HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

RadioTour

Studienarbeit, Abteilung Informatik

RadioTour Anwendung

Hochschule für Technik Rapperswil

Herbstsemester 2017

Autoren	Dominik Good und Urs Forrer
Betreuer	Prof. Dr. P. Heinzmann, Patrick Eichler

Arbeitsperiode	18.09.2017 – 22.12.2017
Arbeitsumfang	240 Arbeitsstunden bzw. 8 ECTS pro Student
Link	github.com/radiotour

Abstract

Die RadioTour Anwendung dient zur Verwaltung der aktuellen Rennsituation bei Radrennen. Sie wird seit mehreren Jahren an der Tour de Suisse eingesetzt. Die seinerzeit bei cnlab entwickelte und im Rahmen von Studienarbeiten erweiterte Version ist in die Jahre gekommen. Im Rahmen der vorliegenden Studienarbeit sollten einerseits die Bedienbarkeit und Performance verbessert werden und anderseits sollten neue Funktionen hinzugefügt werden.

In einer ersten Phase wurde die bestehende Applikation analysiert. Mit «inVision» wurden klickbare Prototypen erstellt, um neue Bedienkonzepte zu diskutieren. Ausserdem wurden neue Ansätze zur Datenspeicherung und für die Verbindung zum TourLive Server evaluiert. Resultate aus der Evaluation ergaben, dass der Einsatz der «Realm» basierten Datenbank einen deutlichen Performancegewinn mit sich bringt. Die Erkenntnisse aus der ersten Phase wurden in einem zweiten Schritt in einer von Grund auf neuen nativen Android Applikation (Java) umgesetzt. Fortlaufende User Tests in Zusammenarbeit mit dem Stakeholder und dem Hauptnutzer, dem RadioTour Speaker, optimierten die Applikation im Detail.

Das Resultat der Studienarbeit ist eine neue RadioTour Anwendung, die bereit ist für ausführliche Tests durch den RadioTour Speaker. Die Hauptmerkmale zu Performance und Funktionserweiterung sind erfolgreich umgesetzt worden. Dazu gehören korrekte Verarbeitung der GPS Informationen der API, sowie der lokale GPS Daten. Des Weiteren Berechnungen zu aktuellen Rangierungen auf verschiedenen spezifischen Daten (Punkte, Zeiten, virtuelle Führende), die Vereinfachung der Datenimportierungsfunktion mit zusätzlichem Fehlerhandling und Einführung eines Demostrationsmodus, der es erlaubt die Anwendung im Offline-Betrieb zu nutzen.

Es ist geplant, in einer nachfolgenden Bachelorarbeit aufgrund der ausführlichen Feedbacks der Tester, die erfassten RadioTour Daten zusätzlich auf der Tour Live Webplattform anzubieten. So hat neben den Tour de Suisse Organisatoren auch die Öffentlichkeit die Möglichkeit Informationen zum Rennverlauf einzusehen.

Aufgabenstellung[1]

Studiengang	Informatik (I)
Semester	HS 2017
Durchführung	Studienarbeit
Fachrichtung	Internet-Technologien und –Anwendungen
Institut	INS
Gruppe	Urs Forrer, Dominik Good
Betreuer	Peter Heinzmann und Patrick Eichler

Ausganslage

Der RadioTour Speaker ist die zentrale Renninformationsstelle bei Profi Radrennen. Er informiert über Zeitabstände und Zusammensetzung von Fahrergruppen, Zwischenklassemente und Wertungen (Sprint, Bergpreis), spezielle Ereignisse (z.B. Sturz, Defekt, Aufgaben) und allgemeine Angaben zum Rennen (z.B. Aufgebote zu Dopingkontrollen). Diese Daten sammelt der RadioTour Speaker via Funk von seinem Chrono Team, welches die Fahrergruppen auf dem Motorrad begleitet.

Seit einigen Jahren verwendet der RadioTour Speaker der Tour de Suisse eine Tablet Anwendung. Diese soll nun erneuert und erweitert werden.

Ziel

Nach dieser Arbeit soll eine robuste, neue RadioTour-Anwendung mit erweiterten Funktionen vorliegen.

Aufgaben

Einarbeitung
- Grundkenntnisse zum Ablauf von Profi Radrennen (Tour Guide, Marschtabellen, Rennreglemente studieren)
- Studium und Testing der vorhandenen RadioTour Anwendung
- Realisierung einer Testing Umgebung

Analyse
- Bestimmung der relevanten Use Cases
- Festlegung der Anforderungen
- Technologieentscheid

Realisierung
- Realisierung der neuen Anwendung

Testing

[1] Gemäss der Aufgabenstellung von avt.hsr.ch

- Ausführliche User Tests

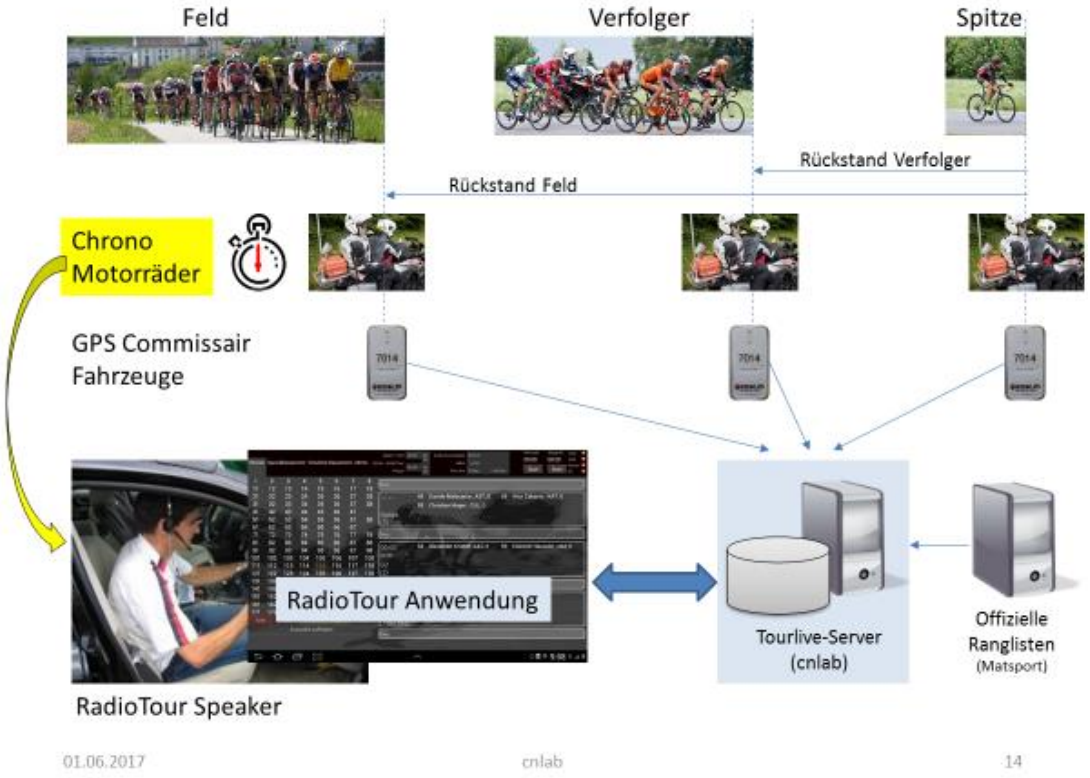


Abbildung 1: Übersicht RadioTour und TourLive

Referenzen und Beispiele

1. **Hinweise zur Durchführung von Studienarbeiten**
   https://drive.switch.ch/index.php/s/Freq5JbeMUJuDQ4
2. Florian Bentele & Daniel Stucki, **Studienarbeit**, Abteilung Informatik **Android Applikation RadioTour**, Hochschule für Technik Rapperswil Frühjahrssemester 2012. http://docplayer.org/2494735-Android-applikation-radiotour.html
3. Steve Bovet, RadioTour Speaker Tour de Suisse, «**The feeling of being in the middle of the race is indescribable**»
   http://www.tourdesuisse.ch/en/news/news-detail/news/the-feeling-of-being-in-the-middle-of-the-race-is-indescribable/
4. **TourLive Anwendung**, www.tourlive.ch

Notizen

Die nur lesen Anzeige der Rennsituation mit virtuellem Klassement wäre auch für die Kommissäre nützlich
Tricots und Wertungen

Erklärung zur Urheberschaft[2]

Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellungen und Beiträgen gemäss folgender Aufstellung:

Gegenstand, Leistung	**Person**	**Funktion**
Studienarbeit «Android Applikation RadioTour»	Florian Bentele & Daniel Stucki	Autoren der Arbeit
Studienarbeit «RadioTour Anwendung»	Dominik Good & Urs Forrer	Autoren der Arbeit
Idee, Aufgabenstellung, Betreuung während der Arbeit	Prof. Dr. P. Heinzmann	Verantwortlicher Professor
Betreuung während der Arbeit, Backend Entwicklung	Patrick Eichler	Mitbetreuer der Arbeit

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt habe,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe,
- dass in der Arbeit verwendete urheberrechtlich geschützte (Copyright) Inhalte (insbesondere Fotografien und Grafiken) klar gekennzeichnet und mit Quellenhinweis versehen sind,
- dass Inhalte die unter Creative-Commons-Lizenz veröffentlicht wurden, klar gekennzeichnet sind.

Rapperswil, den..... .....
Dominik Good

Rapperswil, den..... .....
Urs Forrer

[2] Diese Erklärung basiert auf der Muster-Erklärung in den Richtlinien der HSR zur Durchführung von Projekt-, Studien, Diplom- oder Bachelorarbeiten vom 07. September 2011.

Vereinbarung zur Verwendung und Weiterentwicklung der Arbeit[3]

Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit „RadioTour“ von Dominik Good und Urs Forrer unter der Betreuung von Prof. Dr. P. Heinzmann (für die Arbeit verantwortlicher Professor) geregelt.

Urheberrecht

Die Urheberrechte stehen der Studentin / dem Studenten zu.

Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, d.h. von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Professor sowie vom Industriepartner verwendet und weiterentwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht, aber nicht Pflicht.

Rapperswil, den..... .....

Dominik Good

Rapperswil, den..... .....

Urs Forrer

Rapperswil, den..... .....

Prof. Dr. P. Heinzmann

[3] Diese Vereinbarung basiert auf der Muster-Vereinbarung in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 07. September 2011.

Management Summary

Ausgangslage

Durch Grössen wie Fabian Cancellara sind Radrennen in der Schweiz populär geworden. An den alljährlichen «Tour de Suisse» Rennen nehmen im Schnitt ca. 180 Rennfahrer teil, diese sind aufgeteilt in verschiedene Fahrergruppen welche sich während der Rennen dynamisch verändern. In einem Begleitfahrzeug während des Rennens sitzt der Radio Tour Speaker und trackt die aktuelle Rennsituation (Gruppenbildungen, Wertungen, usw.). Um dem Radio Tour Speaker einen Papierkrieg zu ersparen wurde eine Tablet Anwendung realisiert.

