HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Reisebegleiter App

TravelGuide

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2012

Autoren: Dominik Lüchinger, Nicolas Bigler
Betreuer: Prof. Dr.-Ing. Andreas Rinkel
Experte: Dr. Andreas Jarosch, Swisscom Innovations
Gegenleser: Prof. Eduard Glatz

Abstract

Heute existieren bereits reichlich kleine Helfer oder Apps. Diese erleichtern die Reiseplanung und helfen dem Benutzer sich im Alltag zurechtzufinden. Die meisten Reisebegleiter sind auf regionale Gebiete beschränkt oder bieten nur eingeschränkte Möglichkeiten (z.B. nur Navigation, Fahrplansuche).
Die Frage ist jedoch, wie der Benutzer über Ereignisse und Abläufe seiner Reise auf dem Laufenden gehalten wird.
Die Antwort darauf bietet die Android App TravelGuide. Sie macht es sich zum Ziel, den Benutzer mittels automatisierten Überwachungsprozessen und unterstützenden Features zu entlasten. Die App ermöglicht es eine persönliche Reise zu erstellen. Jede Reise kann eine oder mehrere Etappen beinhalten und jede Etappe wiederum kann mehrere ÖV-Verbindungen und POIs besitzen. An die benötigten Informationen kommt man in der App über verschiedene Schnittstellen, wie SBB oder Google Maps. Die erfassten Daten können übersichtlich in einem Diagramm angezeigt werden und bieten benutzerdefinierte Erinnerungseinstellungen sowie weitere nützliche Features. Mit den Karten von Google Maps ist es möglich Positionen und Routen zu einem POI oder der nächsten Haltestelle auf der Karte anzeigen zu lassen. Bei aktivierter Reise werden alle dazugehörenden Verbindungen in regelmässigen Intervallen automatisch überprüft. Sollten irgendwelche Angaben nicht mehr aktuell sein, wird der Benutzer umgehend mittels einer Benachrichtigung informiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
INFORMATIK

Aufgabenstellung zur Bachelorarbeit FS 2012

"Entwicklung und Implementierung einer Reisebegleit-App"

Gruppe: Nicolas Bigler / Dominik Lüchinger

Ausgangssituation

Heute existieren bereits reichlich kleine Helfer oder Apps, die die Reisplanung erleichtern und helfen die richtige Verkehrsverbindung zu finden. Aber dann, wenn es losgeht wird man wieder alleine gelassen. Im Falle von geänderten Anschlussverbindungen wird der Reisende nur über Lautsprecheransagen in den entsprechenden Verkehrsmitteln und Bahnhöfen informiert. Daher erhält der Reisende die wichtige Informationen vielfach erst in letzer Minute. Die offiziellen Fahrplan-Apps ermöglichen es zwar dem Benutzer jederzeit die Verbindungen zu überprüfen, jedoch fehlt die Möglichkeit über Änderungen und Ausfälle informiert zu werden.

Aufgabe

Ziele der zu entwickelnden App ist es, den Reisenden auf der Fahrt zum Ziel ständig zu begleiten. Nach Übernahme der Reiseplandaten wird der Reisende zum Beispiel erinnert, wann er starten muss, er erhält nach Bedarf Informationen, ob er noch im Zeitplan ist oder sich die Anschlussverbindungen geändert haben – und das unabhängig vom jeweiligen Verkehrsmittel. Alternative Anschlussverbindungen sollen ebenfalls nach Bedarf angezeigt werden. Der Benutzer soll die Möglichkeit haben, eine Wegbeschreibung zur gewünschten Station oder einem POI anzeigen zu lassen.
Die App wird für Android Smartphones entwickelt.

Termine

20. Februar 2012	Arbeitsbeginn
15. Juni 2012	Ababe des Berichts an den Betreuer

Betreuung

Betreuer: Prof. Andreas Rinkel, Email: arinkel@hsr.ch
Während der Durchführung der Arbeit findet nach Möglichkeit regelmässig jede Woche eine Besprechung mit dem Betreuer statt. Dazu werden entsprechende Termine bei Arbeitsbeginn festgelegt.

Rapperswil, den 20. Februar 2012

Andreas Rinkel

Management Summary

Ausgangslage

Smartphones werden immer beliebter in der Bevölkerung. Ein Grund dafür ist die einfache Erweiterbarkeit des Funktionsumfangs mit Hilfe von sogenannten Apps. Für viele Aufgaben des täglichen Lebens gibt es schon Applikationen, welche die Anwender unterstützen oder ihnen Arbeit abnehmen.
Auch für das Reisen sind bereits diverse Apps auf dem Markt. Einige sind Planer für Reisen, Guides für diverse Städte. Andere zeigen Fahrpläne für ÖV an oder erlauben die Suche nach POIs. Dies sind aber meistens statische Anwendungen, die nicht auf sich ändernde Umstände reagieren. Andere wiederum beschränken sich auf ganz bestimmte Regionen (z.B. eine Stadt).
Diese Einschränkungen soll die TravelGuide App beseitigen.

Vorgehensweise

Bei dieser Arbeit wird mit einer agilen Vorgehensweise gearbeitet. Die Analyse und Ausarbeitung der Anforderungen in der Planungssphase erfolgt mit Event- Driven Architecture. Das Projekt hat keinen definitiven Endzustand. Daher ist Scrum mit dem empirischen, iterativen und inkrementellen Ansatz ideal als Vorgehensmodell für die Softwareentwicklung geeignet.
Die Applikation ist für Android entwickelt. Dies aus dem einfachen Grund, da Android eine sehr einfache, weit verbreitete Plattform ist und zudem kostengünstige Bedingungen bietet.

Resultate

Das Ergebnis dieser Projektarbeit stellt einen stabilen Prototyp einer Reisebegleiter App dar. Der Prototyp enthält folgende Funktionalität:

- Erstellen von Reisen und Etappen
- Suchen und Speichern von ÖV-Verbindungen
- Suchen und Speichern von POIs
- Anzeigen einer zeitlichen Übersicht aller Verbindungen und POIs einer Reise
- Automatisches Überprüfen der gespeicherten Verbindungen
- Benachrichtigung bei geänderten Verbindungen
- Anzeigen von POIs und Stationen auf der Karte
- Anzeigen von Routen vom aktuellen Standpunkt zu einem POI oder einer Station

Bewertung

Da sich im Android Play Store die vorhandenen Produkte teils stark voneinander unterscheiden, ist es schwierig eine Bewertung aus Kundesicht einzubringen. Die App ist in sich einzigartig und kann sich gut mit anderen Apps messen.

Kosten

Die App benutzt die folgenden externen Schnitstellen:

- Google Maps Library

- transport.opendata.ch OpenSource JSON-Schnittstelle, SBB-Fahrplandaten
- Google Places API, POI-Suche
- geonames.org Städtedatenbank, Auswahl einer Stadt für die POI-Suche

Die einzigen angefallen Kosten belaufen sich auf die 25 USD des Developer-Accounts von Android. Diese wird für den Google Maps API-Key und Google Places API-Key benötigt. Für den Android-Benutzer fallen nebst den Internetkosten keine weiteren Kosten an. Bei der Anwendung im Ausland sollte man sich jedoch im Klaren sein, dass zusätzliche Roamingkosten anfallen können.

Zusammenfassung & Ausblick

Das stärkt die Kompetenzen und das Know-How in den folgenden Gebieten:

- JSON, Kommunikation mit Server für Verbindungsabfragen
- LaTeX
- Google Maps
- Mobiles Betriebssystem - Android

Mögliche zukünftige Erweiterungen für die Applikation sind:

- GPS-History aufzeichnen und anzeigen
- Sharing von Reisedaten (App2App, Email, NFC, etc.)
- Verknüpfung der einzelnen Elemente einer Reise (Validierung ob keine Überschneidungen vorhanden sind)

Technischer Bericht

1. Einführung in die App ”TravelGuide”

1.1. Problemstellung

Im Jahr 2011 wurden weltweit über 1‘774 Millionen Mobiltelefone verkauft. Gegenüber 2009 entspricht dies einem Zuwachs von über 45% (Gartner, 2012b). Betrachtet man dies aus der Sicht der verkauften Android Geräte kommt man im 1. Quartal 2012 auf einen Marktanteil von 56,1%. Die Entscheidung von Google das Android Betriebssystem den Herstellern kostenlos zur Verfügung zu stellen, stösst also auf grosse Beliebtheit (Gartner, 2012a).
Dadurch steht den Entwicklern ein sehr breites und grosses Zielpublikum zur Verfügung. Im Gegensatz zu den kauffreudigen Apple Kunden liegt das Zielpublikum bei den Google Play Kunden mehrheitlich bei den Gratis Apps.
Bei den Reiseführer/-Planer im Google Play trifft man häufig auf spezifische oder ortsgebundene Apps. Ein Ziel der App ist es diese Lokalitäten zu durchbrechen um eine möglichst vielseitige Plattform zu bieten.
Bei Reisen im In- und Ausland müssen jedoch ständig allfällige Zusatzgebühren im Auge behalten werden. Dazu zählt zum Beispiel die Roaming Gebühr im Ausland.
Aus Sicht der Performance sollte eine Applikation nicht überflüssig viel Akku verbrauchen. Sind also GPS oder andere stromintensive Ressourcen im Spiel, müssen diese mit Bedacht eingesetzt werden.

1.2. Rahmenbedingungen

1.2.1. Administratives

1.2.2. Bereitgestellte Gerätschaften

Für die Durchführung der Bachelorarbeit stehen folgende Geräte zur Verfügung:

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

Tabelle 1: Arbeitsrechner

Google Nexus S		
Prozessor	1GHz Hummingbird-Prozessor	
Netwzerk	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	

Tabelle 2: Google Nexus S

1.2.3. Entwicklungsumgebung und Tools

Für die Entwicklung der Applikation werden diverse Anwendungen eingesetzt (siehe Tab. 3 auf Seite 4).

Logo	Software / Version	Einsatzzweck	Beschreibung
	Android ADT 18.0.0	Mobile App Entwicklung	Eclipse-Plugin für die Android Development Tools Quelle: developer.android.com

Weiter auf der nächsten Seite

Tabelle 3 – Fortsetzung von vorheriger Seite

Logo	Software / Version	Einsatzzweck	Beschreibung
	Android SDK 2.2	Entwicklung	Software Development Kit. Quelle: developer.android.com
eclipse	Eclipse Indigo 3.7.2	Entwicklungs-umgebung	IDE für die Entwicklung in Java 7. Quelle: www.eclipse.org
	Agilo	Zeiterfassung und Projektverwaltung	Webbasiertes Projektmanagement Tool. Quelle: agiloforscrum.com
GIT	Git 1.7.X	Versionierung	Versionsverwaltungs-Software für das Projekt. Quelle: www.git-scm.com
SQLite	SQLite 3.7.7.1	Datenbank	Von Android unterstützte Programmbibliothek. Quelle: www.sqlite.org
Paint.net	Paint.NET 3.5.8	Bildbearbeitung	Bildbearbeitungs-software Quelle: www.getpaint.net
	GIMP 2.6.12	Bildbearbeitung	Bildbearbeitungs-software Quelle: www.gimp.org
TEX/ LATEX	Tex 3.1415926	Textsatzsystem	Software zur Erstellung von Berichten, Dokumenten und Bücher. Quelle: www.latex-project.org

Weiter auf der nächsten Seite

Tabelle 3 – Fortsetzung von vorheriger Seite

Logo	Software / Version	Einsatzzweck	Beschreibung
	Hudson 2.2.0	Buildserver	Serverapplikation zum automatisierten Testen der Software Quelle: www.hudson-ci.org
	Sonar 3.0	Software-Qualitätsmanagement Plattform	Serverapplikation zur automatisierten Codeanalyse. Quelle: www.sonarsource.org

Tabelle 3: Entwicklungsumgebung und Tools

2. Vision der ”TravelGuide”-App

Ziel dieser Arbeit ist es, eine Android Applikation zu erarbeiten, die eine Reise über mehrere Etappen planen und verwalten kann und mit Unterstützung von Google Maps und SBB Daten die Suche von POIs

3. Verfügbare Android Reise-Apps

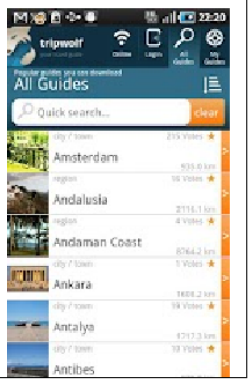
Im Google Play Store finden sich zahlreiche Auswahlmöglichkeiten unter den Reiseführern und den Reiseplanern. Einige beschränken sich auf Benutzereingaben, andere auf vorgefertigte Informationen. Um einen kleinen Einblick in dieses Segment zu erhalten, werden hier ein paar Beispiele angesprochen.

3.1. TravelPlaner

Logo	Beschreibung
	Find your journey with public transport in the western part of Sweden. Our travel planner finds the smartest trips for you. Features are for example: • Plan yor trip between two addresses/stops. • Locate the nearest stop via gps. • Easy presentation of the trips. • Trip presentation on a map. • Price and sms-codes. • Accessibility information. • Disturbance information and new trips suggestions if your trips is not possible due to delays (Real time information is not available everywhere). • Timetable for stops. • Gadgets for stops and trips.
Info	Kategorie: Verkehr Downloads: 100.000-500.000 Bewertungsdurchschnitt: 3.8/5 Preis: gratis Für Android Version 2.1 und höher, unterstützt App2SD
Bericht	
Mit dieser App lassen sich Fahrpläne von öffentlichen Verkehrsmitteln suchen. Die Fahrpläne sind strukturiert aufgebaut und verfügen über zusätzliche Informationen zu den einzelnen Verbindungen. Unterstützt mit Google Maps lässt sich die Route mit den Zwischenstationen auf der Karte darstellen. Die App verfügt auch über eine POI Suche, mit NearBy-Funktion. Der Einsatzbereich begrenzt sich auf Schweden (Västtrafik, 2012).	

Tabelle 4: TravelPlaner App

3.2. Tripwolf

Logo	Beschreibung
	Keine Roaming-Kosten im Ausland: Maps und Bilder können downgeloadet werden. Entdecke die schönsten Städte der Welt mit tripwolf, deinem Reiseführer. Folge den Empfehlungen anderer Reisender und hole dir Reiseinfos aus “Marco Polo”- und Footprint”-Reiseführern. Inklusive Stadtplänen, Augmented Reality Viewer, kostenlosen Updates und GRATIS Guide für Amsterdam.
Info	Kategorie: Verkehr Downloads: 100.000-500.000 Bewertungsdurchschnitt: 3.2/5 Preis: Basic: gratis / Premium 6.05Fr Für Android Version 1.5 und höher
Bericht	
Die App bietet diverse Städteinformationen an. Zum einen bekommt man zu Sehenswürdigkeiten jegliche benötigte Informationen, wie zum Beispiel: Adresse, Öffnungszeiten, Preise und noch vieles mehr. Die Travel Guides müssen separat heruntergeladen werden und sind ca. 200Mb gross. Man kann jedoch auch Fotos und Kartenmaterialien separat laden. Für eine offline Verfügbarkeit und weitere Städte benötigt man einen Premium Account (tripwolf, 2012).	

4. Vorgehensweise

In diesem Projekt wird mit einer agilen Vorgehensweise gearbeitet. In der Planungsphase erfolgt mittels Event-Driven Architecture die Analyse und Ausarbeitung der Anforderungen der Software. Als späteres Vorgehensmodell wird Scrum verwendet.

4.1. Event-Driven Architecture

Die Informationen in diesem Kapitel sind eine Zusammenfassung aus (Bruns u. Dunkel, 2010).

Event-Driven Architecture (EDA) ist ein Architekturstil bei dem Ereignisse ins Zentrum der Softwarearchitektur rücken.
Bei der Softwareentwicklung mit EDA wird also viel Wert auf Ereignisse gelegt. Die Konzentration auf Ereignisorientierung besitzt das Potential die Architektur der Anwendung agiler, reaktionsschneller und echtzeitfähig zu werden.
EDA ist vorallem bei Anwendungen sinnvoll die von Ereignissen abhängig sind oder von Ereignissen gesteuert werden.

4.1.1. Historischer Kontext

Event-Driven Architecture hat sich in den letzten Jahren zu einem selbstständigen Fachgebiet der Ereignisverarbeitung etabliert. Ereignisse spielen aber in der Informatik schon seit vielen Jahren eine Rolle.
Ereignisbasierte Konzepte wurden unabhängig voneinander in unterschiedlichen Bereichen entwickelt. Einige Beispiele dabei sind:

- Diskrete Ereignissimulation
  Die Simulation erzeugt aus Eingabendaten Ereignisse, die die Interaktion zwischen Teilkomponenten nachbilden.
- Computer-Netzwerke
  Das Senden und Empfangen von Datenpaketen stellen jeweils ein signifikantes Ereignis dar.
- Aktive Datenbanken
  Erweitern traditionelle Datenbankmodelle um Techniken zur unmittelbaren Reaktion auf Datenänderungen.
- Softwarearchitektur
  Der Austausch von Ereignissen hat sich in Form von Nachrichten als *Kommunikationsstil* zwischen verteilten Systemkomponenten etabliert.

4.1.2. Bedeutung von Ereignissen

Jeder Vorgang, jede Aktivität und jede Entscheidung in der realen Welt oder in einem Softwaresystem, der bzw. die Auswirkungen auf die betrieblichen Prozesse nach sich ziehen, stellt ein Ereignis dar.

4.1.3. Ereignismodelle

Ereignisse sind das zentrale Konzept und die Basis einer EDA.
Diese werden in unterschiedliche *Abstraktionsebene* unterteilt:

- Physikalische und technische Ereignisse
  Diese Ereignisse beruhen auf Beobachtungen physikalischer Beobachtungen
- Anwendungs- oder Geschäftsereignisse
  Diese Ereignisse spiegeln einen fachlichen Sachverhalt wider und werden durch die Anwenungsyssteme ausgelöst

4.2. Scrum

Die Informationen aus diesem Kapitel sind eine Zusammenfassung aus (Wirdemann, 2009).

Scrum ist ein Framework um komplexe Projekte, welche nicht exakt vorhersagbar sind zu verwalten. Bei Scrum sind die Anforderungen nicht schon zu Beginn des Projektes exakt definiert werden. Schwerfällige Managementstechniken wie zum Beispiel Gantt-Chart-basierte Projektplanung, Detailspezifikationen oder aufwändiges Statusreporting fallen weg. In Scrum konzentriert man sich auf wichtige Anforderungen und arbeitet priorisiert. Anforderungen lassen sich während der Entwicklung agil anpassen, ohne ständig die Dokumentation anpassen zu müssen.

4.2.1. Rollenverteilung

Team

Das Entwicklerteam zusammen mit dem ScrumMaster und dem Product Owner stehen im Zentrum des Geschehens und bilden das Scrum-Team. Eine Rollenverteilung wie Chef oder Projektleiter gibt es nicht. Basis bildet eine gute Kommunikation und gegenseitiges Vertrauen.

ScrumMaster

Der ScrumMaster hat die Aufgabe das Team zum Erfolg zu verhelfen. Seine Hauptaufgaben sind dafür zu sorgen, dass Scrum funktioniert und das Team optimale Arbeitsbedingungen hat. Er schützt das Team vor negativen Einflüssen und Hindernissen.

Product Owner

Der Product Owner schreibt User Stories und verwaltet User Stories im Backlog. Nur er darf neue Stories hinzufügen. Seine Aufgabe besteht darin zu entscheiden, welche Stories im Product Backlog am wichtigsten sind und sie entsprechen zu priorisieren. Er beschliesst und überlegt sich auch was für die Basisnutzung der Anwendung mindestens notwendig ist und identifiziert die ”Must-Have”-Stories, welche für die Anwendung einen elementaren Wert darstellen. Danach folgt die Verarbeitung der ”Should-Have”-Stories, welche einen grossen Mehrwert für die Anwendung liefern. Die ”Must Have”-Stories und ”Should-Have”-Stories werden sinnvoll getrennt in einer Reihenfolge gegliedert.
Aus diesen Stories wird danach eine Aufwandsschätzung durchgeführt. Hierbei wird das Entwicklerteam als Experte miteinbezogen. Dabei ist wichtig, dass die Bewertung nicht die Entwicklungsdauer sondern die Grösse (im Verhältnis zu den anderen Stories) angibt. Die Stories werden einer Grösse von drei Story Points zugeteilt. Der Vorteil wird daraus ersichtlich, dass es so sehr viel einfacher ist die einzelnen Stories in Grössen zu unterteilen, als zu schauen wie lange etwas dauert.

