgend ist der Aufbau und der Inhalt der drei Tables dargestellt.

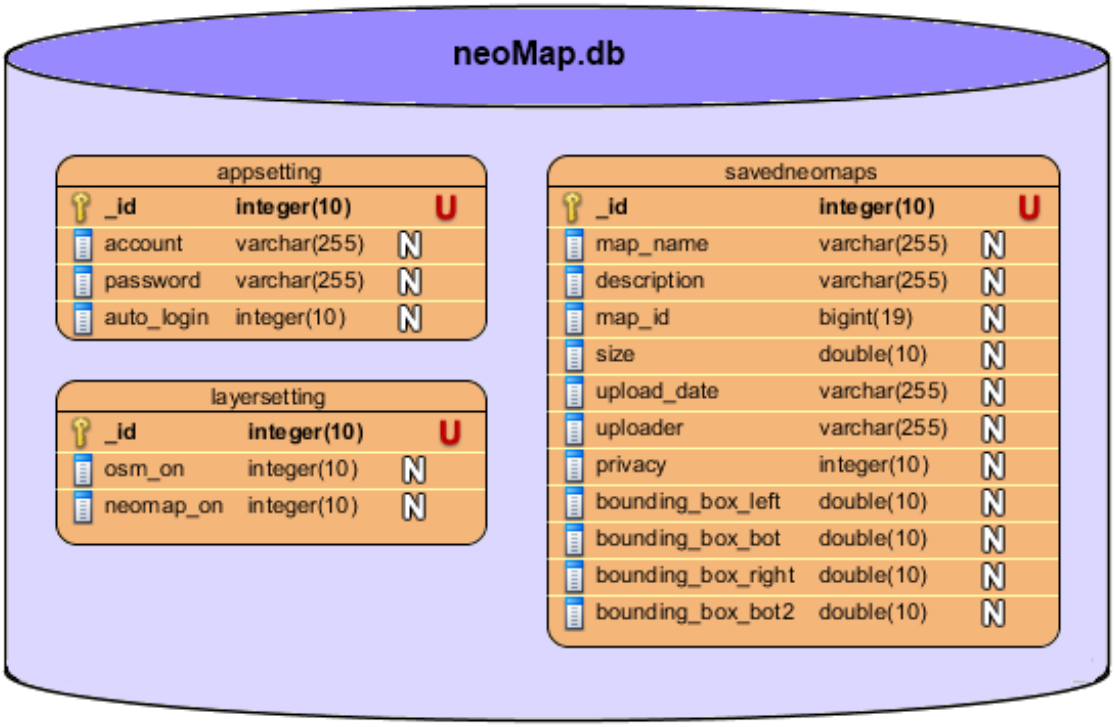


Bild 6: Datenbank Modell

3.3 Android Lifecycle-Modell[7]

Die Kernklassen, die Activities, werden vom Android Betriebssystem gemanagt. Es ist wichtig, dass man die Funktionsweise, oder in diesem Fall den Lebenszyklus, kennt. So lassen sich bei der App-Programmierung Fehler und Bugs schon im Vorhinein vermeiden.

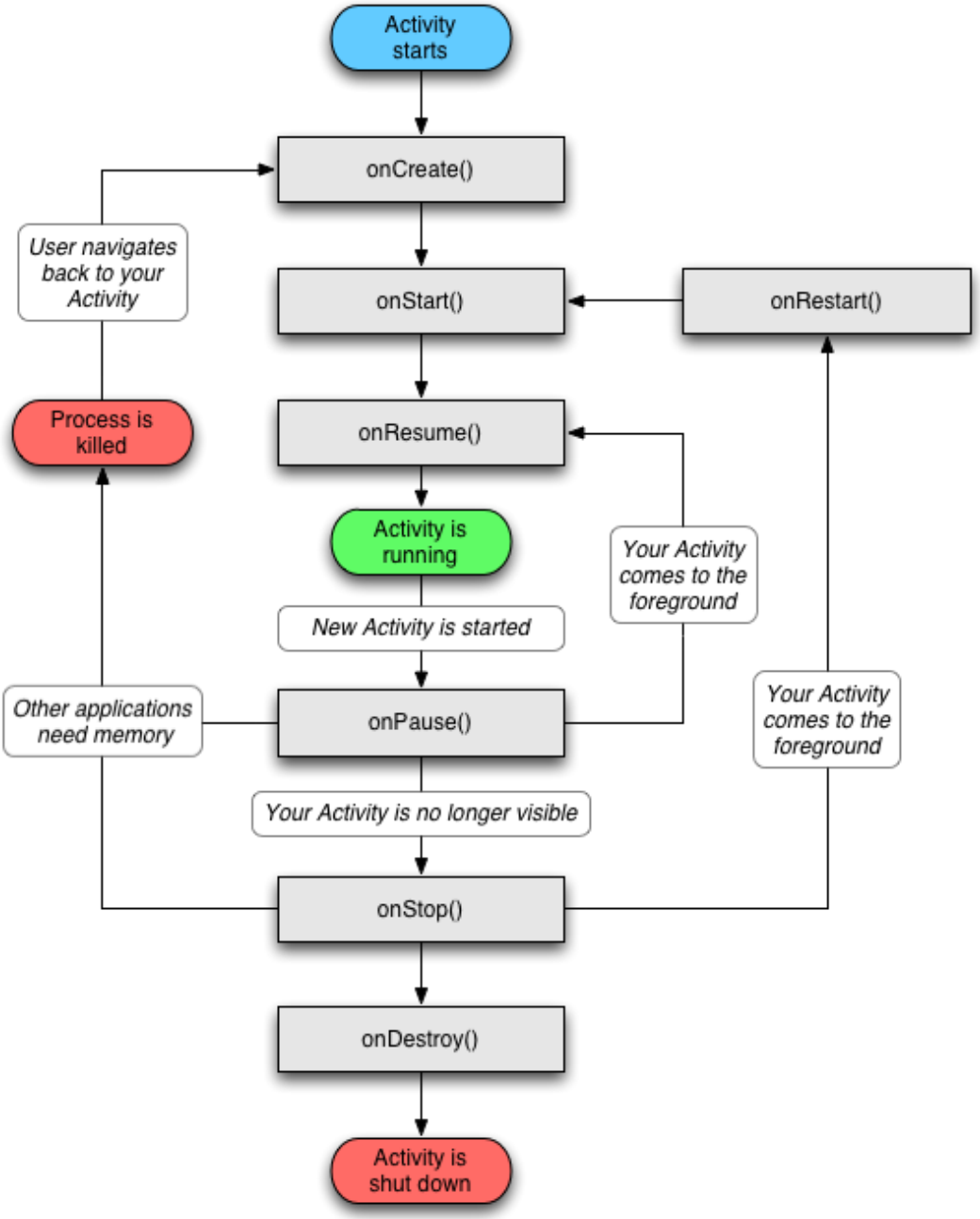


Bild 7: Activity Lifecycle

[7]Quelle Activity Lifecycle: www.androidjavadoc.com/1.0_r1_src/android/app/Activity.html

4. Design

4.1 Domain Modell

Das folgende Domain Modell beinhaltet die Hauptfunktionen der App.

