Columban
-
client-server-basierter mobiler GPS-Tracking-Service für Android-Smartphones zum Einsatz im Tourismusbereich
-
technische Machbarkeitsstudie

Projektdokumentation
Im Rahmen des Moduls
Ausgewählte Themen der Programmierung

Erstellt von:
Jens Frank, Alexander Nöske, Andreas Kondratowicz

Mönchengladbach, den 18.01.2013

Inhaltsverzeichnis

Anlage:
Zwischenpräsentation am 10.12.2012
Abschlusspräsentation am 14.01.2013
Alle Dateien übergeben am 21.01.2013 an Prof. Dr. rer. nat. Claus Brell

Anzahl Anschläge: 58021, Anzahl Worte: 8834

Tabellenverzeichnis

Abbildungsverzeichnis

1 Management Summary

Columban ist eine GPS-Tracking-App mit angegliedertem Server und für Android-Smartphones entwickelt. Die Projektumsetzung geschieht im Rahmen des Moduls „Ausgewählte Themen der Programmierung“ an der Hochschule Niederrhein.

Ziel des Projektes ist es, eine Anwendung für z. B. Wanderer und Mountainbiker, zu erstellen, welche es dem Nutzer erlaubt, seine zurückgelegte Strecke aufzuzeichnen und an den Server zu übertragen. Weitere wichtige Anforderungen an die App sind die Robustheit gegenüber schlechter Verbindung und die Möglichkeit, eine Notfall-SMS mit den aktuellen Koordinaten abzuschicken. Auf der Serverseite werden alle Daten angenommen und dem Nutzer in Form einer Auswertungswebseite zur Verfügung gestellt.

Als Softwareentwicklungsprozess wird Scrum verwendet. Dabei vertritt Andreas Kondratowicz die Rolle des Product Owners, Alexander Nöske wird den Scrum Master darstellen und das Umsetzungsteam wird zusätzlich zu diesen beiden Personen noch durch Jens Frank unterstützt.

Es sind insgesamt vier Sprints á jeweils 2 Wochen geplant und eine Vorbereitungszeit von zwei Wochen zu Beginn des Projekts. Alle Sprints wurden erfolgreich nach dem Zeitplan abgeschlossen.

Das Projekt verwendet eine Client-Server-Architektur. Die Serverseite wird ausschließlich mittels PHP-Skripten und HTML-Seiten umgesetzt, der Client für die mobilen Endgeräte mit dem Android-SDK in Java. Die gesamte System-Architektur wird mittels des Aris-Frameworks dargestellt.

Der erste Schritt der Entwicklung war es, sowohl Client als auch Server grundlegend aufzubauen. Im nächsten Schritt wurde dann der Kommunikationskanal zwischen beiden Seiten aufgebaut, um anschließend aufgenommen Daten übergeben und verarbeiten zu können.

Danach wurde die Auswertungswebseite entwickelt, welche die aufgenommen Daten dem Nutzer mit einer Map-Ansicht und weiteren Auswertungen zur Verfügung stellt.

Für die Installation der Serverskripte wurde im letzten Schritt noch ein Installations-Programm in C# entwickelt.

Über alle Anwendungen hinweg wurde darauf geachtet, dass die Eingewöhnungszeit möglichst gering ist und die Bedienung intuitiv erfolgen kann.

Die Ergebnisse der Planung und Umsetzung sind im vorliegenden Skript festgehalten. Alle wichtigen Anforderungen konnten dabei erfüllt werden.

Der Name der App „Columban“ wird zurückgeführt auf den Heiligen Columban von Luxeuil, welcher für seine Missionierungen viele Reisen auch unter schwierigen Bedingungen unternahm. Aus diesem Grund gilt er heutzutage als Schutzpatron der Motorradfahrer und wurde zum Namensgeber des vorliegenden Projekts bestimmt.

2 Projektziel

2.1 Ausgangslage

Seit geraumer Zeit kann beobachtet werden, dass insbesondere in landschaftlich reizvollen Gebieten Freizeitaktivitäten mit Hilfe von GPS-Geräten unterstützt werden. Einzelne Tourismuszentralen haben darauf z.B. mit der Downloadmöglichkeit von Wandertouren oder Mountainbiketouren als GPX-Dateien reagiert. Seit kurzem sind gängige Smartphones als GPS-Geräte mit zusätzlichen Eigenschaften einsetzbar und werden von den Kunden der Tourismusgebiete auch zunehmend genutzt. Das Potenzial, dass sich insbesondere hierdurch für touristische Angebote eröffnet, ist noch nicht ausgeschöpft. Hierzu gehören insbesondere gemeindeübergreifende Angebote (Touren, Informationen zu Unterkünften und Gastronomie in Abhängigkeit des Standortes und der Bewegungsrichtung des Kunden, Angebote zur Befriedigung des Sicherheitsbedürfnisses und Möglichkeit zur Communitybildung z.B. über den Austausch der selbst gemachten Touren).

Technische Basis zur Erstellung solcher Angebote wäre eine Smartphone und webserverbasierte Umgebung, die die folgenden Anforderungen miteinander verbindet:

- Tour des Kunden mitschreiben (online GPS-Tracking)
- Senden des Aufenthaltsortes an festgelegte Nummer im Krisenfall
- Versorgung des Kunden mit standortspezifischen touristischen Informationen
- Communitybildung für Kunden einer Gemeinde oder eines übergreifenden touristischen Gebietes
- Robustheit gegenüber schlechter Netzanbindung und gegenüber Netzausfällen.

Im Rahmen des Semesterprojektes soll eine Lösung für einige der Anforderungen prototypisch entwickelt werden.

2.2 Zielkatalog

In der folgenden Auflistung finden sich die festgelegten Anforderungen an das Projekt. Die Anforderungen sind sowohl aus eigenen Überlegungen als auch aus Rücksprachen mit dem Kunden entstanden.

Primärziele	1. Erstellung einer App, die es dem Nutzer ermöglicht - Touren zu Erstellen - seinen aktuellen Aufenthaltsort zu speichern und an einen Server zu übertragen - optional: eine Notfall-SMS mit der aktuellen Position zu senden 2. Erstellung einer Server-Anwendung, die - Nutzerdaten / Touren aufnimmt - eine Auswertungswebseite zur Verfügung stellt 3. die mobile Anwendung sollte außerdem in der Lage sein, mit temporären Verbindungsabbrüchen umgehen zu können, z. B. durch automatische Wiederholung einer unterbrochenen Übertragung von Daten
Sekundärziele	1. Einfache Bedienbarkeit der App 2. Erweiterung der eigenen Kenntnisse hinsichtlich: - Projektarbeit, im Speziellen die konkrete Anwendung von z. B. Architektur- und Vorgehensmodellen und Teamarbeit - Programmierung mobiler Anwendungen - Erstellung von Client-Server-Anwendungen

Tabelle 1: Zielkatalog

Da als Vorgehensmodell Scrum gewählt wurde (für nähere Erläuterungen s. Kap. 3.1.1), werden die Primäranforderungen auch etwas detaillierter als User Stories dargestellt, welche einer einfachen Formulierung der Software-Anforderungen entsprechen. Die Gesamtheit der Anforderungen entspricht dem Product-Backlog in der Scrum-Methode.

Jede Anforderung wird mit einem Namen, einer kurzen Beschreibung, einer Priorität und einer ersten Zeitabschätzung erfasst. Im Verlaufe des Projekts können diese Zeiten dann mit Erfahrungen aus den Sprint-Meetings gegebenenfalls noch angepasst werden. Aus Gründen der Übersichtlichkeit befinden sich die User Stories im Anhang.

2.3 Funktionsweise als Netzwerkdiagramm veranschaulicht

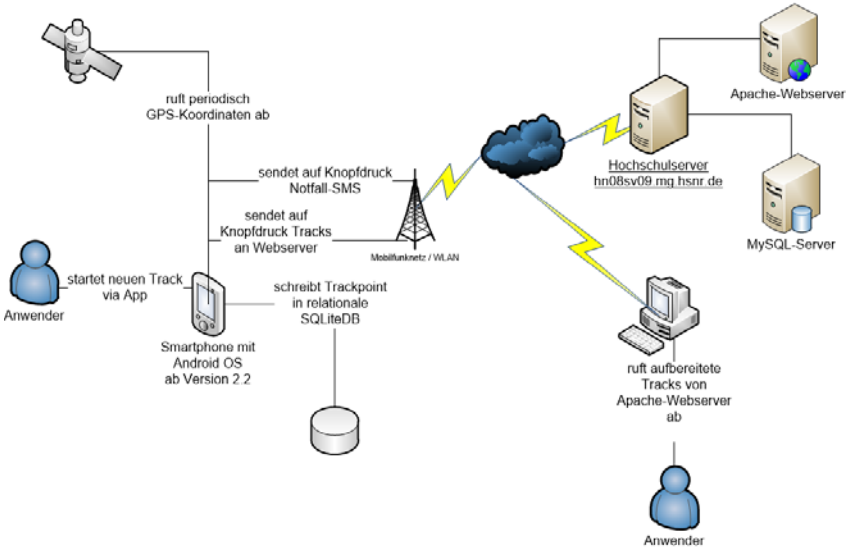

Abbildung 1: Systemarchitektur als Netzwerkdiagramm[1]

Als Voraussetzung für die Aufnahme wird ein Android-Smartphone mit einem Betriebssystem ab Version 2.2 angegeben. Der Anwender wird periodisch GPS-Koordinaten abrufen können. Für den Abruf der GPS-Koordinaten wird kein Internetzugriff benötigt, es wird mit den Rohdaten des GPS-Empfängers gearbeitet. Wird eine Aufnahme (Track) gestartet, so werden die einzelnen Wegpunkte (Trackpoints) in eine Datenbank auf dem Smartphone geschrieben bis die Aufnahme gestoppt wird. Über die App kann im Notfall eine SMS an einen vorher definierten Empfänger gesendet werden, idealerweise sollten die Koordinaten als Weblink enthalten sein, sodass nicht lange gesucht werden muss, an welchem Ort der Sender sich befindet. Die erfassten Tracks können auf Knopfdruck mit dem Columban-Server synchronisiert werden. Nachdem die Synchronisation abgeschlossen ist können die Aufnahmen über den Browser abgerufen und ausgewertet werden.

3 Projektorganisation und -planung

3.1 Vorgehensmodell

Da das gewählte Vorgehensmodell einige Besonderheiten in Aufbau und Ablauf mit sich bringt, wird dessen Vorstellung aus dem ursprünglichen Kapitel 4 vorgezogen. Dies soll das Verständnis von z. B. vergebenen Rollen oder Besonderheiten in der Zeitplanung erleichtern.

3.1.1 Auswahl des Modells

Als Entwicklungsmodell für das GPS-Tracker-Projekt wurde Scrum verwendet. Scrum ist ein agiler und iterativer Prozess. Der Prozess besitzt die drei wesentlichen Rollen des Product Owners, des Scrum Masters und des Teams. Der Product Owner kann auch als „oberster Produktentwickler“ bezeichnet werden und bildet in dieser Rolle eine Schnittstelle zwischen

[1] Quelle: eigene Darstellung

dem Kunden und dem Entwicklungsteam. In Rücksprache mit dem Kunden nimmt er die Anforderungen an das Projekt auf, welche dann im sog. Product Backlog festgehalten, gepflegt und priorisiert werden. Allerdings ist der Product Owner kein Vertreter des Kunden, da sein oberstes Ziel die Wirtschaftlichkeit des Produktes für die eigene Firma sein muss.

Der Scrum Master arbeitet mit Product Owner und Projektteam zusammen und kümmert sich um einen reibungslosen Ablauf von Scrum. Das Team besteht aus allen Beteiligten, die zur Umsetzung des fertigen Produkts benötigt werden, es verwaltet und organisiert sich dabei aber selbst. Allerdings ist es trotzdem verpflichtet, die vereinbarten Ziele zu erreichen.

Die Iterationen in Scrum heißen Sprints und haben in der Regel eine eher kurze Dauer von 1-4 Wochen.

Zu Beginn eines Sprints gibt es ein Sprint-Planning-Meeting, in dem das Team festlegt, welche Anforderungen im Sprint umgesetzt werden sollen. Diese werden im Sprint Backlog festgehalten und in kleinere Arbeitspakete zerlegt. Die Arbeitspakete werden möglichst so klein gehalten, dass sie jeweils an einem Arbeitstag abgearbeitet werden können.