4.2.2. User Story

User Stories sind Anforderungen in der Sprache des Kunden beschrieben. Das heisst, sie liefern für den Kunden einen konkreten und sichtbaren Mehrwert. Der Detailwert einer Story nimmt zu, je näher sie an ihre konkrete Umsetzung rückt. Für die Beschreibung der User Stories hat sich folgendes Muster bisher bewährt:

Als <Benutzerrolle> will ich <das Ziel> [,so dass <Grund für das Ziel>].

Bei der Erarbeitung werden die User Stories auf einer Story-Karte beschrieben. Diese Karten werden im Team besprochen und entsprechend mit Details ergänzt. Mittels Akzeptanzkriterien werden die wesentlichsten Aspekte einer Story festgehalten.

Epics

Epics sind User Stories, welche noch nicht konkretisiert wurden und werden benutzt um grosse Teile des Systems schnell zu beschreiben. Häufig sind Epics auch grosse User Stories, welche noch nicht geschätzt und heruntergebrochen wurden. Sie müssen bevor sie in einen Sprint eingebunden werden können erst noch in mehreren User Stories spezifiziert werden.

4.2.3. Sprints

Ein Sprint ist eines von mehreren Iterationselemenen in Scrum. Diese können je zwischen ein bis vier Wochen dauern. In diesem Zeitraum werden vorher bestimmte User Stories abgearbeitet.

Sprint-Planung

Die erste Sprint-Planung kann erfolgen, sobald das Product Backlog gefüllt und die User Stories geschätzt und priorisiert wurden.
Zu Beginn eines jeden Sprints wird zuerst ein sogenanntes Sprint Planning Meeting mit dem Product Owner und den ScrumMaster durchgeführt. In diesem Meeting geht es darum das konkrete Ziel des Sprints festzulegen. Dazu werden die User Stories miteinbezogen. Nachdem das Team eine Story ausreichend analysiert und verstanden hat, wird diskutiert, ob die Story nun im nächsten Sprint umgesetzt wird oder nicht. Für die Umsetzungsmenge wird die Velocity für ein Team berücksichtigt.
Beim ersten Sprint wird ohne Velocity geplant. Das Team arbeitet also einfach einmal drauf los. Am Ende des Sprint und der geleisteten Arbeit wird dann ersichtlich wie viel die Velocity ungefähr beträgt. Im zweiten Teil, dem Design-Meeting, werden die Stories noch in einzelne Tasks aufgeteilt.

Sprint Durchführung

Bei der Sprintdurchführung organisieren sich die Entwickler selbst. Der ScrumMaster steht dem Team nur als Helfende Hand zur Seite, kümmert sich jedoch nicht um die Aufgabenzuteilung. Zudem steht der Product Owner als Gesprächspartner für zusätzliche Fragen zu den User Stories zur Verfügung. Die enge Zusammenarbeit ermöglicht wenig Schreibarbeit und dafür eine hohe Kommunikation am laufenden System. Tauchen während der Entwicklung neue Tasks auf, werden diese zusätzlich ergänzt.
Die Sprintdurchführung wird mittels eines Sprint-Burndown-Charts* gemessen. Das Ergebnis ist ein treppenförmiges Diagramm. Dadurch, dass die am höchsten Priorisierten Stories als erstes abgearbeitet werden, wird sichergestellt, dass weniger wichtige Stories am Ende liegen bleiben, wenn die Zeit knapp wird.

Sprint Ende

Die Deadline des Sprints wird in jedem Fall eingehalten. Der Lieferumfang ist das Einzige, welches sich dynamisch anpassen lässt. Bei Zeitknappheit fallen User Stories weg und andererseits können bei genügend Raum zusätzliche User Stories hinzugenommen werden. Am Ende eines jeden Sprints werden zwei wichtige Meetings durchgeführt:

Sprint-Review

Der erste Review dient dazu die umgesetzten Stories öffentlich am laufenden System zu demonstrieren. Die Interessenvertreter haben dadurch die Möglichkeit ein Feedback zu den Entwicklungen zu geben. Dabei können wiederum weitere User Stories hinzukommen.

Sprint-Retrospektive

Der ScrumMaster führ mit dem Team ein Meeting durch. Im Meeting wird auf den letzten Sprint zurückgeblickt und allfällige Verbesserungsmassnahmen werden analysiert. Dabei werden sowohl gute als auch weniger gute Erfahrungen gesammelt. Ein weiterer Punkt ist die Prüfung, was erledigt wurde und woran noch weiter gearbeitet werden muss.

Begriffe

Backlog	Enthält Liste mit priorisierten und geschätzten User Stories.
Product Backlog	Ist ein dynamisches Dokument. Im Laufe des Projekts ändern sich Anforderungen, es kommen neue hinzu, Prioritäten ändern sich oder Anforderungen fallen ganz weg.
Story Points	Werden in drei Gewichtungen unterteilt: Klein (1), Mittel (3) oder Gross (5).
Velocity	oder auf Deutsch Geschwindigkeit bestimmt wie viel Story Points ein Team maximal in einem Sprint abarbeiten darf.
Selected Backlog	Bezeichnet die für einen Sprint ausgewählten User Stories.
Task	Organisatorische oder Programmiertechnische Aufgaben, welche innerhalb eines Tages erledigt werden können.
Release-Plan	Projektplanung des mindestens einen Release in einer Zeitspanne von drei bis sechs Monate umfasst.
Sprint Backlog	Enthält alle entschiedenen User Stories und deren Tasks.
Daily Scrum	Tägliches Teammeeting, wobei folgende Themen Besprochen werden: Was habe ich gestern getan? Was plane ich heute? Welche Hindernisse oder Probleme haben sich mir in den Weg gestellt?
Story-Karte	Sichtbarer Teil einer User Story. Sie Bringt den Kern der zu umsetzenden Arbeit in einem Satz auf den Punkt. Story-Karten werden üblicherweise auf A5-Karteikarten geschrieben.

5. Reiseszenarien

5.1. Szenario 1

Ausgangslage

Hans, der in Pfäffikon wohnt, ist ein begeisterter Hockeyfan der Rapperswil-Jona Lakers. Er würde sich heute abend sehr gerne das Playoff-Spiel zwischen seiner Mannschaft und den Kloten Flyers in der Kolping Arena in Kloten anschauen. Doch ihm fehlen noch einige Fanartikel. Bevor er also von Päffikon aus nach Kloten reist, muss er erstmals nach Rapperswil und sich einige Fanartikel kaufen.

Ist

Mühselig stellt er sich den Fahrplan mithilfe der SBB-Seite zusammen. Er schreibt sich alle relevanten Ankunfts- und Abfahrtszeiten übersichtlich auf ein kleines Stück Papier zusamen.
Am frühen Abend macht sich dann Hans auf den Weg nach Rapperswil um sich noch die Fanartikel zu kaufen. Während er am Einkaufen ist, gibt es im Bahnhof Stadelhofen ein technisches Problem, dass zum Ausfall eines kompletten Perrons führt. Alle Züge auf dieser Verkehrsachse verspäten sich auf unbestimmte Zeit.
Leider kriegt Hans dies nicht mit. Als er einige Minuten vor Abfahrt auf dem Bahnhof ankommt, sieht er auf der Anzeigetafel schon, dass etwas nicht in Ordnung ist. Nach der Lautsprecherdurchsage ist auch Hans auf dem neuesten Stand und realisiert schnell, dass es mit diesem Zug wohl zu grossen Verspätungen kommen kann. Er schaut daher schnell im Fahrplan nach einer alternativen Verbindung nach Kloten. Kurze Zeit später hat er eine Umfahrungsroute gefunden. Leider ist aber der Zug vor einigen Minuten abgefahren.
Wenn Hans schon beim Einkaufen erfahren hätte, dass es auf seiner geplanten Route ein Problem gibt, hätte er nicht das 1. Drittel des Matches verpasst.

Soll

Kurzerhand holt Hans sein Smartphone aus der Tasche und startet die ”Traveler Guide” App. Dort gibt er in einer einfach bedienbaren Maske die gewünschte Route ein: Von Päffikon nach Rapperswil, von Rapperswil nach Kloten und von Kloten zurück nach Pfäffikon. Zudem trägt Hans in der App noch den Fanshop mit der Zeit ein.
Nachdem alle Angaben erfasst und im App bestätigt sind, zeigt dass App die Reiseinformationen übersichtlich dar. Während Hans in Rapperswil am Einkaufen ist, gibt es im Bahnhof Stadelhofen ein technisches Problem, dass zum Ausfall eines kompletten Perrons führt. Alle Züge auf dieser Verkehrsachse verspäten sich auf unbestimmte Zeit.
Die ”Traveler Guide” App merkt sofort, dass Hans mit dieser Verbindung wohl nicht

rechtzeitig in Kloten ankommen wird. Mit einer Benachrichtung meldet die App die Störung. Hans kann dadurch rasch eine Alternativroute suchen und die Störung umfahren.
Hans ist sichtlich erleichtert, dass er gewarnt wurde und entscheidet sich den Einkauf etwas zu beschleunigen und die gewählte Alternativroute nach Kloten zu verwenden.

5.2. Szenario 2

5.2.1. Ausgangslage

Anne-Marie und ihre zwei Freundinnen Catelyn und Sarah planen einen Trip nach London. Anne-Marie hat eingewilligt die Organisation des Trips zu übernehmen und reserviert im Reisebüro den Flug von Zürich nach London Heathrow, wo sie dann den Bus in die Innenstadt nehmen wollen. Als Unterkunft wählen sie ein ***Hotel nahe der Kingscross Station.
Einige Tage später bekommen die drei Freundinnen die Informationen zum Flug und Hotel in Papierformat nach Hause geliefert.

Ist

Nachdem sie in London gelandet sind, suchen sie sich einen Bus, der sie in das Stadtzentrum bringt. Als sie ausgestiegen sind kramen sie ihre Unterlagen vom Reisebüro heraus und versuchen sich zu orientieren. Zuerst einmal wo sie sich gerade befinden und danach wohin sie laufen müssen. Da sie den Weg nicht auf Anhieb finden, fragen sie einen Passanten. Nach ein paar Verständigungsproblemen finden sie jedoch den Weg zu ihrem Hotel doch noch. Am nächsten Tag wollen die Mädels zum ”Prim Mart”. Von einer anderen Freundin haben sie erfahren, dass es da eine riesige Auswahl zu günstigen Preisen haben soll. Das Problem ist nur, dass keiner der Kolleginnen weiss, wo sich dieser genau befindet. Nach einer langen Suche und Herumirren erfahren sie von einem Einwohner, in welcher Strasse sich das Shopping Center befindet. Trotzdem, als sie endlich am Ziel angekommen sind, hat es schon geschlossen. Frustriert gehen sie wieder ins Hotel zurück.
Am nächsten Tag finden sie es jedoch auf Anhieb und können ihren Shopping Tag geniessen. Da sie leider schon nächsten Abend zurück müssen, bleibt ihnen aber nicht mehr viel Zeit für die Sehenswürdigkeiten der Stadt und müssen sich entscheiden, was sie sich noch anschauen wollen.

Soll

Ein Tag vor der Abreise fordert Anne-Marie ihre Freundinnen auf, den TravelGuide auf ihr Android Handy zu laden. Als sie es installiert haben, sendet sie ihnen einen detaillierten Ablauf ihres Trips.

Nachdem sie in London gelandet und mit dem Bus ins Stadtzentrum gefahren sind, packt Catelyn ihr Handy aus und lokalisiert ihre Position mittels GPS. Da Anne-Marie die Hoteladresse eingegeben hat können sie mit einem Klick die Wegbeschreibung zu ihrem Hotel abfragen.
Am zweiten Tag können sie auswählen, welches ihrer vorher eingegeben Ziele sie als erstes besuchen wollen. Sie kommen gut voran und können sich am frühen Abend in einem Pub einen Cocktail gönnen. Vorher schauen sie mit Hilfe der App noch kurz nach wo sich eine Cocktailbar in der Nähe befindet. Auf dem Weg zur Bar entdecken sie noch ein interessantes Geschäft. Leider hat es jedoch schon geschlossen. Kurzerhand setzt eine der Freundinnen eine Erinnerung mit der App, damit sie es am nächsten Tag wieder finden.
Zuhause zeigt Anne-Marie ihrem Freund mit der History-Funktion der App, wo sie überall waren.

6. Ablaufmodell

Das Ablaufmodell beschreibt die Szenarien in einer formelleren Weise. Anhand dieses Modells können wir die User Stories ableiten

6.1. Szenario 1

Der Ablauf für dieses Szenario ist relativ einfach gehalten.
Hans plant zuerst seine Reise auf dem Smartphone indem er die nötigen Reiseinformationen - Reiseziele, Aufenthaltsdauer und gewünschte Ankunftszeit bei Zwischenetappen - eingibt. Das Smartphone zeigt nun den idealen Reiseplan. Hans startet die Reise. Während der Reise muss der Reiseplan, aufgrund eines technisches Problems angepasst werden. Der ÖV-Betreiber kann die vorgegeben Zeiten nicht mehr einhalten. Nach der Kenntnisnahme der neuen Informationen setzt Hans seine Reise fort. Trotz Zugverspätung kommt Hans noch rechtzeitig zu Hause an.
Abbildung 1 zeigt den Ablauf dieses Szenarios dar.

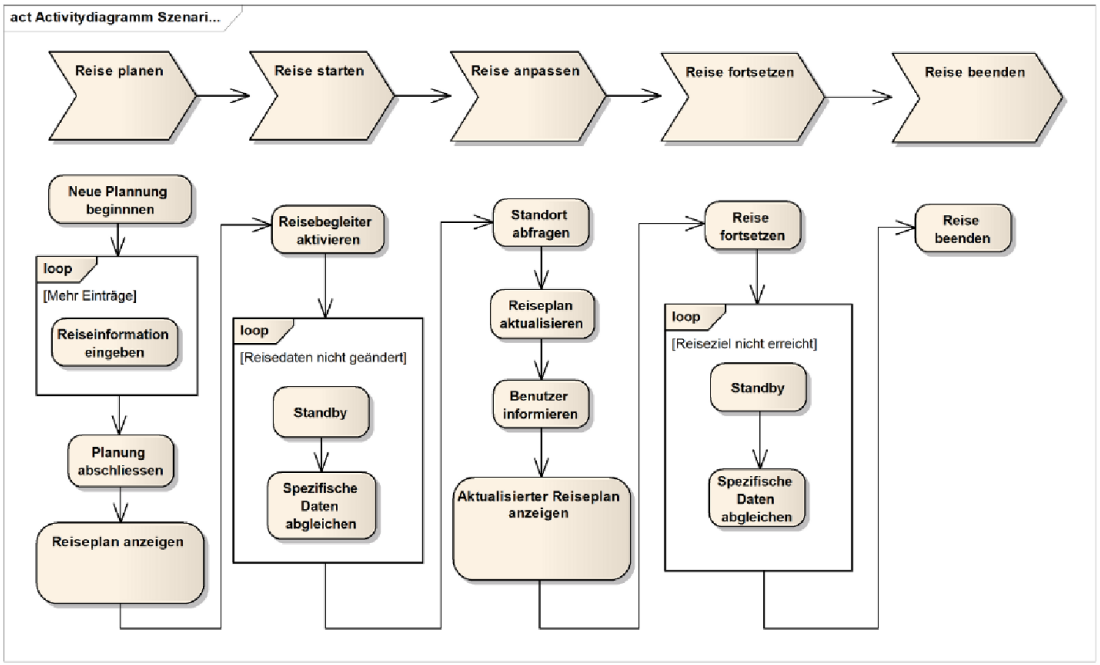


Abbildung 1: Ablaufplan für Szenario 1

6.2. Szenario 2

Der Ablaufplan für das zweite Szenario greift etwas tiefer. Hier machen sich die drei jungen Frauen auf nach London. Die Reiseplanung erfolgt noch klassisch in einem Reisebüro mit Papierinformationen. Mit der App plant Anne-Marie schlussendlich die Reise und gibt die Informationen aus den Reiseunterlagen und ihre eigenen Daten ein. Die Reise teilt sie schliesslich mit ihren Freundinnen. Nachdem die Reise gestartet wurde werden GPS-Aufzeichnungen gemacht.
In London lassen sich die Freundinnen ihren Weg zum Ziel zeigen. Dafür eignet sich am besten die Routenfunktion. Mit der POI-Suche - in Verbindung mit der Orientierungshilfe - finden sie auch ziemlich schnell zu ihre Cocktail Bar. Mit der Möglichkeit jeden Ort auf der Karte als POI zu setzen ist es ihnen einfach möglich zusätzliche Reiseziele zu ihrer Reiseplanung hinzuzufügen. Somit finden sie alle benötigten Informationen später wieder. Um die Shoppingtour-Abdeckung nachzuprüfen gibt es in der App eine GPS-History.

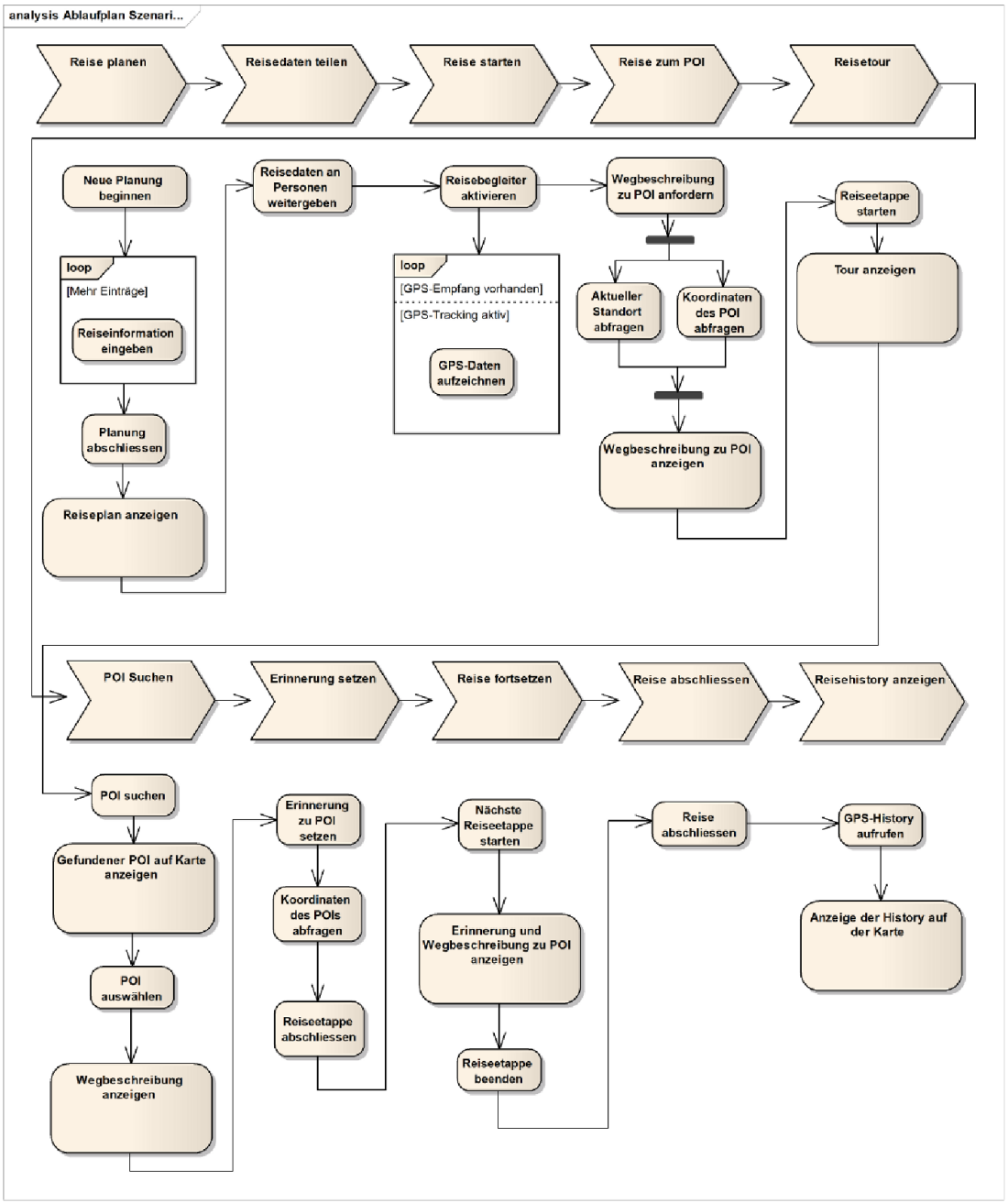


Abbildung 2: Ablaufplan für Szenario 2

6.3. User Stories

Aus den beiden Szenarios lassen sich teils folgende User Stories ableiten:

Eingabemaske für Reise	
US 1	*Als Benutzer will ich meine Daten einfach eingeben können.* Eine Eingabemaske ermöglicht es dem Benutzer die geplante Reise in der App zu erfassen.