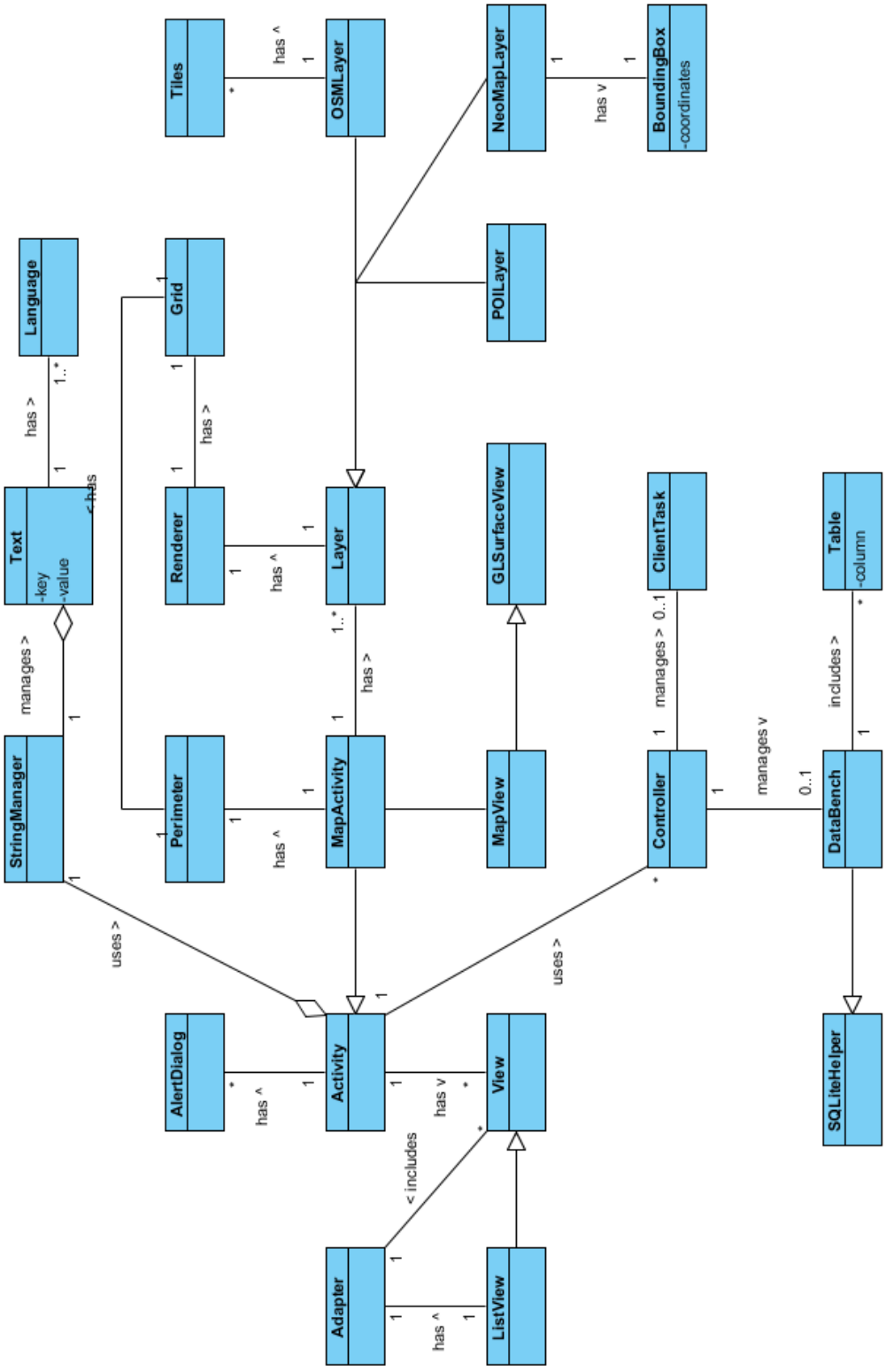


Bild 8: Domain Modell

4.2 Architektur

Bei der NeoMap App handelt es sich um eine Drei-Schichten-Architektur. Die folgenden Ausschnitte zeigen das Zusammenspiel der internen Verwaltung der Applikationsstruktur.

Die momentane Paketstruktur sieht folgendermassen aus:

Da diese sich momentan jedoch noch in kontinuierlicher Entwicklung von Herr Recher befindet ist sie noch nicht endgültig.