Die Anwendung wurde durch cnlab entwickelt und in vorhergehenden Studienarbeiten aktualisiert. Aufgrund der schlechten Reaktionsfähigkeit und nicht mehr 100%iger Erfüllung der Anforderungen wurde entschieden, die Anwendung neu zu entwickeln.

Das Ziel dieser Studienarbeit ist es, die Bedienbarkeit der Anwendung benutzerfreundlich zu gestalten und Funktionalitätserweiterungen umzusetzen.

Vorgehen und Technologie

Das Projekt wurde anhand von RUP (Rational Unified Process) mit den vier Phasen Inception, Elaboration, Construction und Transition geplant. Um während des Projektes agil zu arbeiten wurde alle zwei Wochen ein Feedback-Loop zur Anwendung eingebaut.

In einem ersten Schritt wurden die Anforderungen analysiert und festgelegt. Dies durch Analyse der bereits vorhandenen Software, sowie durch eine Stakeholder-Analyse. Ebenso wurden die nicht funktionalen Anforderungen für das Projekt definiert.

Neue Technologie wie «Realm» wurden evaluiert um den nicht funktionalen Anforderungen gerecht zu werden. In einer weiterführenden Forschung im Netz wurde ein entsprechend für die Anwendungszwecke passendes Tablet evaluiert.

Nach der Festlegung der Architektur der Anwendung und deren spezifischen Eigenschaften wurde die Implementation auf der Android Plattform gestartet. Während der Entwicklung wurden stetig Tests durchgeführt, damit die Codequalität gewährleistet ist. Ebenso wurden Tests mit dem Hauptnutzer arrangiert um eventuelle Schwachstellen der Anwendung zu identifizieren.

Ergebnisse

Abbildung 2: Radio Tour Anwendung

Aus der Studienarbeit resultierte eine neue Anwendung für die Verwaltung eines Radrennens. Für die Benutzeroberfläche wurde das «Material Design» verwendet, um eine ansprechende, innovative Oberfläche zu gestalten. Die Applikation wurde von Grund auf neu aufgebaut und mit neuen Technologien umgesetzt. Sie erlaubt es dem Radio Tour Speaker die Rennsituationen zu managen und sich schnell einen Überblick über aktuelle Ereignisse zu verschaffen. So werden unter anderem im Hintergrund dynamisch die virtuellen Leader der jeweiligen Klassemente berechnet und visualisiert. Dies um Aussagen über die Entwicklung des Rennens mit aktuellen Daten zu machen, ohne die Notwendigkeit von eigenen Berechnungen und Interpretationen. Die Daten für das Radrennen werden von einer API bezogen, welche bereits existiert hat. Die Anbindung an die API erlaubt die aktuell stattfindende Etappe zu importieren. Zudem gibt es einen Mechanismus welcher Veränderungen an der Rennsituation an die API weiterleitet.

Ausblick

Die vorliegende Version der Radio Tour Anwendung ist bereit für ausführliche Tests durch den Radio Tour Speaker, dies ist auch während eines Radrennens möglich. Gegebenenfalls entstehen während dieses Einsatzes weitere Anforderungen an die Anwendung. Die Implementierung des Uploads der Daten an die API und deren Persistierung, sowie weitere Anwendungsanpassungen werden im Rahmen einer weiterführenden Bachelorarbeit erfolgen.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangslage

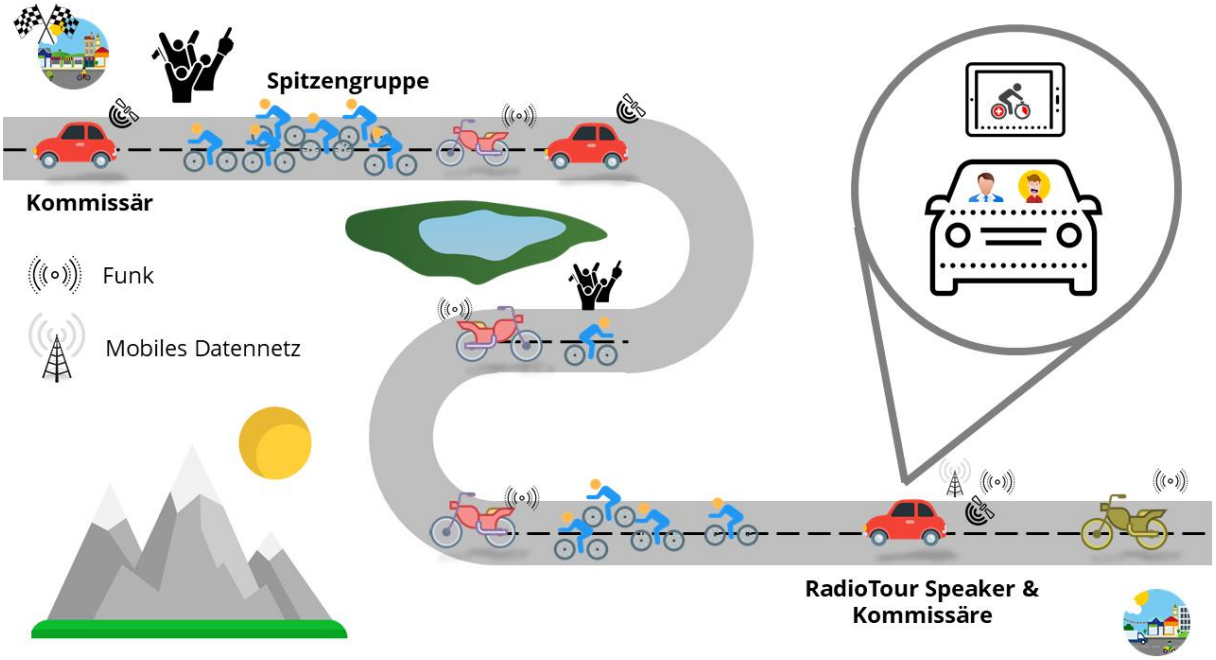


Abbildung 3: Big Picture (Rennkolone)

Während einem Radrennen werden viele Informationen zwischen dem RadioTour Speaker und den anderen Beteiligten (Kommissäre, etc.) ausgetauscht. Die RadioTour Anwendung bietet die Möglichkeit diese Informationen (Fahrergruppen, erreichte Wertungen, Zeitabstände, usw.) festzuhalten. Die «Abbildung 3: Big Picture» stellt eine typische Rennsituation dar und zeigt die Einbindung aller relevanten Aktoren der RadioTour Anwendung.

Neben den Rennfahrern in der Abbildung 3 «Big Picture», welche sich auf die Tour de Suisse bezieht, sind diverse Motoradfahrer und Autos in einer Kolonne unterwegs. Jede Gruppe, welche während eines Rennens vorhanden ist, besteht neben den Radfahrern zusätzlich aus einem Motorrad zusammen mit dessen Fahrer, welche im dauernden Kontakt mit dem RadioTour Speaker stehen. Der RadioTour Speaker (platziert in einem Auto am Ende der Kolone) hält die über Funk erhaltenen Informationen in der Tablet Anwendung fest und besitzt somit immer einen aktuellen Überblick über die Rennsituation. Zusätzlich zu den Motorradfahrern und dem RadioTour Speaker sind noch Kommissäre, Werbekolonen, Mannschaftsfahrzeuge und Streckensicherungsfahrzeuge unterwegs. Zur Übersicht wurden diese in der «Abbildung 3: Big Picture» nicht abgebildet.

1.1.1 Bisherige Anwendung

Die folgenden Bilder zeigen die wichtigsten Ansichten der bestehenden RadioTour Anwendung.