ÖV-Verbindungen	
US 2	*Als Benutzer will ich Fahrpläne suchen können.* Dem Benutzer soll es möglich sein nach öffentlichen Verbindungen suche zu können. Diese sollen zu einer Reise hinzugefügt werden können (CRUD).

Notifikation	
US 3	*Als Benutzer will ich immer über aktuelle Änderungen informiert werden.* Die App soll in regelmässigen Intervallen die volatilen Daten überprüfen (z.B. Zugverbindungen). Bei Änderungen sollen diese Änderungen dem Benutzer mitgeteilt werden.

GPS-Position ermitteln	
US 4	*Als Benutzer will ich meine aktuelle Position ermitteln können.* Damit der Benutzer sich auf der Karte lokalisieren kann und sich einen Überblick verschaffen kann, soll ein GPS-Dienst miteinbezogen werden.

Alarm Funktion	
US 5	*Als Benutzer will ich eine Alarm Funktion für meine Termine, sodass ich auf wichtige Ereignisse erinnert werde.* Mit einer Erinnerungsfunktion soll dem Benutzer geholfen werden, dass er seinen Zeitplan einhalten kann. Solch eine Erinnerungsfunktion soll es sowohl bei POIs als auch bei den Fahrplänen geben.

Reisedaten teilen	
US 6	*Als Benutzer will ich meine Daten an andere Personen weitergeben können, sodass mehrere Personen die selbe Planung für sich benutzen können.* Die Möglichkeit eine Reise zu teilen verfolgt den Vorteil, dass Daten abgeglichen werden können und unnötige Eingaben entfallen.

Suche von POIs	
US 7	*Als Benutzer will ich die Möglichkeit haben POIs zu suchen und diese meiner Reise hinzuzufügen (CRUD).*

GPS-History	
US 8	*Als Benutzer will ich meine Tour zurückverfolgen können.* Die GPS-History soll dem Benutzer einen Rückblick über seine vergangene Reise verschaffen. Dabei soll bei der Aktivierung der Reise GPS-Daten gesammelt und archiviert werden.

InApp-Suche	
US 9	*Als Benutzer will ich auftretende Fragen sofort im Internet suchen können, sodass ich nicht immer die Programme wechseln muss.* Es soll eine Möglichkeit geben dem Benutzer schnellen Zugriff auf Suchmaschinen zu gewähren. Um dies noch etwas effizienter zu bieten sollen einige der umgänglichsten Webseiten ausgewählt werden können.

Karte	
US10	*Als Benutzer will ich meine Ziele auf einer Karte zu finden wissen.* Durch das Integrieren einer Karte soll es dem Benutzer möglich sein, die Position von POIs oder Bahnhöfen zu betrachten.

Route	
US11	*Als Benutzer will ich Möglichkeit haben mich zum Ziel leiten zu lassen.* Mit Hilfe der Karte, dem aktuellen Standort und dem Standort eines POI oder Bahnhofs soll es dem Benutzer ermöglicht werden, mittels Routenfunktion den Weg zum Ziel zeigen zu lassen.

Übersichtfunktion	
US12	*Als Benutzer will ich meine Reise in einer Gesamtübersicht betrachten können.* Eine grafische Übersicht aller Einträge einer Reise soll es dem Benutzer vereinfachen seine Termine zu ordnen.

Übersetzer	
US13	*Als Benutzer will ich mich im Ausland verständigen können.* Die App soll Möglichkeit bieten Wörter oder Sätze in eine andere Sprache zu übersetzen.

Daten Persistierung	
US14	*Als Benutzer will ich meine Daten persistieren können.* Daten sollen in einem non-volatilen Speichermedium verwaltet werden (CRUD).

7. Use Cases der App

Bei den Use Cases wird davon ausgegangen, dass alle externen Dienste der Applikation funktionstüchtig sind und die App richtig installiert wurde. Die Schnittstellen liefern alle benötigten Daten fehlerfrei.

7.1. Use Case Diagramm

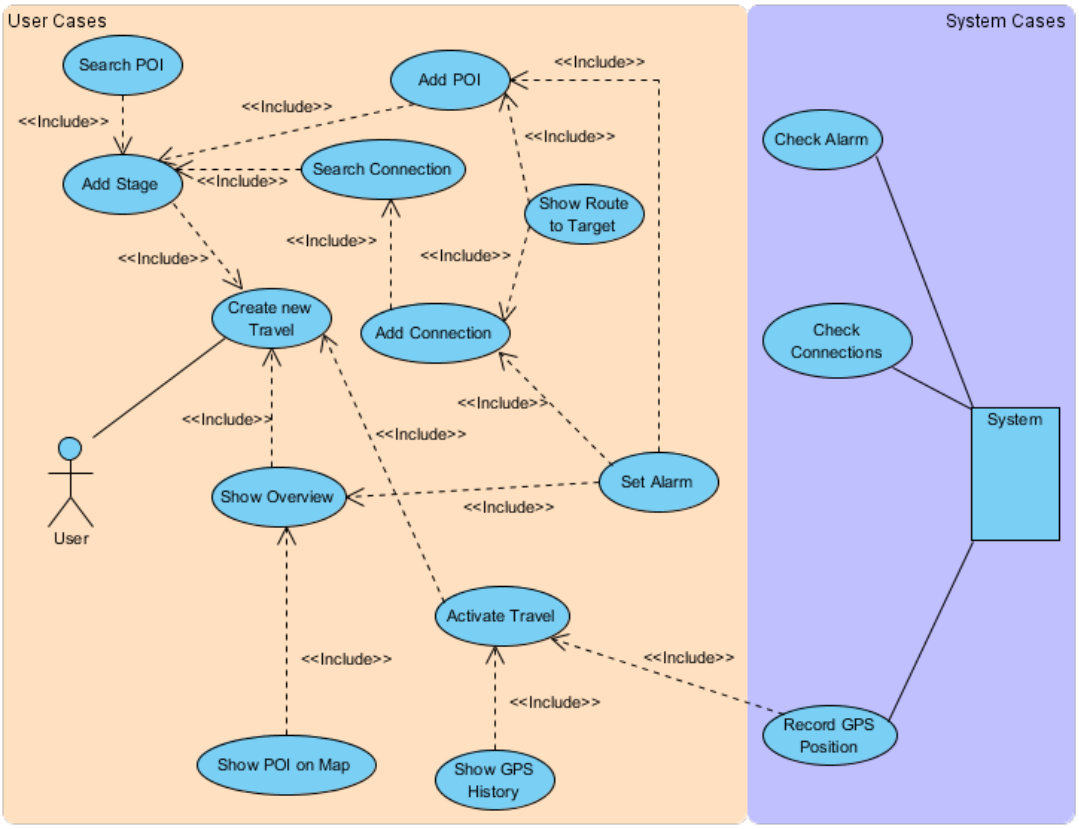


Abbildung 3: Use Cases

7.2. Use Case Beschreibung

Create new Travel	
UC 1	Precondition: - Der Benutzer möchte eine neue Reise planen. Dazu drückt er im Startbildschirm auf den ”Neue Reise hinzufügen”-Knopf. Nachdem er diesen gedrückt hat erscheint eine neue View. Nun kann er den Namen seiner Reise bestimmen und ein Start und Enddatum wählen.

Add Stage	
UC 2	Precondition: Reise ist erstellt (UC 1) Um weitere Daten hinzuzufügen kann der Benutzer auf den Reise-Eintrag in der Liste im Startbildschirm (kurz) drücken. Es erscheint ein Kontextmenü. Hier wählt er die Bearbeiten-Funktion aus. Nun wird er auf die nächste Activity geleitet. Von hier aus kann er mittels dem ”Etappe hinzufügen”-Knopf einen neuen Tagesabschnitt erstellen. Nach der Namens- und Datumseingabe erscheint, nachdem drücken des ”Speichern”-Knopfs, die erstellte Etappe in der Liste.

Search POI	
UC 3	Precondition: Etappe ist erstellt (UC 2) Für die POI Suche gibt es zwei Möglichkeiten. Dies lässt sich mit einem Klick auf eine Etappe mit der Auswahl ”Neuen POI hinzufügen” weiter verfolgen. • Die erste Möglichkeit besteht darin einen POI in Umgebung eines Ortes zu suchen. Mit der Auswahl ”Spezifischer Ort” erscheint ein weiteres Eingabefeld. Gibt man hier einige Buchstaben ein, erhält man Vorschläge zu dessen Vervollständigung. Dazu muss der zweite Punkt ausgeführt werden... • Die einfachste Weise, wie man nach POIs suchen kann, ist dass man die Checkbox ”In der Nähe” auswählt. Nach dem einfüllen des gewünschten Suchbegriffs (z.B. Hotel) erhält man eine Liste von Resultaten auf die Suchanfrage. . . .

Add POI	
UC 4a	Precondition: POI Resultate vorhanden (UC 3) Ein POI kann mit einem Klick auf die POI-Resultateliste hinzugefügt werden. In dieser Variante ist es nur möglich POIs hinzuzufügen, die bei einer Suchanfrage als Resultat erscheinen.
UC 4b	Precondition: POI oder Connection ist erstellt (UC 4a/6) Weiter kann ein benutzerspezifischer POI mit Hilfe der Karte hinzugefügt werden. Über die Übersicht gelangt man zu der Google Maps Karte. Hat man sich einen gewünschten Punkt als POI ausgewählt, so klickt man im Menüpunkt auf "POI hinzufügen". Nach Möglichkeit den POI mit einem Namen und einer Zeit zu versehen, wird die Adresse und Koordinaten des Bildschirmmittelpunktes zu der ausgewählten Etappe hinzugefügt.

Search Connection	
UC 5	Precondition: Etappe ist erstellt (UC 2) Wie auch beim POI kann durch anklicken eines Etappenelements und der Auswahl ”Fahrplan hinzufügen” nach einer Verbindung suchen. Es erscheint eine neue View. Nachdem die Standarddaten eingegeben wurden kann die Suche mit einem Klick auf den ”Verbindung Suchen” gestartet werden. Dabei werden mehrere Verbindungen auf die Anfrage aufgelistet.

Add Connection	
UC 6	Precondition: Resultate von Verbindungssuche (UC 5) Um eine Verbindung einer entsprechenden Etappe zuweisen zu können, klickt der Benutzer auf ein gewünschtes Resultat, welches aus der Suche hervor ging.

Show Overview	
UC 7	Precondition: Reise ist erstellt (UC 1) Der Benutzer kann sich eine Übersicht seiner Reise darstellen lassen, indem er im Startbildschirm auf einen Reiseeintrag klickt. Es erscheint ein Kontextmenü, indem er den Menüpunkt ”Zeige Übersicht” auswählt.

Set Alarm	
UC 8	Precondition: POI oder Connection vorhanden (UC 4/6) Der Alarm zu einem POI- oder Verbindungselement ist per default ausgeschaltet. Um den Alarm auf eine gewünschte Zeit zu setzen, klickt man auf das Icon des jeweiligen Elements. Es Öffnet sich ein Dialog mit der Option ”Alarm”. Durch antippen dieser Option lässt sich die Zeit einstellen.

Show POI/Station on Map	
UC 9a	Precondition: POI ist vorhanden (UC 4) Um dem Benutzer die Möglichkeit zu bieten einen POI auf der Karte zu lokalisieren, besteht die Option im Dialogmenü eines POI-Übersichtelements das Auswahlkriterium ”Karte” anzuwählen. Damit stellt die Applikation den POI auf der Karte mit einem Pin dar.
UC 9b	Precondition: Connection ist vorhanden (UC 6) Bei Verwendung der Kartenfunktion mit einem Fahrplan wird der zeitlich passenste Bahnhof einer Linienverbindung markiert. Das Vorgehen bleibt gleich.

Show Route to Target	
UC10a	Precondition 1: POI ist vorhanden (UC 4) Precondition 2: GPS eingeschaltet, GPS-Daten empfangen Die Routen-Funktion bietet eine sichere Geleitung an das Ziel. Im Übersichtsdialog befindet sich eine Option ”Route”. Durch Anwahl zeigt es die Route vom aktuellen Standort bis zum POI.
UC10b	Precondition 1: Connection ist vorhanden (UC 6) Precondition 2: GPS eingeschaltet, GPS-Daten empfangen Mit dem selben Vorgehen bei einem Fahrplan wird jeweils die Route zum zeitlich nächsten Bahnhof angezeigt.

Activate Travel	
UC11	Precondition: Reise vorhanden (UC 1) Will der Benutzer die Erinnerungsfunktion aktivieren, so kann er mittels Longclick auf ein Reiseelement im Startbildschirm ein weitere Kontextmenü aufrufen. Drückt er auf ”Reise aktivieren” wird die Reise als aktiv markiert. In diesem Fall wird ein neuer Android Service erstellt. Dieser läuft auch weiter, falls sich die App nicht mehr im aktiven Zustand befindet.

Record GPS Positions (System)	
UC12	Precondition: Reise ist aktiv (UC 11) Wenn die Reise aktiv ist, sammelt das System während der ganzen Reise die empfangenen GPS-Daten. Dies Funktion kann auch separat auf Sparmodus (GPS-Requests werden nur abgefragt, wenn andere Dienste dies explizit verlangen) oder ganz ausgeschaltet werden.

Show GPS History	
UC13	Precondition: GPS Daten sind vorhanden (UC 12) Möchte der Benutzer seinen zurückgelegten Weg zurückverfolgen bietet sich diese Funktion an. Mit dem Klick auf das entsprechende Reiseelement lässt sich ”GPS Verlauf” auswählen. Der Benutzer wird auf eine Kartenansicht weitergeleitet, wobei alle zu der Reise vorhandenen GPS-Koordinaten mittels farbigen Punkten markiert werden.

Check Alarm (System)	
UC14	Precondition: Reise ist aktiv (UC 11) Alle 15 Minuten überprüft ein Dienst alle gesetzten Erinnerungen. Ist eine Erinnerung abgelaufen, wird der Benutzer per Benachrichtigung auf die Erinnerung aufmerksam gemacht.

Check Connections (System)	
UC15	Precondition 1: Reise ist aktiv (UC 11) Precondition 2: Connections vorhanden (UC 6) Im 15 Minuten Takt werden alle Verbindungen überprüft. Unterscheiden sich die aktuellen Verbindungen mit den neu abgefragten Verbindungen, so wird der Benutzer mit einer Benachrichtigung über die Änderungen informiert.

8. Domainmodell

Ein wesentlicher Teil der Analyse ist das Erstellen eines Domainmodells. Es beschreibt in einer einfach verständlicher Weise den zukünftigen Aufbau der zu entwickelnden Applikation.
Das Domainmodell ist zum Teil aus den User Stories und Use Cases abgeleitet und ist in Abbildung 4 aufgezeichnet.
Um eine Reise erfassen zu können, benötigt es erstmals ein Reise-Objekt. In diesem Fall ”Travel” genannt. Eine Reise besteht aus einer oder mehreren Etappen. Das Objekt ”Stages” repräsentiert solche Etappen. Des weiteren kann jeder Etappe ein oder mehrere Fahrpläne oder POIs hinzugefügt werden. Diese werden durch die Objekte ”POI” sowie ”Timetable” festgehalten. Um diese Informationen anschaulich dem Benutzer anzeigen zu können, benötigt es eine Reiseübersicht. Für diese Aufgabe ist das Objekt ”TravelOverview” zuständig. Diese Übersicht befasst sich mit dem Anzeigen von POIs, Routen und Stationen auf der Karte. Die Karte ist in diesem Fall das Objekt ”Map”.

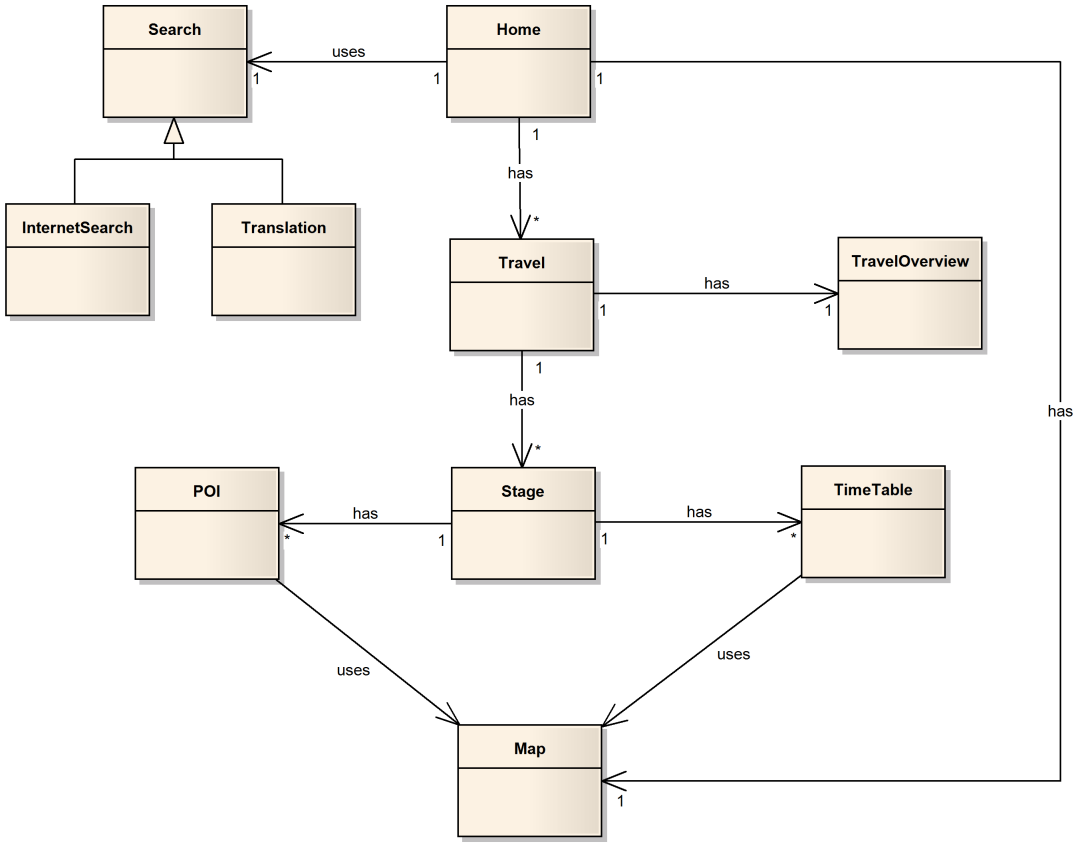


Abbildung 4: Domainmodell

9. Evaluation Online Schnittstellen

In diesem Kapitel werden die verschiedenen eingesetzten Technologien und deren Alternativen verglichen und evaluiert. Für jede Variante werden die Vor- und Nachteile aufgelistet. Zum Schluss wird kurz erläutert, welche Technologie für die App ausgewählt wurde.

9.1. SBB-Schnittstelle

Die Einbindung des SBB-Fahrplans ist ein wichtiger Aspekt der App. Es ermöglicht dem Benutzer effizient und mit minimaler Eingabe eine Reise zu planen. Für die Einbindung gibt es vier verschiedene Möglichkeiten. Diese werden nachfolgend genauer beschrieben und evaluiert.

9.1.1. Einbindung SBB-App

Die SBB-App ist eine beliebte und einfach zu bedienenbare Anwendung um Fahrpläne für die ganze Schweiz abzufragen.
Beim Planen einer Reise in der ReiseApp, wird automatisch die SBB-App gestartet. Dort gibt der Benutzer, in der bereits bekannten Eingabemaske, die nötigen Daten ein. Die Resultate werden dann direkt an die ReiseApp übergeben und die SBB-App wird geschlossen.

Vorteile	Nachteile
UI mit Eingabemaske ist bereits vorhanden	Benutzer ist gezwungen die SBB-App zu installieren
Abfragelogik wird durch SBB-App gesteuert.	Das Abrufen der Informationen aus der SBB-App ist nicht dokumentiert. Daher ist nicht ersichtlich ob eine Schnittstelle überhaupt existiert.

Tabelle 5: Vor- und Nachteile SBB-App

9.1.2. Einbindung Mobile-Webseite SBB

Die Mobile-Webseite der SBB ist speziell für Smartphone-Browser ausgelegt. Die Seite besteht aus HTML-Code.

Vorteile	Nachteile
Einfach zu implementieren	Ausgabe ist reines HTML dass geparst werden muss. Eine Anpassung der Seite kann zu einer nicht funktionsfähigen ReiseApp führen.
Eingabemaske ist bereits vorhanden.	
Abfragelogik wird durch SBB-Seite gesteuert.	