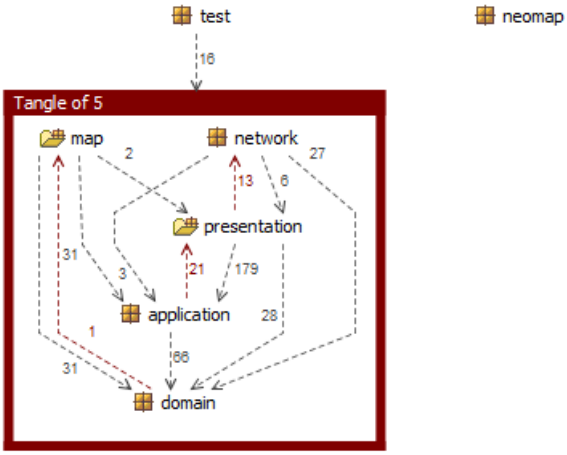


Bild 9: Paketstruktur

4.2.1 Klassenstruktur

activity Packet

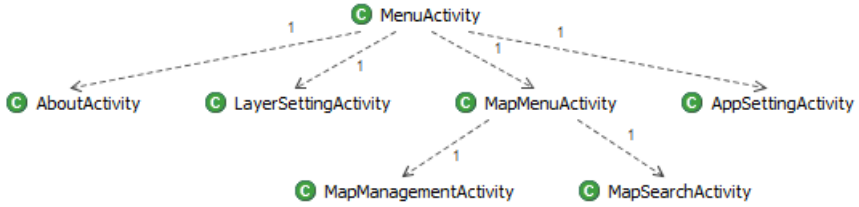


Bild 10: activiy Packet

view Packet

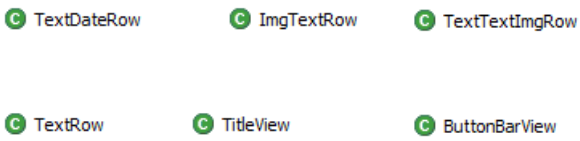


Bild 11: view Packet

application Packet

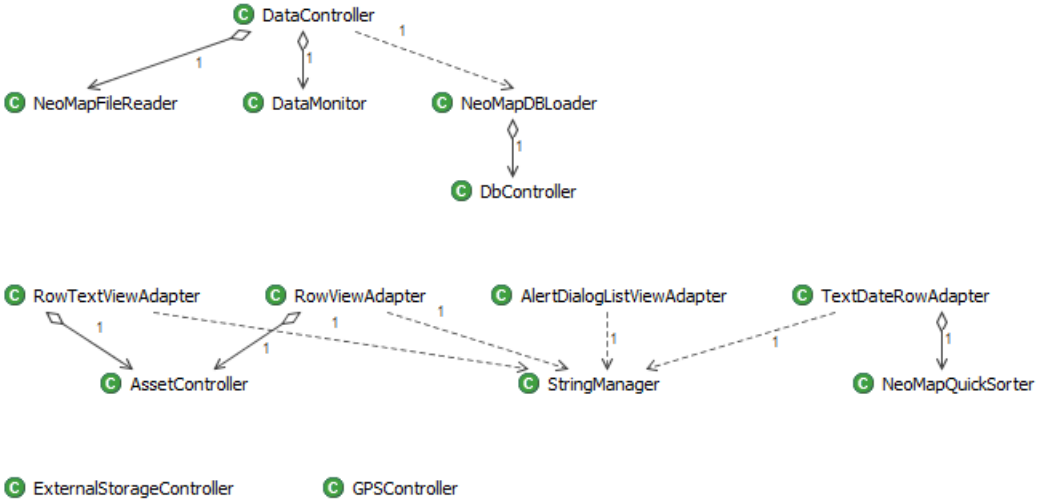


Bild 12: application Packet

map Packet

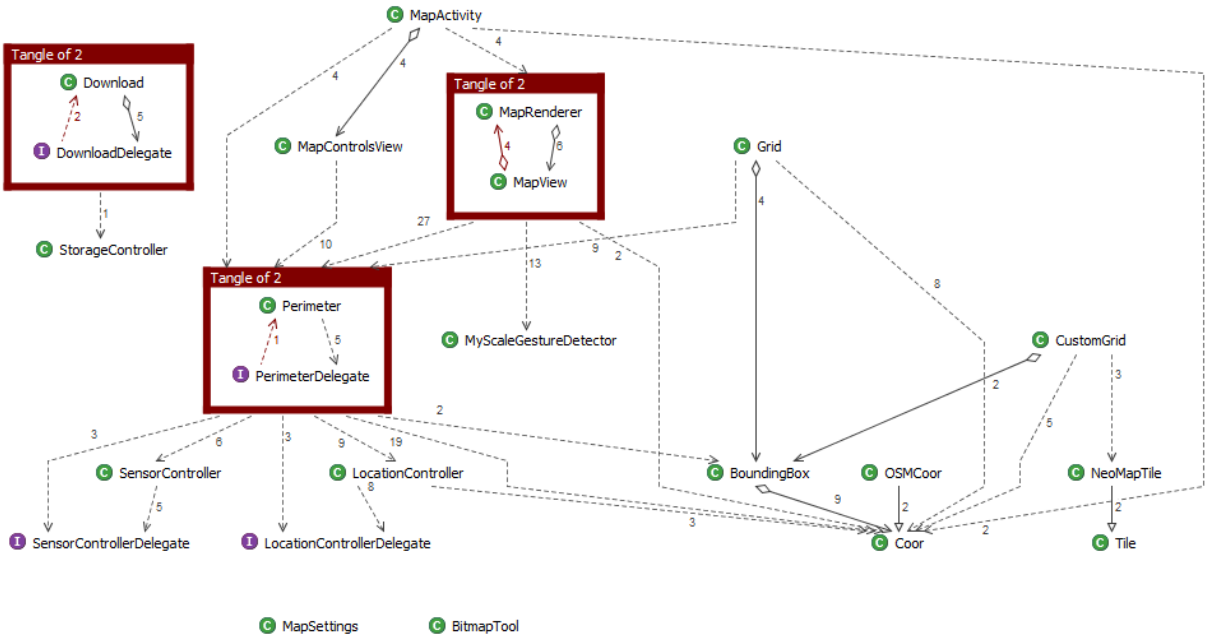


Bild 13: map Packet

layers Packet	osm Packet	neomap Packet

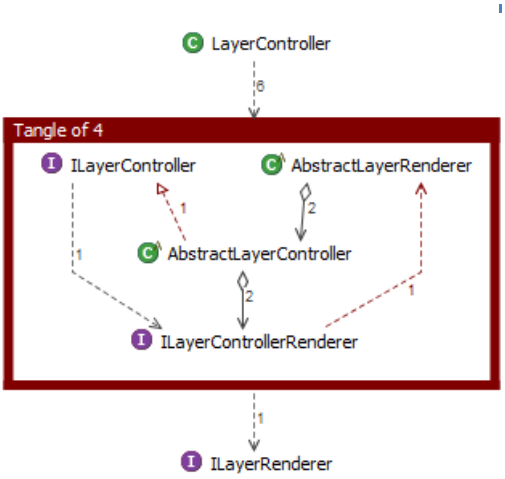


Bild 14: layers Packet

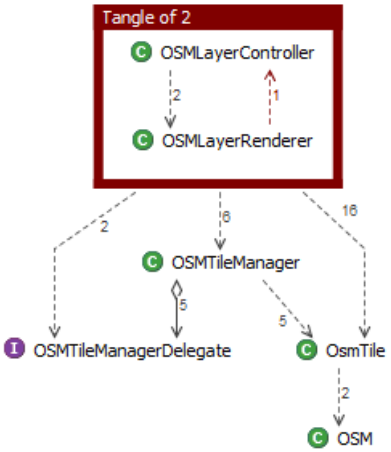


Bild 15: osm Packet

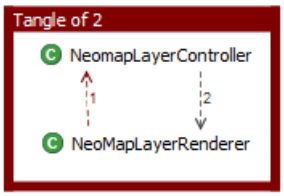


Bild 16: neomap Packet

network Packet

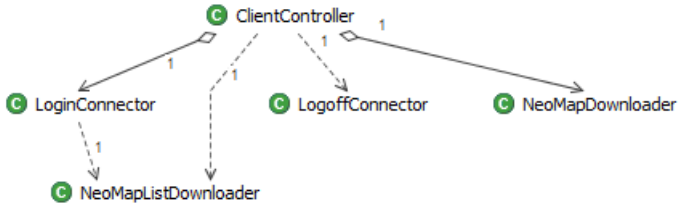


Bild 17: network Packet

Domain Packet

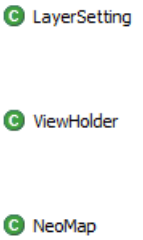


Bild 18: domain Packet

4.2.2 Dependency Graph

Hier noch ein Dependency Graph von der Activity NeoMapSearch. Die Klasse delegiert andere Klassen, die Datenbankanbindungen, Serveranbindungen und Filesystemverbindungen herstellen müssen. Sie ist also eine der komplexesten Klasse. Da die Struktur zu viel zu gross ist wird nur die onCreate Methode gezeigt, welche die beste Übersicht über die Objekte bietet.