Während der Sprints gibt es jeden Tag ein so genanntes Daily-Scrum-Meeting, in dem das den Fortschritt seit dem letzten Meeting besprochen werden, in der Regel sollte dies nicht mehr als 15 Minuten betragen. In jedem Sprint wird auch ein Burndown Chart geführt, welches den noch zu erbringenden Restaufwand graphisch darstellt und täglich aktualisiert wird. Am Ende eines Sprints sollte der Restaufwand auf null stehen.

Zusätzlich wird am Ende jedes Sprints ein Sprint-Review- und Retrospective-Meeting abgehalten, bei dem die Ergebnisse und der Prozess des letzten Sprints beurteilt werden. Auf Basis der Feedbacks der verschiedenen Beteiligten werden dann Strategien entwickelt, wie die zukünftigen Sprints noch optimiert werden können.

Eine wichtige Besonderheit von Scrum ist, dass durch den Ansatz, ständig Änderungen zuzulassen, eine feste Zeitplanung zu Beginn des Projekts kaum möglich ist. Um trotzdem eine Zeit- und Budgetplanung für die Gesamtentwicklung vornehmen zu können, wird häufig mit Timeboxing gearbeitet: Es wird ein bestimmtes Zeit-/Budget-Kontingent zur Verfügung gestellt, ohne jedoch die Projekt-Ergebnisse vertraglich abstimmen zu können. Eine Abschätzung, was in diesem Zeitraum realisiert werden kann, kann trotzdem vorgenommen werden – allerdings nie so detailliert wie es bei klassischen Modellen getan wird.

In diesem Projekt wurde sich für die Umsetzung nach Scrum entschieden, da dieses Modell gut mit Anforderungsänderungen bzw. noch nicht vollständig bekannten Anforderungen zu Beginn des Projekts umgehen kann. Eine Änderung hat keinen Einfluss auf den aktuellen Sprint und trotzdem keine negativen Auswirkungen auf das Projekt, da die Sprints sehr kurz gehalten werden. Hierdurch können Änderungen trotzdem schnell behandelt werden. Ein weiterer Grund für Scrum ist, dass die für den Kunden wichtigsten Anforderungen als Erstes abgearbeitet werden. Dadurch sind nach den ersten Sprints die Hauptanforderungen implementiert und der Kunde hat nach jedem Sprint ein Zwischenprodukt. Zudem werden die wichtigsten Anforderungen dadurch am häufigsten getestet.

Durch die Burndown-Charts haben zudem die Stakeholder immer einen guten Überblick über den tagesaktuellen Fortschritt des Projektes.

Hervorzuheben sind weiterhin die Sprint-Review- und Retrospective-Meetings am Ende eines Sprints, da man hier den vergangenen Sprint kritisch betrachten und daraus Verbesserungen für den nächsten Sprint ableiten kann.

Für das aktuelle Projekt mussten allerdings einige Abweichungen vom klassischen Vorgehen nach Scrum in Kauf genommen werden. Dies betrifft vor allem die Rollenverteilungen und Größe innerhalb des Teams.

3.1.2 Modellierung

Die oben beschriebene Funktionsweise von Scrum kann grafisch wie in Abbildung 2 dargestellt werden:

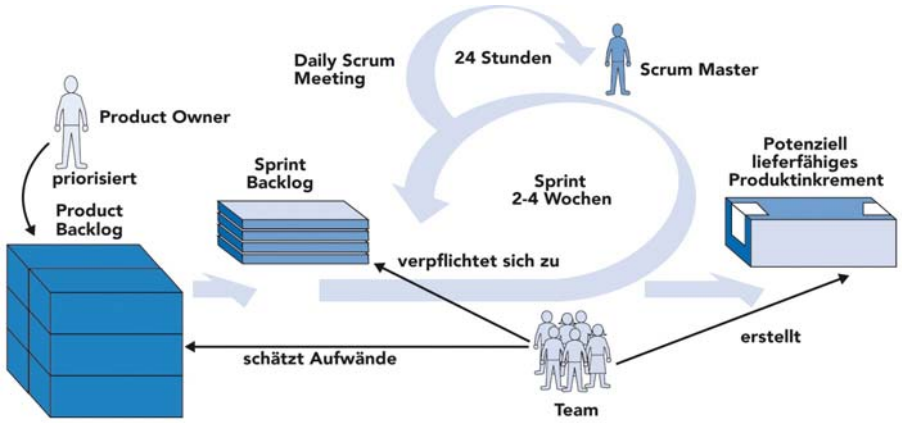

Abbildung 2: Scrum-Modell[2]

Die genaue Rollenverteilung zu dem vorliegenden Projekt wird in Kapitel 3.4.2 näher erläutert.

Neben einer grafischen Darstellung können zudem noch die einzelnen Sprints modelliert werden. Aus Gründen der Übersichtlichkeit wird dies jedoch gemeinsam mit dem tabellarischen Zeitplan in Kapitel 3.3.1 vorgenommen.

3.2 Projektstrukturplan

Das Projekt ist im Projektstrukturplan in drei Teilaufgaben aufgeteilt, im Einzelnen das Projektmanagement, die Projektplanung und die Projektdurchführung. Sowohl das Projektmanagement als auch die Projektplanung wurden von allen Teammitgliedern gleichermaßen durchgeführt. Bei der Projektdurchführung wurde das Paket der Programmierung aufgeteilt in 3 Kernbereiche. Die Installation der Software wurde individuell von jedem selbst gestaltet, da verschiedenste Software zum jeweiligen Arbeitspaket von Nöten war. Die Einrichtung des Servers wurde von Jens Frank übernommen, da die Arbeit hauptsächlich auf seinem privaten Server durchgeführt wurde. Bei der Programmierung kam es zur folgenden grundsätzlichen Aufteilung der Arbeitspakete:

Jens Frank → App Programmierung

Alexander Nöske → Client/Server Programmierung

Andreas Kondratowicz → Programmierung der Auswertungsseite (Web)

Trotz dieser Aufteilung der Arbeitspakete, hat jeder den anderen bei der Fertigstellung des Projekts unterstützt. Die weiteren Arbeitspakete wie „Einrichten des Smartphones“, „Testphase“ und „Design einrichten“ wurden dementsprechend wieder gemeinsam gemeistert.

[2] Entnommen aus: http://www.computerwoche.de/a/sprinten-mit-dem-v-modell-xt,1228447, 10.12.2012

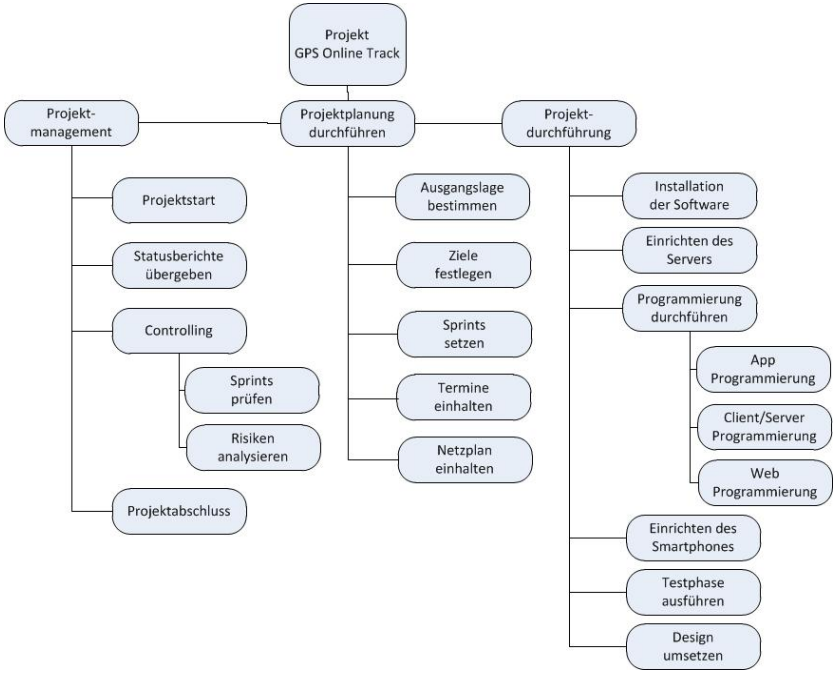

Abbildung 3: Projektstrukturplan[3]

3.3 Projektplan

3.3.1 Tabellarischer Projektplan

Die Scrum-Sprints in diesem Projekt sind in der Regel 2 Wochen lang. Die grobe Sprintplanung des Projekts ist in der nachfolgenden Tabelle beschrieben:

Woche	Von	Bis	Arbeitstage	Phase	Beschreibung
KW46	12.11.2012	25.11.2012	10	Vorbereitung	Einrichtung der Arbeits- / Entwicklungsumgebungen, Projektplanung, Anforderungssammlung
KW47					
KW48	26.11.2012	09.12.2012	10	Sprint 1	Einarbeitung in die Programmierung, Datenmodellierung, Erstentwurf Clientoberfläche, Serverkommunikation, Erstellung Zwischenpräsentation
KW49					
10.12.2012: Zwischenpräsentation					
KW50	10.12.2012	23.12.2012	10	Sprint 2	Serverkommunikation, Abrufen und Speichern von GPS-Daten, Automatisierte Übertragung von Daten, Auswer-
KW51					

[3] Quelle: eigene Darstellung

					tungswebseite
KW52 KW1	24.12.2012	06.01.2012	2	Sprint 3	Login-Funktion, Design, optionale Anforderungen (SMS-Versand, Auswertungen auf Client, etc.)
KW2	07.01.2013	13.01.2013	5	Sprint 4	Zeitpuffer für offene oder neue Aufgaben, Erstellung Abschlusspräsentation
14.01.2013: Abschlusspräsentation					
KW3	14.01.2013	20.01.2013	5	Sprint 4	Abschluss der Dokumentation
21.01.2013: Abgabe der Dokumentation					

Tabelle 2: Projektplan

In den nachfolgenden Kapiteln wird nun ein Überblick über die einzelnen Sprints gegeben.
Die gesamte Zeitplanung betrachtet jedoch nicht den Bereich der Vorbereitung, welcher vom gesamten Team gemeinsam bearbeitet wurde, als einzelnen Sprint. Dies liegt in der Tatsache begründet, dass die diesem Zeitpunkt des Projektes noch kein Vorgehensmodell festgelegt war und daher auch noch keine Methodik zur Planung vorlag.

3.3.1.1 Sprint 1

Für den ersten Sprint waren folgende User-Stories und Aufgaben geplant:

Kennziffer	Name	Geschätzte Zeit in Personentage	Benötigte Zeit in Personentage	Durchgeführt von
US1	Der Nutzer kann seine Nutzerdaten eingeben	1	1	Frank
US10	Der Server kann Nutzerdaten verwalten	1	1	Nöske
US14.1	Erste Entwürfe für das Design	2	3	Frank, Kondratowicz, Nöske
US15	Aneignen des benötigten Vorwissens durch die Entwickler	9	8	Frank, Kondratowicz, Nöske
-	Scrum-Arbeiten / -Meetings	1,5	1	Nöske, Frank, Kondratowicz
-	Datenmodellierung	1	1	Frank
-	Prozessmodellierung	2	2	Kondratowicz
-	Fortführung Dokumentation	3	4	Frank, Kondratowicz, Nöske
Summe		**20,5 PT**	**21 PT**	

Tabelle 3: Sprint 1 – Detailplan

Unsere ersten Schätzungen hatten sich demnach zunächst als ganz gut erwiesen. Trotzdem gab es einige Abweichungen. So konnten wir das Aneignen des Vorwissens etwas schneller Abschließen als zunächst geschätzt, allerdings erforderten die ersten Designentwürfe innerhalb der App doch etwas mehr Übung und Abstimmung. Die eigentlichen Arbeiten an Scrum

konnten ebenfalls schneller abgewickelt werden, jedoch nahm die Dokumentation der gesamten Änderungen mehr Zeit in Anspruch als gedacht. Die Sprint-Review ergab daher, dass wir unsere Schätzungen für den schriftlichen Teil in Zukunft noch etwas höher ansetzen sollten und gerade bei den Programmierarbeiten unsere Schätzungen mit einem Puffer versehen müssen.