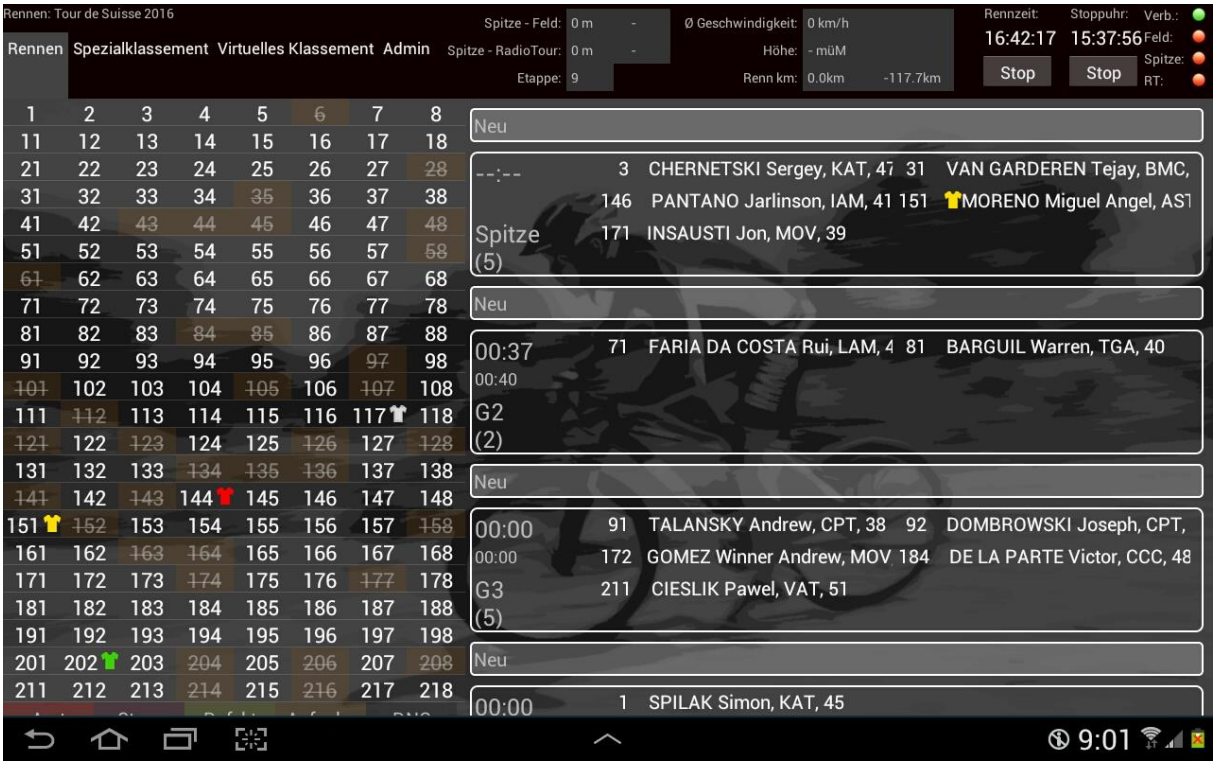


Abbildung 4: Bestehende Applikation Rennen

Abbildung 5: Bestehende Applikation Spezialklassement

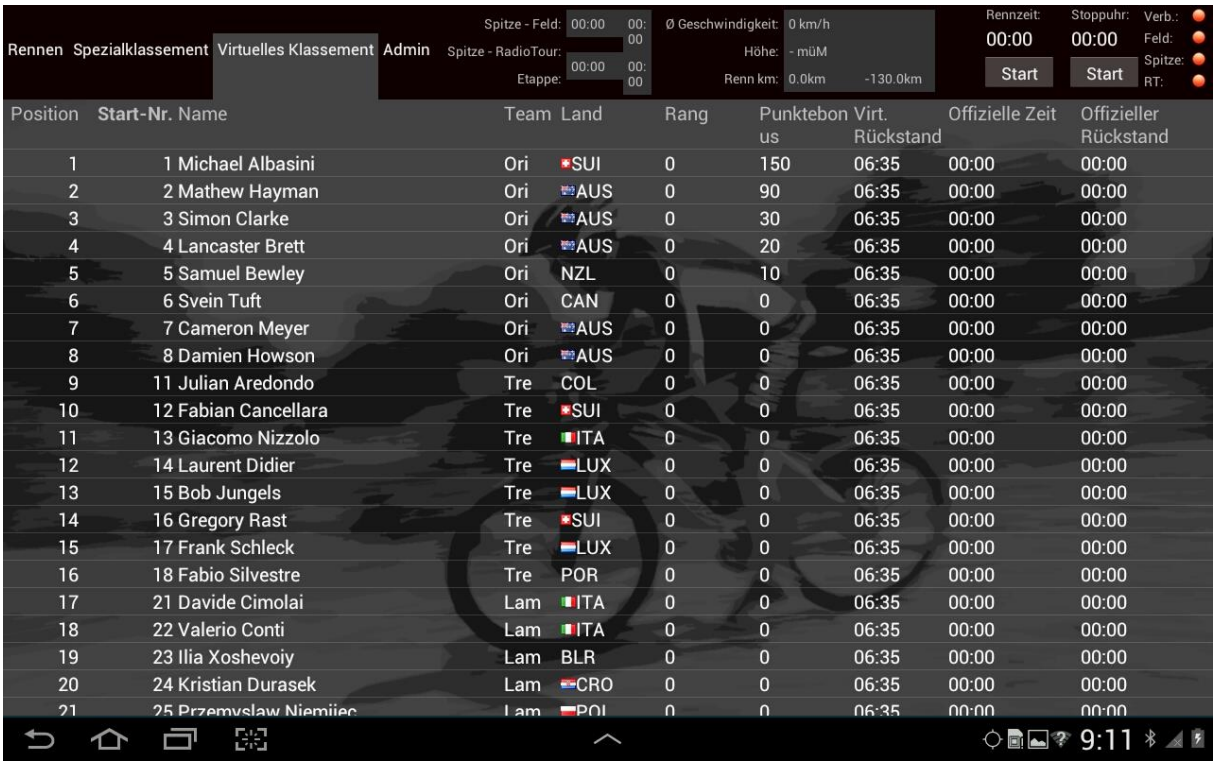


Abbildung 6: Bestehende Applikation Virtuelles Klassement

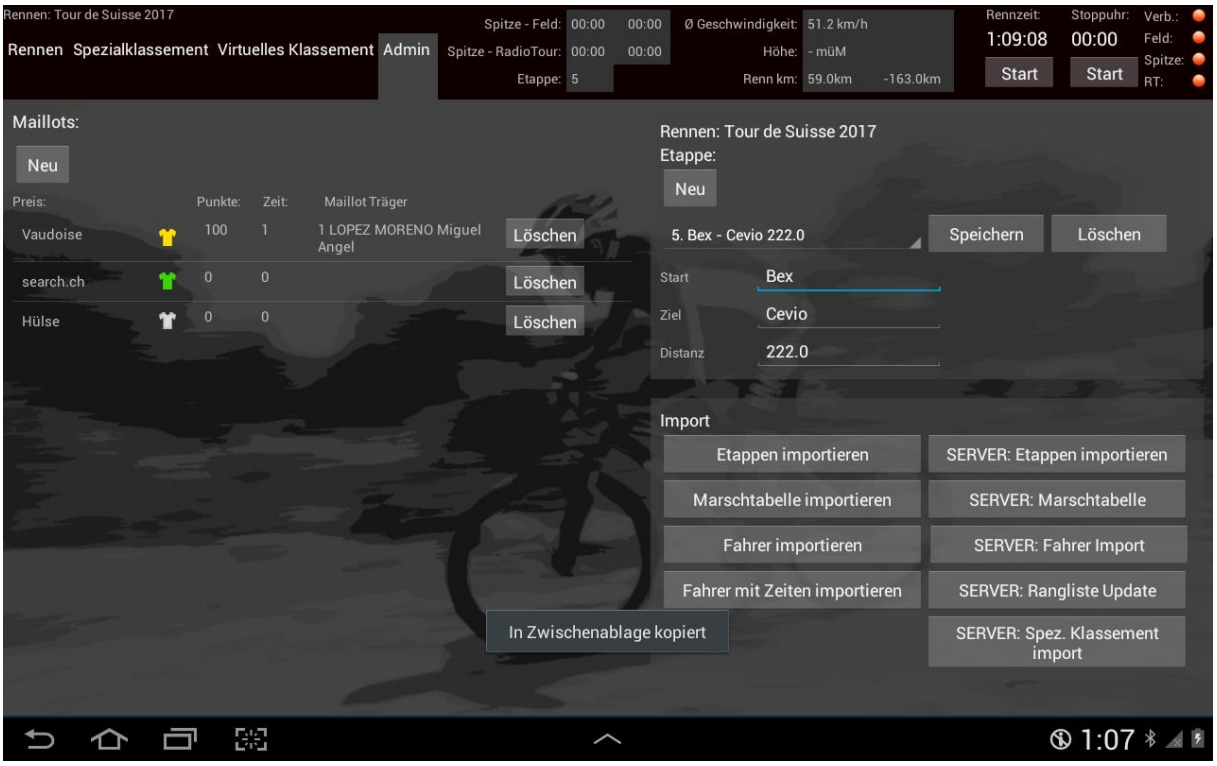


Abbildung 7: Bestehende Applikation Admin

1.1.2 Umfeld

Die RadioTour Anwendung stellt einen kleinen Teil eines grösseren Systems rund um Radrennen dar, welches durch cnlab betrieben und zur Verfügung gestellt wird.

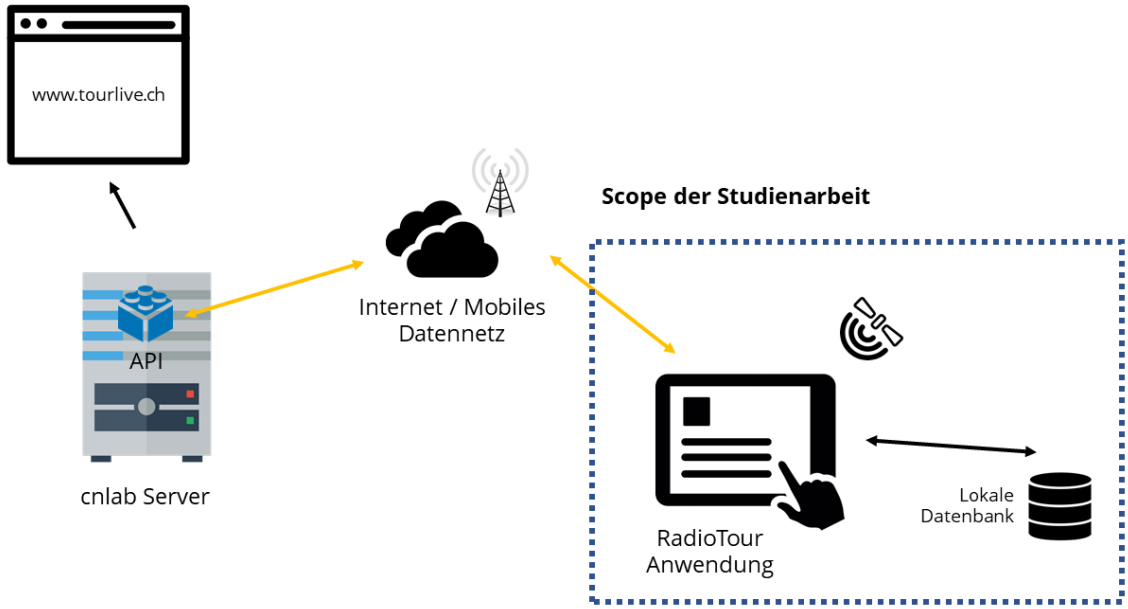


Abbildung 8: Umsysteme

Die durch den RadioTour Speaker eingegeben Informationen auf dem Tablet werden in der lokalen Datenbank abgespeichert. Im gleichen Schritt werden Daten wie Rennkilometer und GPS Lokationen von der API bezogen. Zu Beginn jedes Rennens besteht die Möglichkeit aktuelle Stammdaten (Fahrer, Leistung, usw.) von der API zu importieren. Der cnlab Server bzw. die API dient als Kommunikationspunkt zur RadioTour Anwendung. Die Übertragung geschieht über das Mobile Datennetz (während dem Rennen) oder im Start-/Zielbereich über WLAN.

Die Daten der API dienen des Weiteren als Datenbasis für diverse Webanwendungen (erreichbar unter tourlive.ch). Die Webseite bereitet die Daten entsprechend auf und stellt die Informationen weiteren Beteiligen (SRF, usw.) zur Verfügung.

1.2 Vision

Diese Arbeit bringt die RadioTour Anwendung auf ein neues Niveau. Die Anwendung ist auf den aktuellsten verfügbaren Systemen umgesetzt und verwendet den State-Of-The-Art. Zudem weist sie eine bessere User Experience auf. Bedienelemente sind für einen versierten RadioTour Speaker einfach zu begreifen. Es wird dabei immer darauf geachtet, ein angenehmes Verhältnis aus bereits bewährten Benutzerkonzepten und neuen Interaktionsaspekten zu haben.

1.3 Aufbau der Arbeit

Die Arbeit ist in verschiedene Bereiche gegliedert. In einem ersten Teil, dem Kapitel 2, wird die realisierte Anwendung mit deren Interaktionsmöglichkeiten im Detail erläutertet. Dieses Kapitel richtet sich prinzipiell an die Anwender der App. Dieser Teil verfolgt das Ziel dem Anwender die wichtigsten Interaktionen zu beschreiben und kann als kurzes Handbuch dienen. Der zweite Teil der Arbeit beinhaltet das Requirements Engineering und die Analyse, das Software Engineering, die Realisierung sowie die Qualitätsmassnahmen. Die Kapitel 3 – 6 richten sich somit nicht an den Anwender. Diese Kapitel richten sich an technische Fachleute. Mit den Kapitel 3 – 6 wird die Anwendung so dokumentiert, dass diese in einem späteren Stadium durch andere Entwickler weiterentwickelt werden kann. Abgerundet wird die Arbeit mit einer Schlussfolgerung im Kapitel 7.