Tabelle 6: Vor- und Nachteile Mobile-Webseite SBB

9.1.3. Verwendung der XML-Schnittstelle

Die inoffizielle Schnittstelle zur Fahrplanabfrage ist nicht dokumentiert. Daher muss die Funktionsweise ”reverse-engineered” werden.
Mithilfe der SBB-App und Wireshark können die übermittelten Daten abgefangen werden. Aus diesen Daten kann herausgefunden werden, welche Daten an die Schnittstelle gesendet werden müssen und welche Daten als Antwort zu erwarten sind.
Grundsäztlich besteht eine Fahrplanabfrage aus zwei HTTP-POST ”Requests”. Der erste Request beinhaltet den Abfahrtsort sowie den Ankunftsort. Der folgende Code zeigt einen möglichen Request:

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <ReqC lang="EN" prod="Android 4.0.2" ver="2.3" accessId="sbuR5...
       lKDnN">
3    <LocValReq id="fromInput" sMode="1">
4      <ReqLoc match="Rapperswil" type="ALLTYPE">
5      </ReqLoc>
6    </LocValReq>
7    <LocValReq id="toInput" sMode="1">
8      <ReqLoc match="Bern" type="ALLTYPE">
9      </ReqLoc>
10   </LocValReq>
11 </ReqC>
```

Listing 1: Location Request

Als ”Response” erhält man darauf alle Stationen, welche der Eingabe entsprechen. Jedem Treffer wird dabei ein Wert zugewiesen. Dieser Wert entspricht der Genauigkeit der Eingabe. Jede Station enthält zudem eine `externalId`, welche für die zweite Abfrage benötigt wird. Der folgende Codeausschnitt zeigt die Antwort auf die vorhin getätigte Abfrage:

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <ResC xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:
       noNamespaceSchemaLocation="http://demo.hafas.de/hafas-res/sbb/xml
```

```
    /47r/hafasXMLInterface.xsd" ver="3.0.5" prod="5.30.SBB.4.7v2 [Sep
     13 2011]" lang="EN">
  <LocValRes flag="FINAL" id="fromInput">
    <Station name="Rapperswil" score="101"
     externalId="008503110#85" externalStationNr="008503110"
     type="WGS84" x="8816727" y="47224884"/>
  </LocValRes>
  <LocValRes flag="FINAL" id="toInput">
    <Station name="Bern" score="101" externalId="008507000#85"
     externalStationNr="008507000" type="WGS84"
     x="7439122" y="46948825"/>
  </LocValRes>
</ResC>
```

Listing 2: Location Result

Aus dieser Antwort kann, dann entweder automatisch die passende Station ausgewählt werden oder der Benutzer kann eine gewünschte Station selber auswählen.
Nach getroffener Abfahrts- und Ankunftsstationswahl wird eine zweite Abfrage gestartet. Der XML-Code der Abfrage sieht wie folgt aus:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<ReqC lang="EN" prod="Android 4.0.2" ver="2.3" accessId="sbuR5...
   lKDnN">
  <ConReq>
    <Start>
      <Station name="Rapperswil" externalId="008503110#85"></Station>
      <Prod prod="1111111111000000"></Prod>
    </Start>
    <Dest>
      <Station name="Bern" externalId="008507000#85"></Station>
    </Dest>
    <ReqT a="0" date="20120402" time="11:15"></ReqT>
    <RFlags b="0" f="4" sMode="N"></RFlags>
  </ConReq>
</ReqC>
```

Listing 3: Connection Request

Der Response dieser Abfrage ist eine ca. 800kb (unkomprimiert) grosse XML-Datei.

9.1.4. OpenSource SBB JSON-Schnittstelle

http://Transport.opendata.ch bietet eine OpenSource Schnittstelle an, um Fahrplandaten der SBB abzufragen. Bei der Schnittstelle handelt es sich um eine PHP-Serveranwendung welche HTTP-Requests akzeptiert. Diese werden dann verarbeitet und an die XML-Schnittstelle der SBB weitergeleitet. Der XML-Response der SBB wird dann verarbeitet und als JSON-Array an den Benutzer zurückgegeben.

Vorteile	Nachteile
Ausgabe als reines XML	Schnittstelle ist nicht dokumentiert.
Anwender benutzt für alle Eingaben die selbe Applikation.	Funktionsweise muss durch Reverse-Engineering erlernt werden.
	Abfragelogik muss selbst implementiert werden.
	Abfragen verursachen viel Datenverkehr.

Tabelle 7: Vor- und Nachteile XML-Schnittstelle SBB

Diese Schnittstelle hat einige entscheidende Vorteile gegenüber der XML-Schnittstelle: Die selbe Abfrage analog zur XML-Variante in Listing 3 sieht in JSON wie folgt aus:

```
GET
http://transport.opendata.ch/v1/connections?from=Rapperswil&to=Bern&
    date=2012-04-02&time=11:15
```

Listing 4: JSON Connection Request

Der Response dieser Abfrage ist lediglich 20kb gross. Listing 5 zeigt Ausschnitte der Antwort.

```
{
  "connections": [
    {
      "from": {
        "station": {
          "id": "008503110",
          "name": "Rapperswil",
          "score": null,
          "coordinate": {
            "type": "WGS84",
            "x": 8.816727,
            "y": 47.224884
          }
        },
        "arrival": null,
        "departure": "2012-04-02T11:44:00+0200",
        "platform": "4",
        // more elements ...
      },
      "to": {
        "station": {
          "id": "008507000",
          "name": "Bern",
          "score": null,
          "coordinate": {
            "type": "WGS84",
            "x": 7.439122,
            "y": 46.948825
```

```
29          }
30        },
31        "arrival": "2012-04-02T13:28:00+0200",
32        "departure": null,
33        "platform": "4",
34        // more elements ...
35      }
36      "sections": [
37        {
38          "journey": {
39            "name": "S1519542",
40            "category": "S15",
41            "number": "19542",
42            "operator": null,
43            "to": null,
44            "passList": [
45              {
46                "station": {
47                  "id": "008503110",
48                  "name": "Rapperswil",
49                  // more elements ...
50                },
51                "arrival": null,
52                "departure": "2012-04-02T11:44:00+0200",
53                ...
54              },
55              {
56                "station": {
57                  "id": "008503120",
58                  "name": "Jona",
59              // more elements ...
60                },
61                // more elements ...
62              },
63              // more elements ...
64            }
65        }
66      ]
67    },
68    {
69    // more elements ...
70    }
71  ]
72  ...
73 }
```

Listing 5: JSON Connection Response

Vorteile	Nachteile
JSON ist sehr einfach zu parsen.	Noch in Entwicklung
Die JSON-Responses sind wesentlich kleiner als die XML-Responses.	Unstabil und unzuverlässig
Die Datenstruktur wurde vereinfacht und unnötige Informationen wurden entfernt.	Enthält nur einen Teil der Informationen der XML-Schnittstelle
Eine Verbindungsabfrage kann mit einem einzigen Request getätigt werden.	

Tabelle 8: Vor- und Nachteile JSON-Schnittstelle transport.opendata.ch

9.1.5. Fazit

Keine dieser Lösungsvorschläge ist ideal. Trotzdem muss eine Schnittstelle für die Fahrplananbindung gewählt werden. Die Einbindung der SBB ist, falls überhaupt möglich, nicht praktikabel. Die Benutzer wären gezwungen die offizielle SBB App zu installieren um die TravelGuide Applikation verwenden zu können.
Die Einbindung der Mobile-Webseite der SBB ist die einfachste Möglichkeit der Einbindung der Fahrplandaten. Jedoch ist diese Variante auch sehr anfällig auf Änderungen der Webseite. Änderungen in der Struktur der Webseite können zu Problemem beim parsen führen und die App würde nicht mehr funktionieren.
Die XML-Schnittstelle der SBB bietet am meisten Freiheiten bei der Gestaltung der Fahrplansuche. Jedoch ist die API nicht öffentlich und daher auch nicht dokumentiert. Des weiteren benötigt die XML-Schnittstelle einen relativ grossen Datenverkehr.
Die OpenSource JSON Schnittstelle von transport.opendata.ch ist ein RESTful Webservice. Die API greift ebenfalls auf die XML-Schnittstelle der SBB zu. Der Informationsgehalt ist aber deutlich kleiner und enthält nur die wichtigsten Angaben zu den Verbindungen. Nachteil dieser Lösung ist, dass diese API noch relativ neu ist und ständig weiterentwickelt wird. Daher kommt es gelegentlich vor, dass die API nicht funktioniert.

Der Prototyp dieser App verwendet die JSON-Schnittstelle von transport.opendata.ch. Sie ist einfach einzubinden und der Traffic der durch regelmässigen Abfragen einsteht, ist nicht überflüssig gross. Um die Stabilität der API zu erhöhen und fehlende Informationen selber hinzufügen zu können, läuft die API auf einem internen Server an der HSR.

10. Umsetzungskonzept

10.1. Domainobjekte

Nachfolgend werden alle für die App benötigten Domainobjekte näher beschrieben.

10.1.1. TravelEntry

Eine Reise wird mittels einer `TravelEntry` erzeugt. Eine Entry kann sich über einen Tag oder mehr erstrecken. Nachfolgend sind die einzelnen Elemente einer Reise näher beschrieben.

10.1.2. Stage

Eine Reise kann mehrere `Stages` beinhalten. Jede `Stage` kann mehrere `Pois` oder `Connections` beinhalten. Eine Etappe hat eine fixe Dauer von einem Tag. So kann eine Reise etwas feiner unterteilt werden. Ausserdem lässt es sich so einfacher an die begrenzten Displaygrösse der Mobiltelefone skalieren.

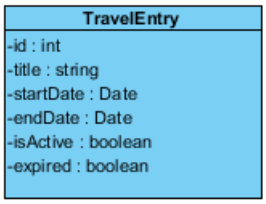


Abbildung 5: TravelEntry

10.1.3. Connection

Eine `Connection` beinhaltet Fahrplandaten der öffentlichen Verkehrsmittel. Das Objekt wird weiter unterteilen in:

- ConnectionSection
  Hier können die verschiedenen Verbindungslinien (z.B. Bern - Zürich, Zürich - St. Gallen) herausgelesen werden.

- Station
  In den `Station`-Objekten werden Informationen zu einem Bahnhof gespeichert. Da eine Verbindungstrecke aus einem Ausgangs- und Zielbahnhof besteht, sind in einer ConnectionSection mindestens zwei `Station`-Objekte vorhanden.

10.1.4. POI - Point of Interests

In den POI-Objekten werden benutzerdefinierte Orte oder Immobilien festgehalten. In der App können vordefinierte POIs verwendet werden. Dafür wird dem Benutzer eine Suche bereitgestellt. Die POIs werden von Google Places geliefert. Frei wählbare POIs kann man in der Overview-Kartenansicht erstellen.

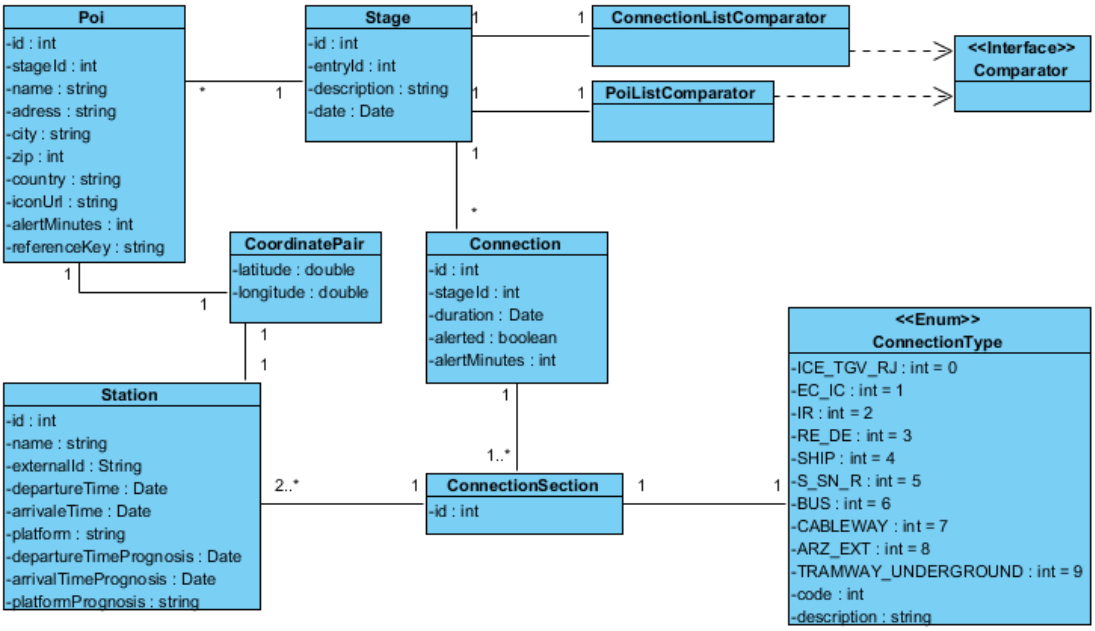


Abbildung 6: Stage, Connection und POI

10.2. Datenverwaltung

Die Datenverwaltung wird durch eine Manager-Klasse verwaltet. Die Aufgabe des Managers ist es die Domainobjekte der App für andere Klassen leicht zugänglich zu machen und für dessen Verfügbarkeit und Aktualität zu garantieren. Zudem dient er als Observer. Das ermöglicht bei Bedarf ein automatisiertes Datenupdate.

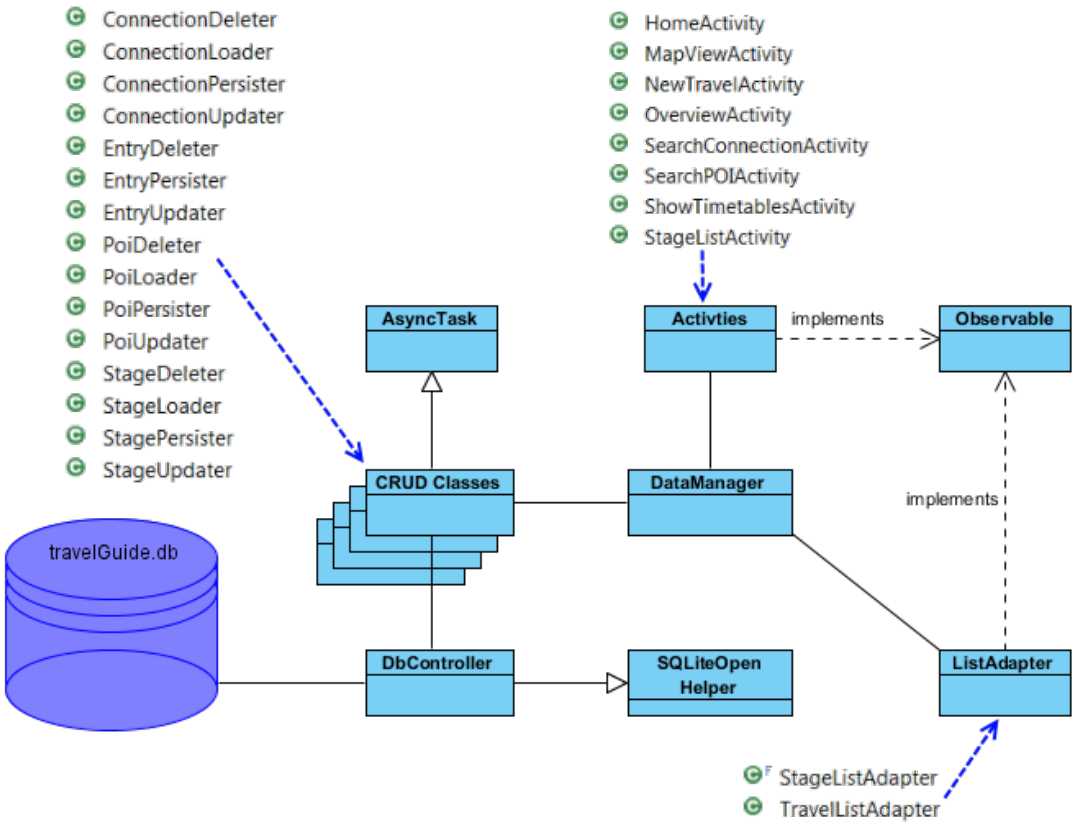


Abbildung 7: DatenManager und Umfeld

Der `DataManager` erstellt bei Bedarf asynchrone CRUD-Klassen. Diese Klassen sind für Datenbankoperationen zuständig und greifen auf den `DbController` zu. Die Aufgaben des `DBControllers` sind:

- Deklaration der Tables und der SQLite Commands
- Ausführung der SQL Commands.
- Definition CRUD-Methoden für die Domainobjekte
- Verwalten der SQLite Datenbank (z.B. Versionsmanagement)

Als Return-Wert der Datenbank-Querys wird ein `Cursor` verwendet. Das Konzept des `Cursors` ist verwandt mit dem eines `Iterators`. Ein `Cursor` ist eine Art `Pointer`, der genau eine Row einer Table referenziert. Er kann jedoch auf andere Rows verschoben werden. Mit der Benutzung der von Android bereits vordefinierten Klasse: `SQLiteOpenHelper` entfallen einige weitere Aufwände, wie zum Beispiel die Deklaration des `Cursors` oder die Implementierung eines Write-Lockers grösstenteils.

10.3. Datenbankmodell

Die Domainobjekte werden in einer SQLite Datenbank gespeichert. Im `DbControl ler` werden die SQLite-Befehle deklariert. Hier im Beispiel des `TravelEntry`-Objekts:

```
1  private static final String TRAVEL_ENTRY_TABLE = "travelEntries";
2
3  public static final String ID = "_id";
4  public static final String TITLE = "title";
5  public static final String START_DATE = "start_date";
6  public static final String END_DATE = "end_date";
7  public static final String ACTIVE = "active";
8
9  public static final String TRAVEL_ENTRY_TABLE_CREATE =
       TRAVEL_ENTRY_TABLE + " ("
10     + ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
11     + TITLE + " STRING NOT NULL, "
12     + START_DATE + " TIMESTAMP, "
13     + END_DATE + " TIMESTAMP, "
14     + ACTIVE + " INTEGER NOT NULL);";
```

Listing 6: TravelEntry SQLite

Daraus ergibt sich dann das folgende ORM (Abb. 8):

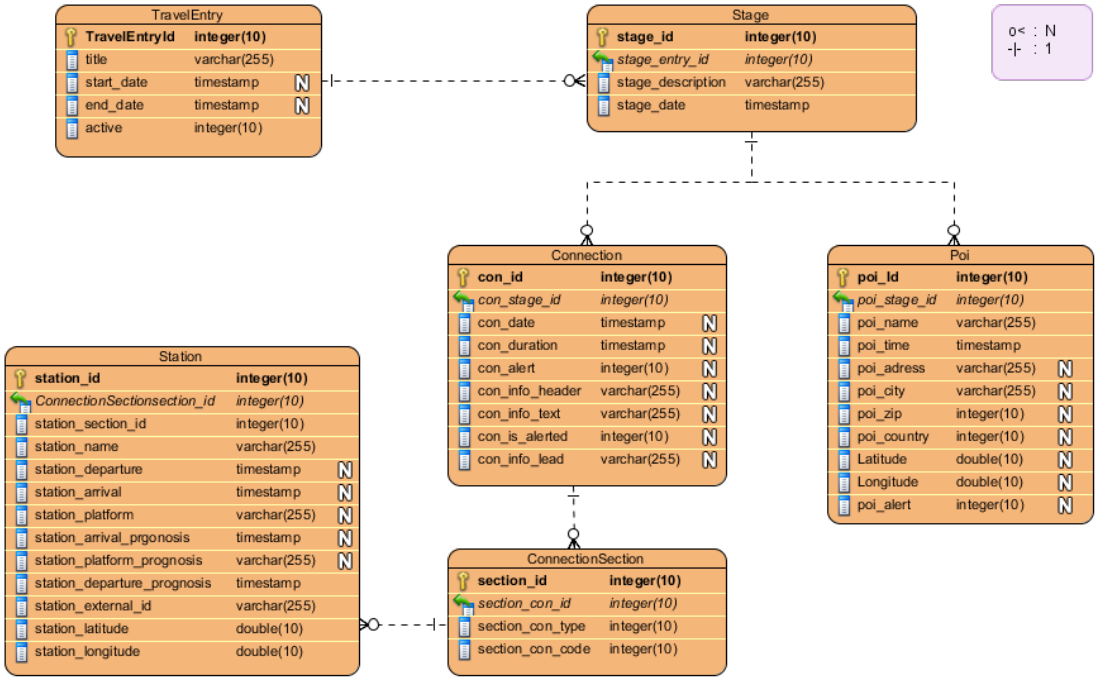


Abbildung 8: Object Relation Model

10.4. Fahrplandaten

Die App bezieht Informationen zu Verbindungen und Stationen direkt aus dem Internet. Dies garantiert, dass die Daten stehts aktuell sind. In diesem Kapitel wird beschrieben, wie die Applikation auf diese Daten zugreift und sie benutzt.

10.4.1. Vorschläge bei der Eingabe

In der Suchmaske für die Verbindungssuche wird dem Benutzer bei der Eingabe automatisch eine Liste von passenden Vorschlägen angezeigt (siehe Abb. 9). Bei jeder Eingabe bzw. Löschung eines Zeichens wird die Liste neu geladen und angezeigt. Das Sequenzdiagramm in Abbildung 10 auf Seite 40 zeigt den Ablauf für das Laden und Anzeigen der Vorschläge.
Bei jeder Benutzereingabe in das entsprechende Feld wird die Methode `onTextChanged()` des Textfeld-Objekts ausgeführt. Durch den Aufruf der Methode wird ein neues Objekt `GetStationListTask` erstellt. Das Objekt erbt von `AsyncTask` und ermöglicht das Ausführen von Aufgaben im Hintergrund und läuft getrennt vom UI Thread. In der `doInBackground()`-Methode wird die Methode `getStations()` der Klasse `SBBControllerJSON` ausgeführt. Als Parameter wird dabei ein String erwartet. Die Methode gibt alle Stationen in einer Liste zurück die zum angegebenen Parameter passen.

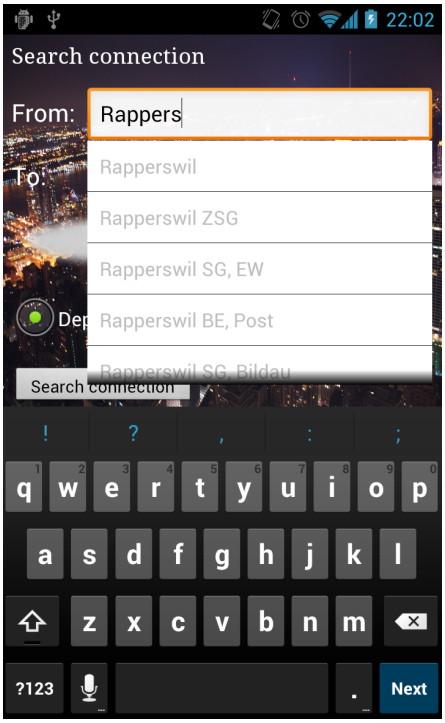


Abbildung 9: Vorschläge für Stationen

Der Abruf der Informationen erfolgt dabei mittels HTTP-GET Request. Der Server antwortet dabei mit den passenden Stationen als JSON-Code. Listing 7 zeigt einen Ausschnitt der Antwort für den Request: `/locations?type=station&query=Rappers`.