Das Burndown-Chart zu unserem ersten Sprint sieht daher folgendermaßen aus:

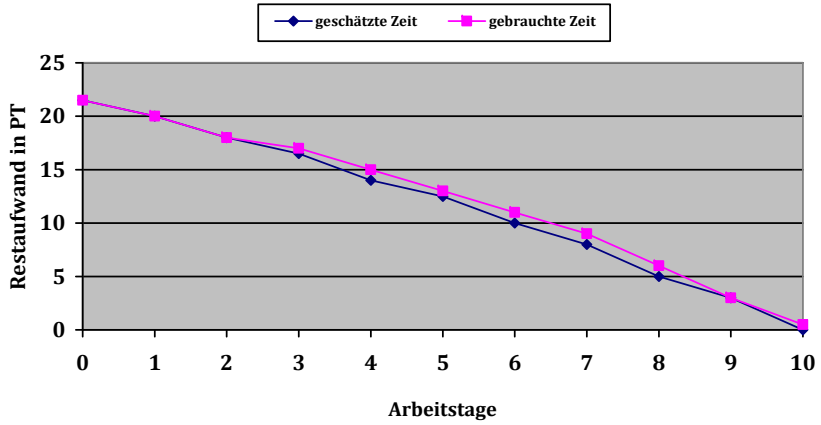

Abbildung 4: Burndown-Chart Sprint 1[4]

Wie bereits in Kapitel 3.1.1 erläutert soll das Burndown-Chart dabei helfen, den aktuellen Projektstatus graphisch für alle Teammitglieder darzustellen. Wir haben jedoch festgestellt, dass diese graphische Darstellung uns bei unserer täglichen Arbeit nur bedingt weiterhilft. Daher haben wir für die weiteren Sprints beschlossen, dieses Hilfsmittel nicht mehr weiter fortzuführen.

3.3.1.2 Sprint 2

Im 2. Sprint wurden durch das Team folgende Userstories umgesetzt:

Kennziffer	Name	Geschätzte Zeit in Personentage	Benötigte Zeit in Personentage	Durchgeführt von
US3	Der Nutzer kann einen neuen Track starten	2	2	Frank
US4	Der Client nimmt GPS-Koordinaten auf	2	2	Frank
US6	Der Nutzer kann Trackingdaten manuell senden	1	2	Frank
US7	Der Nutzer kann sich historische Daten ansehen (Vorbereitung)	2	2	Kondratowicz

[4] Quelle: eigene Darstellung

US9	Der Client kann Trackingdaten übertragen	2	2	Frank, Kondratowicz
US10	Der Server kann Nutzerdaten verwalten	1	1	Nöske
US12	Der Server kann Trackingdaten speichern	3	3,5	Nöske
-	Scrum-Arbeiten / -Meetings	2	2	Nöske, Frank, Kondratowicz
-	Datenmodellierung	0	1	Nöske, Frank
-	Fortführung Dokumentation	2	2,5	Frank, Kondratowicz, Nöske
Summe		**17 PT**	**20 PT**	

Tabelle 4: Sprint 2 – Detailplan

Auch im 2. Sprint zeigte sich, dass die vorherigen Schätzungen relativ gut eingehalten werden konnten. Da wir aus dem vorherigen Sprint unsere Erfahrungen mit der Server- und App-Programmierung bereits verwenden konnten, wurden einige Userstories bereits von Vornherein zeitlich neu eingeschätzt. Diese Maßnahme hat sich im Nachhinein als genau richtig erwiesen, wie sich an den benötigten Zeiten zeigt. Zudem wurde die Zielsetzung für US9 etwas abgeändert. Eine automatische Übertragung alle X Sekunden erschien dem Team im Nachhinein nicht mehr sinnvoll umzusetzen. Dies lag zum einen an größeren Problemen bei Verbindungsabbrüchen, zum anderen auch daran, dass dies nicht zwingend benötigt wird. Ein Live-Tracking ist derzeit noch nicht implementiert und die aktuelle Position kann im Notfall über die Notfall-SMS übertragen werden.

Trotz dieser Vorsichtsmaßnahme gab es jedoch einige Probleme bei der Datenübertragung von der App zum Server. Infolgedessen musste die Datenmodellierung noch einmal komplett überarbeitet werden.

3.3.1.3 Sprint 3

Der 3. Sprint des vorliegenden Projekts weicht vom sonstigen Rhythmus etwas ab. Durch Feiertage und Betriebsferien blieben im Endeffekt nur 2 Arbeitstage übrig. Trotzdem wurde auch in dieser Zeit mit Hochdruck am Projekt gearbeitet. Dabei wurden folgende Userstories behandelt:

Kennziffer	Name	Geschätzte Zeit in Personentage	Benötigte Zeit in Personentage	Durchgeführt von
US13	Der Nutzer kann sich eine Auswertungswebseite ansehen	3	2	Kondratowicz
US14	Weitere Arbeiten an der GUI der App	2	2	Frank
US8	Der Nutzer kann eine Notfall-SMS senden	-	1	Kondratowicz
-	Administration / Einrichtung des Servers	1	1	Nöske
-	Scrum-Arbeiten / -Meetings	0,75	0,75	Nöske, Frank, Kondratowicz
-	Fortführung Dokumentation	1	1	Frank, Kondratowicz, Nöske
Summe		**7,75 PT**	**7,75 PT**	

Tabelle 5: Sprint 3 – Detailplan

Der Ablauf dieses dritten Sprints wurde im Meeting als nahezu optimal eingeschätzt. Die vorherigen Planungen konnten alle eingehalten werden, was allerdings auch durch den eher geringen Umfang der zu planenden Aufgaben erklärbar ist. Da die Auswertungswebseite in ihrem Umfang etwas reduziert wurde, konnte diese schneller umgesetzt werden. Zusätzlich wurde daher noch die Implementierung der Notfall-SMS aufgenommen, welche gleichfalls wesentlich weniger aufwendig war als zuvor geschätzt.

3.3.1.4 Sprint 4

Der letzte Sprint erforderte eine sehr umsichtige Planung, da an seinem Ende die Abgabe des Projekts stand. In Anbetracht unseres Zeitplans und der bisherigen Umsetzungen hat sich das Team im Sprint-Planning-Meeting daher unter anderem auch dazu entschieden, die Login-Funktionalität des Servers nicht mehr umzusetzen. Die Grundfunktionalitäten des Logins und Logouts waren zu diesem Zeitpunkt zwar bereits implementiert, eine vollständige Implementierung inkl. Verwaltung von Session-IDs hätte jedoch einen erheblichen Mehraufwand zur Folge. Da die sonstigen Funktionalitäten der App jedoch unabhängig von einer Login-Abhandlung ausführbar sind, ist dies für das Projekt zunächst nicht entscheidend.

Stattdessen sollte jedoch noch ein Programm erstellt werden, welches die Installation der Dateien und Datenbanken auf dem Server unterstützt. Die Zeitplanung des 4. Sprints gestaltete sich daher wie folgt:

Kennziffer	Name	Geschätzte Zeit in Personentage	Benötigte Zeit in Personentage	Durchgeführt von
US13	Der Nutzer kann sich eine Auswertung ansehen	2	2	Kondratowicz
US14	Abschluss der GUI der App	2,5	3	Frank, Nöske
-	Erstellung eines Installationsprogramms	4	4	Nöske
-	Scrum-Arbeiten & sonstige Meetings	4	4	Nöske, Frank, Kondratowicz
-	Erstellung Abschlusspräsentation	4	4	Frank, Kondratowicz, Nöske
-	Abschluss Dokumentation	6	7	Frank, Kondratowicz, Nöske
Summe		**23,5 PT**	**25 PT**	

Tabelle 6: Sprint 4 – Detailplan

Der 4. Sprint erwies sich im Vergleich als der Umfangreichste im gesamten Projekt.

Dieser Umfang ergibt sich aber hauptsächlich aus administrativen und projektplanerischen Tätigkeiten wie der Erstellung der Abschlusspräsentation und der Finalisierung der vorliegenden Dokumentation. Ansonsten war neben kleineren Bugfixes, welche in die obige Darstellung des Sprints nicht extra aufgenommen wurden, an dem Endprodukt lediglich noch Arbeiten an der GUI offen. Da diese nicht nur von dem Hauptzuständigen für die Android-

Entwicklung sondern auch von anderen Teammitgliedern vorgenommen wurden, welche weniger erfahren in diesem Bereich waren, dauerten diese Tätigkeiten etwas länger als geplant. Die übrigen geplanten Punkte konnten jedoch nahezu optimal umgesetzt werden.

3.3.1.5 Gesamtbetrachtung

In der Gesamtbetrachtung über alle Sprints hinweg kann das Projekt zeitlich als voller Erfolg gewertet werden. Angesichts des gewählten Vorgehensmodells kann dies allerdings auch keine Überraschung sein. Jeder Sprint wird für sich geplant und nötigenfalls Funktionalitäten gestrichen, um das vorgegebene Zeit- und Kostenbudget nicht zu überschreiten. Über alle Sprints hinweg war eine Umsetzungszeit von ca. 69 Personentagen (PT) á ca. 5 Stunden geplant. Die tatsächlich benötigte Zeit liegt zwar mit 73,75 PT etwas darüber, allerdings durchaus noch im Rahmen. Dies lag vor allem auch daran, dass für die einzelnen Sprints genügend Puffer eingeplant war um die übrigen Sprints nicht zu beeinflussen.

Die Verteilung der benötigten Zeit auf die einzelnen Teammitglieder zeigt, dass Herr Frank mit 23,5 PT im Vergleich zu Herrn Kondratowicz mit 22,25 PT und Herrn Nöske mit 22,5 PT insgesamt etwas mehr Zeit benötigte. Dies kann aber durch die Komplexität der Android-Entwicklung erklärt werden.

3.3.2 Netzplan

Für das gesamte Projekt wurde zudem ein einfacher Netzplan erstellt (s. Abbildung 5). Da die Sprints einer nach dem anderen abgewickelt wird und die konkrete Planung der Umsetzung innerhalb dieser Sprints stattfindet, ist der kritische Pfad auch gleichzeitig er einzige Pfad. Für die einzelnen Sprints wurde aus Gründen der Übersichtlichkeit auf eigene Netzpläne verzichtet.

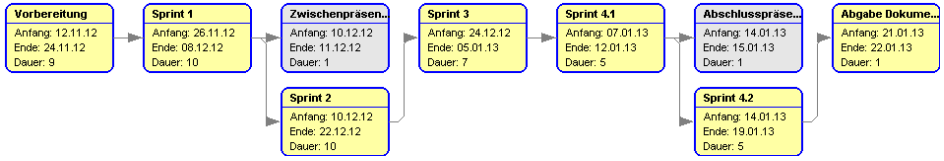

Abbildung 5: Netzplan[5]

[5] Quelle: eigene Darstellung

3.3.3 Gantt-Diagramm

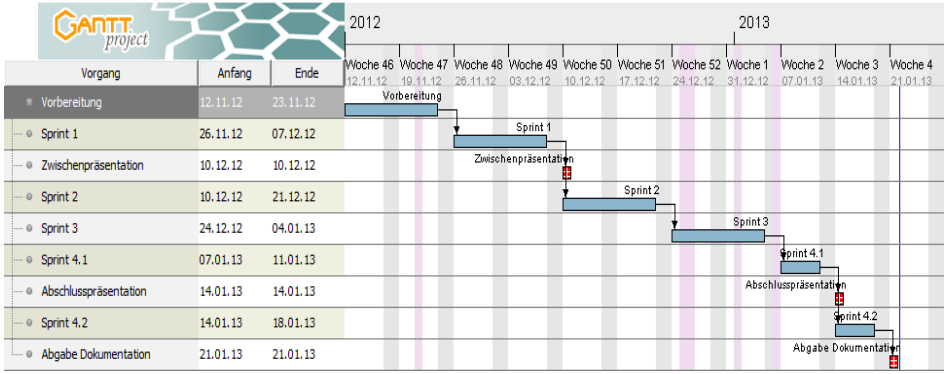

Abbildung 6: Gantt-Diagramm[6]

3.4 Ressourcen

Im diesem Kapitel soll ein Überblick über die am vorliegenden Projekt beteiligten Personen, deren Rollenverteilung und verwendete Ressourcen gegeben werden. Da das gesamte Projektteam nur aus 3 Personen bestand, müssen diese jeweils mehrere Projektrollen übernehmen.

3.4.1 Personen

Das Projektteam besteht aus Jens Frank, Andreas Kondratowicz und Alexander Nöske. In **Fehler! Verweisquelle konnte nicht gefunden werden.** wird zunächst ein Überblick über die bisherigen Erfahrungen und Kenntnisse hinsichtlich Programmierung und Projektarbeit der Teammitglieder gegeben.