Informationen zum Aufbau der Arbeit sind dem Kapitel 8.2 Projektmanagement im Anhang zu entnehmen.

2 Anwendung

In diesem Kapitel werden die einzelnen Interaktionsmöglichkeiten mit den verschiedenen Bedienelementen der Applikation kurz erläutert. Dies dient dazu einen Überblick über den Funktionsumfang zu liefern. Das Kapitel zielt auf die allgemeine Nutzergruppe ab, technische Details werden in den Folgekapiteln erläutert.

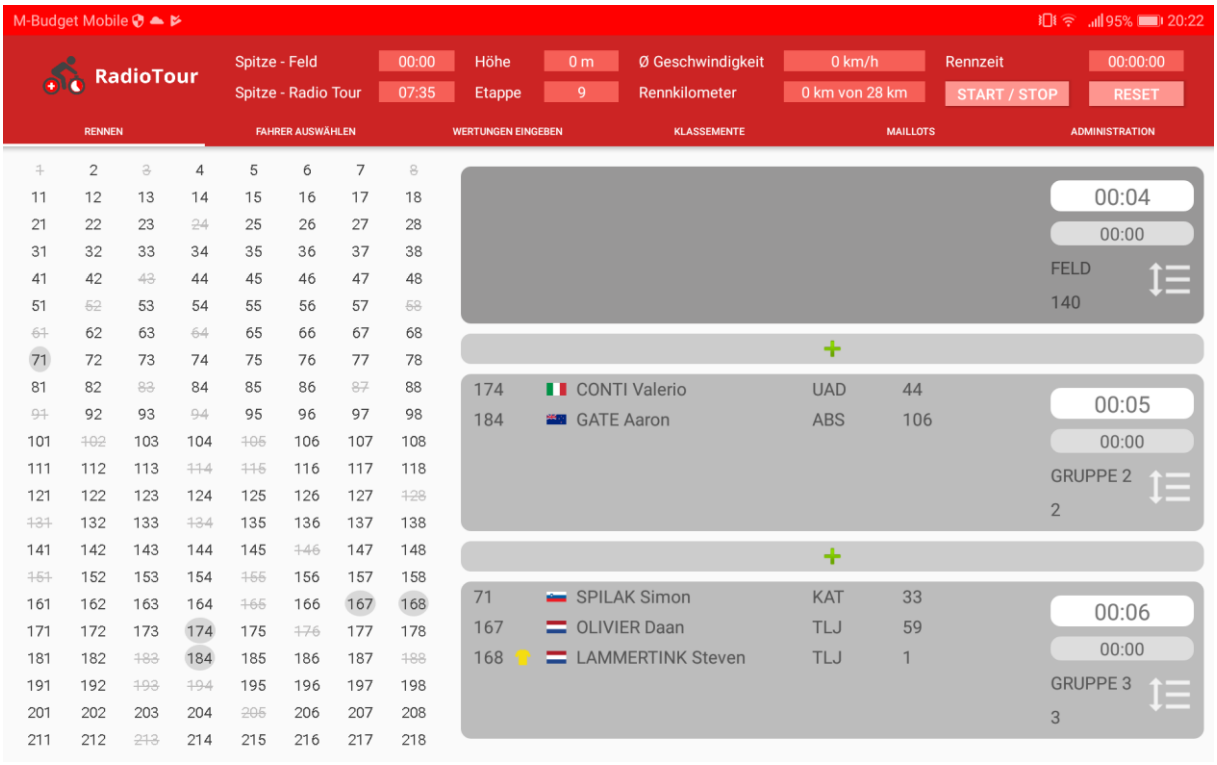


Abbildung 9: Gesamtübersicht

2.1 Infoleiste

Abbildung 10: Infoleiste

In der Infoleiste sind diverse Angaben zum aktuellen Rennen sichtbar. Dabei werden zum einen GPS Tracker von der API ausgewertet und zum anderen lokale GPS Daten verwendet. Über die Felder «Start / Stop» und «Reset» ist es möglich das Rennen zu starten, zu stoppen und zurückzusetzten. Die GPS Daten werden nur während eines aktiven Rennens aktualisiert. Nach dem Start der Anwendung bzw. dem Start des Rennes über den entsprechenden Button dauert es etwa 60 Sekunden, bis die GPS Position verfügbar ist. Dies ist durch die Anlaufphase des GPS geschuldet.

In der Tab-Bar ist es möglich zwischen verschiedenen Anzeigeelementen umzuschalten, welche in den folgenden Kapiteln genauer beschrieben werden.

2.2 Rennen

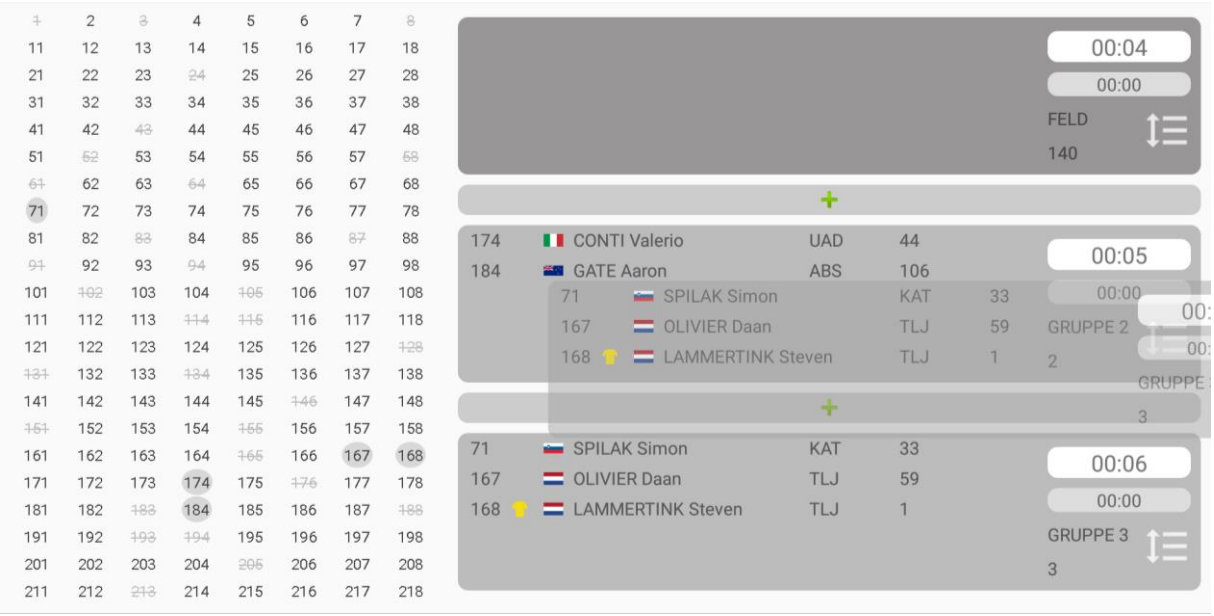


Abbildung 11: Rennen

Diese Ansicht ist die Standardansicht des Rennens. Es bietet einen Überblick über Fahrerpositionen und zeigt an welche Fahrer sich in welcher Gruppe befinden. Grau hinterlegte Fahrer befinden sich in einer Gruppe die nicht zum Feld gehört. Hellgrau dargestellte Fahrer nehmen nicht mehr aktiv am Rennen teil, sie sind ausgeschieden oder haben (Status DNC und Defekt) aufgegeben. Die Status Sturz, Defekt und Art werden in dieser Ansicht nicht dargestellt (mehr dazu im Kapitel 2.3 Fahrergruppen erstellen und bearbeiten).

Auf der rechten Seite befindet sich die Gruppenansicht. Bei dieser können Fahrer verschoben werden. Durch ein «Drag und Drop» [1] auf einen einzelnen Fahrer ist es möglich diesen einer neuen Gruppe zuzuweisen. Diese Funktionalität wird auch auf die ganze Gruppe angeboten, indem der Drag und Drop auf den Gruppenbeschreibungsbereich unten rechts gemacht wird (Abbildung 11: Rennen, Schattierung der ausgewählten Gruppe durch Drag und Drop).

Bei den Gruppen gibt es die Funktion, ihre Abstandszeit zur Spitzengruppe zu definieren. Dabei beschränkt sich die Auswahlmöglichkeit auf nur grössere Zeiten als die direkt davorliegende Gruppe. Durch diese Gegebenheit werden die nicht anwählbaren Zeiten ausgegraut dargestellt.

Abbildung 12: Zeiteinstellung der Gruppe

Im zweiten abgerundeten Rechteck unterhalb der auswählbaren Zeit wird die vorherige Zeit dargestellt, damit ein direkter Vergleich der Zeiten möglich ist. Das heisst, es ist ersichtlich, ob die Gruppe Zeit aufgeholt oder verloren hat.

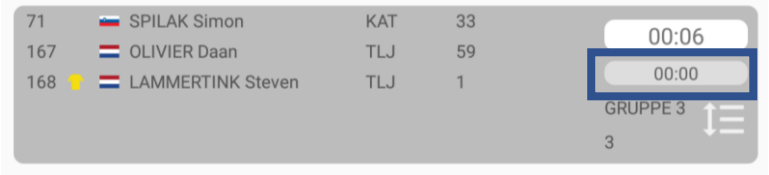
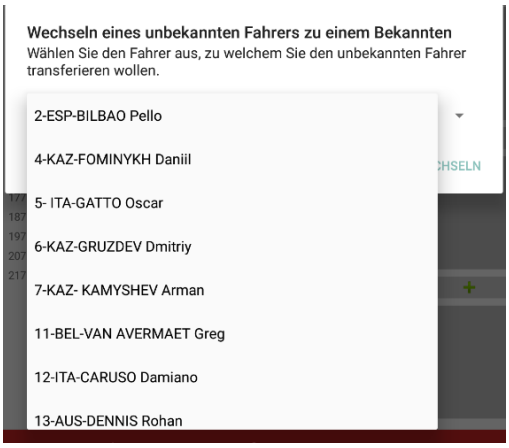


Abbildung 13: Zeiten der Fahrergruppen



Als letzte Einstellung in dieser Ansicht ist es möglich zuvor als unbekannte Fahrer erfasste Rennteilnehmer (einer Gruppe) einem spezifischen Fahrer zuzuweisen und damit zu löschen. Zu diesem Fenster gelangt man über einen Klick auf einen unbekannten Fahrer.

Abbildung 14: Unbekannte Fahrer zuweisen

2.3 Fahrergruppen erstellen und bearbeiten

Mittels Funksprüche gelangen Informationen zum RadioTour Speaker und damit zum Tablet. Die meisten Informationen resultierten in einer Eingabe auf diesem Screen.