Aufbauschema Klassen:

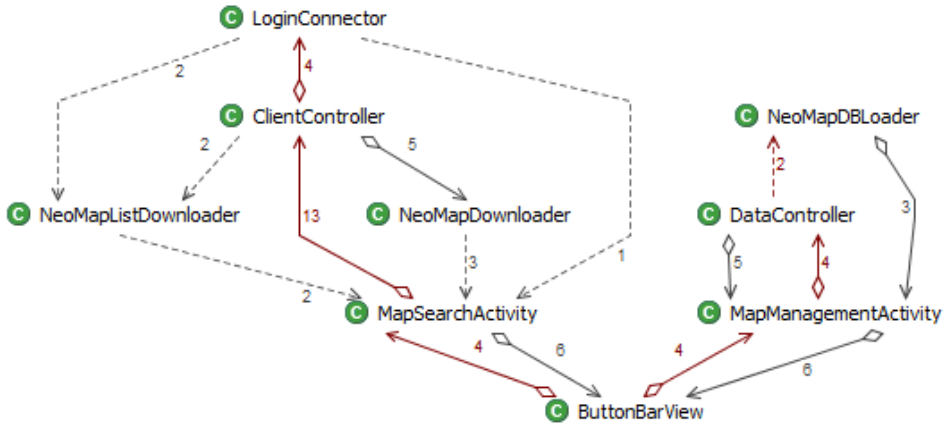


Bild 19: Class Dependency

Aufbauschema onCreate Methode:

Bild 20: onCreate() Method

4.3 Sequenz Diagramme

Da für die interessanten Klassen ein riesiges Sequenzdiagramm notwendig wäre wurde dies vereinfacht dargestellt. Siehe „4.2.3 Dependency Graph".

5. Implementation & Tests

5.1 Implementationsdetails

Da in diesem Sinne nur auf meiner Seite ein Codefreeze vorhanden ist, ist die endgültige Code-Struktur noch nicht fix. Zudem könnte die Testabdeckung könnte noch etwas erhöht werden. Die konnte leider aus Zeitmangel nicht in dem Mass ausgeführt werden wie sie eigentlich gewollt war.

Wie in den Paketstrukturen zu sehen ist, kann man die Klassenabhängigkeiten noch etwas mehr verringern. Jedoch ist es manchmal sehr schwer, da man für gewisse Aktionen immer einen Context benötigt oder verlangt wird.

6. Resultate

6.1 Resultate

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

6.2 Codestatistik

Packages	12
Classes	69
Methods	314
Lines of Code	4661

7. Projektmanagement

7.1 Projektorganisation

Name	Kontaktdaten	Rolle
Keller Stefan	sfkeller@hsr.ch	Auftraggeber und Betreuer
Binna Tobias	vertraulich	Mitarbeiter des IFS, Entwickler NeoMap Server
Lüchinger Dominik	vertraulich	Student, Entwickler der NeoMap App
Recher Patrick	vertraulich	Student, Mitentwickler der NeoMap App

Die Studienarbeit wurde alleine vollzogen. Unterstützt wurde das Projekt von Binna Tobias. Er hat sich um den Serverteil gekümmert.
Ein weiteres Mitglied war Recher Patrick, ein erst Semester Teilzeit-Student, der mir programmierttechnisch unter die Arme gegriffen hat.

7.2 Besprechungen

Damit der Betreuer laufend auf dem neusten Stand bleibt, wurde jede Woche eine Besprechung durchgeführt. In diesen Besprechungen wurden momentane Arbeiten, Problemen und die nächsten Arbeitsschritte besprochen. Falls es von Herr Rechers Seite aus möglich war, wurde jeweils am Dienstag eine Meeting durchgeführt. Dies diente dazu um Arbeiten etwas koordinieren und Neuerungen besprechen zu können. Mit Herrn Binna wurde je nach Bedarf entweder jeweils am Besprechungstag mit Herr Keller oder spontan via E-Mail kommuniziert.

7.3 Projektvorgehen

7.3.1 RUP

Durch das Verwenden von RUP für das Projektvorgehen, konnten gewisse Erfahrungswerte von früheren Projekten mitgenommen werden. Da jedoch noch nie alleine in einem grösseren Projekt gearbeitet wurde musste man sich in jeden Teil nochmals vertiefen. Dabei wurde mit dem Buch *UML2 und Patterns*[8] gearbeitet.

Im Nachhinein betrachtet jedoch, empfiehlt sich eine agilere Vorgehensweise (z.B. SCRUM). Dadurch wird man etwas entlastet vom ganzen Management, welches sich in einer Einzelarbeit zudem nicht aufteilen lässt. Damit erreicht man, dass mehr Zeit in die Software investiert werden kann, anstatt sich mit Dokumentationen zu belasten. Für ein Android Projekt ist es wichtig eine Grobstrukturierung des Projektes zu machen.

7.3.2 Iterationsplanung

Iteration	Beschreibung / Artefakte	Meilenstein	Dauer
Inception	• Vision • Beginn Glossar • Entwickler Tools bestimmen/einrichten • Projekt Management	Definition Anforderungen KW 41	03.10.2011 bis 13.10.2011 (KW 40-41)
Elaboration	• Analyse • Design • Projekt Management • Core Elements • Paper Prototype • Requirement	Anforderungen und Analyse KW 43	14.10.2011 bis 30.10.2011 (KW 41-43)

	Specification		
Construction 1	• Implementation • Prototyp • Ausbau Prototyp • Ausbau GUI	End of Elaboration KW 44	31.10.2011 bis 13.11.2011 (KW 44-45)
Construction 2	• Serveranbindung • Datenbank	Architektur & Design KW 46	14.11.2011 bis 27.11.2011 (KW 46-47)
Construction 3	• Reserve • Implementation zusätzlicher Features	Abklärung zusätzlicher Features KW 48	28.11.2011 bis 11.12.2011 (KW 48-49)
Transition	• Bedienungsanleitung / Hilfe • Webseite / Wiki • Abgabe vorbereiten	Abgabe KW 51	12.12.2011 bis 23.12.2011 (KW 50-51)

7.3.3 Meilensteine

Definition Anforderungen **12.10.2011**

Im ersten Meilenstein wurden die funktionalen Anforderungen definiert. Diese sind unter dem Kapitel „2.1 Funktionale Anforderungen" im Teil II zu finden. Weiter wurde das Logo bestimmt:

Das Logo wurde von Greiter Robert für das Bachelor Projekt „IndoorGuide4Android2"[9] designt.

Es wurde ausserdem entscheiden, dass für die NeoMap App die MIT Lizenz verwendet wird (siehe auch Kapitel „2.5.3 Lizenzanforderungen").
Zum Schluss ist die Iterationsphase Construction in drei Phasen unterteilt worden (C1, C2 und C3).

Anforderungen und Analyse **26.10.2011**

Eine überarbeitete Fassung der Anforderungen wurde vorgelegt. Diese beinhalten nun auch die nichtfunktionalen Anforderungen und die detaillierte Beschreibung der Funktionalen.
Die Use Cases wurden auch überarbeitet und abgeschlossen. Weiter werden Server API und Wishlist vorbereitet. Zu Analyse und informationenzwecken wurde eine Paperprototyp erstellt.

End of Elaboration **04.11.2011**

Im Fokus der End of Elaboration Phase ist der erste Prototyp. Die ersten Menüs der App sind verfügbar und können erste Aktionen auslösen. Der Prototyp der OSM Karte ist leider noch nicht verfügbar.
Die Server API ist fast fertig. Am nächsten Montag werden die wenigen

Verbesserungen dann abgeschlossen sein. Die Datenübertragung erfolgt dann mit JSON.

Architektur & Design	**16.11.2011**

Herr Binna hat mit der Implementierung des Server begonnen. Die SQLite Datenbank der App ist funktionsfähig.