Frank, Jens (Fachinformatiker / SI)	
Programmierung	Semi-professionell: Java-Kenntnisse, PHP, HTML, Visual-Basic, SQL
Projektarbeit	Erste Erfahrungen in kleineren Software-Projekten. (bis 120 Stunden Gesamtumfang) Professionell: Hardware-Projekte
Kondratowicz, Andreas (Fachinformatiker / SI)	
Programmierung	Gute Kenntnisse: Java, C/C++, SQL, PHP, HTML, VB .NET
Projektarbeit	Erste Erfahrungen in Software-Projekten. Sehr gute Erfahrungen in Hardware-Projekten.
Nöske, Alexander (Fachinformatiker / AWE)	
Programmierung	Sehr gute Kenntnisse: Java, C#, Microsoft Dynamics NAV Gute Kenntnisse: mobiler Programmierung (Objective C), SQL, PHP, HTML, VB .Net
Projektarbeit	Professionelle Erfahrungen auch in größeren Software-Projekten, erste Erfahrungen mit Projektmanagementmethoden.

Tabelle 7: Personen

[6] Quelle: eigene Darstellung

3.4.2 Rollen

In Anlehnung an das Scrum-Modell aus Kapitel 3.1.1 gibt es in diesem Projekt 3 Hauptrollen, den Product Owner, den Scrum Master und das Team. Diese sind nachfolgend mit der jeweiligen Personen- und Rollenzuordnung aufgelistet:

- **Product Owner:** Kondratowicz, Andreas
- **Scrum Master:** Nöske, Alexander
- **Team:**

Name	Rolle(n)/Zuständigkeiten
Frank, Jens	App-Erstellung, Architekturmodell, Systemarchitektur, Materialplanung, Dokumentation & Präsentation
Kondratowicz, Andreas	Projektstrukturplan, eEPKs, Auswertungswebseite, Dokumentation & Präsentation
Nöske, Alexander	Projektplanung (Ziele, Risiken, Vorgehensmodell, Zeitplanungen), Server-Programmierung, ggfs. Auswertungen in App, Dokumentation & Präsentation

Tabelle 8: Rollen

3.4.3 Material

Für die Umsetzung wurden folgende Ressourcen verwendet:

1. Kommunikationskanäle: WhatsApp, Facebook, Skype, persönliche Meetings
2. Dateiaustausch: Dropbox
3. Smartphones für Debugging und Testing:
   a. Sony Xperia Mini Pro
   b. HTC Hero
   c. HTC One S
   d. HTC Sensation XL
   e. LG Nexus 4
   f. Vodafone 858 / Huawei 8160
4. Entwicklungs-PCs:
   a. Medion Akoya P6622
   b. Lenovo IdeaPad Y570
   c. MacBook
5. Entwicklungsumgebung:
   a. Eclipse Juno V 4.2.1
   b. Android Development Toolkit V21.0.0.v201210310015-519525
   c. Microsoft Visual Studio 2010 Professional
   d. Notepad++ 6.2.2
6. Server:
   a. Test-Server der HS-Niederrhein, zur Verfügung gestellt von Prof. Brell
   b. Testserver auf eigener Domain www.roflrofl.eu
      i. PHP-Version: 5.3.18
      ii. mySQL-Version: 5.1.66
7. verwendete Literatur:
   a. Java für Android: Native Android-Apps programmieren mit Java und Eclipse von Christian Bleske

b. Android Dokumentation, http://developer.android.com/index.html

Die Entwicklungen fanden sowohl in der Hochschule als auch im Home-Office statt.

3.5 Projektrisiken

Für das vorliegende Projekt konnten einige Projektrisiken identifiziert werden, die jeweils mit einer kurzen Beschreibung, Häufigkeit ihres Auftretens, ihrer Auswirkung und mit Lösungsvorschlägen erfasst wurden. Diese sind in aller Ausführlichkeit im Anhang aufgelistet.

Zur besseren Veranschaulichung können die Projektrisiken zudem noch in einem sog. Risikoportfolio grafisch dargestellt werden. Hierdurch wird auf einen Blick schnell klar, welche Risiken die größte Aufmerksamkeit verdienen.

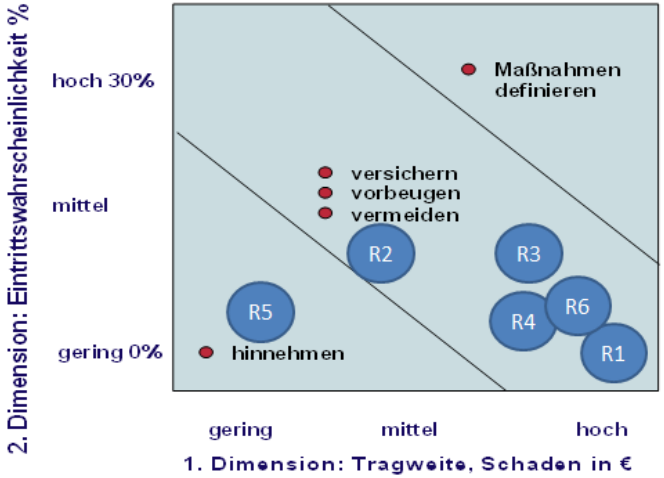

Abbildung 7: Risikoportfolio[7]

3.6 Lessons Learned

1. PHP ist auf dem Gebiet Programmtests und Debugging wenig komfortabel, oft ungenaue Fehlermeldungen. Im Rahmen der Entwicklung strengen die häufig vorkommenden und notwendigen Fensterwechsel an.
2. Im Bereich mobile Anwendungsentwicklung bietet Eclipse verglichen mit Xcode (iOS) oder Visual Studio (Windows Phone) wenig Komfort und Performance im Bereich des Debuggings. Tritt eine nicht behandelte Aufnahme auf, so stürzt die App ab mit der Fehlermeldung „Leider wurde z.B. Columban beendet“. Daraufhin muss im Logcat, welches sämtliche Aktivitäten des angeschlossenen Android-Phones aufzeichnet, zunächst Schritt für Schritt der Fehlerstack durchgangen und dort der Fehlercode gelesen werden. Zum Lesen der Fehlercodes bedarf es jedoch einiger Übung.
3. Im Bereich SQL gibt es keinen Bedarf größere Änderungen durchzuführen. Lediglich auf der Serverseite fallen erneut die häufigen Wechsel zwischen den Tools für die jeweiligen Zwecke auf. SQLite in Verbindung mit Android arbeiten gut zusammen und sind auch gewohnt gut dokumentiert. Ein großer Nachteil liegt in der Einschränkung des Entwick-

[7] Quelle: eigene Darstellung

lers in diesem Bereich, denn ohne Rootzugriff ist es überhaupt nicht möglich die gewünschte Datenbank für Betrachtungszwecke zu exportieren.

4. Die Abwärtskompatibilität der entwickelten App gestaltete sich als schwer zu nehmende Hürde, so musste mithilfe eines Workarounds ein Header implementiert werden. Dieser wäre ohne weiteres ab Android OS 4.0 vorhanden gewesen. Ergänzend spielen die vielen unterschiedlichen Auflösungen in Android eine wesentliche Rolle, um für jeden Anwender eine qualitativ hochwertige Bedienerfahrung zu gewährleisten. Im Bereich Auflösungsunabhängigkeit gibt es zwar von Google bereitgestellte Tricks und Kniffe, die allerdings für jeden Anwendungsfall unterschiedlich komplex in der Handhabung und Anwendung sind (Beispielsweise können Bilder nicht auf die gleiche Weise wie andere Steuerelemente auflösungsunabhängig konfiguriert werden).
5. Als Architekturmodell wurde das eher im angloamerikanischen Bereich anzutreffende Zachman-Framework aufgrund seines Umfangs und der daraus resultierenden schwierigen Handhabung durch das im europäischen Raum häufig anzutreffende ARIS-Modell abgelöst. ARIS wirkt strukturiert, übersichtlich und gleichzeitig einfach in der Handhabung. Dieser Eindruck bestätigte sich auch im weiteren Verlauf der Hausarbeit.
6. Für die Modellierung des Gantt-Diagramms wurde bewusst nach kurzem Einblick in die Software auf MS-Project verzichtet. Da in dieser Software keine Export-Funktionen für die Dateiformate .jpg- oder .png vorgesehen sind, war eine unnötig erschwerte Veröffentlichung der modellierten Daten zu erwarten. Gantt-Project erfüllte jedoch sogar als Freeware alle geforderten Kriterien und wurde daher im vorliegenden Projekt verwendet.
7. Microsofts Visio überzeugte erneut mit dem äußerst umfangreichen Paket an Standard-Shapes und wurde für die Erstellung des Projektstrukturplans, des Funktionsbaums sowie die Systemarchitektur verwendet. Für die ER-Modellierung erwies sich Visio jedoch als wenig vorteilhaft, sodass an dieser Stelle auf die Freeware UMLet zurückgegriffen wurde.
8. Unlösbare Problemfälle sind während der Projektdurchführung nicht aufgetreten

4 Systemarchitektur

4.1 Architekturmodell

Ursprünglich war für die Fertigstellung dieses Projekts die Verwendung des Zachman-Frameworks geplant. Von diesem Plan wurde jedoch frühzeitig wieder Abstand genommen. Die Gründe hierfür und welches Framework stattdessen verwendet wurde, werden im folgenden Abschnitt erläutert.

4.1.1 Auswahl des Modells

Als Architekturmodell wird das Zachman-Framework nicht genutzt, da dieser Ordnungsrahmen die Perspektiven des Planers, des Anwenders, des Architekten, des Designers, des Implementierers und schließlich das im Betrieb befindliche System erfasst. Für jede Perspektive sind die Fragestellungen Was? Wie? Wo? Wer? Wann? sowie Wieso? zu beantworten. Man erhofft sich im Allgemeinen durch diese sehr detailliert ausgeprägten Vorgaben einen geringeren zeitlichen Aufwand für das Architekturmodell sowie eine möglichst ganzheitliche Beleuchtung der projektrelevanten Sachverhalte. Die grafische Darstellung erfolgt als 6 x 6 Matrix. Als Architekturmodell bevorzugt man jedoch für kleinere Projekte das Modell Architektur integrierter Informationssysteme (ARIS) nach Scheer, um von der Gesamtheit und Einfachheit in der Anwendung und Betrachtung der Geschäftsprozesse zu profitieren.

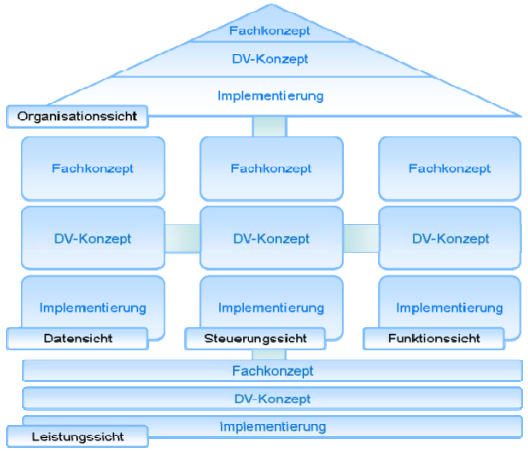

Abbildung 8: nicht ausgefülltes ARIS-Haus[8]

Das ARIS-Modell besteht aus der Organisations-, Daten-, Funktions-, Leistungs- und Steuerungssicht, die als zentrale Sicht alle anderen Sichten miteinander in Beziehung setzt. Auf der Organisationssicht wird man keine grafische Ausarbeitung niederlegen, da kein Unternehmen involviert ist, welches modelliert werden könnte.

Durch die Nutzung des ER-Modells, welches die im Projekt vorkommenden Informationen zueinander in Beziehung setzt, soll die Datensicht abgebildet werden. Die Datensicht umfasst alle Informationen und deren Verhältnis zueinander. Die Funktionssicht soll mit Hilfe eines Funktionsbaums die Ziele und Funktionen der Anwendung GPS-Tracker charakterisieren. Auf Ebene der Leistungssicht wird auf eine Modellierungsmethode verzichtet, da die Anwendung GPS-Tracker ein immaterielles Produkt sein wird. Man wird sich auf die Leistungsbeschreibung beschränken. Die Steuerungssicht wird alle Sichten durch die Modelliertechnik eEPK in Relation zueinander setzen.