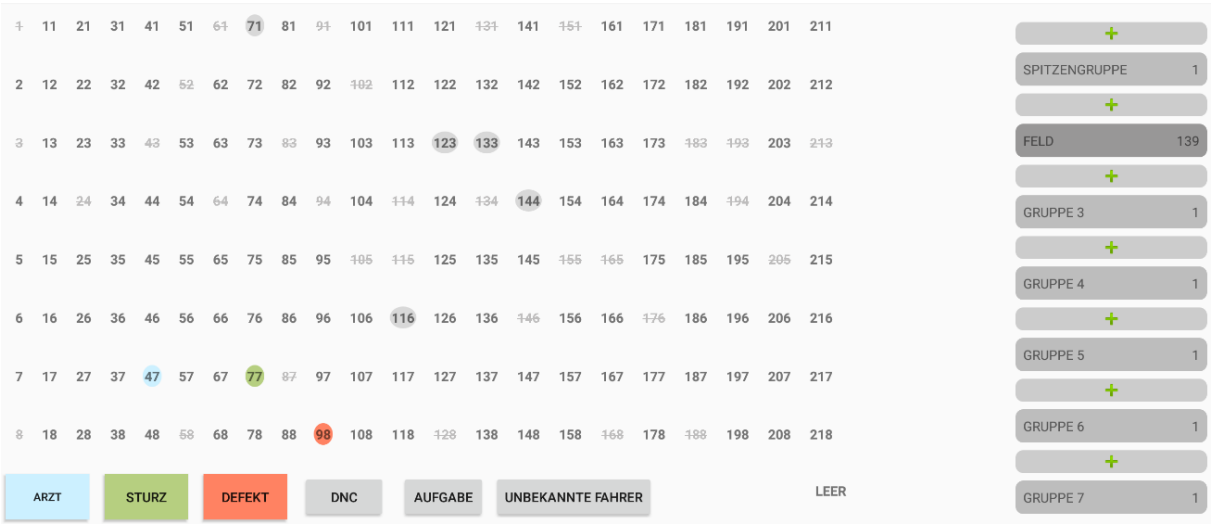
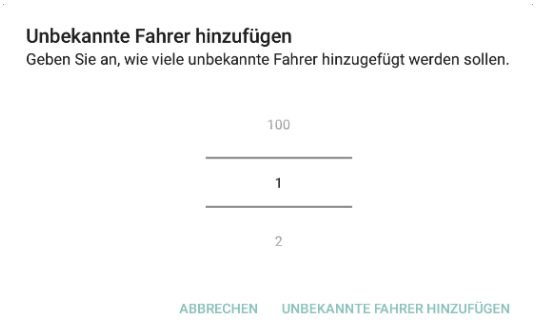


Abbildung 15: Fahrer auswählen

Diese Ansicht dient dazu einzelne Fahrer während eines Rennens zu verwalten. Bei einem Sturz, Defekt, DNC, Aufgabe oder Arzt ist es möglich diesen Status dem Fahrer zuzuweisen.

Es ist möglich Fahrer einer Gruppe zuzuweisen, indem diese selektiert werden und anschliessend einer Gruppe auf der rechten Seite zugewiesen werden. Mit den «+» Symbolen ist es möglich eine neue Gruppe zu erstellen.



Durch Klick auf den Button «unbekannte Fahrer» ist es möglich eine bestimmte Anzahl unbekannter Fahrer zu erfassen und diese anschliessend einer Gruppe zuzuweisen.

Abbildung 16: Unbekannte Fahrer erfassen

2.4 Wertungen

Für Fahrer gibt es zu bestimmten Zeitpunkten auf der Strecke Wertungen. Diese Wertungen sind in diesem Screen zuweisbar. Durch Selektion der Wertung erscheint auf der rechten Seite die Auswahlmöglichkeit mit den Punkteinformationen für die jeweilige Rangierung. Durch einen Klick auf den Positions-Button und anschliessender Auswahl der Fahrernummer wird diesem die Wertung zugewiesen.

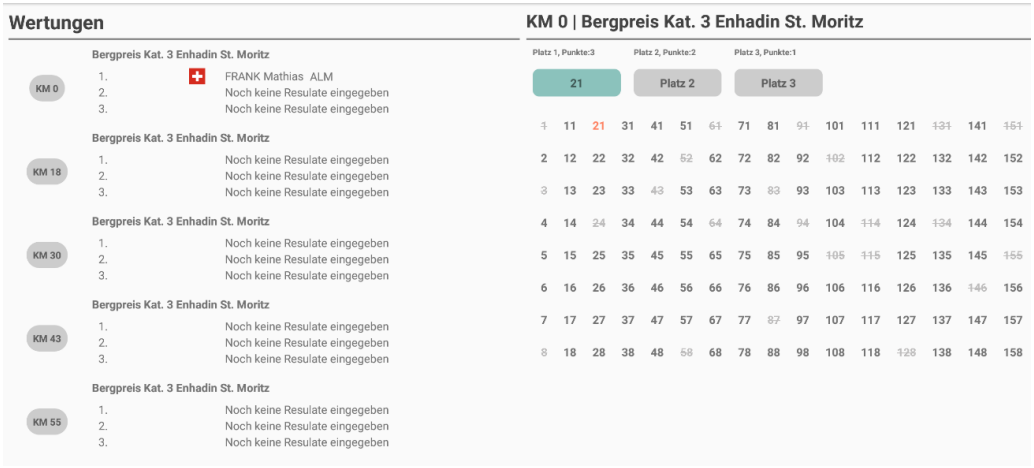


Abbildung 17: Wertungen zuweisen

2.5 Klassemente

In diesem Tab wird einer immer aktuelle Rangliste (Klassement) aller Fahrer angezeigt. Sortierung je Spalte ist möglich. Bei einem angewählten Fahrer wird jeweils die ganze Zeile eingefärbt.

Abbildung 18: Klassemente

2.6 Maillots

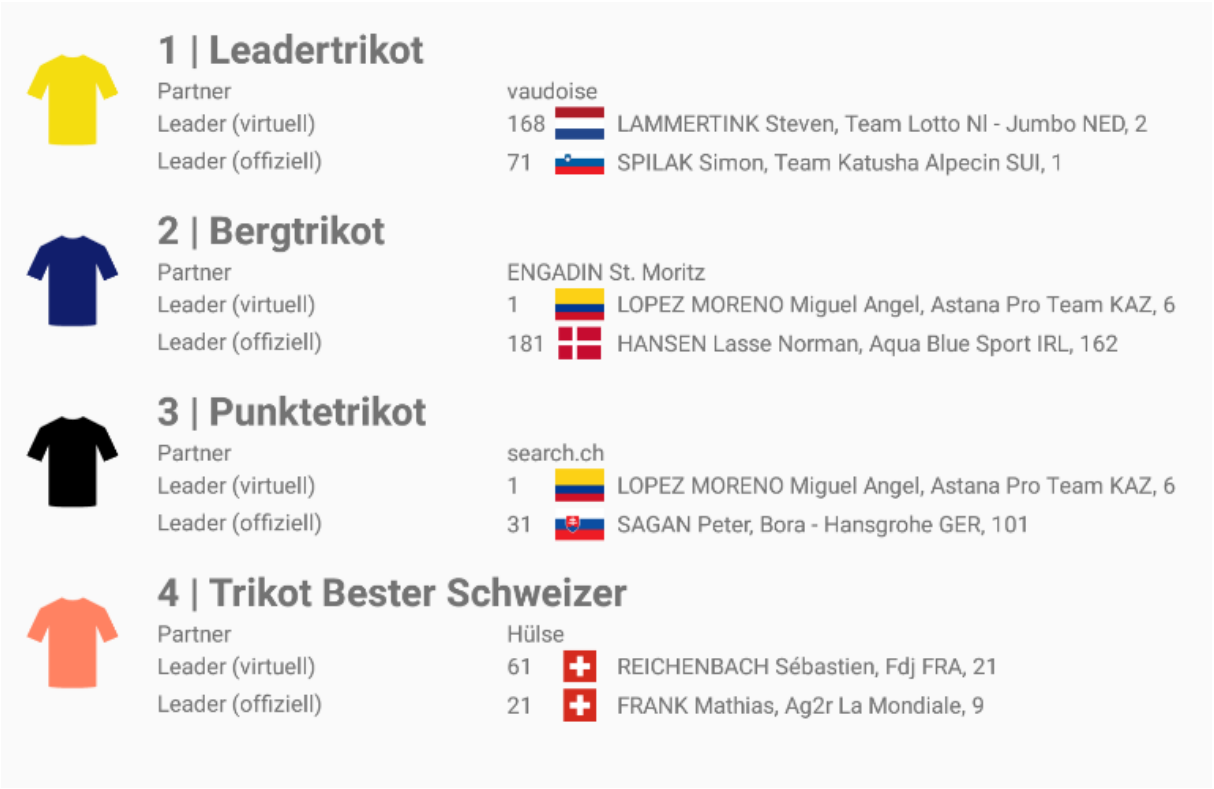


Abbildung 19: Maillots Übersicht

Diese Ansicht bietet einen Überblick über alle möglichen Trikots die während der Tour vergeben werden. Dabei wird zwischen den offiziellen Leadern (offizielles Klassement beim Start einer Etappe) und den virtuellen Leadern (berechnet aus Zeitabständen und Punkten während der Etappe) unterschieden.

2.7 Administration

In der Administration sind nebst einer globalen Übersicht zur Version und dem Rennen weitere Interaktionen möglich. So ist es möglich einen Demomodus zu starten, bei dem ausschliesslich lokale Daten verwendet werden. Dieser Demomodus ist auch ohne eine Internetverbindung bedienbar. Als Hauptfunktion bietet dieser Screen die Möglichkeit die aktuellen Daten aus der API zu importieren.

Abbildung 20: Administration

3 Requirements Engineering und Analyse

Dieses Kapitel widmet sich voll und ganz dem Requirements Engineering und der Analyse. Die bestehende Anwendung ist in die Jahre gekommen. Zudem sind neue Anforderungen entstanden. In diesem Zug sollen nun die gesamten Anforderungen überarbeitet werden. Um die Anwendung im sich immer ändernden Umfeld möglichst aktuell zu implementieren, wurde die Plattform sowie das dazu passende Betriebssystem neu evaluiert. Die Realisierung der Anwendung erforderte zudem verschiedene Technologien zur Speicherung der Daten und weiteren Anwendungsfällen. Für sämtliche Themen, wie Plattform, Betriebssystem, Datenbank, und Tablet wurden umfassende Analysen erstellt, Kandidaten untersucht und in den Vergleich zu den jeweiligen Mitbewerber-Technologien gestellt.

3.1 Funktionale Anforderungen (Use Cases)

Die funktionalen Anforderungen wurden auf Grund der bestehenden RadioTour App sowie weiteren Anforderungen seitens Herrn Heinzmann und Herrn Eichler ermittelt.