Abklärung zusätzlicher Features	**30.11.2011**

Die zusätzlichen Features müssen erst mal auf die Seite gelegt werden. Es gibt noch viel zu tun bei den Grundfunktionen. Momentan hat Herr Recher die OSM Kartenansicht implementiert. Sie beschränkt sich jedoch vorerst auf die Zoomfunktion. Der Server sollte nächste Woche bereit stehen, sodass erste Verbindungstests von Client – Server stattfinden können. Ein paar Grundfunktionen beim Client wurden bereits vorbereitet und getestet. Sobald der Server online ist, werden Praxistests hoffentlich auch erfolgreich verlaufen. Wir werden das nun so einteilen, dass Herr Reicher an der OSM – Kartenansicht weiter arbeitet und ich alle benötigten Funktionen wie Netzwerkkommunkikation und Datensicherung bereitstelle.

Abgabe	**23.12.2011**

Um 17:20 wird das Projekt abgeschlossen. Nachträglich werden zusätzliche Test nachgeführt.

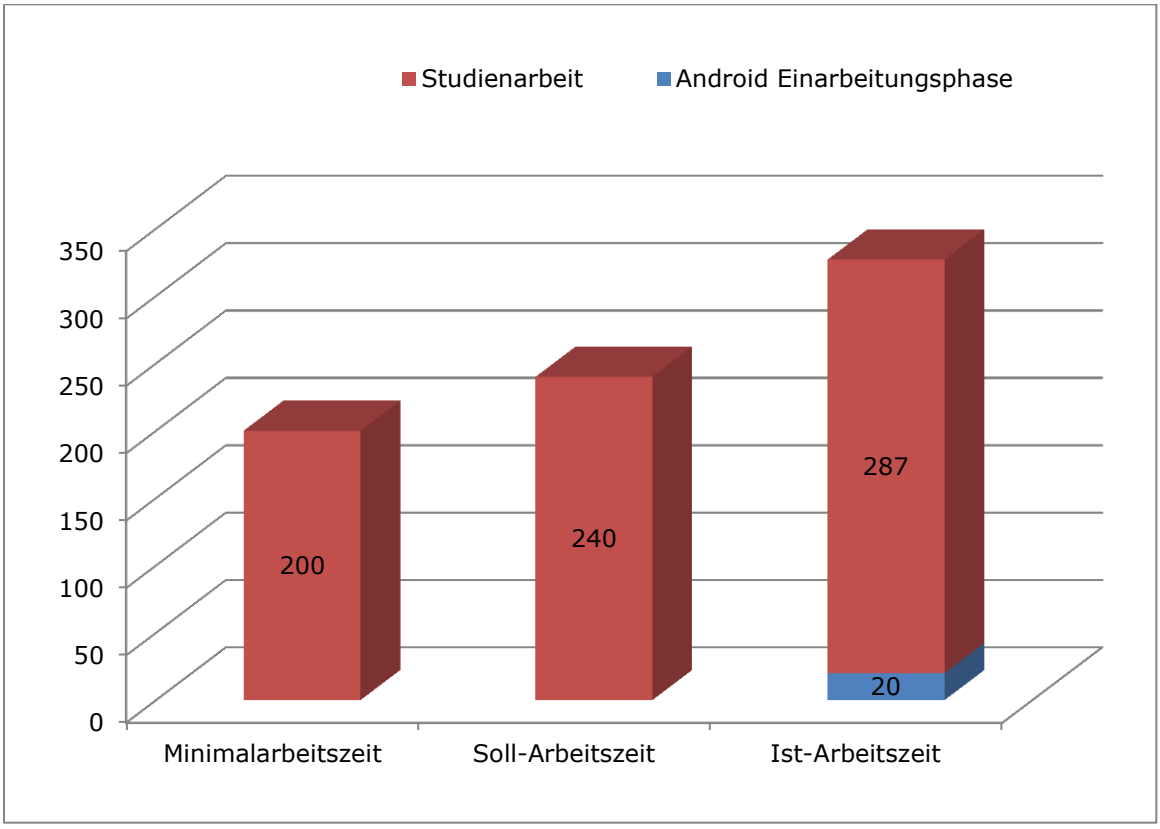
[9]Quelle: eprints3.hsr.ch/119/

7.4 Auswertung der Arbeitszeiten

7.4.1 Arbeitsumfang

Das ECTS[10]-System an der Hochschule Rapperswil fordert, dass pro ECTS-Punkt 30 Arbeitsstunden geleistet werden müssen. Die Rektorenkonferenz der Schweizer Universitäten[11] rechnet den Arbeitsaufwand pro Punkt auf 25-30 Stunden. Bei erfolgreichem Abschluss der Studienarbeit werden 8 ECTS-Punkten gutgeschrieben. Es steht einem zusätzlich zwei Wochen Einarbeitungszeit zur Verfügung, sollte man

Unter der Berücksichtigung dieser Daten ergibt dies also einen Gesamtaufwand von (8 x 30 Stunden =) 240 Stunden. Entsprechend ist dies ein durchschnittlicher Arbeitsaufwand von ca. 17 Stunden pro Woche. Die ersten zwei Wochen wurden genutzt um sich mit der Technologie Android auseinander zu setzen. Diese Einarbeitungszeit ist deshalb getrennt, weil sie in der späteren, eigentlichen Projektplanung nicht mit integriert wurde.



Die Mehraufwand, ohne die Einarbeitungszeit miteinzurechnen, beträgt also 47 Stunden oder +19,6%.

Die der Mehraufwand kommt daher, dass die Verwaltung von Daten und Views der Applikation mehr Aufwand, respektive mehr Funktionen benötigten als zuerst angenommen. Zudem war kein Erfahrungswert vorhanden. Dadurch war es sehr schwer den Aufwand einzuschätzen.
Theoretisch müsste man noch sehr viel mehr Zeit in die App stecken damit am Ende eine optimale Version auf den Markt gebracht werden kann. In der App-Programmierung ist es jedoch gut möglich dies auf spätere Updates zu auslagern. Releast man die App jedoch zu früh, so hat dies negative Auswirkungen durch unzufriedene User-Bewertungen. Deshalb sollte man sich das zuerst gut überlegen.