[8] Entnommen aus: http://commons.wikimedia.org/wiki/File:ARIS-Modell.png, 27.12.2012

4.1.2 Modellierung

4.1.2.1 Organisationssicht

Da im Rahmen dieses Projekts kein Kunde vorhanden ist, entfällt die grafische Darstellung der Organisationssicht.

4.1.2.2 Datensicht

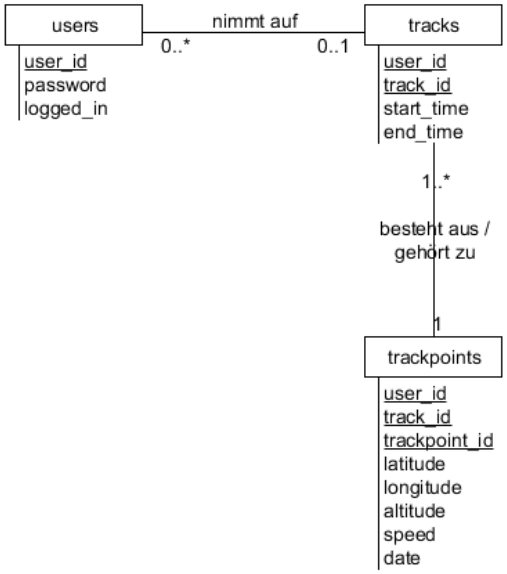

Abbildung 9: serverseitiges ER-Modell[9]

Ein Benutzer existiert, sobald er mindestens eine user_id besitzt. Die user_id entspricht der Android-ID, welche einzigartig ist. Ein Benutzer nimmt keinen oder mehrere Tracks auf. Ein Track gehört zu genau einem und zu mindestens einem Benutzer.

Ein Track besteht aus dem Primärschlüssel, der sich aus Nutzer-ID und Track-ID zusammensetzt. Des Weiteren wird das Datum sowie die Uhrzeit aufgenommen.

Ein Track besteht aus mindestens einem, oder mehreren Trackpoints. Ein Trackpoint gehört zu genau einem aufgenommenen Track. Der Trackpoint besteht aus dem Primärschlüssel, der sich wiederum aus der Nutzer-ID, Track-ID sowie der Trackpoint-ID zusammensetzt. Ferner wird ein Trackpoint durch Latitude, Longitude, Altitude und Speed charakterisiert.

Um serverseitig die Zuordnung der Tracks zu gewährleisten wird die Tabelle Benutzer benötigt, die auf der Client-Seite im Sinne der Gerätebindung einer App von marginaler Bedeutung ist.

[9] Quelle: eigene Darstellung

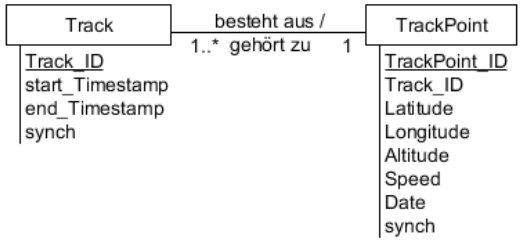

Abbildung 10: clientseitiges ER-Modell[10]

Mit Ausnahme der Tabelle Benutzer gleicht das nachstehende clientseitige ER-Modell dem serverseitigen Modell im Bereich der Tabellen. Durch die nicht benötigte Benutzertabelle - da die Benutzerdaten in den Einstellungen der App enthalten sind – findet die user_id sich nicht in der Tabelle Track und Trackpoint wieder. Um zwischen bereits synchronisierten und nicht synchronisierten Datensätzen zu differenzieren beheimatet jede Tabelle ein Feld „synch“, welches zur Filterung dient.

[10] Quelle: eigene Darstellung

4.1.2.3 Funktionssicht

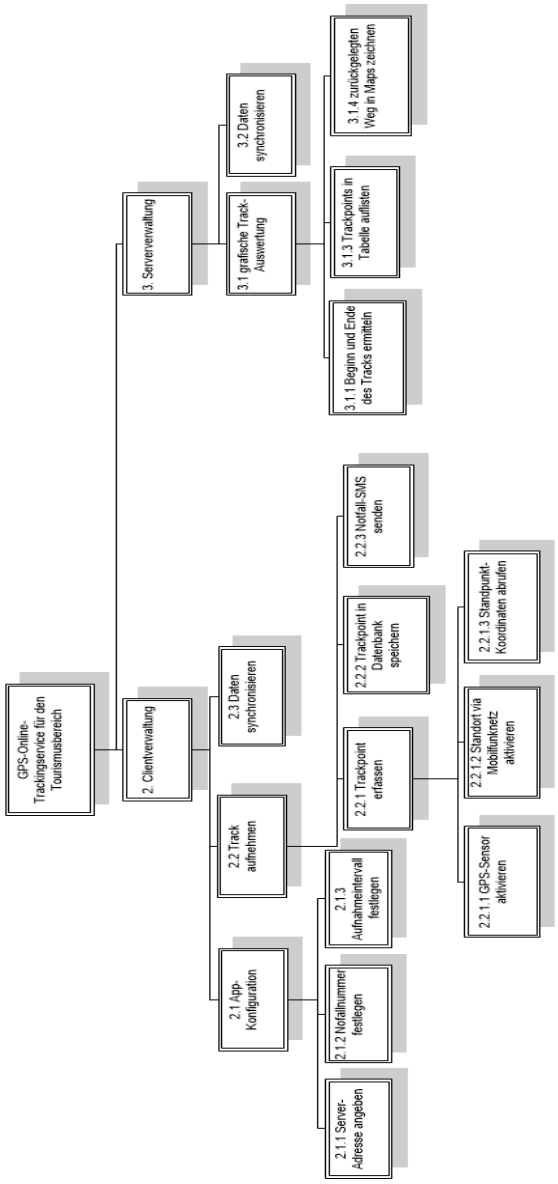

Abbildung 11: Funktionsbaum GPS-Online-Trackingservice für den Tourismusbereich[11]

[11] Quelle: eigene Darstellung

4.1.2.4 Steuerungssicht

Die Steuerungssicht des ARIS-Rahmens wird in Form von erweiterten Ereignisgesteuerten Prozessketten (eEPK) etikettiert (s. Kapitel 4.2).

4.1.2.5 Leistungssicht

Da keine materiellen Güter als Ergebnis dieses Projekts erwartet werden, ist es nicht möglich die Leistungssicht zu modellieren. Die Leistungssicht wird daher für das vorliegende Projekt nicht ausgefüllt.

4.2 Prozessanalyse und -modellierung

4.2.1 Beschreibung der Prozesse aus Kunden- bzw. Benutzersicht

Bei der Prozessanalyse und -modellierung ist es wichtig für den Kunden bzw. den Benutzer die Prozesse so einfach wie möglich zu gestalten. Diese Prozesse befinden sich dabei in einem ständigen Zyklus und wurden regelmäßig zur Verbesserung überarbeitet. Zu Beginn des Projektes gab es noch keinen direkten externen Partner, zu einem späteren Zeitpunkt fand jedoch ein Meeting mit Vertretern der GEMIT statt.

Die GEMIT ist ein Forschungsinstitut der Hochschule Niederrhein und befasst sich mit anwendungsorientierter Forschung in den Bereichen Logistik und IT. Die GEMIT hat mit einem Niederländischen Partner eine Applikation geplant, welche den Bereich des Grenzlands am Niederrhein und Südholland, auch bekannt als Euregio Rhein Maas Nord, übergreifend touristisch erschließen soll. Im Gespräch stellte sich jedoch schnell heraus, dass die Anforderungen an diese App von der bisher geplanten Umsetzung relativ stark abweichen bzw. in der Projektzeit nicht umgesetzt werden kann. Als Ergebnis des Meetings konnte aber festgehalten werden, dass die Entwicklung von Columban als ein Teilmodul oder Grundlagenforschung in die gewünschte Applikation einfließen könnte.

Daher werden an dieser Stelle nur die Prozesse modelliert, die für den ersten Prototypen umgesetzt werden sollen. Dies umfasst die Prozesse der Anmeldung, der Aufzeichnung und der Auswertung der Daten.

4.2.2 Auswahl der Modellierungssprache

Es wurde das Modell der ereignisgesteuerten Prozesskette gewählt, um die Prozesse zwischen dem Kunden und dem System zu beschreiben. Diese Geschäftsprozesse gelten als zeitlich-logische Abfolge betriebswirtschaftlicher Aufgaben mit dem Ziel einer Leistungserstellung.

Die zentralen Elemente zur Modellierung des Kontrollflusses sind:

- Ereignisse: Betriebswirtschaftlich relevante Zustandsübergänge und Bedingungen (passive Komponenten) mit einem Zeit-, Daten-, Bearbeitungs-, Benutzer- oder Nachrichtenbezug (Ereignisalgebra).
- Funktionen: Fachliche Aufgabe bzw. Tätigkeit zur Unterstützung eines oder mehrerer Unternehmensziele (aktive Komponenten) verbunden mit Zeit- und Ressourcenverbrauch.
- Verknüpfungsoperatoren (Konnektoren): Verknüpfung von Ereignissen und Funktionen und somit grafische Abbildung komplexer Geschäftsregeln. Die entsprechenden Konnektoren werden vereinfacht als AND-, OR- bzw. XOR-Operatoren bezeichnet.
- Kontrollflusskanten: Gerichteter Pfeil zwischen Ereignissen, Funktionen und Verknüpfungsoperatoren.
- Zur Bezeichnung der Funktionen wird das jeweilige Objekt der Bearbeitung und ein Verb im Infinitiv zur Kennzeichnung der zu verrichtenden Tätigkeit verwendet (z. B. Kundenauftrag anlegen). Bei Ereignissen wird das Objekt, welches eine Zustandsän-

derung erfährt, mit einem Verb im Partizip Perfekt verbunden (z. B. Kundenauftrag (ist) angelegt).

- Informationsobjekte stellen Dokumente oder sonstige Datenspeicher dar. Das Symbol für ein Informationsobjekt ist ein Rechteck. Beispiel: „Kundendatenbank“
- Organisationseinheiten (Symbol: Ellipse, die vor dem linken Rand eine senkrechte Linie enthält) symbolisieren Rollen oder Personen, die für den Prozess verantwortlich sind. Organisationseinheiten werden per durchgehender Linie mit Funktionen verbunden. Beispiele für Organisationseinheiten sind „Kunde“ oder „Marketingabteilung“.

4.2.3 Modellierung

4.2.3.1 Prozess 1: Benutzer Anmeldung Server/Smartphone

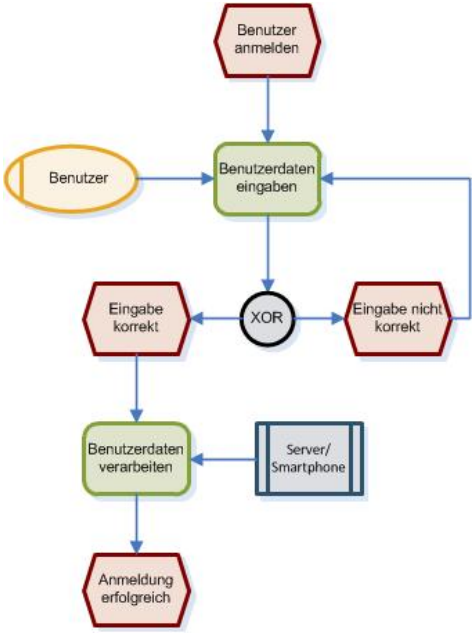

Abbildung 12: EPK „Benutzer Anmeldung“[12]