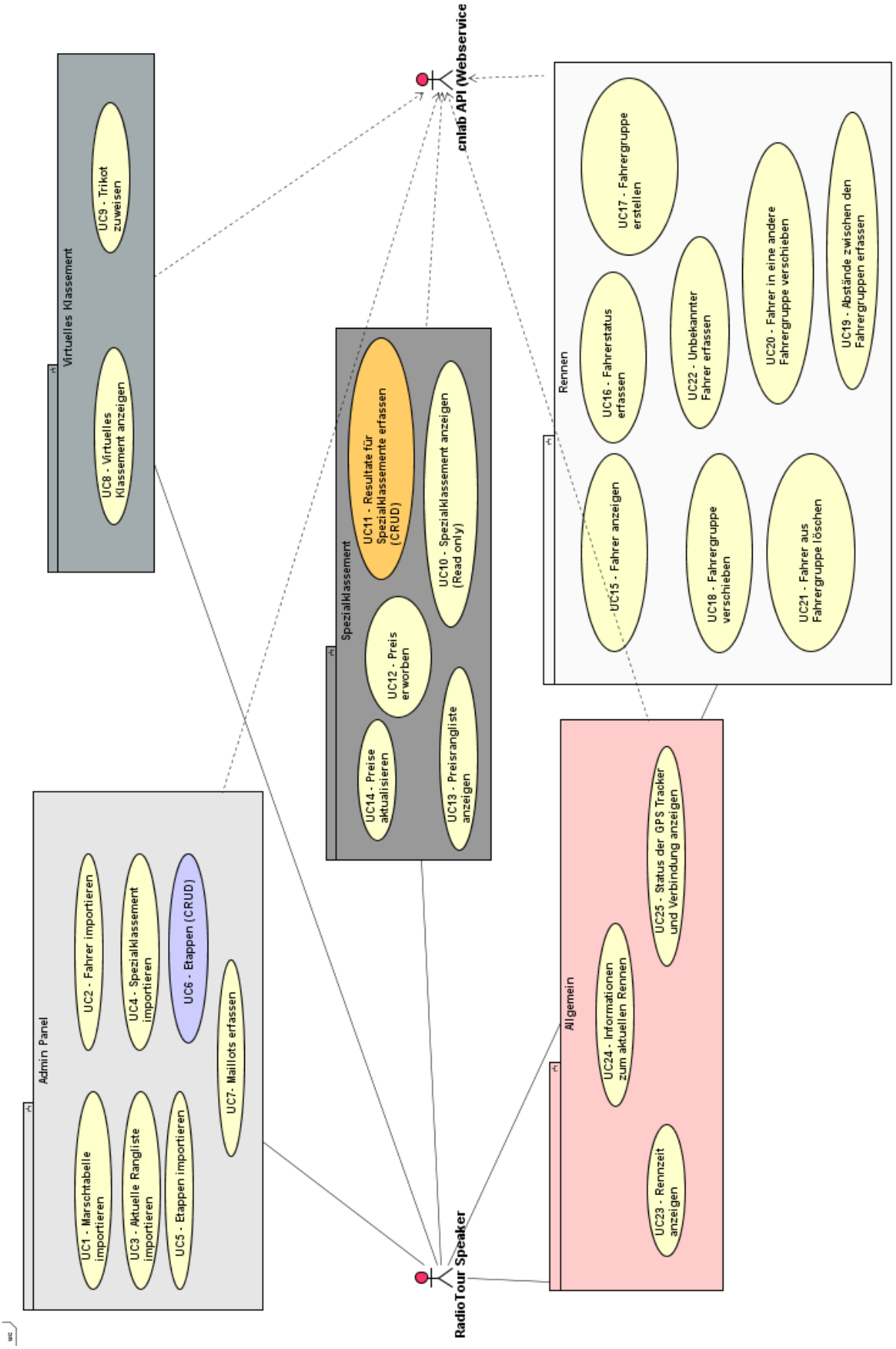


Abbildung 21: Use-Case Diagramm

Im abgegeben Dokumentenverzeichnis sind sämtliche Use Cases in Casual-Form detailliert aufgelistet und beschrieben. Eine Übersicht über dieses Verzeichnis ist im Kapitel 9 zu finden. Der folgende Abschnitt beschreibt die Anforderung in kurzer Form. Anforderungen mit einem * wurden innerhalb der Studienarbeit nicht implementiert. Den funktionalen Anforderungen wurden entsprechende Prioritäten zugewiesen (Text in der Klammer).

3.1.1 Admin Funktionen (Import)

- # Marchstabelle importieren (niedrig)
- # Fahrer importieren (hoch)
- # Aktuelle Ranglisten importieren (hoch)
- # Spezialklassemente importieren (hoch)
- # Etappen importieren (hoch)
- # Etappen CRUD* (optional)
- # Maillots importieren (hoch)

3.1.2 Virtuelles Klassement

- # Virtuelles Klassement anzeigen (hoch)
- # Trikots zuweisen (hoch)

3.1.3 Spezialklassement

- # Spezialklassemente anzeigen (hoch)
- # Resultate für Spezialklassemente erfassen (hoch)
- # Preise erwerben und aktualisieren (hoch)

3.1.4 Rennen

- # Fahrer anzeigen (hoch)
- # Fahrerstatus erfassen (hoch)
- # Fahrergruppe erstellen (hoch)
- # Fahrergruppe verschieben (hoch)
- # Zeitabstände zwischen den Fahrergruppen erfassen (hoch)
- # Fahrer in andere Fahrergruppen verschieben (hoch)
- # Fahrer aus einer Fahrergruppen löschen (hoch)
- # Unbekannte Fahrer erfassen (hoch)

3.1.5 Allgemein

- # Rennzeit anzeigen (mittel)
- # Informationen zum aktuellen Rennen anzeigen (hoch)
- # Status der GPS Tracker und Verbindungen anzeigen (mittel)

3.2 Nicht-funktionale Anforderungen (NFA)

Die RadioTour Anwendung wird während Radrennen wie der Tour de Suisse eingesetzt. Dort spielen die nicht funktionalen Anforderungen eine sehr grosse Rolle. Die Merkmale wurde nach ISO9126 definiert. Die relevantesten, nicht funktionalen Anforderungen für diese Anwendung sind folgende:

3.2.1 Reaktionszeit & Zeitverhalten

Die Reaktionszeiten der Anwendung sollen geringgehalten (weniger als 300ms) werden, damit Eingaben in kurzen Abständen möglich sind. Länger andauernde Tasks wie Aktualisierungen von Tabellen sollen in den Hintergrund ausgelagert werden, damit der RadioTour Speaker die Anwendung weiterhin bedienen kann.

3.2.2 Stabilität

Falls ein unerwartetes Problem mit einer Version der RadioTour App auftritt, muss es innerhalb von fünf Minuten möglich sein, auf eine alte und stabile Version zurückzugreifen.

3.2.3 Fehlertoleranz

Die Anwendung darf während dem Einsatz nicht abstürzen und muss daher möglichst fehlertolerant sein. Fehleingaben sollen wo möglich bereits im User Interface mit Validierungen abgefangen und unterbunden werden.

Das System muss auch im Offline-Betrieb funktionsfähig sein, da auf den Rennstrecken Mobilfunkempfang nicht immer gegeben ist.

3.2.4 Analysierbarkeit

Allfälliges Fehlverhalten in der Anwendung wird geloggt um Fehler und Ursachen auch nach einem Rennen noch nachvollziehen zu können. Mit den Nutzungsdaten kann zudem das Verhalten des Nutzers nachvollziehen.

3.2.5 User Experience

Die RadioTour Anwendung muss so konzipiert und aufgebaut sein, dass für den Benutzer eine erfolgreiche Bedienung jederzeit möglich ist. Allfällige Fortschritte für länger andauernde Tasks sollen im UI sichtbar sein. Der Anwender soll zu jederzeit wissen, welchen Status die Anwendung aufweist.

Ein gesamter Auszug der nicht funktionalen Anforderungen ist im abgegeben Dokumentenverzeichnis zu finden.

3.3 Evaluation | Programmiersprache / Technologie

Für die Weiterentwicklung der RadioTour Anwendung ist die Wahl der Programmiersprache bzw. der Zielplattform von grosser Bedeutung. Wie die bisherige Anwendung soll auch die neue Anwendung auf einem Tablet betrieben werden. Windows wird aufgrund der Verbreitung im Tablet Markt von Beginn weg ausgeschlossen. Für den Vergleich wurden die drei verbreitetsten Programmiersprachen / Technologien gewählt.

Legende

 Anforderung erfüllt

 Anforderung nicht erfüllt

 Performance in Ordnung

 Performance nicht in Ordnung

	iOS nativ	Android nativ	Xamarin
Programmiersprache	Objective C, Swift	Java, Kotlin	C#
Bestehende App			
Entwicklungsumgebung	XCode (nur macOS	Android Studio (macOS & Windows)	Visual Studio (macOS & Windows)
Vorhandene Kenntnisse der Studenten			
Native Programmierung			
Vorhandene Dokumentationen			
Möglichkeit für lokale Datenbanken			

Tabelle 1: Vergleich Programmiersprachen / Technologien

3.3.1 Entscheid

Auf Grund der Evaluation haben wir uns für Android als Zielplattform entschieden. Dabei am meisten ausschlaggebend waren unsere bereits vorhandenen Programmierkenntnissen (Java, C#, C++) sowie die Einwicklungsumgebung. Nur einer von uns beiden besitzt einen Mac was die Entwicklung erschwert hätte, da ein Mac Grundvoraussetzung für die Entwicklung mit iOS ist. Die Programmierung in Android werden wir nativ vornehmen.

3.4 Evaluation | Tablet

Das bisherige Tablet der RadioTour Anwendung wurde 2012 gebaut und ist daher sechs Jahre alt. Dies ist spürbar bei der Bedienung des Tablets. Das Weitern ist das Betriebssystem des Tablets aus dem Jahre 2012/2013 und lässt sich nicht weiter updaten. Daher wird ein neues Tablet evaluiert um den heutigen Anforderungen gerecht zu werden und um ein aktuelles App auf dem neusten Betriebssystem abliefern zu können. Die evaluierten Geräte basieren auf der zuvor evaluierten Programmiersprache. Tablets anderer Programmiersprachen werden nicht evaluiert.

3.4.1 Kriterienkatalog

Die Auswahl des Tablets beruht auf verschiedensten Kriterien. Die Kriterien wurden in diesem Kriterienkatalog zusammengefasst. Die Kriterien ergaben sich aus der bisherigen Nutzung (Radrennen) sowie aus funktionalen Anforderungen.