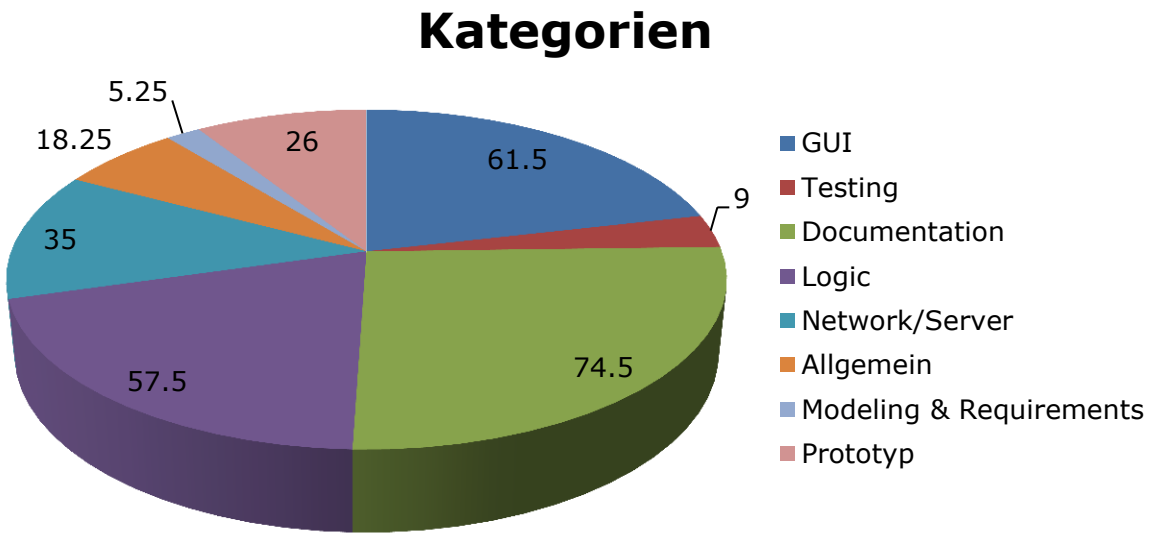
[11]Quelle: www.ects.ch

7.4.2 Übersicht Kategorien und Arbeitspakete

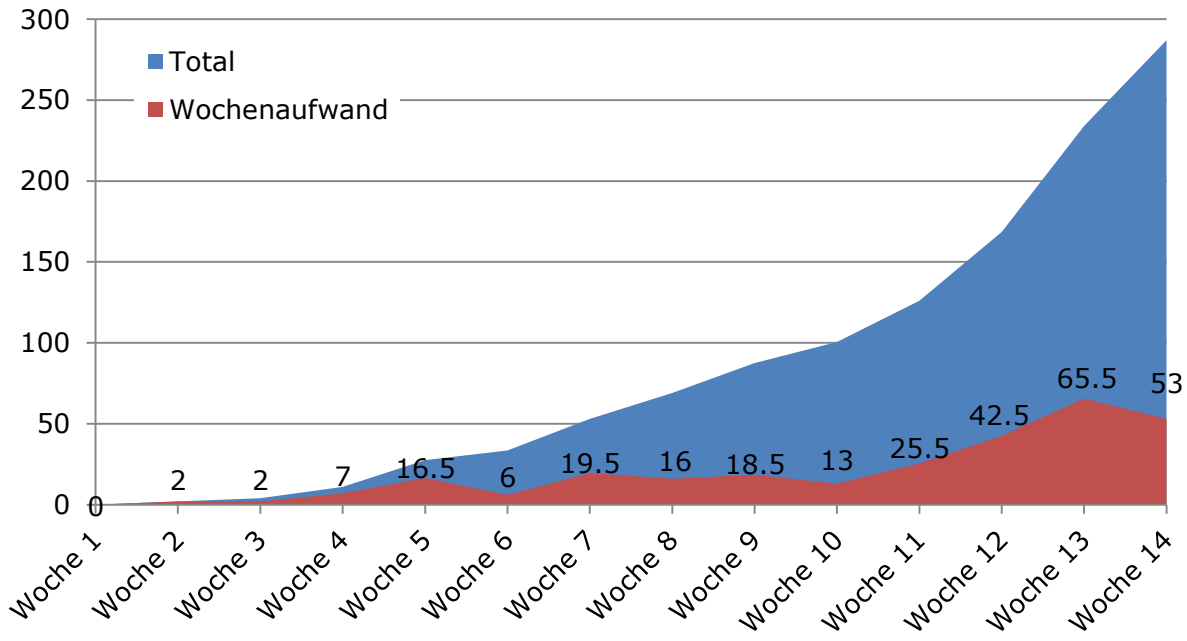
Nachfolgende Grafiken und Listen sind in Stunden angegeben und beinhalten die Einarbeitungszeit nicht.

Kategorie	Ticket	2011-9	2011-10	2011-11	2011-12	Gesamtzahl
GUI				*19.00*	*42.50*	*61.50*
	Feature #52: Ausbau GUI			19.00	42.50	61.50
Testing					*9.00*	*9.00*
	Feature #46: Testing & Refactoring				9.00	9.00
Documentation			*19.50*	*2.00*	*53.00*	*74.50*
	Feature #30: Projektplan		7.00			7.00
	Feature #33: Kapitel ohne Nummerierung		2.50		9.00	11.50
	Feature #38: Requirements		4.50			4.50
	Feature #40: Ziele, Vision		2.00	1.00		3.00
	Feature #41: Entwicklertools		1.50		0.50	2.00
	Feature #43: Zielgruppe & Lizenzanforderung		2.00			2.00
	Feature #48: Wishlist			1.00		1.00
	Feature #61: Dokumenation Teil I				25.50	25.50
	Feature #62: Dokumenation Teil II				18.00	18.00
Logic				*12.00*	*45.50*	*57.50*
	Feature #51: Datenbank			12.00	28.00	40.00
	Feature #55: File Management				17.50	17.50
Network/Server				*26.00*	*9.00*	*35.00*
	Feature #47: Network			26.00	9.00	35.00
Allgemein		*2.00*	*6.75*	*7.50*	*2.00*	*18.25*
	Feature #29: Sitzung 12.10.2011		2.00			2.00
	Feature #31: Sitzung 19.10.2011		1.25			1.25
	Feature #39: Protokollierung (Sitzungen)		1.50	2.00		3.50
	Feature #44: Entwicklertool Management	2.00	2.00			4.00
	Feature #49: Sitzung 09.11.2011			1.00		1.00
	Feature #50: Sitzung 16.11.2011			0.50		0.50
	Feature #53: Sitzung 23.11.2011			2.00		2.00
	Feature #54: Sitzung 30.11.2011			2.00		2.00
	Feature #60: Sitzung 21.12.2011				2.00	2.00
Modeling & Requirements			*5.25*			*5.25*
	Feature #32: Use Cases		3.25			3.25
	Feature #35: Architecture Model		2.00			2.00
Prototyp			*6.00*	*20.00*		*26.00*
	Feature #45: Prototyp (Core Imlementierung)		6.00	20.00		26.00
Gesamtzahl		2.00	37.50	86.50	161.00	287.00

7.4.3 Übersicht Kategorie



7.4.4 Aufwandsverlauf



7.4.5 Risikomanagement

ID	Risiko	Massnahmen	Max. Schaden	W'keit	Gewichtung [h]
R1	Serverausfall (Git & Redmine Fallen aus)	Lokale Versionierung & Zeit-planung	15	5%	2.25
R2	Datenverlust (Schwerwiegende Fehler legen Programme lahm)	Regelmässiges Backup	15	10%	1.5
R3	Ausfall der Hardware (Notebook)	Zusätzlich PC im SA-Raum mit allen Resourcen ausrüsten	20	5%	1
R4	Verplanung	genügend Pufferzeit einrechnen	20	25%	5
R5	Problem beim Umsetzen/Implementierung		35	20%	7
R6	Unfall/Krankheit	vitaminreiche Ernährung	25	10%	2.5
				Total Reserve:	19.25

8. Qualitätsmassnahmen

8.1 Sitzungsprotokolle

Die Wöchentlichen Besprechungen wurden protokolliert. Die Ergebnisse der Sitzung wurden zusammengefasst und können auch noch später nachvollzogen werden. Man findet darüber hinaus auch noch eine Übersicht der Gesprächs-themen und die beteiligten Personen.