[12] Quelle: eigene Darstellung

4.2.3.2 Prozess 2: Track Aufzeichnung

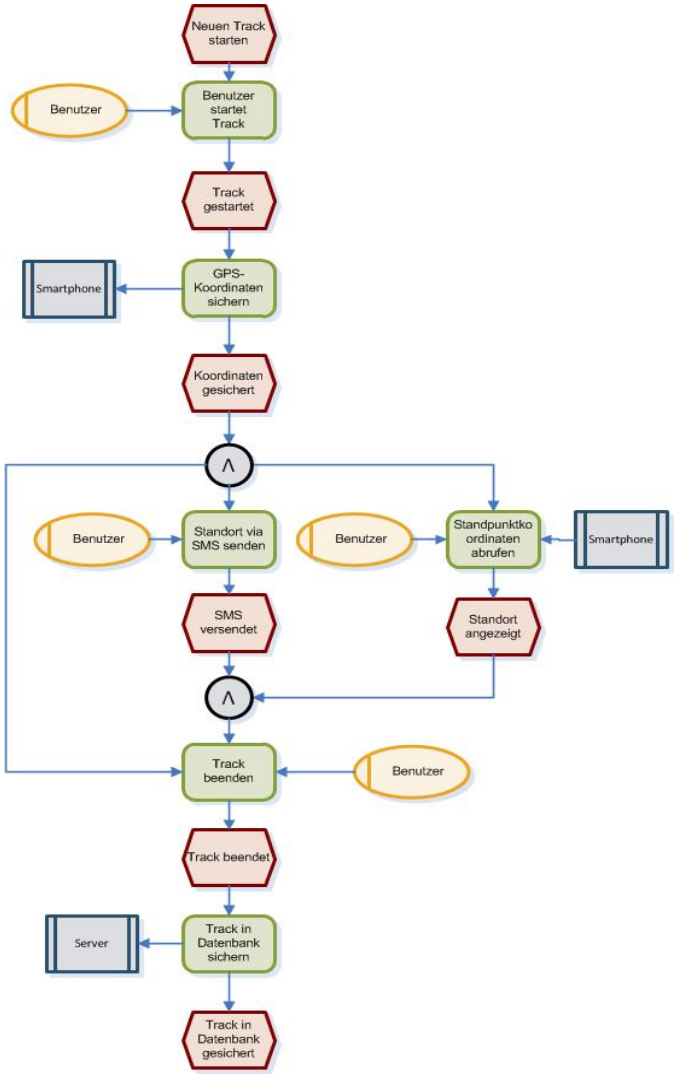

Abbildung 13: EPK „Track aufzeichnen“[13]

[13] Quelle: eigene Darstellung

4.2.3.3 Prozess 3: Track Auswertung

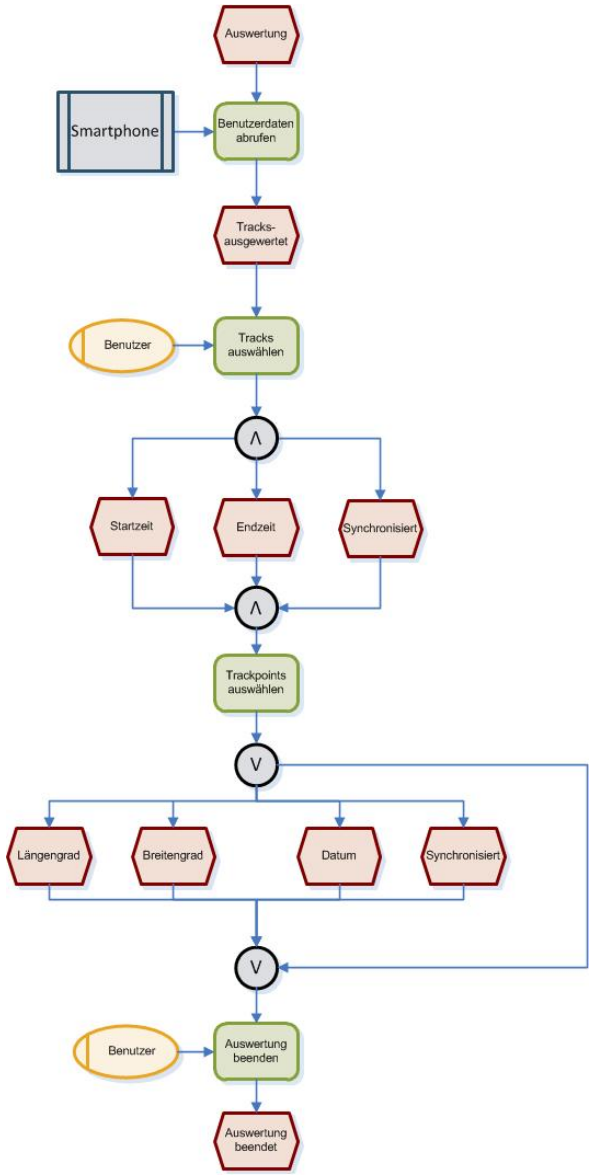

Abbildung 14: EPK „Auswertung“[14]

[14] Quelle: eigene Darstellung

5 Realisierung

In diesem Abschnitt werden kurz die Besonderheiten der jeweiligen Komponenten der Umsetzung erläutert. Der Quellcode ist im Anhang einzusehen.

5.1 Server

Der Columban-Server besteht aus insgesamt 11 PHP-Skripten, wovon jedoch 4 derzeit noch nicht aktiv verwendet werden, für zukünftige Entwicklungen im Bereich der Auswertung innerhalb der App jedoch bereits umgesetzt wurden:

1. **db_config.php**: Enthält alle nötigen Daten für die SQL-Verbindung zur Datenbank
2. **db_connect.php**: Wird in jedes weitere PHP-Skript eingebunden und stellt eine Verbindung zur Datenbank her
3. **create_database.php**: Wird einmal vom Installer aufgerufen und erstellt die Columban-Datenbank
4. **create_user.php**: Erstellt einen User aus den übergebenen Daten (ID & Passwort)
5. **create_track.php**: Erstellt einen Track mit User-ID, Track-ID und Startzeit
6. **create_trackpoint.php**: Erstellt zu einem Track einen Trackpoint mit allen übergebenen GPS-Daten und aktualisiert gleichzeitig den Endzeitpunkt des Tracks
7. **update_track.php**: Zuständig für das Aktualisieren des Endzeitpunkt eines Tracks

Noch ohne aktive Funktion innerhalb der App:

8. **login.php**: Ermöglicht es, einen User einzuloggen
9. **logout.php**: Ermöglicht es, einen User auszuloggen
10. **get_all_tracks.php**: Gibt alle Tracks oder alle Tracks zu einem bestimmten User inkl. Trackpoints als JSON-Objekt zurück
11. **get_trackpoints.php**: Gibt alle Trackpoints zu einem Track als JSON-Objekt zurück

Die Skripte arbeiten alle nach einem ähnlichen Schema. Zunächst werden die Datenbankparameter aus der Config-Datei gelesen und eine Verbindung zur Datenbank aufgebaut. Anschließend werden mittels SQL-Abfragen die benötigten Daten ausgelesen bzw. in der Datenbank eingefügt oder geändert. Das Ergebnis der SQL-Abfrage wird dann mittels JSON-Objekt ausgegeben. In den meisten Fällen besteht dieses Objekt aus einem „success“-Tag aus 0 bzw. 1 und einer kurzen Nachricht. So werden bei Fehlern in der Abfrage oder fehlenden benötigen Feldern im POST-Parameter eine Meldung mit dem Success-Code „0“ zurückgegeben, auf die in der App dann reagiert wird.

```
// benötigte Felder fehlen
$response["success"] = 0;
$response["message"] = "Benoetigte Felder fehlen.";

// JSON-Response ausgeben
echo json_encode($response);
```

Abbildung 15: Beispielhafter JSON-Response[15]

[15] Quelle: eigene Darstellung

Etwas aufwendiger gestaltete sich der Aufruf eines PHP-Skripts mit der POST-Methode aus PHP heraus, welches innerhalb der create_trackpoint.php durchgeführt werden muss um die Endzeit des Tracks zu aktualisieren. Dies konnte jedoch über das Kommandozeilen-Werkzeug cURL gelöst werden.

Der Quellcode befindet sich im Anhang im Ordner „htdocs“.

5.2 JAVA / Android-Programmierung

Um den Zugriff auf den integrierten GPS-Empfänger zu gewährleisten, werden zunächst die Berechtigungen „android.permission.ACCESS_FINE_LOCATION“ (GPS) sowie „android.permission.ACCESS_COARSE_LOCATION“ und „android.permission.ACCESS_NETWORK_STATE“ (für den Standort via Netzwerk) im App-Manifest ergänzt. Im Programmcode selbst wird der Location-Manager des Geräts mitsamt der Methode „requestLocationUpdates“ verwendet. Damit der Anwendung sich erschließt auf welche Weise die Standorterfassung durchgeführt und aktiviert werden soll, wird hier der jeweilige Provider mitgegeben. Es werden an dieser Stelle erstens der GPS-Empfänger sowie zweitens der Netzwerkempfänger genutzt. Für einen Übergabe-Parameter wird die Einstellung „userInt * 1000“ genutzt, diese entspricht dem Aufnahmeintervall für die Standortaktualisierung.

Eine Kern-Anforderung an die Applikation wird in der Synchronisation der Daten beschrieben. An dieser Stelle wird erneut eine Berechtigung (android.permission.INTERNET) im App-Manifest ergänzt, sodass der Zugriff auf das Internet gewährleistet wird. Im Bereich der Datenübertragung wird im http-Header einer http-Request jeweils ein nicht synchronisierter Datensatz aus der Datenbank an ein PHP-Skript gesendet. Das aufgerufene PHP-Skript antwortet mit einem String im JSON-Format, anschließend wird der String mithilfe eines JSON-Parsers ausgewertet und auf das Ergebnis des PHP-Skripts reagiert. Damit clientseitig die Übertragung jedes einzelnen Datensatzes gewährleistet wird und Überschneidungen verhindert werden, nimmt man Performanceeinbußen bei der Übertragung in Kauf. Nach 5 fehlerbehafteten Versuchen wird die Übertragung abgebrochen und zum nächsten Datensatz übergegangen.

Im Notfall kann eine SMS aus der Applikation gesendet werden, die Kontaktnummer wird aus den App-Einstellungen ausgelesen. Ferner wird die Berechtigung „android.permission.SEND_SMS“ im App-Manifest und der SMS-Manager des Betriebssystems benötigt, um auf die SMS-Funktionalitäten des Smartphones zuzugreifen. Der Sendevorgang wird unter Zuhilfenahme der Methode „send“ gestartet, diese erhält als Parameter die Telefonnummer aus den Einstellungen und dem Nachrichten-String. Daraufhin werden diverse Ergebnisnachrichten mithilfe von Broadcast-Empfängern abgeprüft und als Kurznachricht des Betriebssystems (Toast) ausgegeben, damit der Anwender feststellen kann, ob die SMS erfolgreich versendet wurde.

Der Quellcode befindet sich im Anhang im Ordner „Columban“.

5.3 Auswertungswebseite

Die Auswertungswebseite besteht aus zwei Hauptmodulen, der Map-Anbindung und der Auswertung der Tracks. Die Skripte und die Hauptprogrammierung befinden sich in der „tracking.php“. Die Webapp braucht zudem „chmod 777“, also den Vollzugriff für alle Benutzergruppen, um voll funktionsfähig zu sein. Nun werden die Besonderheiten der Auswertungswebseite beschrieben:

1. Zuerst wurde ein Login zum Schutz der Daten eingerichtet. Die Zugangsdaten erhält man in der „Users“ Tabelle in der Datenbank. An jener Stelle können die „user_id“ und das „password“ ausgelesen werden.
2. Bei der Map-Anbindung wurde mit Hilfe von Google Maps gearbeitet. An dieser Stelle müssen ein paar Punkte beachtet werden. Wichtig ist zunächst, dass die Daten aus der Datenbank zuerst in XML übertragen werden, da hierdurch das Laden der Seiten schneller, die Kartenanwendung flexibler und das Debugging unproblematischer wird. Wie die Daten verarbeitet werden, ist im nächsten Abschnitt erläutert. Wenn die XML-

Ausgabe funktioniert hat, wird die Karte mit JavaScript erstellt. Um eine Karte im öffentlichen Netz zu benutzen, muss zunächst eine Registrierung bei Google erfolgen. Der erhaltene Schlüssel wird anschließend im Quellcode hinterlegt.

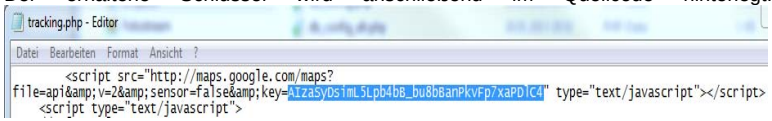

Abbildung 16: Google Key im JavaScript

Im JavaScript werden die Markierungen, die Karte, der Startpunkt, das Infofenster und die XML Daten miteinander verknüpft. Wenn die Markierungen, Symbole und XML verknüpft sind, wird beim Laden der Seite die „load“-Funktion aufgerufen. Diese Funktion richtet die Karte ein und ruft anschließend „GDownloadUrl“ auf, die zum Verarbeiten der XML Daten benötigt wird.