```
1  {
2    "stations": [
3      {
4        "id": "008503110",
5        "name": "Rapperswil",
6        "score": "100",
7        "coordinate": {
8          "type": "WGS84",
9          "x": 8.816727,
10         "y": 47.224884
11       }
12     },
13     {
14       "id": "008503667",
15       "name": "Rapperswil ZSG",
16       "score": "96",
17       "coordinate": {
18         "type": "WGS84",
```

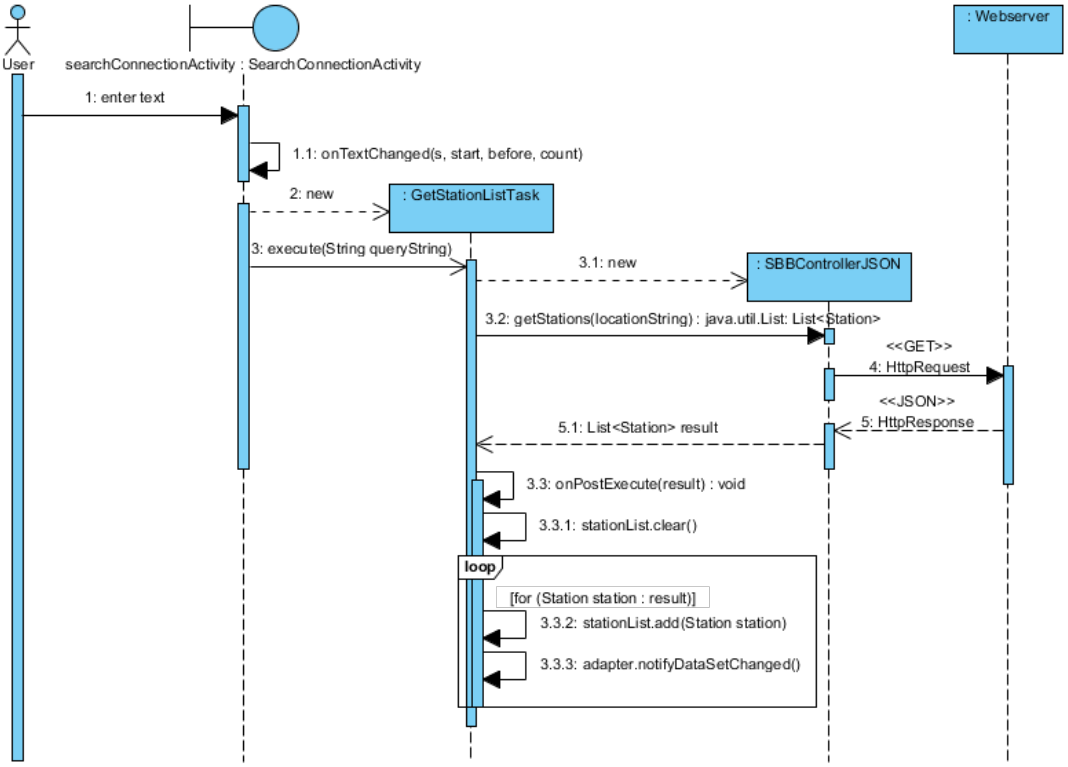


Abbildung 10: Sequenzdiagramm: Vorschläge für Stationen

```
      "x": 8.813554,
      "y": 47.225702
    }
  },
  {
    "id": "008591748",
    "name": "Rapperswil SG, EW",
    "score": "95",
    "coordinate": {
      "type": "WGS84",
      "x": 8.833178,
      "y": 47.232282
    }
  },
  ...
```

Listing 7: getStations

10.4.2. Verbindungen suchen

Nach Eingabe des Abfahrt- und Zielbahnhofs sowie der gewünschten Abfahrts- oder Ankunftszeit sucht die App analog zur SBB-App und SBB-Webseite die vier nächsten Verbindungen. Das Sequenzdiagramm in Abbildung 11 zeigt den internen Ablauf bei der Suche einer Verbindung. Die Suche wird durch den Benutzer ausgelöst, nachdem Abfahrts- und Ankunftsstation sowie die Abfahrts- oder Ankunftszeit eingegeben wurden.
Die App überprüft zuerst ob eine Verbindung zum Internet vorhanden ist. Android bietet dafür eine einfache Möglichkeit mit dem `ConnectivityManager`. Die Methode in Listing 8 überprüft ob eine Netzwerkverbindung vorhanden ist und bei Verfügbarkeit ob auch eine Verbindung aufgebaut ist.