8.2 Iteratives Vorgehen

Durch die iterative Planung mit RUP ist es möglich das Projekt in kleinere Etappen zu unterteilen. Durch diese kleineren Zieletappen können Abweichungen schneller erkannt und Fehler minimiert werden.

8.3 Einsatz eines Versionierungssystem

Der Einsatz mit dem GIT Versionierungssystem bietet mehrere Vorteile. Zum einen kann man das Projekt auf einem externen Server backupen. Zum anderen kann man auch Offline commiten. Damit ist auch eine Lokale Versionierung möglich. Durch die Versionierung kann man auch auf ältere Commits zugreifen.

8.4 Testing

Um ein hohe Codequalität garantieren zu können braucht es Tests. Dabei beziehen sich die Test vor allem auf die Logik des Programms. Siehe „5.2 Implementationsdetails“ für mehr Informationen.

8.5 Coderichtlinien

Für die Weiterentwicklung ist es wichtig, dass der Code leserlich ist. Darum wurde auf eine logische Namensgebung und Codestruktur geachtet. Für die Formatierung des Codes wurden die standardmässigen Eclipse Einstellungen verwendet.

9. Anleitungen und Tutorials

9.1 Einführung in die Applikation

Um einen kurzen Einblick in die App zu gewähren, ist hier die Struktur der GUI-Elemente dargestellt.

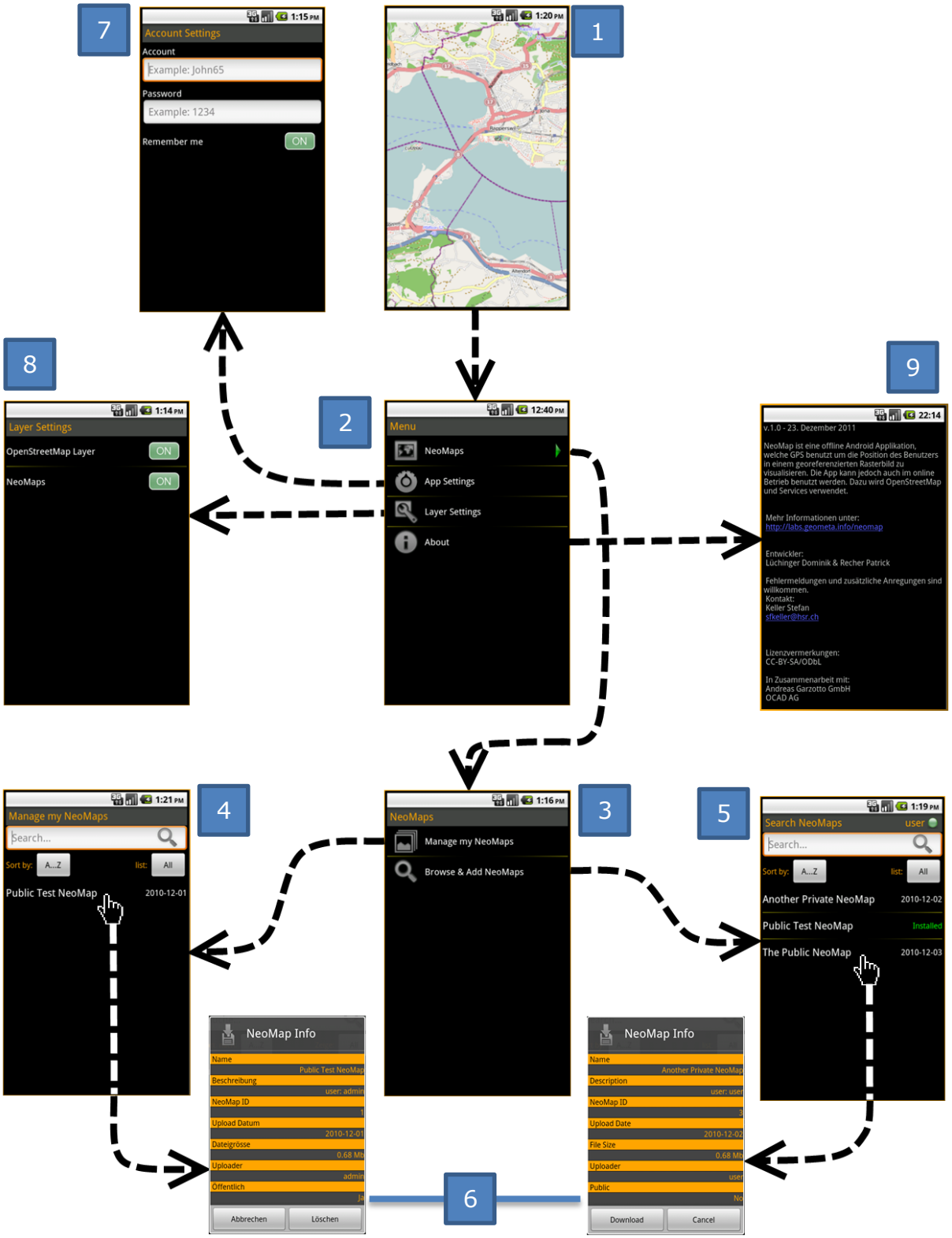
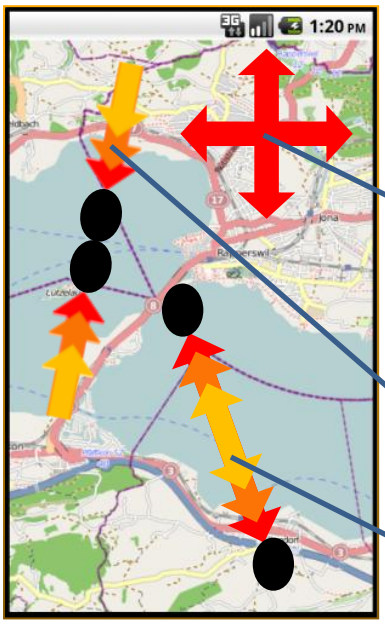


Bild 21: NeoMap GUI-Elemente

Nr.	Beschreibung
1	**Map View:** Hier wird im Hintergrund der OSM Layer dargestellt. Darüberliegende Layer zeigen POIs und NeoMaps.
2	**Main Menu:** Das Hauptmenu bietet die ersten Auswahlmenüs in der Verwaltung der App.
3	**NeoMap Menu:** Im zweiten Menü kann man zwischen dem Verwalten seiner NeoMaps oder dem Suchen neuer NeoMaps entscheiden.
4	**NeoMap Management:** NeoMaps, welche man bereits heruntergeladen hat, können hier verwaltet werden.
5	**NeoMap Search:** Man kann hier neue NeoMaps downloaden. Sind NeoMaps bereits im Besitz, so werden sie mit „Installed" markiert.
6	**Info Dialog:** Durch klicken auf die NeoMap Listenelemente können zusätzliche Aktionen und Informationen angezeigt werden.
7	**App Settings:** Durch Eingabe von Benutzername und Passwort wird der Benutzer automatisch bei **[4]** eingeloggt. Vorausgesetzt er hat die „Remember me" Funktion aktiviert.
8	**Layer Setting:** Hier können Grundfunktionen der Kartenansicht eingestellt werden. Wird der OSM Layer deaktiviert, werden keine OSM Tiles mehr geladen. Damit wäre in der **[1]** der Offline-Modus aktiv.
9	**About:** Für weitere Informationen über die App kann das About aufgerufen werden.