Zwingende Kriterien

- **Android** Betriebssystem, Version >= 6.0
- **Bildschirmgrösse** >= 10.1”
- **Auflösung** 1920 * 1200
- **Speicher** erweiterbar
- **GPS** vorhanden
- **Mobilfunknetz** vorhanden
- **USB** Schnittstelle vorhanden
- **Akkukapazität** mindestens 6 Stunden (Renndauer) *
- **Lesbarkeit** bei Sonnenlichteinstrahlung

* Die Akkukapazität ist eine Anforderung, da während eines Rennens nicht die Möglichkeit besteht das Gerät zu laden. Grundsätzlich wären 12V Buchsen in den Autos verfügbar. Die Buchsen sind durch anderweitige Verwendung (für Funk, GPS) allerdings blockiert und damit nicht verfügbar.

Optionale Kriterien

- Die Anzahl der Prozessoren sollte >= 4 sein (Damit flüssiges Arbeiten möglich ist)
- Preis < CHF 800.-
- Erscheinungsjahr (nicht älter als 2 Jahre)
- Gute Feedbacks/Erfahrungen von Usern

3.4.2 Tablet-Vergleich

Die Informationen der Kandidaten wurden von dem verschiedenen Hersteller zusammengetragen und einander gegenübergestellt. In einem weiteren Schritt wurden die Geräte nach der Erfüllung der Anforderungen geprüft. Informationen zur Erfüllung der verschiedenen Kriterien sind direkt in der Tabelle zu finden (siehe dazu Legende).

Legende

Farbe / Markierung	Beschreibung
Text	Anforderung erfüllt
Text	Anforderung nicht erfüllt

Tabelle 2: Legende Tablet-Vergleich

	Samsung Galaxy Tab A (10.1", 16GB, 4G, Metallic Black)	Huawei MediaPad T2 Pro (10.10", 16GB, 4G, Charcoal Black)	Lenovo Tab 4 (10.1", 16GB, 4G, Slate Black)
Zwingende Anforderungen			
Betriebssystem	6.0 Marshmallow	**5.0 Lollipop**	7.0 Nougat
Bildschirmgrösse	10.1”	10.1”	10.1”
Auflösung	1920 * 1200	1920 * 1200	**1280 * 800**
Speicher	16GB, erweiterbar	16GB, erweiterbar	16GB, erweiterbar
GPS	Ja	Ja	Ja
Mobilfunknetz	4G	4G	4G
USB	Ja	Ja	Ja
Akkukapazität	7300 mAh, 10.5h	6660 mAh, 13h	7000 mAh, 12h
Optionale Anforderungen			
Prozessoren	8	4	4
Veröffentlicht in	2016	2016	2017
Preis	249Fr. (Digitec)[2]	214Fr. (Digitec)[3]	279Fr. (Digitec)[4]

Tabelle 3: Tabletvergleich 1

	Panasonic Toughpad FZ-A2	**Samsung Galaxy Tab S3**	**Huawei Tablet MediaPad M3 Lite 10**
Zwingende Anforderungen			
Betriebssystem	6.0 Marshmallow	7.0 Nougat	7.0 Nougat
Bildschirmgrösse	10.1"	**9.7"**	10.1"
Auflösung	1920 * 1200, Anti Reflect Screen	2048 * 1536, Super AMOLED	1920* 1200
Speicher	32GB, erweiterbar	32GB, erweiterbar	32GB, erweiterbar
GPS	Ja	Ja	Ja
Mobilfunknetz	4G	4G	4G
USB	Ja	Ja	Ja
Akkukapazität	**2720 mAh, max. 9h**	6000 mAh, 12.5h	6500 mAh, 16h
Optionale Anforderungen			
Prozessoren	4	4	8
Veröffentlich in	2016	2017	2017
Preis	1709Fr. (Brack)[5]	649Fr. (Digitec)[6]	339Fr. (Brack)[7]

Tabelle 4: Tablet-Vergleich 2

	Huawei Tablet MediaPad M2
Zwingende Anforderungen	
Betriebssystem	**5.1.1 Lollipop**
Bildschirmgrösse	10.1"
Auflösung	1920 * 1200
Speicher	16GB, erweiterbar
GPS	Ja
Mobilfunknetz	4G
USB	Ja
Akkukapazität	6660 mAh, 6h Volllast
Optionale Anforderungen	
Prozessoren	8
Veröffentlich in	2016
Preis	359Fr. (Brack)[3]

Tabelle 5: Tablet-Vergleich 3

3.4.3 Statements zur Benutzung der Tablets unter Sonnenlichteinstrahlung

Für die Benutzung der Tablets bei Sonnenlichteinstrahlung gibt es keinen Wert, welcher der Hersteller in seinen Data Sheets führt. Daher verlassen wir uns in diesem Bereich auf Testberichte aus dem Internet.

Galaxy Tab A
Das Display ist geeignet für den Outdoor-Gebrauch. Jedoch kann direkter Sonneinfluss das Display auswaschen. Für häufige Verwendung im Sonnenlicht sollte ein Model mit AMOLED Screen gewählt werden, wie z.B. das Samsung Galaxy Tab S3.[8] ➔ **Fazit**: nicht geeignet

Huawei Media Pad M3 Lite 10
Um ein Lesen im Outdoor Bereich zu ermöglichen muss das Display auf maximale Helligkeit eingestellt werden. Allerdings erschwert das Hochglanzdisplay den Gebrauch des Tablets im Sonnenlicht über längere Zeit.[9] ➔ **Fazit**: geeignet

Samsung Galaxy Tab S3
Die Helligkeit ist ein gewinnbringender Punkt dieses Tablets. In diesem Fall ist es möglich sowohl im direkten Sonnenlicht zu arbeiten als auch im Dunkeln. Verschiedene Ansichtsperspektiven sowie die Helligkeit sind eindrucksvoll.[10]
➔ **Fazit**: sehr gut geeignet

Huawei Media Tab M2
Der Outdoor Gebrauch ist keine grosse Herausforderung für dieses Tablet. Allerdings können der automatische Sonnenlichtmodus sowie die hohe Helligkeit die direkte Sonneneinstrahlung nicht kompensieren. Dies auch auf Grund des Glanzdisplays.[11] ➔ **Fazit**: nicht geeignet

Panasonic Tablet Toughpad FZ-A2
Das Tablet kann ohne Handschuhe oder andere Hilfen im direkten Sonnenlicht, sowie im Regen ohne Problem genutzt werden.[12]
➔ **Fazit**: Perfekt für das Arbeiten unter Sonnenlichteinstrahlung

3.4.4 Entscheid

Auf Grund der oben genannten Kriterien und nach Absprache mit Herrn Heinzmann und Herrn Eichler haben wir uns für das **Huawei Tablet Media Pad M3 Lite 10** entschieden. Zusätzlich wurde die Option auf das **Samsung Galaxy Tab S3** offengehalten. Die aktive Nutzung mit Sonneneinstrahlung wird auf Grund von Erfahrungsberichten anderer Nutzer ausgewertet. Jedoch kann ein definitiver Test erst durchgeführt werden, sobald das Tablet vorliegt.

3.5 Technologie | Lokale Datenspeicherung

Während der Verwendung der RadioTour App müssen lokal zahlreiche Daten gespeichert werden (Fahrer, Fahrergruppen, usw.). Zudem müssen die Daten auf ein anderes Tablet übertragen werden können.

3.5.1 Verfügbare Technologien

Shared Preferences[13]
Shared Preferences ist eine Möglichkeit zur Datenspeicherung, welche bei Android bereits von Haus aus integriert ist. Gespeichert werden können nur Key-Value Sets. Die gesamten Shared Preferences Objekte werden im Hintergrund in einer einzigen Datei abgelegt. Über Getter und Setters kann auf die Daten zugegriffen werden.

SQLite[14]
SQLite ist ebenfalls als Bestandteil von Android bereits von Haus aus verfügbar. Es ist eine relationale Datenbank und lässt daher auch die Speicherung von Beziehungen auf eine einfache Art und Weise zu. Grundsätzlich kann man von einer abgespeckten SQL Datenbank sprechen. Tabelle und Objekte in der Datenbank müssen mit der SQL-Syntax verwaltet werden. Durch OR-Mapper ist es möglich, dass die Objekte in Java definiert werden können. Eine Umwandlung auf das DB-Schema nimmt dann der OR-Mapper (ORMLite, SugarORM, usw.) vor.

Realm.io Mobile Database[15]
Realm.io Mobile Database ist neuartige Speicherung der Daten. Die Library ist in C++ sehr systemnah geschrieben und damit schnell. Die Definition der Objekte bzw. des Schemas findet durch Java-Klassen statt, welche mit entsprechenden Annotationen versehen sind. Die Datenbanken können mit Realm Object Server synchronisiert werden und stehen somit potentiell mehreren Benutzern zur Verfügung. Die Verwendung von Realm.io kann Plattform übergreifen stattfinden. Die Syntax ist je nach Programmiersprache anders, die Datenbank ist allerdings bei den gängigsten Plattformen verwendbar.

Vergleich der Technologien

Legende

 Anforderung erfüllt

 Anforderung nicht erfüllt

 Gute Performance

 Mässige Performance

 Unzureichende Performance

	Shared Preferences 	**SQLite** 	**Realm.io** 
Art der Datenspeicherung	Key-Value	Relationale Datenbank	Relationale Datenbank
Beziehungen			
Hauptbestandteil von Android			
Schnelligkeit			
Deklaration über Objekte		 (mit OR Mapper)	
Dokumentation			
Cross-Plattform fähig			
Verfügbar seit	unbekannt	2000	2010
Verschlüsselung			
Reaktive Programmierung			
Einfachheit der Benutzung			

Tabelle 6: Vergleich | Lokale Datenspeicherung

3.5.2 Entscheid der Technologie

Bei der lokalen Datenspeicherung werden wir die Realm.io Mobile Database einsetzen. Ausschlaggebend sind dafür Performance, die objektnahe Programmierung sowie die reaktive Programmierung. Realm ist zwar nicht Bestandteil von Android, kann aber als Plugin nachinstalliert werden. Die Tatsache, dass viele der Top 500 Fortune Unternehmen Realm brauchen, spricht für unsere Entscheidung.

Shared Preferences sind auf Grund der grossen Datenmengen nicht geeignet. Zudem können mit Key-Values Stores nur schlecht Beziehungen zwischen Objekten abgebildet werden (hohes Risiko bei der Umsetzung).

Eine Umsetzung mit einer SQLite Datenbank wäre auch möglich. Bei der Benützung eines OR-Mappers ist SQLite ähnlich zu verwenden wie der Zugriff auf eine Realm-Datenbank. Der entsprechende Overhead der Architektur macht sich aber in der Performance (Schnelligkeit) bemerkbar. Zudem wäre ein Cross-Plattform-Ausbau nicht gegeben.