```
1  public boolean isNetworkAvailable() {
2    ConnectivityManager cm = (ConnectivityManager)
3      getSystemService(Context.CONNECTIVITY_SERVICE);
4    NetworkInfo networkInfo = cm.getActiveNetworkInfo();
5    // if no network is available networkInfo will be null
6    // otherwise check if we are connected
7    if (networkInfo != null && networkInfo.isConnected()) {
8      return true;
9    }
10   return false;
11 }
```

Listing 8: Netzwerk-Verfügbarkeit

Nach erfolgreicher Überprüfung der Verbindung wird ein neuer Thread gestartet. Der Thread ruft die Methode `getTimeTableEntries()` mit den vier Parametern Abfahrtsbahnhof, Ankunftsbahnhof, Zeit und Zeitmodus. Der Zeitmodus ist ein Boolean Wert und definiert wie die Zeit interpretiert wird. Ist der Wert auf `true` gesetzt, wird die Zeit als Ankunftszeit interpertiert andernfalls wird die Zeit als Abfahrtszeit interpretiert.
Die Methode `getTimeTableEntries()` der Klasse `SBBControllerJSON` löst einen HTTP-GET Request beim Webserver aus und erhält als Antwort die nächsten vier Verbindungen im JSON Format. Listing 10 auf Seite 64 im Anhang zeigt den Ausschnitt für eine Verbindung der Abfrage:
`/connections?from=Rapperswil&to=Zurich&date=2012-06-01&time=16:00`
Die Verbindungen im JSON-Code werden dann im `SBBControllerJSON` geparst und als Liste vom Typ `Connection` gespeichert. Die Liste wird an einen Dialog zur Anzeige weitergegeben. Der Benutzer kann nun eine der vier `Connection`'s zum speichern auswählen (siehe Abb. 12(a) auf Seite 43). Die ausgewählte Verbindung wird dem `DataManager` zur Persistierung (in Datenbank) übergeben.

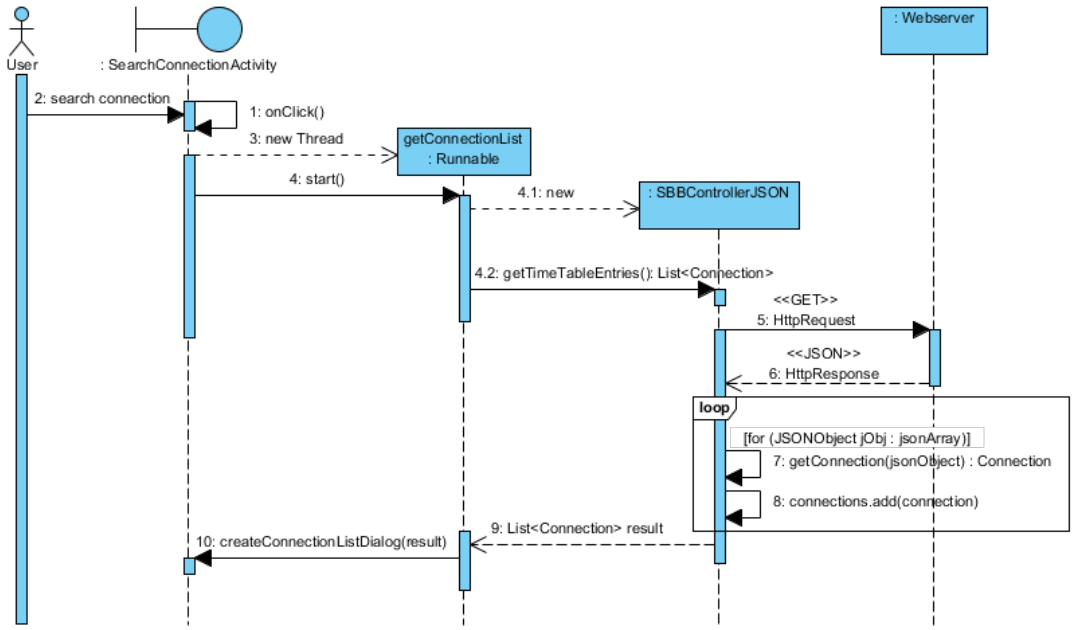


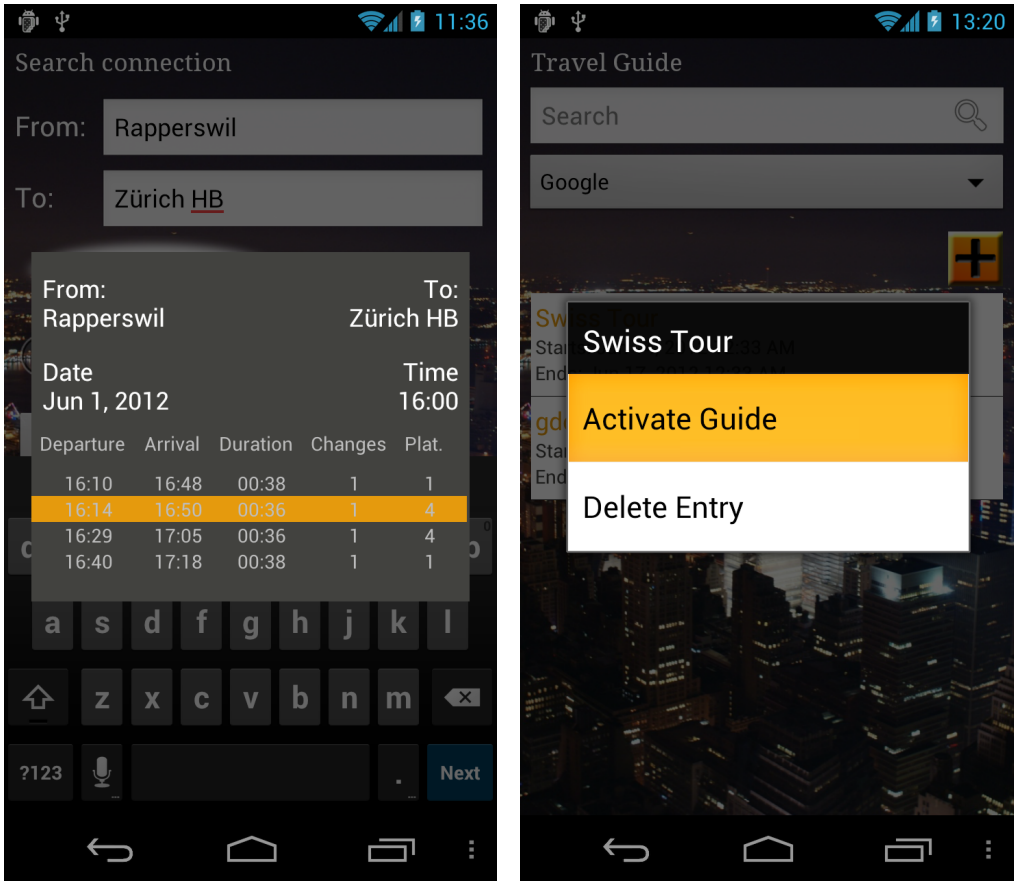
Abbildung 11: Sequenzdiagramm: Verbindungssuche

10.4.3. Benachrichtigung bei Verspätungen oder Unterbrüchen

Erfasste Reisen können im Homescreen mit einem Longclick aktiviert werden (siehe Abb. 12(b) auf Seite 43).

Dieser Vorgang startet einen Hintergrundprozess (`Service`), der alle 15 Minuten alle vorhandenen Verbindungen der aktivierten Reise überprüft. Falls sich eine Verbindung geändert hat (Plattform, Verspätungen) oder Störungsmeldungen vorhanden sind, wird für diese Verbindung eine Benachrichtigung bzw. `Notification` angezeigt. Das Sequenzdiagramm in Abbildung 13 auf Seite 44 zeigt den systeminternen Ablauf für die Überprüfung und Anzeige der `Notification`. Das anklicken des Menupunktes "Activate Guide" startet die Methode `onContextItemSelected()`. Diese Methode ruft die Methode `setRepeatingAlarm()` mit der Id der aktivierten `TravelEntry` auf. Damit der Service alle 15 Minuten aufgerufen wird, kreiert `setRepeatingAlarm()` den AlarmManager und setzt die dazu nötigen Werte. Anschliessend wird die Methode `setRepeating()` des Objekts `AlarmManager` aufgerufen. Dies setzt den Service `TravelGuideService` im `AlarmManager` des Betriebssystems fest.

Ab sofort wird im Viertelstundentakt die Methode `onStart()` des Services aufgerufen. Der Aufruf wird auch ausgeführt wenn die Hauptapplikation nicht läuft oder nur im Hintergrund ist. Der Service lädt sich alle vorhandenden Verbindungen der aktiven Reise und überprüft sie mit aktuellen Daten im Internet. Dies wird mit dem `AsyncTask` `CheckTimetableTask` erreicht. Der Task führt die Methode `isConnec`



(a) Liste der Verbindungen (b) Guide aktivieren

Abbildung 12: Verbindungsliste und Guide aktivieren

`tionOnTime()` des Controllers `SBBControllerJSON`. Dabei wird die Methode `getTimeTableEntries()` ausgeführt. Dessen Vorgehen ist im Abschnitt 10.4.2 auf Seite 41 erläutert.

Da eine Abfrage immer vier Verbindungen liefert, muss die zu überprüfende Verbindung mit der Liste verglichen werden und die korrekte `Connection` eruiert werden. Bei Übereinstimmung werden die Felder `departureTimePrognosis`, `arrivalTimePrognosis`, `platformPrognosis` der Klasse `Station` sowie das Feld `informationHeader` der Klasse `Connection` überprüft. Falls eines dieser Felder nicht leer ist, wird die Verbindung mit den neuen Informationen zurückgegeben. Andernfalls wird `null` zurückgegeben. Die `PostExecute`-Methode des `CheckTimetableTask` erstellt bei erhaltener Verbindung eine neue `Notification` und gibt diese mit der `notify()` Methode der Klasse `NotificationManager` mit. Die `notify()`-Methode fügt die Benachrichtigung in die Statusleiste und in das Benachrichtigungsfenster (siehe Abb. 14(a) auf Seite 45).

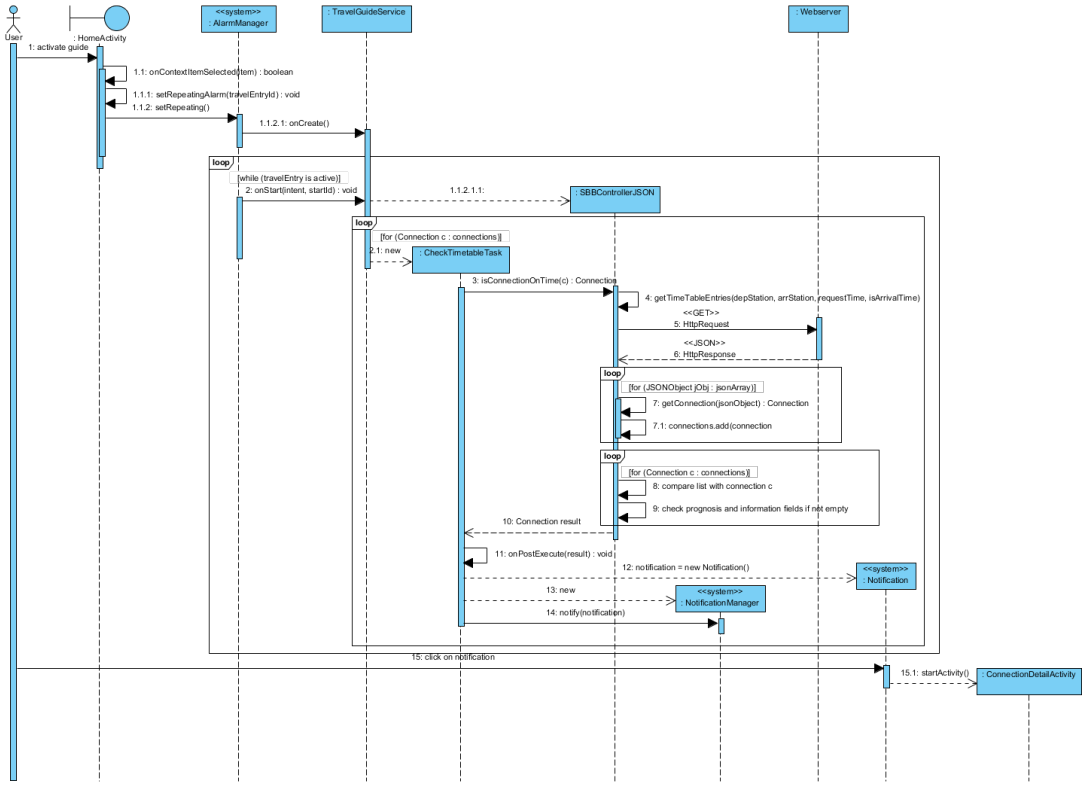


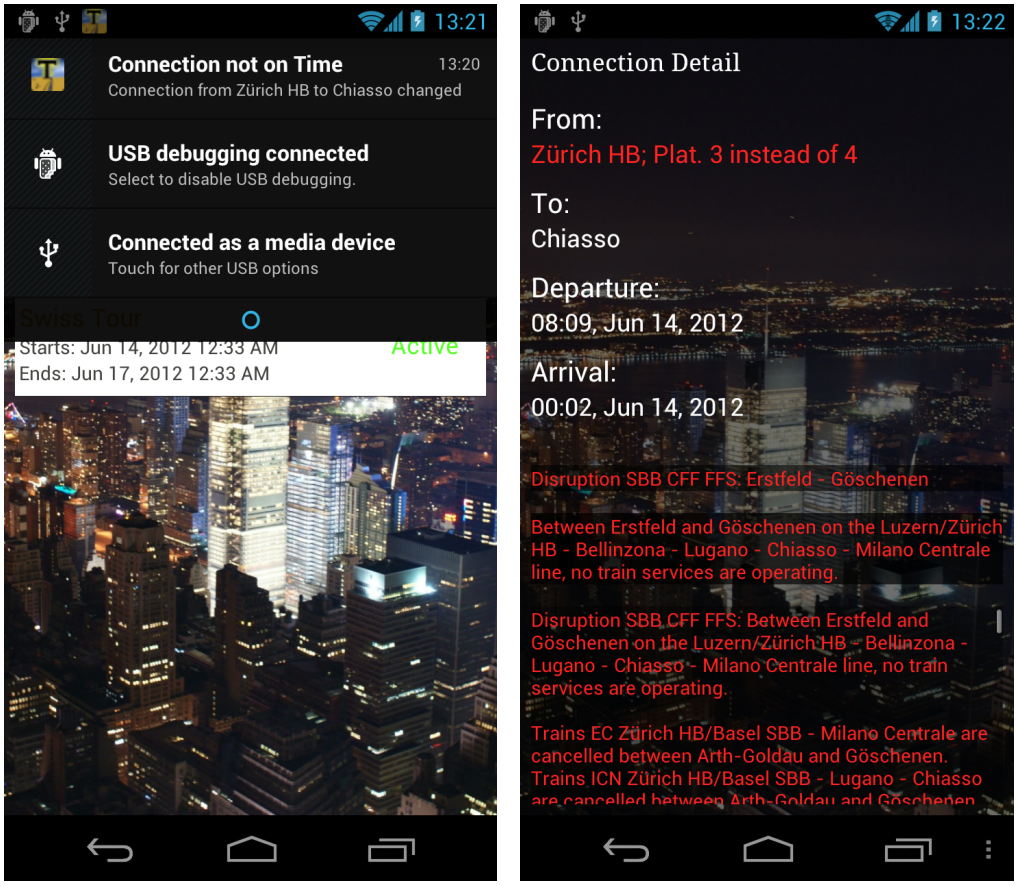
Abbildung 13: Sequenzdiagramm: Benachrichtigung bei Verspätungen

Der Benutzer hat jederzeit die Möglichkeit die Benachrichtigung aufzurufen. Mit einem Klick werden die detaillierten Informationen der betroffenen Verbindung angezeigt. Das anklicken der `Notification` startet Klasse `ConnectionDetailActivity` und zeigt eine Detailansicht der Verbindung an (siehe Abb. 14(b) auf Seite 45).

10.5. POI-Daten

Die App ermöglicht das Suchen und Speichern von POIs. Den zu speichernden POIs muss dabei ein Datum und eine Uhrzeit mitgegeben werden, damit diese in der Reiseübersicht eingeordnet werden können.
Der Benutzer hat zwei Möglichkeiten nach einem POI zu suchen. Die erste Möglichkeit besteht darin, POIs in der Nähe seines aktuellen Standortes zu suchen. Dies erfordert die Ermittlung der aktuellen Position.
Alternativ kann der Benutzer auch einen Ort bestimmen. Der POI wird dann im Umkreis dieses Ortes gesucht.

Die Abbildung 15 auf Seite 46 beschreibt beide Varianten in einem Sequenzdiagramm.



(a) Notification in der Statusleiste und im Benachrichtigungsfenster

(b) Detailseite der benachrichtigten Verbindung

Abbildung 14: Notifications und Detailansicht

10.5.1. Benutzerdefinierter Ort

In der Suchmaske kann der Benutzer mit dem Auswahl-Button entscheiden, einen POI im Umkreis eines selbst gewählten Ortes zu suchen. Nach Betätigung des Auswahlbuttons erscheint ein Eingabefeld. Der Benutzer wird dort aufgefordert, den Ort einzugeben (siehe Abb. 16(a) auf Seite 47). Jegliche Eingaben in das Textfeld lösen die Methode `onTextChanged()` aus. Der Methodenaufruf erstellt das `AsyncTask` Objekt `GetPlaceListTask`. Diese Klasse ruft im Hintergrund und getrennt vom UI Thread die Methode `getCities()` der Klasse `PoiSearchController` auf. Die Methode führt einen HTTP-GET Request auf die Adresse:
`http://ws.geonames.org/searchJSON?featureClass=P&maxRows=`

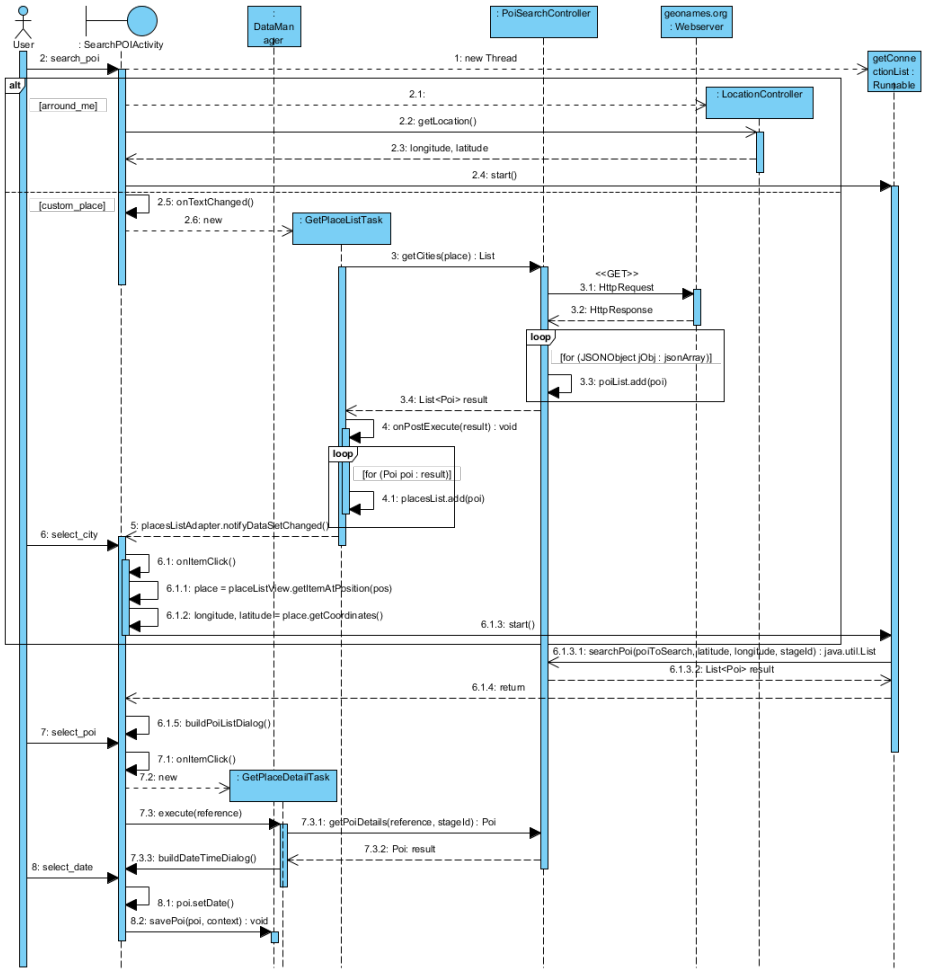


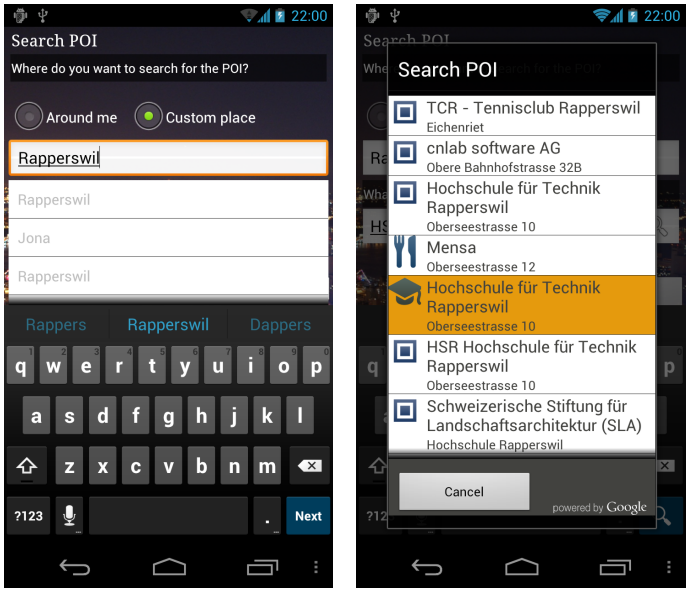
Abbildung 15: Sequenzdiagramm: POI Suche

`10&country=CH&q=[CityString]` aus. Als Antwort erhält die Methode eine Liste von Ortschaften die zum String 'CityString' passen. Die Liste mit den Orten wird zur Dropdown-Liste hinzugefügt und ermöglicht dem Benutzer die einfache Auswahl des passenden Ortes.

10.5.2. Suche in der Umgebung

Für die Suche eines POIs in der aktuellen Umgebung des Benutzers muss die App die Koordinaten feststellen. Dies geschieht mit der Klasse `LocationController`. Die Klasse besitzt die Methode `getLocation()` und erwartet als Parameter eine Callback-Methode. Sobald der Controller die Position herausgefunden hat, wird die Callback-Methode `gotLocation` ausgeführt. Diese Methode speichert die aktuellen Koordinaten in der Klasse ab.

10.5.3. POI Suche



(a) Benutzerdefinierter Ort für POI Suche (b) Liste der gefundenen POIs

Abbildung 16: POI Suche

Nachdem die Koordinaten des benutzerdefinierten Ortes oder der aktuellen Position ermittelt sind, wird die eigentliche Suche ausgelöst. Dabei wird zuerst ein neuer Thread gestartet und darin die Methode `searchPoi` des Controllers `PoiSearchController` ausgeführt. Die `searchPoi`-Methode führt einen HTTP-Get Request auf die Google Places API aus und erhält eine Liste von passenden POIs im JSON-Format als Antwort zurück. Die Antwort wird geparst und als Liste von POIs zurückgegeben.
Ein Dialog zeigt anschliessend die Liste übersichtlich für den Benutzer dar (siehe Abb. 16(b) auf Seite 47). Sobald ein POI aus der Liste ausgewählt wird, muss noch eine Zeit und ein Datum angegeben werden, bevor der POI in der Datenbank gespeichert wird.

10.6. Übersicht der Domainelement

Eines der aufwändigsten Klassen der App ist die `OverviewActivity`. Sie ist zuständig für die Darstellung der Domainobjekte. Mit dieser Klasse wird das Ziel angepeilt, dem Benutzer eine möglichst übersichtliche Darstellung und ein bequemes Handling für seine Reiseplanung zu bieten.
Die Relationen der `OverviewActivity` werden hier kurz erläutert:

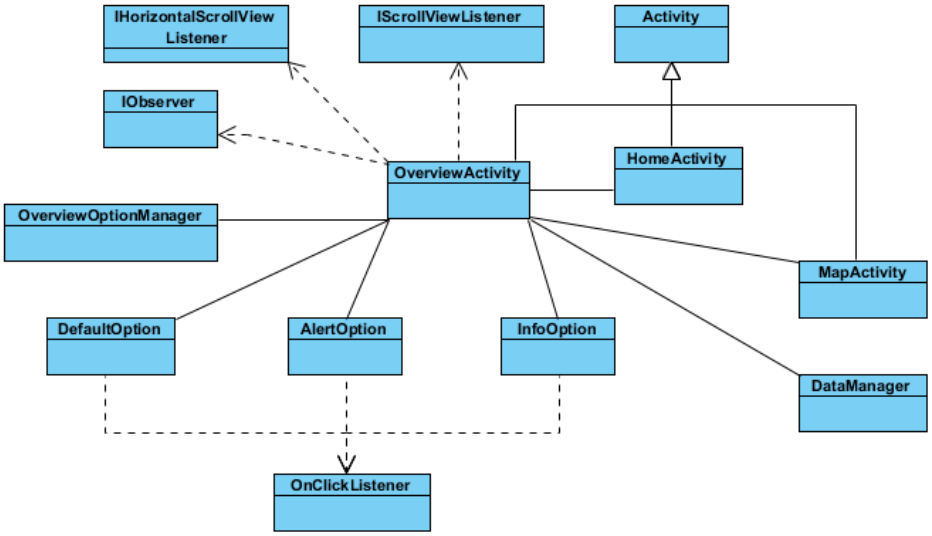


Abbildung 17: Sequenzdiagramm: OverviewActivity

Options

Die verschiedenen Option-Klassen (`DefaultOption`, `AlertOption`, `InfoOption`) werden benötigt um zusätzliche Informationen darzustellen (siehe Abb. 18). Je nach Betätigung der Button-Elemente wird eine zusätzliche View hinzugefügt. Für die Route, Map und Delete kommt eine `DefaultOption`-, für weitere Informationen zum Domainelement die `InfoOption`- und für den Alert eine `AlertOption`-View hinzu.

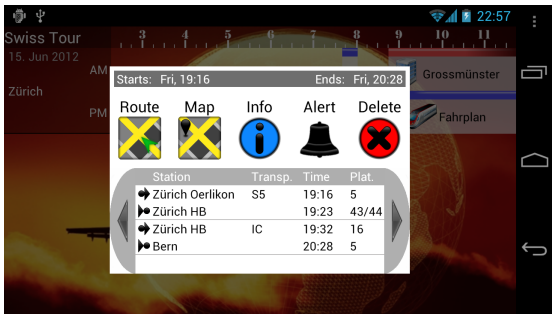


Abbildung 18: Option-Dialog in der Übersicht

OverviewOptionManager

Die ganzen Options werden vom `OverviewOptionManager` verwaltet. Dieser hat die Aufgabe die Grösse der Views zu bestimmen und die aktiven Option Views zu organisieren. Der `OverviewOptionManager` ist eine Art Hilfs- und Managerklasse für den Dialog.

10.6.1. Initialisierungsvorgang der Domainobjekte

Von der `HomeActivity` wird die `TravelEntry`-ID mitgegeben. Damit werden in der Übersicht alle `Stage`s und die dazugehörigen Objekte über den `DataManager` geladen.
Weiter müssen die `Poi`s und `Connection`s auf der Übersicht platziert werden.

`initStagListView()`:
Zuerst werden die `Stage`s in der linken `ScrollView` initialisiert. Besitzt eine `Stage` keine Elemente, so wird sie nicht dargestellt. Die Anderen werden einer `View` zugeteilt und mit Datum, Beschreibung versehen.

`initGraficElements()`:
Nachdem Punkt eins abgeschlossen ist, wird die Initialisierung der `Poi`s und `Connection`s eingeleitet. Jede Stage besitzt zwei `LinearLayout`s. Einen für AM und den andern für PM. auf diesen beiden Layouts werden später die Elemente platziert. In dieser Methode werden dann auch die Abstände (oder Position auf dem Layout) bestimmt. Die Abstände sind jeweils Abhängig von dem voran gegangenen Element. Der Abstand und die Elementlänge minus den Abstand des nächsten Elements ergeben dann die nächste Position. Das Ganze wird in "Zeitpixel" gerechnet. Diese Einheit ergibt sich aus der Grösse der View-Element und des 12 Stunden Bereichs. Folgende Berechnung zeigt wie ein Element zu seiner Position kommt:

$$\frac{(element.hour*60+element.min)*timline.pxlWidth}{12*60} - prevDistance + prevElementSize \quad (1)$$

Nachfolgend sind in einem Modell die Entscheidungen dargestellt, damit die Übersichtselemente verschiedenen Grössen zugeteilt werden können (Abb. 19). Die Grösse der `Poi`s ist auf best fit ausgelegt. Bei den `Connection`s ist die Grösse abhängig von deren Dauer.

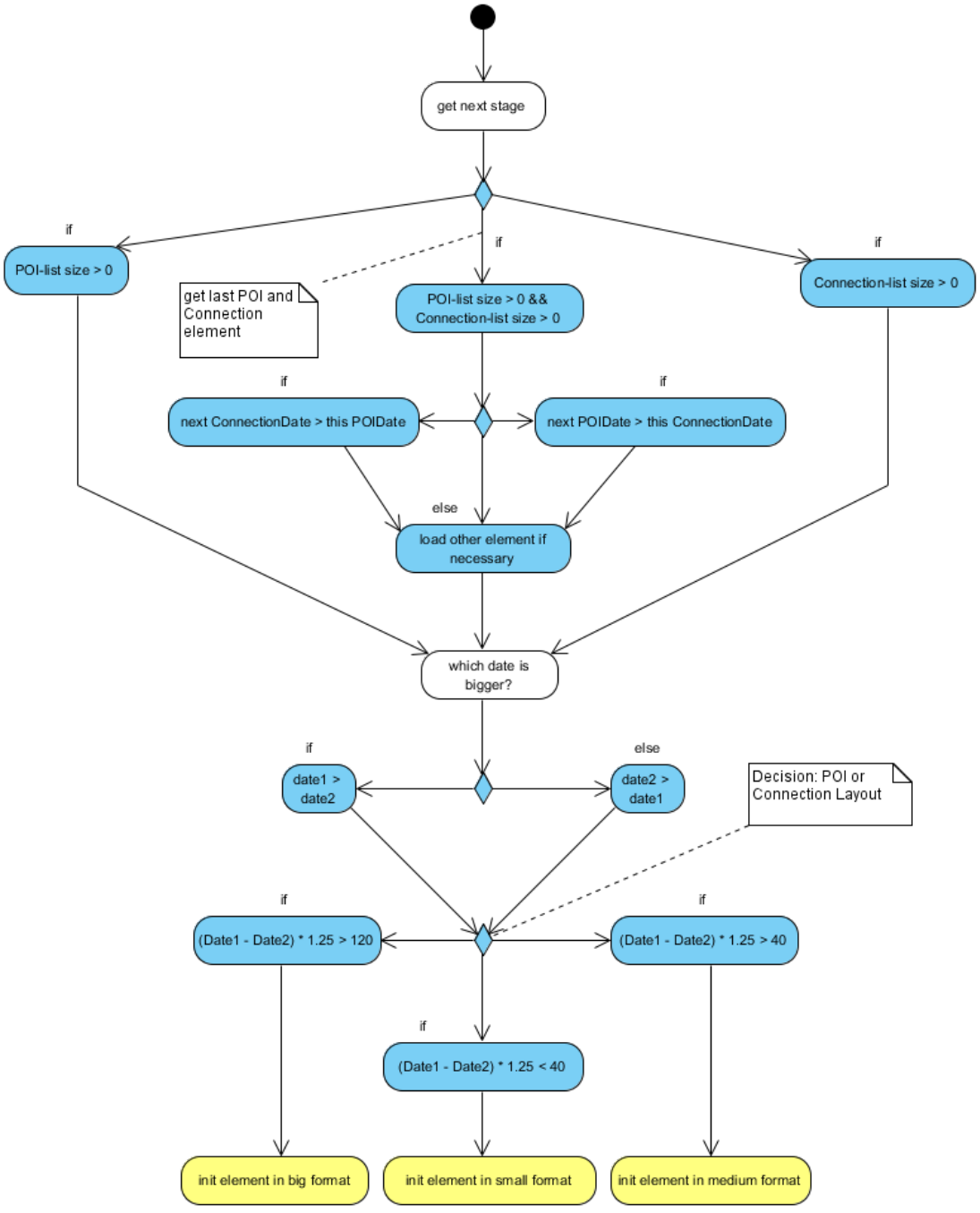


Abbildung 19: Initialisierung der POIs und Fahrpläne

10.7. Karte

In der App gibt es zwei Varianten zu der Kartenansicht zu gelangen. Die Eine ist in der `HomeActivity`. Im Menu gibt einen Punkt Karte. Diese Karte besitzt die Standard Google Maps Funktionen. Zu der zweiten Karte gelangt man, indem man bei einem Element in der `OverviewActivity` die Optionen öffnet. Hier wählt man Karte oder Route aus und man gelangt in eine etwas andere Google Map-Ansicht. Dieser Teil wird im folgenden Abschnitt weiter erklärt.

Die Grundlagen der `MapView` sind bereits durch die Google Maps API(Google, 2012) gegeben. Der Aufbau der der `MapViewActivity` sieht folgendermassen aus:

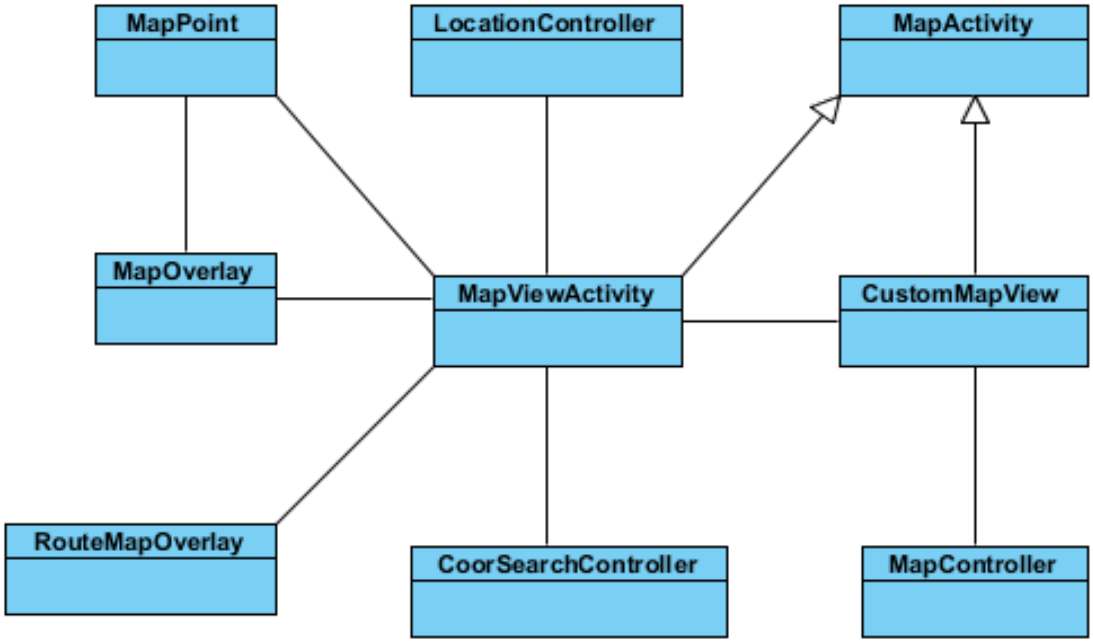


Abbildung 20: Aufbau MapViewActivity

MapPoint

Das `MapPoint`-Objekt wird benötigt um Koordinaten auf der Karte zu beschreiben.

Route- und MapOverlay

`Overlays` werden benutzt um zusätzliche Aktionen oder Bilder (Markierungen) auf der Karte darzustellen.

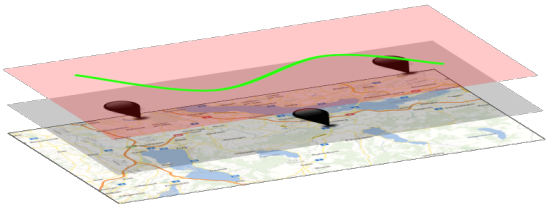


Abbildung 21: Map Overlays

CoorSearchController

Mit diesem asynchronen Controller wird nach Adressen gesucht. Mittels: `HttpGet("http://maps.googleapis.com/maps/api/geocode/json?address="+place+"&sensor=true")` wird dann eine Anfrage an Google gesendet. Aus der Antwort können dann sämtliche Informationen entnommen werden.