9.2 Map View



Auf der Kartenansicht kann folgendermassen agiert werden:

Navigation von nach links, rechts, oben, unten erfolgt mit dem ziehen des Fingers in die entgegengesetzte Richtung. Die Karte verhält sich gleich, wie wenn man ein Blatt Papier vor sich hin schieben würde.

In die Karte kann rausgezoomt werden. Indem man zwei Finger auf einen Punkt zu bewegt.

In die Karte kann reingezoomt werden. Indem man zwei Finger von einem Punkt aus weg bewegt.

9.3 Georeferenzierte Karten

Um Offline-Karten in der App richtig anzeigen zu können müssen sie georeferenziert sein. Georeferenziert heisst, dass ein Karten-, oder Bildausschnitt mit den dazu passenden Koordinaten versehen wird.

Eine Einführung in die Georeferenzierung findet man in diesem Tutorial[11]. In diesem Tutorial wird ArcGIS[12] verwendet. Es gibt jedoch auch noch andere Möglichkeiten.

Eine fertige georeferenzierte Karte[13] sieht dann zum Beispiel so aus:

Bild 23: Georeferenzierte Karte

[11]Tutorial: www.youtube.com/watch?v=xVVdZOQiBuQ
[12]ArcGIS: www.esri.com/software/arcgis/arcgis10/index.html
[13]Karte von: OCAD AG - www.ocad.com/en/index.htm

Anhang

1. ANHANG A: Inhalt der CD

1.1 Applikation

- NeoMap.apk
- Sourcefolder

1.2 Dokumentation

- Sitzungsprotokolle
- NeoMap Bericht.docx
- NeoMap Bericht.pdf

2. ANHANG A: Glossar und Abkürzungsverzeichnis

Begriff	Erklärung
Android	Ein Betriebssystem/Softwareplattform für mobile Geräte.
NeoMap	So wird die offline Karte für diese App bezeichnet. Sie ist nach Norden ausgerichtet und georeferenziert in einer .kml Datei vorhanden.
.kml/.kmz (-Datei)	Keyhole Markup Language - Ist ein Dateiformat, welches Geodaten beinhaltet.
RUP	Rational Unified Process - Ist ein Vorgehensmodell der Softwareentwicklung.
SQLite	Ist eine Programmbibliothek, welche ein relationales Datenbankmodell enthält.
Use Case	Ein Anwendungsfall, der Szenarien beinhaltet, welche eintreten können, wenn ein Akteur mit dem System ein Ziel zu erreichen versucht.
App	Applikation – Meistens werden Smartphone-Programme so genannt.
GPS	Global Positioning System – Ist ein globales Navigations-system zur Positionsbestimmung.
MIT	Massachusetts Institute of Technology – Ist eine Tech-nische Hochschule und Universität in Cambridge (USA).
HSR	Hochschule für Technik in Rapperswil
IFS	Institut für Software an der HSR
OSM	OpenStreetMap – Open Source Dienst
Bounding Box	Beschreibt ein geographisches Rechteck, bestehend aus je zwei X-Koordinaten und zwei Y-Koordinaten.
JSON	Javascript Object Notation ist ein einfach lesbares Daten-format mit wenig Overhead.

Layer	Mit Layer ist eine virtuelle Ebene in der Z-Achse gemeint.
Latitude	Geographische Breitenbezeichnung der Erde
Longitude	Geographische Längenbezeichnung der Erde
Paperprototype	
Overhead	Zusatzinformationen zu den Daten
Parsen	
POI	Point of Interest – Orte welche ein gewisses Interesse aufweisen.
RFID	Radio Frequency Identification – Besteht aus einem passivem Microschip und einem Lesegerät und wird eingesetzt um einen Gegenstand mit Informationen zu versehen.
ECTS	Ein International Leistungssystem für Hochschulstudenten
SQLite	Programmbibliothek für eine relationale Datenbank
Use Case	Ist ein Anwendungsfall, der mögliche Szenarios zeigt, die eintreten können.
GUI	Graphical User Interface – Wir entspricht zusammen-fassend den grafischen Komponenten einer Software.
XML	Extensible Markup Language – Verbreitete Auszeichnungssprache mit hierarchischer Darstellung
SCRUM	Agiles Verfahren zum Entwickeln von Software
Tutorial	Gebrauchsanweisung eines Computerprogramms
commiten	Ein GIT Befehlt, der die Daten lokal manuell versioniert
GSM /UMTS	Standards bei der Mobilfunkübertragung

3. ANHANG B: Literatur- und Quellenverzeichnis

3.1 Bücher und Artikel

1	Prof. Stefan Keller, „Zitat der originalen Aufgabenstellung für die Studienarbeit", 2011
2	Thomas Künneth, „Android 3", 2011, ISBN: 978-3-8362-1697-5
8	Craig Larman, „UML2 und Patterns", ISBN: 978-3-8266-1453-8

3.2 Links und Informationen

3	**Alternativen zu GPS:** knol.google.com/k/alternative-ortungsm%C3%B6glichkeiten-zu-gps-speziell-auch-in-outhouse-lokalisierung#
4	**IndoorWPS:** gis.hsr.ch/wiki/IndoorWPS
5	**MIT-Lizenz:** de.wikipedia.org/wiki/MIT-Lizenz
6	**Android Guidelines:** developer.android.com/guide/practices/ui_guidelines/index.html
7	**Acitivty Lifecycle:** www.androidjavadoc.com/1.0_r1_src/android/app/Activity.html
9	**IndoorGuide4Android2:** eprints3.hsr.ch/119/
10	**Rektorenkonferenz der Schweizer Universitäten:** www.ects.ch
11	**Tutorial:** www.youtube.com/watch?v=xVVdZOQiBuQ
12	**ArcGIS:** www.esri.com/software/arcgis/arcgis10/index.html
13	**OCAD:** www.ocad.com/en/index.htm

3.3 Weitere Quellen

q1	**Handy Nutzung Statistik:** en.wikipedia.org/wiki/List_of_countries_by_number_of_mobile_phones_in_use
q2	**Android Market Statistiken:** www.androlib.com/appstats.aspx
q3	**GoogleMaps:** market.android.com/details?id=com.google.android.apps.maps&hl=de
q4	**MapDroyd:** www.androidpit.de/de/android/market/apps/app/com.osa.android.mapdroyd/MapDroyd
q5	**Maps (-):** market.android.com/details?id=coderminus.maps&feature=search_result#?t=W251bGwsMSwyLDEsImNvZGVybWludXMubWFwcyJd