3. Die Informationen werden wie folgt abgearbeitet. Wie schon im vorigen Punkt beschrieben wurde müssen die Daten in XML eingelesen werden um diese im Nachhinein zu verarbeiten. Dies folgt im „phpsqlajax_genxml5.php“ Skript. Ein wichtiger Punkt ist hier die Übergabe des „$_POST['track_id']“, damit nicht alle Daten ausgelesen werden müssen sondern nur die die im aktuellen Track gesehen werden möchten. Diese wird dann mit einem SQL Befehl gefiltert. Die benötigten Daten werden dann in XML Format übertragen und verarbeitet. Angezeigt werden dann wie folgt die Start, Endzeit des Tracks sowie die Benutzerdaten des Users. Bei den weiteren Informationen werden die „Trackpoints“ einzeln aufgelistet mit den jeweiligen Werten aus der Tabelle.
4. Der Graph für die Höhenauswertung wurde mit Hilfe von JPGraph[16] fertiggestellt. Es wurden die Werte der Höhe in einem Array eingelesen, um diese in dem Graphen zu verarbeiten.

Der Quellcode befindet sich im Anhang im Ordner „htdocs/Webapp“.

5.4 Installer

Um die Benutzerfreundlichkeit des gesamten Produkts auch bei der Installation zu gewährleisten, wurde zum Projektende hin beschlossen, ein eigenes Installationsprogramm zu entwerfen. Der Installer wurde komplett in C# geschrieben. Er unterstützt den Administrator, indem er alle erstellten Server-Skripte auf den angegebenen Server lädt, die Angaben zur Datenbank-Verbindung übernimmt und anschließend das Skript zum Erstellen der Columban-Datenbank aufruft.

Die gesamte Anwendung besteht aus ca. 800 Zeilen Code. Bei der Umsetzung wurden folgende Besonderheiten und Finessen verwendet:

1. Da die Verbindung zum FTP-Server unter Umständen auch über eine verschlüsselte Verbindung stattfinden können muss (sftp), werden 2 Möglichkeiten der Verbindung implementiert.
   a. Verbindung ohne Verschlüsselung: Hierfür können Standardklassen verwendet werden. Zunächst wird ein Webrequest mit den angegebenen Server-Daten erstellt. Diesem werden die FTP-Nutzerdaten übergeben und anschließend können die Skripte mittels eines Streams auf den Server geladen werden.
   b. Verbindung mit Verschlüsselung: Visual Studio bzw. C# bietet keine Standardklassen für die Herstellung einer verschlüsselten Verbindung. Da die eigene Umsetzung in diesem Fall jedoch den Zeitrahmen deutlich gesprengt

[16] http://jpgraph.net/

hätte, wurde für diesen Fall auf eine externe DLL zugegriffen. Diese DLL wurde von Tamir Gal entwickelt und ist bekannt als SharpSSH[17]. Über diese DLL kann nun auch eine verschlüsselte Verbindung aufgebaut werden.

2. Vor dem Hochladen der Skripte müssen im Skript „db_config.php“ noch die angegebenen Daten zur SQL-Verbindung ergänzt werden. Dies wurde über Platzhalter durchgeführt. Das Programm öffnet die Datei, scannt diese und ersetzt die Platzhalter für den Datenbankserver („COLUMBANSERVER“), die Datenbank („COLUMBAN-DATABASE“), den Nutzer („COLUMBANUSER“) und das Passwort („COLUMBAN-PASSWORD“) durch die im Installer angegebenen Werte und speichert die Datei anschließend wieder ab.
3. Um die Handhabung des Installers möglichst einfach zu halten, soll dieser nur aus einer einzigen Datei bestehen. Da jedoch sowohl die Skripte als auch die benötigten externen DLLs eigentlich extern abgelegt sind, werden diese als Ressourcen direkt innerhalb des Installers abgelegt. Die benötigte DLL wird dann bei Start der Anwendung direkt aus den Ressourcen eingebunden. Die Skripte werden nach Betätigen des „Installieren“-Buttons zunächst ins temporäre Verzeichnis des Users gelegt, dort gegebenenfalls bearbeitet (s. Punkt 2), von dort auf den Server geladen und anschließend wieder gelöscht.
4. Bei der Eingabe der Daten wird der User durch Tooltips zu jedem Feld unterstützt. Diese Tooltips werden beim Start der Anwendung für jedes Feld festgelegt.
5. Außerdem führt das Auslassen von abschließenden „/“ oder führenden „http://“ zu keinem Fehler im Prozess sondern werden automatisch im Hintergrund ergänzt

Die aktuelle Umsetzung erfordert leider noch ein erneutes Hinzufügen der Skripte zu den Ressourcen und ein Rekompilieren des Installers, sollte sich an den Skripten etwas ändern.

Der Quellcode befindet sich im Anhang im Ordner „Columban-Installer“.

6 Manuale

In den folgenden Abschnitten soll kurz Erläutert werden, wie Columban installiert, betrieben und genutzt werden kann.

6.1 Implementierungsanleitung

6.1.1 Benötigte Ressourcen

Zum Einrichtung und Nutzung von Columban werden folgende Ressourcen benötigt:

1. Server mit folgenden Eigenschaften:
   a. PHP ab Version 5 getestet
   b. mySQL ab Version 5 getestet und der Möglichkeit, eigene Datenbanken mittels Skript zu erstellen
   c. FTP-Server
   d. cURL ab Version 7
2. PC mit folgenden Eigenschaften:
   a. Windows ab XP
   b. .NET Framework 4 (zum Ausführen des Installers)
   c. FTP-Client (zur manuellen Installation)
   d. Web-Browser mit aktiviertem JavaScript
3. Smartphone mit folgenden Eigenschaften:

[17] http://www.tamirgal.com/blog/page/SharpSSH.aspx

   a. Android ab Version 2.2
   b. GPS
   c. WLAN / 3G (zum Übertragen der Tracks, 3G ist hierbei jedoch kein Muss)

6.1.2 Notwendige Schritte

Für die Installation von Columban sind sowohl Arbeiten am Server als auch am Client (dem Smartphone) durchzuführen.

6.1.2.1 Installation Server

Wie bereits in Kapitel 5.4 erläutert, wurde ein spezieller Installer für Columban entwickelt, im Anhang unter „Columban-Installer.exe“ zu finden. Der Installer unterstützt den Anwender bereits durch Tooltips zu jedem Feld, trotzdem wird hier eine kurze Aufstellung gegeben:

1. Bereich FTP-Daten:
   a. Server: ftp-Server (z. B. hn08sv09.mg.hsnr.de)
   b. Verzeichnis: das Unterverzeichnis auf dem FTP-Server (z. B. Team2/)
   c. Port: FTP-Port des Servers
   d. SSH: bei einer verschlüsselten Verbindung muss der Haken gesetzt werden
2. Bereich mySQL-Daten: Eingabe der Daten des SQL-Servers und der Datenbank. Diese Daten werden auch in die db_config.php übertragen
3. Bereich Server: Webadresse des Servers, über den die Skripte ausgeführt werden können (z. B. http://hn08sv09.mg.hsnr.de:82/Team2/

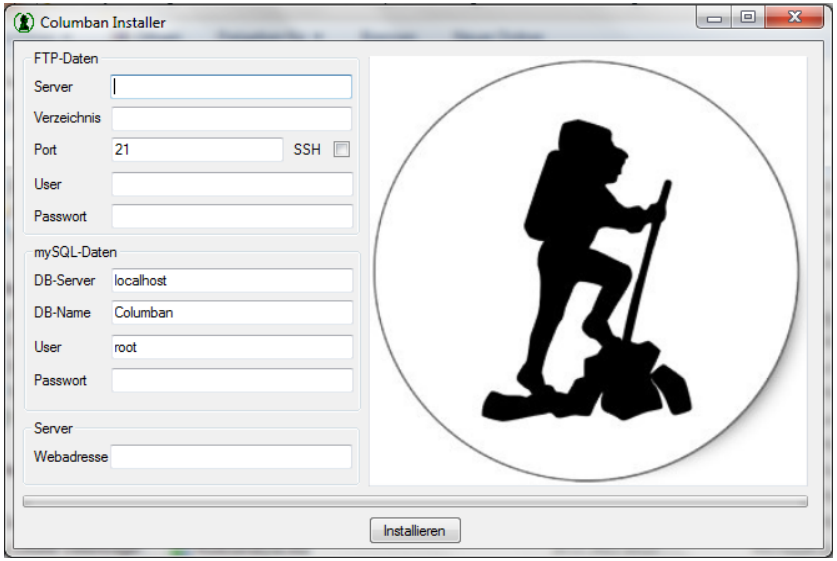

Abbildung 17: Columban-Installer[18]

[18] Quelle: eigene Darstellung

Sollte die Installation über den Installer fehlschlagen, können nicht durchgeführte Teile der Installation (siehe Status-Text im Installer) auch manuell durchgeführt werden.

Hierfür müssen zunächst die benötigten Daten zum SQL-Server und der Datenbank in der Datei db_config.php angepasst werden. Anschließend werden alle Skripte auf den Server geladen. Sollte der gewählte Hoster die Möglichkeit geben, mittels Skript die Datenbank zu erstellen, muss anschließend nur noch das create_database.php-Skript ausgeführt werden. Andernfalls muss die Columban-Datenbank manuell erstellt werden. Die genauen Feldnamen und -werte können dem create_database-Skript entnommen werden.

Die Notwendigkeit zur manuellen Installation trifft derzeit noch auf die Webapp / Auswertungswebseite zu, da eine Aufnahme in den Installer nicht mehr im Zeitbudget lag und einige größere Änderungen noch kurz vor der Abgabe vorgenommen wurden:

1. Den kompletten „Webapp"-Ordner mittels FTP-Verbindung in das gewünschte Verzeichnis legen.
2. Die Datenbankeinstellungen müssen manuell in der Datei „webapp/db_config.php" angepasst werden.
3. Sowohl der „Webapp"-Ordner als auch die Unterordner benötigen Schreibzugriff bzw. „chmod 777". Hierfür in z. B. Filezilla den Ordner auf dem Server mit einem Rechtsklick anwählen und über den Punkt „Dateiberechtigungen…" den numerischen Wert vergeben.
4. Nun kann die Auswertungswebseite über „/webapp/index.php" auf dem Server gestartet werden.

6.1.2.2 Installation Client

Die Anwendung liegt im .apk-Dateiformat vor und ist noch nicht über den Google Play Store erhältlich.

1. Die Datei muss auf das Android-Smartphone übertragen werden, im Anhang unter „Columban.apk" zu finden
2. Wie bereits angedeutet wird die App nicht über den Market bezogen. Hieraus resultiert für den Anwender, dass die Einstellung „Installation von Nicht-Market Anwendungen zulassen" aktiviert werden muss.
3. Im weiteren Verlauf der Installation sollten die benötigten Berechtigungen beachtet werden, da u. a. beim SMS-Versand Kosten entstehen können.
4. Nach der Installation bitte auf die Benutzeranleitung zurückgreifen.

6.1.3 Implementierungsaufwand

Die gesamte Implementierung sowohl des Servers als auch des Clients sollte auf Grund des mitgelieferten Installers für den Server und der simplen Handhabung der Installation von Android-Apps nicht länger als eine halbe Stunde dauern.

6.2 Benutzeranleitung

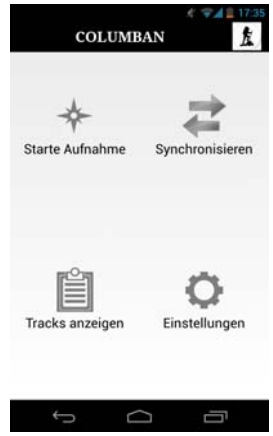

Abbildung 18: Columban – Startbildschirm[19]

Das Startmenü visualisiert übersichtlich die Funktionalitäten der App.

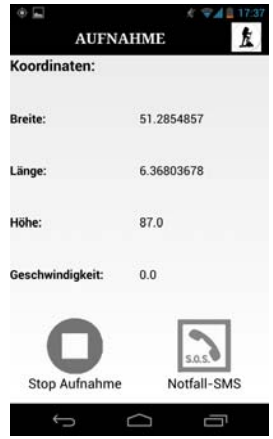

Abbildung 19: Columban – Standorterfassung[20]

Benutzerfreundlichkeit durch gleichzeitige Übersicht und Beschränkung auf die wesentlichen Programminhalte, eine SMS wird auf Knopfdruck gesendet.