```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "1600",
          "short_name" : "1600",
          "types" : [ "street_number" ]
        },
        {
          "long_name" : "Amphitheatre Pkwy",
          "short_name" : "Amphitheatre Pkwy",
          "types" : [ "route" ]
        },
        {
          "long_name" : "Mountain View",
          "short_name" : "Mountain View",
          "types" : [ "locality", "political" ]
        },
        {
          "long_name" : "Santa Clara",
          "short_name" : "Santa Clara",
          "types" : [ "administrative_area_level_2", "political" ]
        },
        {
          "long_name" : "California",
          "short_name" : "CA",
          "types" : [ "administrative_area_level_1", "political" ]
        },
        {
          "long_name" : "United States",
          "short_name" : "US",
          "types" : [ "country", "political" ]
        },
```

```
35          {
36            "long_name" : "94043",
37            "short_name" : "94043",
38            "types" : [ "postal_code" ]
39          }
40        ],
41        "formatted_address" : "1600 Amphitheatre Pkwy, Mountain View,
              CA 94043, USA",
42        "geometry" : {
43          "location" : {
44            "lat" : 37.42291810,
45            "lng" : -122.08542120
46          },
47          "location_type" : "ROOFTOP",
48          "viewport" : {
49            "northeast" : {
50              "lat" : 37.42426708029149,
51              "lng" : -122.0840722197085
52            },
53            "southwest" : {
54              "lat" : 37.42156911970850,
55              "lng" : -122.0867701802915
56            }
57          }
58        },
59        "types" : [ "street_address" ]
60      }
61    ],
62    "status" : "OK"
63  }
```

Listing 9: Location Anfrage (Response)

Momentan werden für die Suche jedoch nur die Koordinaten benötigt. Die Position wird danach im Kartenzentrum zentriert.

CustomMapView

Die `CustomMapView` beinhaltet die Logik für den Doppelklick-Zoom.

11. Tests

Tests sind ein wesentlicher Bestandteil der Qualitätssicherung bei der Softwareentwicklung. Die Testabdeckung am Ende des Projekts beträgt ca. 10%. Abbildung 22 zeigt die Testabdeckung während des gesamten Projektablaufs.

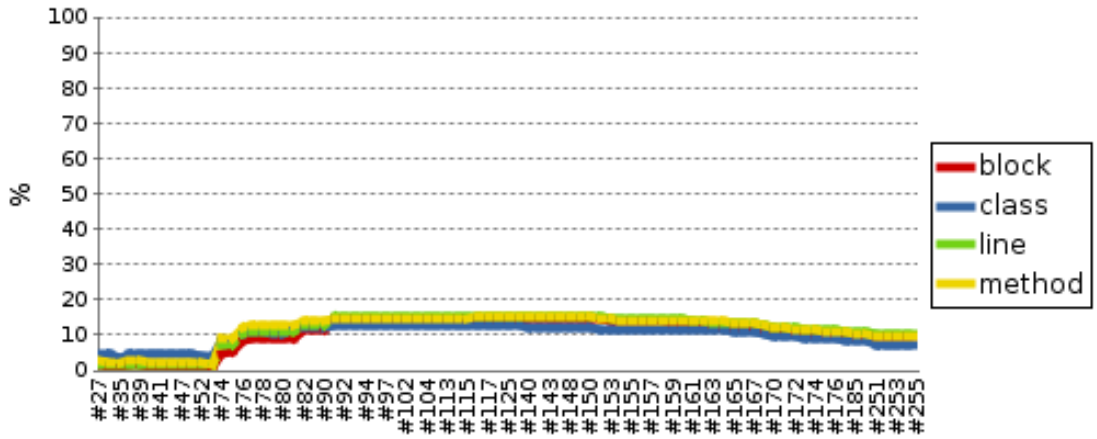


Abbildung 22: Testabdeckung

Getestet wird hauptsächlich die Anbindung an die Datenbank. Der GUI-Test ist dem Android Monkey Test erfolgreich durchgeführt worden:

```
Events injected: 10000
:Dropped: keys=229 pointers=591 trackballs=0 flips=0
## Network stats: elapsed time=98192ms (0ms mobile, 0ms wifi,
98192ms not connected)
```

Ein Usability-Test wurde ebenfalls mit einer Testperson anhand folgendem Szenario durchgeführt:

1. Neue Reise erstellen
2. Zwei Stages hinzufügen
3. Stage No.1 einen POI hinzufügen
4. Stage No.2 einen POI und eine Fahrplan hinzufügen
5. In die Übersicht wechseln
6. Kartenfunktion aufrufen
7. Einen neuen POI hinzufügen

Die Auswertung des Tests:

- Benutzer wusste bei Punkt 3 nicht genau, wo er jetzt weitere Elemente hinzufügen kann.
- Der POI und die Connection werden etwas zu direkt gespeichert. Der Benutzer wusste nicht recht, ob sie nun gespeichert sind oder nicht und dies obwohl ein `Toast` eine Statusmeldung anzeigt. Dieser ist jedoch leicht übersehen. Ausserdem war ihm nicht klar ob durch ein Klick auf das Element noch weitere Informationen dargestellt werden.
- Bei der Alarmzeit setzen in der Übersicht wusste der Tester nicht genau, auf welche Zeit der Alarm nun genau gesetzt wurde.

Diese Punkte müssen bei der Weiterentwicklung noch etwas näher betrachtet werden und eventuell mit zusätzliche Informationen versehen werden.

12. Zusammenfassung und Auswertung der TravelGuide App

12.1. Erfahrungen und Herausforderungen

Dieses Abschnitt befasst sich mit den Herausforderungen und Erfahrungen, welche die App bei der Umsetzung geboten hat. Genauere Informationen zu der Umsetzung sind im Abschnitt 10 auf Seite 34 zu finden.

12.1.1. Overview

Die Übersicht war eine der am herausfordernsten Activities der App. Android bietet zum jetzigen Zeitpunkt keine Vorlage, welche eine dynamische Übersicht wie in der TravelGuide App ermöglicht. Normale `ScrollView`s sind in horizontale oder vertikale Scrollrichtung begrenzt. Für die Übersicht musste man also eine ganz neue Richtung einschlagen. Mittels der `onTouchEvent`-Methode musste jede Bewegung des Benutzers erfasst, eingeordnet und durch Berechnungen kontrolliert oder eingeschränkte werden. Zudem kommen noch zwei weitere eigene `Scrollview`s. Die obere dient als Zeitstrahl, die Linke als Etappen Auflistung. Diese beiden `Scroll views` mussten zudem mit den Bewegungen der Hauptsicht synchron gehalten werden und umgekehrt.
Ein ständiges Problem bei Android sind auch die verschiedenen mobilen Geräte. Jedes Geräte hat verschiedene Auflösungen. Die Übersicht muss also mit allen Geräten kompatibel sein. Das heisst die `Views` und Aktionen mussten dynamisch gebaut werden, damit es keine Rolle spielt, ob man nun eine Auflösung von 800x480 oder 1280x720 besitzt. Doch je dynamischer die gesamte View, desto statischer das deklarative GUI. Android bietet hier sogenannte ”Dense Pixel” (`dp`). Damit lassen sich GUI-Elemente auf eine bestimmte Grösse binden und passen sich trotzdem dynamisch im Verhältnis zu den verschiedenen Bildschirmauflösung an.

Die POI- und Fahrplanelemente der Übersicht waren die zweite Herausforderung. Am Anfang ging es darum die Elemente richtig unter dem Zeitstrahl zu platzieren. Dazu musste man zuerst die Distanz vom linken Rand berechnen, welche sich von den Stunden und Minuten des Elements ableiten lässt.
Weiter ging es mit mehreren Elementen auf einer Reihe. Da ein `LinearLayout` benutzt wird muss die Distanz des vorangehenden Elements von der Gesamtdistanz des nächsten Elements subtrahiert werden, damit es in die richtige Position erhält.
Da POIs und Fahrpläne völlig unterschiedliche Objekte sind müssen diese beim durchlaufen des Initialisierungsprozesses separat behandelt werden. Dabei kann es jedoch vorkommen, dass Elementüberlappungen vorkommen. Somit werden alle Elemente in verschiedene Grössen eingeteilt. Das heisst man muss nun in der endgültigen Berech-

nung zusätzlich die Grössen miteinfliessen lassen.

12.1.2. Karte

Bei der Karte wurde der Aufwand etwas unterschätzt. Zu Anfang gab es einige Schwierigkeiten bei der Einarbeitung in das etwas andere Fachgebiet. Vorallem musste man zuerst sehr viele Informationen sammeln, damit man einen Schritt weiter kommen konnte. Das Grundgerüst ist jedoch schnell zu verstehen (`MapView` und `Overlays`). Die Funktionalitäten, welche man der Kartenansicht hinzufügen kann sind sehr weitreichend. Darum musste darauf geachtet werden, dass man sich nicht zu sehr vom aktuellen Fokus entfernt und versucht immer noch ein paar Features zusätzlich zu implementieren.
Am Schluss hat vor allem die Suche von weit entfernten Orten für etwas Aufregung gesorgt. Auf vielen Webseiten wurde erwähnt, dass man nur POIs oder Places in der Nähe oder zu einer bestimmten Region suchen kann (Google Places). Später jedoch konnte dies über eine Google Web-API auch anders gelöst werden.

12.1.3. SBB-Schnittstelle

Die SBB-Schnittstelle war ebenfalls eine Knacknuss die unterschätzt wurde. Die offizielle SBB-App verwendet eine XML-Schnittstelle die nicht dokumentiert ist. Die Funktionsweise musste daher mühsehlig durch ”Reverse-Engineering” gelernt werden. Glücklicherweise haben wir während des Projektverlaufs eine OpenSource-Alternative entdeckt, die die Daten aus der XML-Schnittstelle parst und in einfach verständlichem JSON-Format ausgibt. Leider ist dieses Projekt noch relativ jung und daher instabil und unvollständig. Daher wurde der Code der Webanwendung auf einem eigenen Server installiert und entsprechend modizfizert, damit auch alle nötigen Daten für die App vorhanden sind.

12.2. Auswertung User Stories

In der folgenden Tabelle ist der Status der User Stories aufgelistet.

ID	User Story	Status
US 1	Eingabemaske für Reise Als Benutzer will ich meine Daten einfach eingeben können.	O.K.

Weiter auf der nächsten Seite

Tabelle 9 – Fortsetzung von vorheriger Seite

ID	User Story	Status
US 2	ÖV-Verbindungen Als Benutzer will ich Fahrpläne suchen können.	O.K.
US 3	Notifikation Als Benutzer will ich immer über aktuelle Änderungen informiert werden.	O.K.
US 4	GPS-Position ermitteln Als Benutzer will ich meine aktuelle Position ermitteln können.	O.K.
US 5	Alarm Funktion Als Benutzer will ich eine Alarm Funktion für meine Termine, sodass ich auf wichtige Ereignisse erinnert werde.	90%*
US 6	Reise teilen Als Benutzer will ich meine Daten an andere Personen weitergeben können, sodass mehrere Personen die selbe Planung für sich benutzen können.	Rejected
US 7	Suche von POIs Als Benutzer will ich die Möglichkeit haben POIs zu suchen und diese meiner Reise hinzuzufügen (CRUD).	O.K.
US 8	GPS-History Als Benutzer will ich meine Tour zurückverfolgen können.	Rejected
US 9	InApp-Suche Als Benutzer will ich auftretende Fragen sofort im Internet suchen können, sodass ich nicht immer die Programme wechseln muss.	Rejected
US 10	Karte Als Benutzer will ich meine Ziele auf einer Karte zu finden wissen.	O.K.
US 11	Route Als Benutzer will ich die Möglichkeit haben, mich zum Ziel leiten zu lassen.	O.K.
US 12	Übersichtfunktion Als Benutzer will ich meine Reise in einer Gesamtübersicht betrachten können.	O.K.
US 13	Übersetzer Als Benutzer will ich mich im Ausland verständigen können.	O.K.
US 14	Daten Persistierung	O.K.

Weiter auf der nächsten Seite

Tabelle 9 – Fortsetzung von vorheriger Seite

ID	User Story	Status
	Als Benutzer will ich meine Daten persistieren können.	

Tabelle 9: Auswertung User Stories

*CRUD, Logik und Einbindung in der App vorhanden. Es fehlt die Alarm Registrierung beim Android System.

12.3. Future Work und bekannte Bugs

ID	Future Work/Bug	Typ
TODO 1	Refactoring & Testing Um Erwartungen an die Funktionalitäten zu erfüllen, musste das Refactoring der Klasse etwas zurückgestuft werden.	Future Work
TODO 2	Alarm Funktion US 5 Die Anforderungen für die Alarm Funktion wurde schon grösstenteils implementiert. In einer nächsten Version muss die Abfrage der gestellten Alarme überprüft und mittels `AlarmManager` dem System übergeben werden.	Future Work
TODO 3	Reise teilen US 6 Eine grosser Teil der Weiterentwicklung wird die Share-Funktion sein. Die Herausforderung wird darin bestehen die Reisedaten von einem Mobilgerät zu einem Anderen zu versenden. Dabei kommen folgende Übertragungsmedien in Frage: Bluethooth, Wi-Fi und UMTS(GSM). Die Übertragen soll bestenfalls per JSON stattfinden.	Future Work
TODO 4	GPS-History US 8 Bei der GPS-History handelt es sich um ein eher spezielles Feature. Dabei kann der `LocationController` benutzt werden, dass bei einer aktiven Reise zyklisch die neusten GPS-Positionen abgefragt werden und diese persistiert werden. Bei dieser Story muss noch eine neue `MapActivity` eingerechnet werden, da die Punkte später auf einer Kartenansicht dargestellt werden sollen.	Future Work
TODO 5	Kartenfunktionen	Future

Weiter auf der nächsten Seite

Tabelle 10 – Fortsetzung von vorheriger Seite

ID	Future Work/Bug	Typ
	Das Hinzufügen der POIs auf der Karte sollte noch etwas verbessert werden.In der Kartenansicht gibt es noch sehr viele zusätzliche Funktionen, welche man hinzufügen könnte. Einge sind: • Wechsel von Auto, ÖV oder Route zu Fuss • Mehrere Pins setzen • Mehrere Routen (Gesamtroute) darstellen • Anpassen der Suchfunktion (GUI) • Notizfunktion auf der Karte • Position von befreundeten Personen auf der Karte darstellen • u.v.m.	Work
TODO 6	Liste von POIs und Connections in `StageListActivity` Um dem Benutzer eine einfachere und schnellere Bearbeitungsmöglichkeit seiner Einträgen zu bieten müssten eine bis zwei neue Views erstellt werden. In diesen Views sollen die Elemente übersichtlich dargestellt werden.	Future Work
TODO 7	Eingabevalidierung Bei den Eingabefeldern müssen die Validierungen noch erweitert werden. Somit ist dem Benutzer eine unterstützende Umgebung geboten und Fehler werden vermieden.	Future Work
TODO 8	Allgemeine Settings Für Spätere Weiterentwicklungen, wie die GPS-History sollten allgemeine Settings zur Verfügung stehen. In diesen werden unter anderem folgende Einstellungen ermöglicht: • GPS-Abfragezyklus (min) • Fahrplancheck-Intervall (min) • Offlinemod	Future Work
TODO 9	Frühere/spätere Verbindungen suchen in `OverviewActivity` Das Gerüst dafür ist schon Vorhanden. Man müsste die bereits bestehende Fahrplanabfrage noch etwas erweitern und mit den GUI-Elementen in der `OverviewActivity` verbinden.	Future Work
TODO 10	Overview	Future

Weiter auf der nächsten Seite

Tabelle 10 – Fortsetzung von vorheriger Seite

ID	Future Work/Bug	Typ
	Die `OverviewActivity` muss noch etwas angepasst werden. Ein Teil davon ist das erneute Laden der Elemente (nur den `LinearLayout` den es betrifft), sollte in der Kartenansicht ein weitere POI hinzugefügt worden sein. Dazu soll der Spinner beim initialisieren nicht unterbrochen werden können.	Work
TODO 11	Overview - Views In der Overview wird beim Scrollen teils die linke und obere View verschoben und blockiert. Dies kann beispielsweise ausgelöst werden durch abrollen des Fingers beim Scrollen oder bei zu schnellem Scrollen. Dieser Bug muss noch genauer untersucht werden und wird wahrscheinlich noch einige Zeit in Anspruch nehmen.	Bug
TODO 12	Overview - Elemente Sollten sich (da noch keine korrekte Validierung vorhanden) zwei Elemente von der Zeit her überlagern, so sorgt ein negativ berechneter Wert dafür, dass das nächste Element nicht korrekt gesetzt wird.	Bug
TODO 13	Overview - Connection Bei der Berechnung des nächsten Bahnhofs kann es vorkommen, dass er den Endbahnhof auswählt, obwohl noch genügend Zeit vorhanden ist um den Startbahnhof zu erreichen.	Bug
TODO 14	Keine Datenverbindung Über die Übersicht zu der Kartenansicht kann es zum Absturz kommen, falls der Anwender über keine Datenverbindung verfügt. Einige Sicherheiten wurden bereits eingebaut. Jedoch muss dies noch beim Starten der Kartenansicht berücksichtigt werden.	Bug

Tabelle 10: Future Work und bekannte Bugs

A. Projektmanagement

A.1. Organisation

Die Bachelorarbeit wird von den zwei Bachelorstudenten Dominik Lüchinger und Nicolas Bigler bearbeitet. Betreut wird die Arbeit durch Prof. Dr. Andreas Rinkel.

A.2. Besprechungen

Damit alle Parteien regelmässig Informationen austauschen können, wird jede Woche ein Meeting geplant und durchgeführt. Normalerweise findet das Meeting jeden Montag um 13:00 Uhr statt. Die Diskussionsthemen, Erkenntnisse und das weitere Vorgehen werden abwechselnd von den beiden Studenten protokolliert.

A.3. Risiko Management

Bei jedem Projekt gibt es Risiken und Gefahren. Damit auf diese Risiken reagiert werden kann, muss eine entsprechende Reserve eingeplant sein. Das Risikomanagment hilft dabei die zu planende Reserve festzulegen. Die Tablle 11 führt die für die Bachelorarbeit relevanten Risiken auf. Aus dieser Tabelle folgt, dass ca. 36 Stunden als Reserve eingeplant werden sollten. Das

ID	Risiko	Massnahmen	Max Schaden	W'keit	Gew [h]
R1	Verplanung: Zuwenig Erfahrung mit Android-Programmierung	Vor dem Sprint mehr Zeit in Einarbeitung einrechnen	40	30%	12
R2	Ausfall Repository	Lokale Verwaltung der Daten	10	10%	1
R3	Sprintabbruch wegen internen Komplikationen oder missverstandenen Anforderungen	Retrospektive und Plannung des nächsten Sprint	90	10%	9
R4	Projektumfang unterschätzt	Wöchentliche Meetings, Daily Scrum, Sprint Review	50	10%	5
R5	Informationsquellen nicht zugänglich oder öffentlich	Frühzeitig abklären, ev. Alternativen suchen	30	30%	9
				Total	36

Tabelle 11: Risiko Management

ECTS-System definiert, dass ein Kreditpunkt studentischen einem Arbeitsaufwand von 25-30 Stunden entspricht (CRUS, 2012). Der erfolgreiche Abschluss der Bachelorarbeit gibt 12 Kreditpunkte. Dies entspricht einem Arbeitsauwand von 300-360 Stunden pro Student. Die Dauer der Bachelorarbeit ist auf 16 Wochen beschränkt. Mit Berücksichtigung der Reserven

aus dem Risikomanagement, ergibt dies einen durchschnittlichen Arbeitsaufwand von ca. 18 - 21 Stunden pro Woche.

Abbildung 23 zeigt den effektiven geleisteten Arbeitsaufwand. Der durchschnittliche Aufwand entspricht in etwa den geforderten Werten. Wird jedoch in den letzten Arbeitswochen deutlich übetroffen.

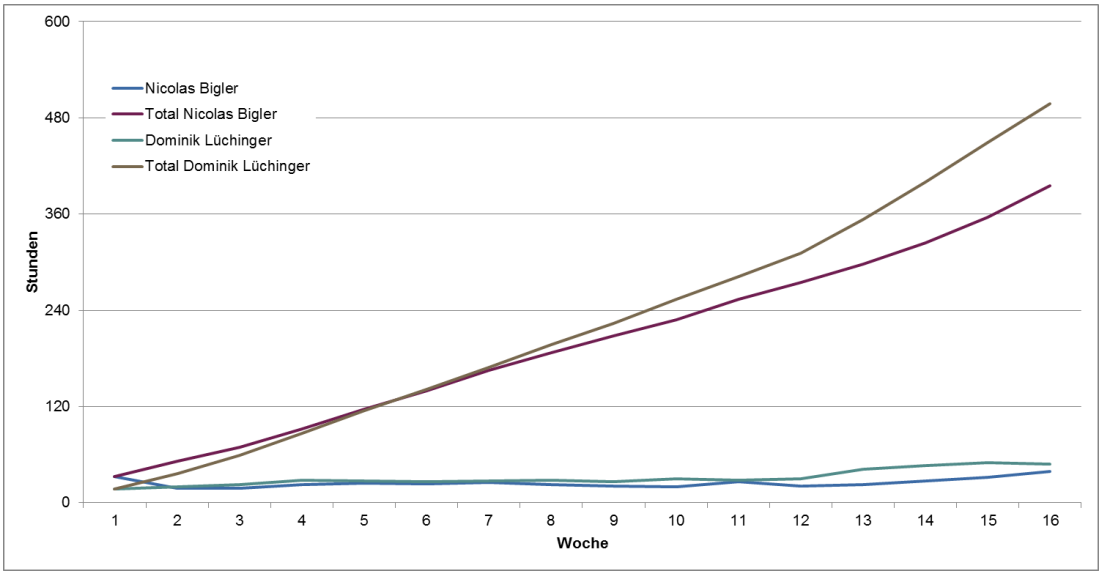


Abbildung 23: Projektaufwand

Die Abbildung 24 zeigt Arbeitsaufwand für die verschiedenen Bereiche auf. Die Grafik zeigt klar auf, dass am meisten Zeit mit der GUI-Programmierung aufgewendet wurde.

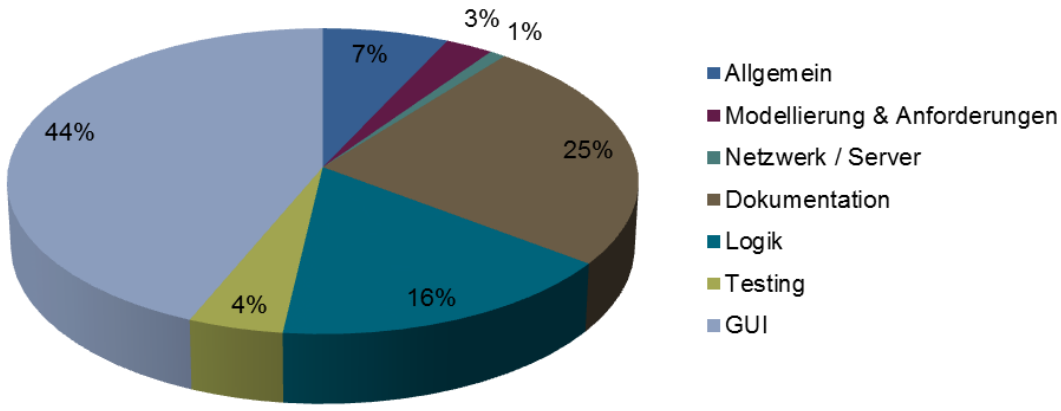


Abbildung 24: Aufwand nach Bereichen

B. Code Listings


```
{
  "connections":
  [
     {
        "from":
        {
           "station":
           {
              "id": "008503110",
              "name": "Rapperswil",
              "score": null,
              "coordinate":
              {
                 "type": "WGS84",
                 "x": 8.816727,
                 "y": 47.224884
              }
           },
           "arrival": null,
           "departure": "2012-06-01T16:10:00+0200",
           "platform": "1",
           "prognosis":
           {
              "platform": null,
              "arrival": null,
              "departure": null,
              "capacity1st": null,
              "capacity2nd": null
           }
        },
        "to":
        {
           "station":
           {
              "id": "008503000",
              "name": "Zürich HB",
              "score": null,
              "coordinate":
              {
                 "type": "WGS84",
                 "x": 8.540192,
                 "y": 47.378177
              }
           },
           "arrival": "2012-06-01T16:48:00+0200",
           "departure": null,
           "platform": "41/42",
           "prognosis":
           {
```

```
            "platform": null,
            "arrival": null,
            "departure": null,
            "capacity1st": null,
            "capacity2nd": null
         }
      },
      "transfers": 0,
      "duration": "00d00:38:00",
      "information":
      {
         "header": "",
         "lead": "",
         "text": "",
         "locType": 0,
         "section": 0,
         "dep": "",
         "arr": "",
         "symbol": 0,
         "channel": 0
      },
      "sections":
      [
         {
            "journey":
            {
               "name": "S7 18764",
               "category": "S7",
               "code": 5,
               "number": "18764",
               "operator": null,
               "to": null,
               "passList":
               [
                  {
                     "station":
                     {
                        "id": "008503110",
                        "name": "Rapperswil",
                        "score": null,
                        "coordinate":
                        {
                           "type": "WGS84",
                           "x": 8.816727,
                           "y": 47.224884
                        }
                     },
                     "arrival": null,
                     "departure": "2012-06-01T16:10:00+0200",
                     "platform": "",
                     "prognosis":
```

```
                        {
                            "platform": null,
                            "arrival": null,
                            "departure": null,
                            "capacity1st": null,
                            "capacity2nd": null
                        }
                    },
                    {
                        "station":
                        {
                            "id": "008503112",
                            "name": "Kempraten",
                            "score": null,
                            "coordinate":
                            {
                                "type": "WGS84",
                                "x": 8.814291,
                                "y": 47.238385
                            }
                        },
                        "arrival": "2012-06-01T16:12:00+0200",
                        "departure": "2012-06-01T16:12:00+0200",
                        "platform": "",
                        "prognosis":
                        {
                            "platform": null,
                            "arrival": null,
                            "departure": null,
                            "capacity1st": null,
                            "capacity2nd": null
                        }
                    },
                    ...
                ]
            },
            "departure":
            {
                "station":
                {
                    "id": "008503110",
                    "name": "Rapperswil",
                    "score": null,
                    "coordinate":
                    {
                        "type": "WGS84",
                        "x": 8.816727,
                        "y": 47.224884
                    }
                },
                "arrival": null,
```

```
                    "departure": "2012-06-01T16:10:00+0200",
                    "platform": "1",
                    "prognosis":
                    {
                       "platform": null,
                       "arrival": null,
                       "departure": null,
                       "capacity1st": null,
                       "capacity2nd": null
                    }
                },
                "arrival":
                {
                    "station":
                    {
                       "id": "008503000",
                       "name": "Zürich HB",
                       "score": null,
                       "coordinate":
                       {
                          "type": "WGS84",
                          "x": 8.540192,
                          "y": 47.378177
                       }
                    },
                    "arrival": "2012-06-01T16:48:00+0200",
                    "departure": null,
                    "platform": "41/42",
                    "prognosis":
                    {
                       "platform": null,
                       "arrival": null,
                       "departure": null,
                       "capacity1st": null,
                       "capacity2nd": null
                    }
                }
            }
        ]
    },
    ...
  ],
  ...
}
```

Listing 10: Verbindungs-Request

C. Benutzerhandbuch

Dieser Abschnitt beschreibt kurz die Verwendung der ”TravelGuide”-App. Eine Sammlung von Screenshots mit Navigation durch die einzelnen Elemente ist in den Abbildungen 25 und 26 auf den Seiten 69 und 70 zu finden. Die folgenden Erklärungen beziehen sich auf diese Abbildungen.

Beim ersten Aufruf der App ist der Homescreen noch ziemlich leer. Als erster Schritt muss durch anklicken des ”+”-Zeichen eine neue Reise hinzugefügt werden (1). Auf der neuen Seite kann nun ein Titel für die Reise eingeben werden. Bevor die Reise abgespeichert werden kann, muss noch eine Start- und Endzeit eingegeben werden (2). Nachdem die Reise erfolgreich gespeichert ist, gelangt man wieder auf den Homescreen und die neue angelegte Reise wird in der Liste aufgeführt.
Durch Anklicken der Reise öffnet sich ein Menü (3). Durch Auswählen von ”Edit Entry” wird die Etappenansicht der Reise geöffnet (4). Diese ist zu Beginn noch ziemlich leer. Mit dem ”+”-Zeichen kann eine neue Etappe eingegeben werden (5) + (6). Nach der Eingabe einer Beschreibung sowie eines Datums kann die Etappe abgespeichert werden (7). Danach gelangt man zurück zur Etappenübersicht. Die Schritte 5-7 können für weitere Etappen wiederholt werden (8).
Um einen Fahrplan für eine Etappe hinzuzufügen, muss der entsprechende Eintrag angeklickt werden (9). Im Menü wählt man dann die Option ”Add Timetable” um eine Verbindung hinzuzufügen (10). In den Textfeldern ”From” und ”To” muss der Abfahrts- bzw. Ankunftsort eingegeben werden. Eine Liste mit Vorschlägen unterstützt den Anwender bei der Eingabe. Zum Schluss muss noch eine Zeit eingegeben werden und ausgewählt werden ob die Zeit die Abfahrtszeit oder Ankunftszeit sein soll (11). Beim betätigen des ”Search Connection” Buttons wird eine Verbindung gesucht und in einer Liste angezeigt (12). Durch Auswahl einer Verbindung wird die automatisch abgespeichert und man gelangt wieder zurück zur Etappenübersicht (13). Die Schritte 9-13 können für weitere Fahrpläne wiederholt werden.
Um einer Etappe einen POI hinzuzufügen, muss die entsprechende Etappe angeklickt werden und im Menü der Punkt ”Add POI” ausgewählt werden (9). In der Suchmaske kann ein POI entweder im der Umgebung des Benutzers oder in der Umgebung eines frei wählbaren Ortes. Wird die Option ”Custom place” ausgewählt, erscheint ein Textfeld. Dort muss der gewünschte Ort eingegeben werden. Eine Liste mit Vorschlägen unterstützt dabei die Eingabe (14). Unabhängig von der gewählten Option muss ein Suchbegriff für die POI-Suche eingegeben werden (15). Anschliessend kann die Suche mit einem Klick auf den ”Search” Button ausgelöst werden (16). Aus der Resultatliste kann der gewünschte POI ausgewählt werden. Zum Schluss muss noch eine gewünschte Besuchszeit für den POI angegeben werden, bevor man wieder auf die Etappenübersicht gelangt (17). Um weitere POIs hinzuzufügen müssen die Schritte 14-17 wiederholt werden. Der Name und das Datum einer Etappe kann im Etappenmenü mit der Option ”Edit Stage” (9) geändert werden. Eine Etappe kann mit der Option ”Delete Stage” im Menü gelöscht werden.

Nachdem die Etappen, Verbindungen und POIs erfasst sind, kann auf dem Homescreen die Reise angeklickt werden und mit der Option ”Show Live Overview” eine aktuelle Übersicht der Reise angezeigt werden (Abbildung 26). In der Übersicht sind POIs mit einem roten Balken und ÖV-Verbindungen mit einem blauen Balken versehen. Durch Antippen des linken Bereichs

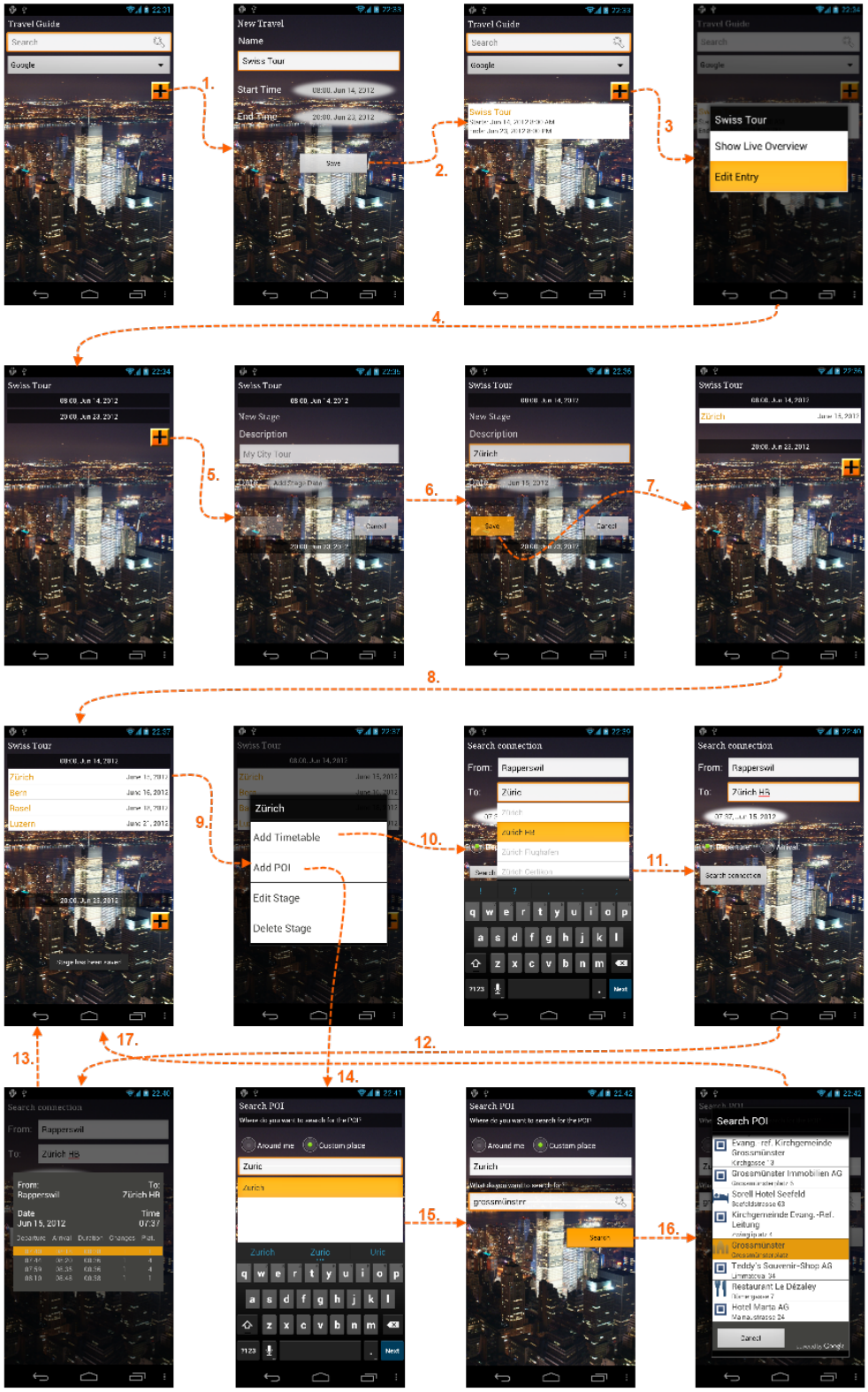


Abbildung 25: Benutzung des Guides 1

eines Elements wird eine Liste von Optionen angezeigt (18). Das Info-Feld zeigt für POIs die Adresse des POIs an (19) und für Verbindungen die Detailansicht der Verbindung (24). Das Route-Feld erlaubt es das anzeigen einer Route vom aktuellen Standort zum ausgewählten POI bzw. der Station der ausgewählten Verbindung (20) + (21). Mit dem Map-Feld kann für POIs und Verbindungen der Standort auf der Karte angezeigt werden (22) + (23).

Abbildung 26: Benutzung des Guides 2

D. Glossar

Bug	Ein Programm oder Softwarefehler
EDA	EDA ist ein Software Architektur Muster, welches das Zusammenspiel der Softwarekomponenten Ereignisbasiert analysiert.
Google Play	Google Play ist der Online Store von Google für Apps, Musik, Filme und Bücher
POI	Point Of Interest: Ein "interessanter Ort"
Refactoring	Erhöhung der Codequalität, durch Umstrukturierung (Aufräumen) des Codes. Resultat: Verbesserte Lesbarkeit, Verständlichkeit, Erweiterbarkeit und Wartbarkeit. Bei der weiteren Entwicklung sollten noch zusätzliche Tests geschrieben werden.
Scrum	Scrum ist ein empirisches, inkrementelles und iteratives Vorgehensmodell der Softwareentwicklung.

E. Referenzen

[Bruns u. Dunkel 2010] BRUNS, Ralf ; DUNKEL, Jürgen: *Event-Driven Architecture*. Springer, 2010. – ISBN 978–3–642–02439–9

[CRUS 2012] CRUS: *Was ist ECTS?* http://www.crus.ch/information-programme/bologna-lehre/was-ist-ects.html?L=2. Version: 2012. – [Online; Zugriff 6. Juni 2012]

[Gartner 2012a] GARTNER: *Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009*. http://www.gartner.com/it/page.jsp?id=2017015. Version: 2012. – [Online; Zugriff 18. Mai 2012]

[Gartner 2012b] GARTNER: *Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth*. http://www.gartner.com/it/page.jsp?id=1924314. Version: 2012. – [Online; Zugriff 18. Mai 2012]

[Google 2012] GOOGLE: *Google Maps API*. https://developers.google.com/maps/?hl=de. Version: 2012. – [Online; Zugriff 1. Juni 2012]

[tripwolf 2012] TRIPWOLF: *tripwolf - dein Reiseführer*. https://play.google.com/store/apps/details?id=com.tripwolf. Version: 2012. – [Online; Zugriff 8. April 2012]

[Västtrafik 2012] VÄSTTRAFIK: *Travel planner*. https://play.google.com/store/apps/details?id=de.hafas.android.vasttrafik. Version: 2012. – [Online; Zugriff 8. April 2012]

[Wirdemann 2009] WIRDEMANN, Ralf: *Scrum mit User Stories*. Hanser, 2009. – ISBN 978–3–446–41656–7