[19] Quelle: eigene Darstellung

[20] Quelle: eigene Darstellung

Abbildung 20: Columban – Einstellungen[21]

Die Einstellungen ermöglichen den Abruf der Geräte-ID sowie des zufällig erzeugten Initialkennwortes. Des Weiteren wird hier die Telefonnummer für den Notfall oder aber das Intervall der Standortaktualisierung eingetragen. Die Synchronisation erfordert zwingend die Angabe einer Web-Adresse, unter dieser muss der Columban-Server eingerichtet sein.

Abbildung 21: Columban – vorhandene Aufnahmen

Die bisher aufgezeichneten Tracks werden zwecks Abwärtskompatibilität klassisch modern und gleichzeitig übersichtlich dargestellt, bereits synchronisierte und für die Synchronisation ausstehende Datensätze werden selbstsprechend markiert. Durch Klick auf die gewünschte Aufzeichnung werden die einzelnen Wegpunkte geladen und ähnlich klassisch aufbereitet (s. Abb. 22).

[21] Quelle: eigene Darstellung

Abbildung 22: Columban – Auflistung Wegpunkte einer Aufnahme[22]

6.3 Administratoranleitung

Die Administrierung der Datenbank kann auf dem Server z. B. mittels phpMyAdmin erfolgen. Für den reibungslosen Betrieb sollte dies jedoch nicht nötig sein. Auch die Bearbeitung der Skripte kann direkt auf dem Server durchgeführt werden. Im Problemfall sollten jedoch zunächst die Entwickler benachrichtigt werden. Eine weitere Administration ist zunächst nicht nötig und daher auch nicht vorgesehen.

7 Fazit und Ausblick

Zum Abschluss bleibt zu sagen, dass die Arbeiten am vorliegenden Projekt insgesamt gut verlaufen sind. So haben wir unser bisher vorhandenes Wissen vor allem in den Bereichen der Client-Server-Programmierung, Erstellung von Android-Apps und Projektmanagement vertiefen und erweitern können. In diesem Rahmen war es besonders interessant, bisher nur theoretisch Erlerntes über Vorgehensmodelle wie Scrum oder Architekturmodelle wie dem Aris-Framework an Hand eines konkreten Projektes selber praktisch anwenden zu können. Dies erforderte zwar einiges an Einarbeitungsaufwand und ist wahrscheinlich auch noch nicht komplett modellkonform durch uns umgesetzt worden, für zukünftige Projekte konnte jedoch einige Erfahrung gesammelt werden.

Auch die Zusammenarbeit im Team hat sich über das gesamte Projekt hinweg als sehr positiv dargestellt. Durch regelmäßige Treffen und aktuelle Kollaborationstechnik war der jeweilige Status der Arbeiten stets einsehbar und Änderungs- bzw. Anpassungsbedarf konnte im Bedarfsfall schnell kommuniziert werden. Hierdurch und vor allem durch das gewählte Vorgehensmodell sind wir zu keinem Punkt in Zeitdruck geraten und die Arbeiten konnten somit entspannt abgeschlossen werden.

Die Zukunft der Anwendung Columban gestaltet sich derzeit folgendermaßen:

Zunächst werden noch ein paar kleinere Verbesserungen und Bugfixes vorgenommen, um die App insgesamt noch etwas abzurunden. Aus persönlichem Interesse wird voraussichtlich zudem eine Umsetzung für Windows Phone als auch für das iPhone erfolgen. Wie bereits angedeutet könnte Columban zudem auch als Basis für eine umfangreichere Touristik-App

[22] Quelle: eigene Darstellung

genutzt werden, welche möglicherweise in einer Projektarbeit im Bereich des Masterstudiums oder durch das Institut GEMIT erfolgen könnte.

8 Anhang

8.1 User Stories

US1	**Der Nutzer kann seine Nutzerdaten eingeben**		**Client**
Beschreibung:	Der Nutzer muss im Client seine Nutzerdaten (min. Nutzer-ID) eingeben können. Ist die ID bereits vergeben, wird ihm dies mitgeteilt.		
Gruppe:	Login	**Priorität:**	hoch
geschätzte Zeit:	2 Tage		
US2	**Der Nutzer kann sich einloggen**		**Client**
Beschreibung:	Der Nutzer muss für weitere Funktionen erst eingeloggt sein.		
Gruppe:	Login	**Priorität:**	niedrig
geschätzte Zeit:	1 Tag		
US3	**Der Nutzer kann einen neuen Track starten**		**Client**
Beschreibung:	Der Nutzer muss einen neuen Track starten können		
Gruppe:	Tracking	**Priorität:**	hoch
geschätzte Zeit:	1 Tag		
US4	**Der Client nimmt GPS-Koordinaten auf**		**Client**
Beschreibung:	Der Client nimmt alle X-Sekunden die aktuellen Koordinaten auf und speichert diese in einer Datenbank. Abhängig von US5		
Gruppe:	Tracking	**Priorität:**	hoch
geschätzte Zeit:	2 Tage		
US5	**Der Nutzer kann Einstellungen festlegen**		**Client**
Beschreibung:	Der Nutzer kann verschiedene Einstellungen in der App hinterlegen. Dazu gehören: - Serveradresse - Intervall für Logging - Intervall für das Senden der Daten - Bonus: Notfallnummer		
Gruppe:	Einstellungen	**Priorität:**	hoch
geschätzte Zeit:			
US6	**Der Nutzer kann Trackingdaten manuell senden**		**Client**
Beschreibung:	Der Nutzer kann Trackingdaten manuell an den Server übertragen. Abhängig von US5		
Gruppe:	Kommunikation	**Priorität:**	mittel
geschätzte Zeit:	1 Tag		
US7	**Der Nutzer kann sich historische Daten ansehen**		**Client**
Beschreibung:	Der Nutzer kann sich historische Daten aus der lokalen Datenbank anzeigen lassen (inkl. Statistiken & Map-Ansicht)		
Gruppe:	Auswertung	**Priorität:**	optional
geschätzte	7 Tage		

Zeit:			
US8	**Der Nutzer kann eine Notfall-SMS senden**		**Client**
Beschreibung:	Der Nutzer kann über eine Notfalltaste eine SMS mit seinen aktuellen Koordinaten an eine festgelegte Nummer senden.		
Gruppe:	Bonus	**Priorität:**	niedrig
geschätzte Zeit:	3 Tage		
US9	**Der Client kann Trackingdaten übertragen**		**Client**
Beschreibung:	Der Client überträgt alle X Sekunden die aufgenommenen und noch nicht übertragenen Trackingdaten an den Server. Abhängig von US5		
Gruppe:	Kommunikation	**Priorität:**	hoch
geschätzte Zeit:	2 Tage		
US10	**Der Server kann Nutzerdaten verwalten**		**Server**
Beschreibung:	Der Server nimmt Nutzerdaten entgegen und speichert diese		
Gruppe:	Login	**Priorität:**	hoch
geschätzte Zeit:	1 Tag		
US11	**Der Server kann einen Login abhandeln**		**Server**
Beschreibung:	Der Server kann einen Login abhandeln und aktuell angemeldete Nutzer verwalten. Weitere Funktionalitäten sind abhängig vom Login.		
Gruppe:	Login	**Priorität:**	niedrig
geschätzte Zeit:	2 Tage		
US12	**Der Server kann Trackingdaten speichern**		**Server**
Beschreibung:	Der Server nimmt Trackingdaten entgegen und speichert diese.		
Gruppe:	Tracking	**Priorität:**	hoch
geschätzte Zeit:	1 Tag		
US13	**Der Nutzer kann sich eine Auswertung ansehen**		**Server**
Beschreibung:	Der Nutzer kann sich auf einer Webseite seine bisherigen Tracks und weitere Statistiken ansehen.		
Gruppe:	Auswertung	**Priorität:**	mittel
geschätzte Zeit:	5 Tage		
US14	**Der Nutzer kann die App intuitiv bedienen**		**Client**
Beschreibung:	Die Benutzeroberfläche sollte so gestaltet sein, dass sie übersichtlich ist und der Benutzer sich möglichst ohne fremde Hilfe zurechtfinden kann. Neben der intuitiven Bedienung sollte die App zudem auch optisch ansprechend sein		
Gruppe:	Optik	**Priorität:**	mittel
geschätzte Zeit:	7 Tage		
US15	**Aneignen des benötigten Vorwissens durch die Entwickler**		**Client**
Beschreibung:	Die Teammitglieder müssen sich vor dem eigentlichen Start der Umsetzung / Programmierung das nötige Grundwissen (z. B. im Bereich der Android-Programmierung oder Modellierung) aneignen		
Gruppe:	Vorbereitung	**Priorität:**	sehr hoch
geschätzte Zeit:	7 Tage		

Tabelle 9: User Stories

8.2 Projektrisiken

R1	**Missverständnisse im fachlichen Kontext**		
Beschreibung:	Der Entwickler hat eine Anforderung im fachlichen Kontext falsch verstanden und umgesetzt		
Effekt:	katastrophal	**Häufigkeit des Auftretens:**	selten
Vorbeugung und Behandlung:	Bevor eine solche Anforderung umgesetzt wird, wird zuerst ein Beispiel erstellt und besprochen, um sicher zu stellen, dass die Anforderung richtig verstanden wurde		
R2	**Schwierigkeiten durch neue Technologien**		
Beschreibung:	Es werden Technologien (z. B. Programmiersprachen oder Bibliotheken) verwendet, die für den Entwickler unbekannt sind		
Effekt:	mittel	**Häufigkeit des Auftretens:**	mittel
Vorbeugung und Behandlung:	Man sollte für eine Aufgabe, die unbekannte Technologien benötigt, mehr Zeit einplanen. Falls eine Aufgabe nicht in der geplanten Zeit fertiggestellt werden kann, kann die noch benötigte Zeit aus einem Zeitpuffer genommen werden, der bei der Sprintplanung eingeplant wurde. ist keine Zeit mehr im Puffer vorhanden, müssen andere Arbeitspakete in den nächsten Sprint verschoben werden.		
R3	**Zu geringe Aufwandsschätzung**		
Beschreibung:	Eine Aufgabe ist nach vorher abgeschätzter Zeit nicht fertiggestellt		
Effekt:	kritisch	**Häufigkeit des Auftretens:**	mittel
Vorbeugung und Behandlung:	Man sollte bei der Sprintplanung einen Zeitpuffer einplanen. Die noch benötigte Zeit kann aus dem Zeitpuffer genommen werden. Ist keine Zeit mehr im Puffer vorhanden, müssen andere Arbeitspakete in den nächsten Sprint verschoben werden. Zudem kann am Ende des Sprints dies noch einmal betrachtet werden und daraufhin im nächsten Sprint die Aufwandsschätzung verbessert werden.		
R4	**Schlechte Architekturentscheidungen**		
Beschreibung:	Eine architekturrelevante Entscheidung (z.B. für ein Framework oder eine Bibliothek) hat sich als schlechte Wahl herausgestellt.		
Effekt:	kritisch	**Häufigkeit des Auftretens:**	selten
Vorbeugung und Behandlung:	Um das Risiko minimieren zu können, sollten bei der Entscheidung mehrere Alternativen in Betracht gezogen werden. Die Entscheidung sollte dann mit Argumenten und den Alternativen dokumentiert werden. Hat sich eine Entscheidung früh als schlecht erwiesen, kann eventuell noch eine Alternative verwendet werden.		
R5	**Späte Anforderungsänderungen**		

Beschreibung:	Eine Anforderung ändert sich während eines Sprints		
Effekt:	wenig bedenklich	**Häufigkeit des Auftretens:**	selten
Vorbeugung und Behandlung:	Durch die Verwendung von Scrum ist der aktuelle Sprint nicht von der Anforderungsänderung betroffen. Die kurzen Sprints machen es möglich, dass die Änderung trotzdem relativ schnell beachtet werden kann		
R6	**Krankheitsbedingter Ausfall**		
Beschreibung:	Ein Mitarbeiter wird krank und kann nicht zur Arbeit kommen		
Effekt:	kritisch	**Häufigkeit des Auftretens:**	selten
Vorbeugung und Behandlung:	Dem Problem kann nicht vorgebeugt werden. Die verlorene Zeit kann aus dem Zeitpuffer genommen werden, der bei der Sprintplanung erstellt wurde. Ist keine Zeit mehr im Puffer vorhanden, müssen Arbeitspakete in den nächsten Sprint verschoben werden.		

Tabelle 10: Projektrisiken