4 Software Engineering

4.1 Software Architektur

4.1.1 Die Anwendung und deren Umsysteme

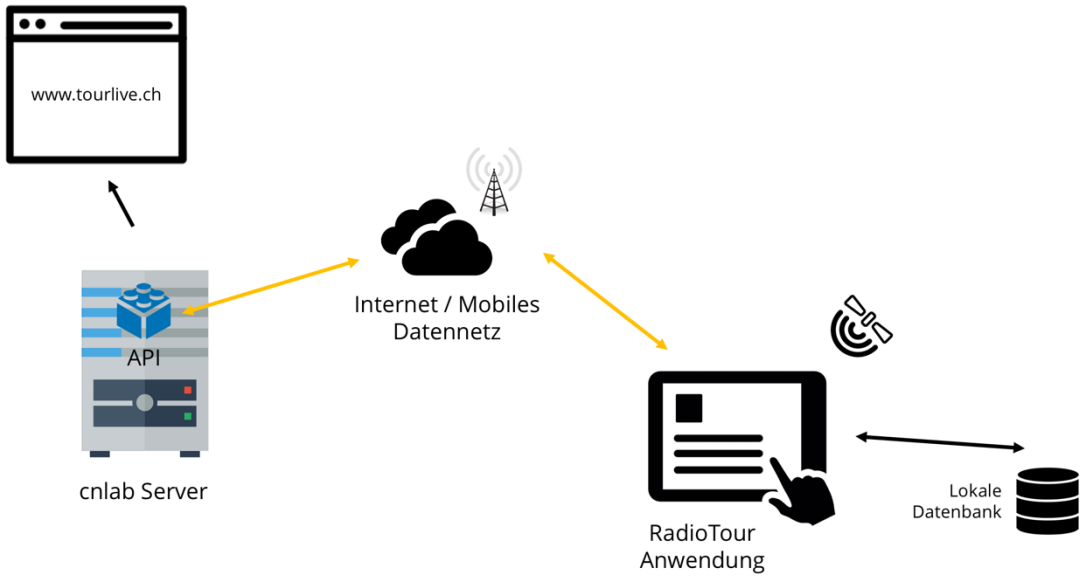


Abbildung 22: Anwendung und Umsysteme

Die RadioTour Anwendung (Android) ist eine nativ programmierte Anwendung auf einem Tablet. Die Anwendung speichert ihre Daten persistent in einer lokalen Datenbank. Über eine von der Firma «cnlab» zur Verfügung gestellte Web API findet der Datenaustausch zwischen dem Tablet und dem cnlab Server statt.

Die Daten der API werden zudem zur Darstellung gewisser Informationen auf der Webseite www.tourlive.ch verwendet. Der Umfang der Software Architektur bezieht sich nur auf die Tablet Anwendung.

4.1.2 Schichtendiagramm

Die RadioTour Anwendung wurde mit einer vier Schichtenarchitektur umgesetzt.

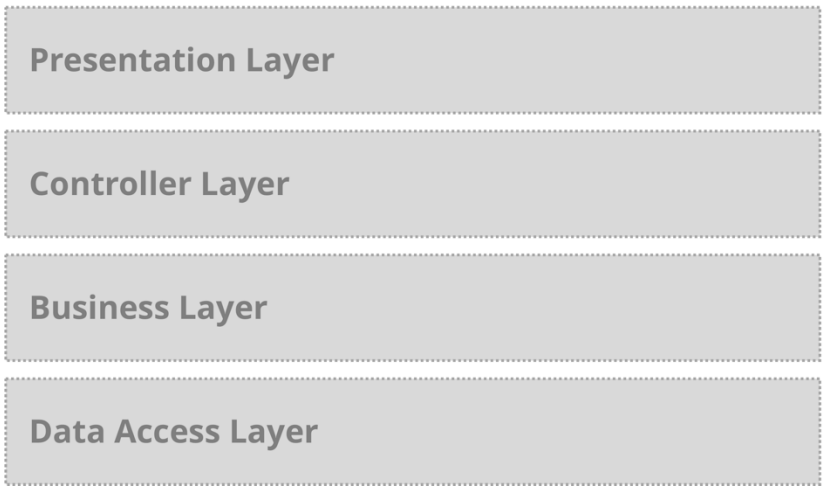


Abbildung 23: Schichtendiagramm

Presentation Layer
Der Presentation Layer ist für die Repräsentation der Daten und die Benutzereingaben verantwortlich. Des Weiteren dient sie als Schnittstelle zum Benutzer.

Controller Layer
Die Klassen innerhalb des Controller Layer sind für die Aufbereitung der Daten für den Presentation Layer verantwortlich und decken die Funktionalität zur API ab.

Business Layer
Der Business Layer enthält die für das MVP Pattern benötigten Presenter und dient allgemein zur Verarbeitung und Vorbereitung der Daten für den unteren Layer (zum Beispiel das Parsen der Daten).

Data Access Layer
Dieser Layer enthält die Datenbank und ist verantwortlich für die Speicherung und das Laden der Daten aus der Datenbank. Daher enthält dieser Layer ebenfalls die zugehörigen Models.

4.1.3 Packages

Die RadioTour Anwendung besteht aus den folgenden Packages und Klassen.

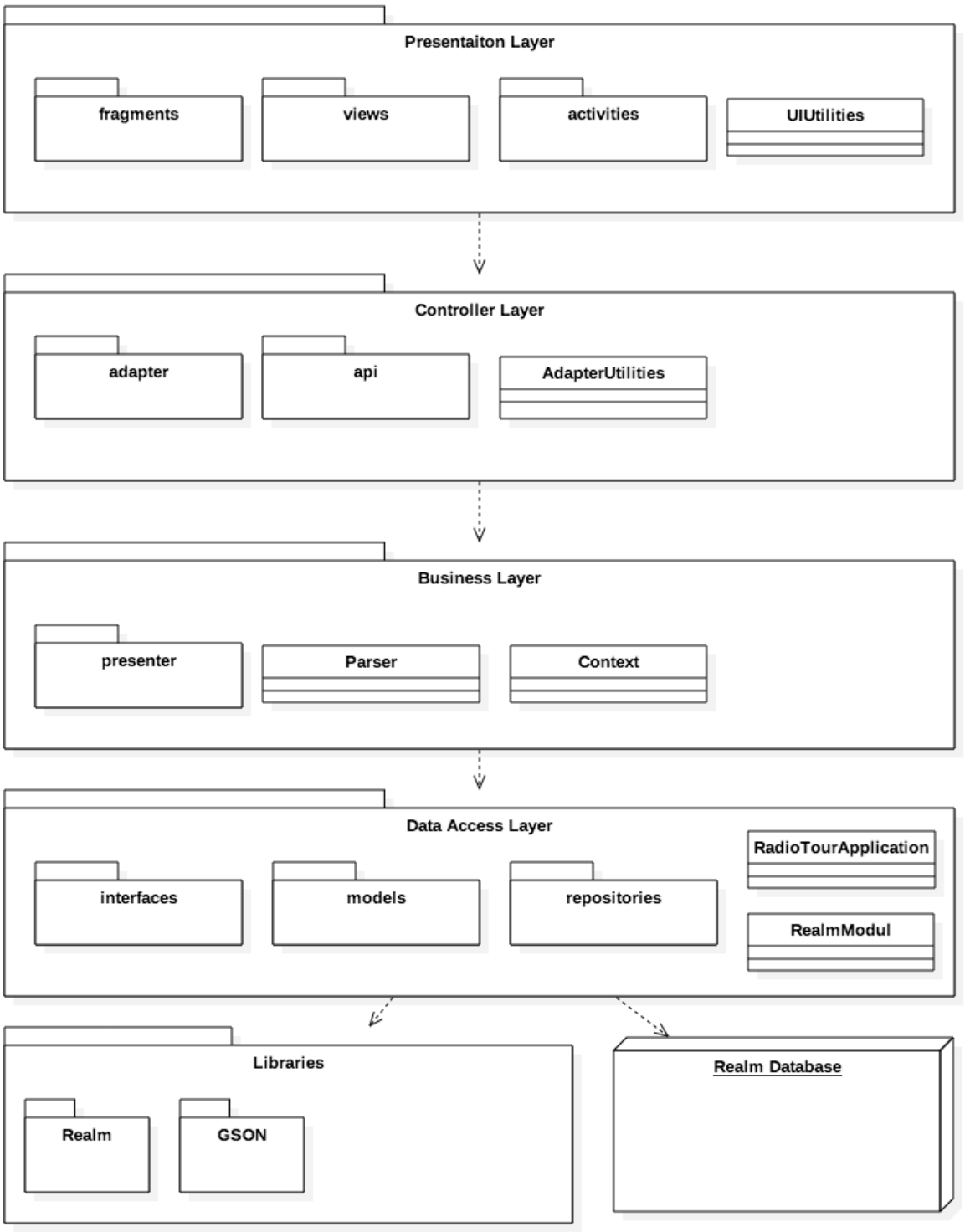


Abbildung 24: Packages und die wichtigsten Klassen

Die Tabelle 7: Packages und Klassen zeigt den Nutzen / Zweck jeder Instanz auf.

Package/Klasse	Beschreibung
Package „Fragments“	Fragments der Anwendung
Package „Views“	Eigene Views zur Darstellung der Daten innerhalb der App
Package „Activities“	Activities der RadioTour Anwendung. Die Anwendung besteht aus zwei Activities (MainActivity für alle Screens und die SplashActivity für den Start der App).
Klasse „UIUtilities“	Häufig verwendete Hilfsmethoden im UI
Package „Adapter	Adapter als Halter der Daten im UI
Package „API“	Verantwortlich für die Kommunikation zur Kommunikation der Web API (GET und POST)
Klasse „AdapterUtilities“	Häufig verwendete Hilfsmethoden in den Adaptern
Package „Presenter“	Presenter nach dem MVP-Pattern (Bindeglied zwischen den Views und den Models)
Klasse „Parser“	Konvertierung der Daten der API in das Format der lokalen Datenbank
Klasse „Context“	Context-Klasse für den Parser
Package „Interfaces“	Interfaces für die Respositories
Package „Models“	Das gesamte Datenmodell der RadioTour Anwendung. Siehe Kapitel 4.3.
Package „Repositories“	Alle Repositories, welche die Daten für den Presenter zur Verfügung stellen.
Klasse „RadioTourApplication“	Enthält die Realm-Datenbank Instanz
Klasse „RealmModul“	Enthält die verschiedenen Realm – Module

Tabelle 7: Packages und Klassen

4.1.4 Ablauf einer Eingabe auf dem Tablet

Folgendes Sequenzdiagramm zeigt den Ablauf und das Zusammenspiel der verschiedenen Packages sowie externen Schnittstellen. Diese Ablaufreihenfolge gilt für alle Interaktionen mit dem Tablet. Ein Bespiel einer Eingabe könnte ein Abstand einer Renngruppe sein. Diese Interaktion wird wie jede andere Aktion alle Schichten durchlaufen.

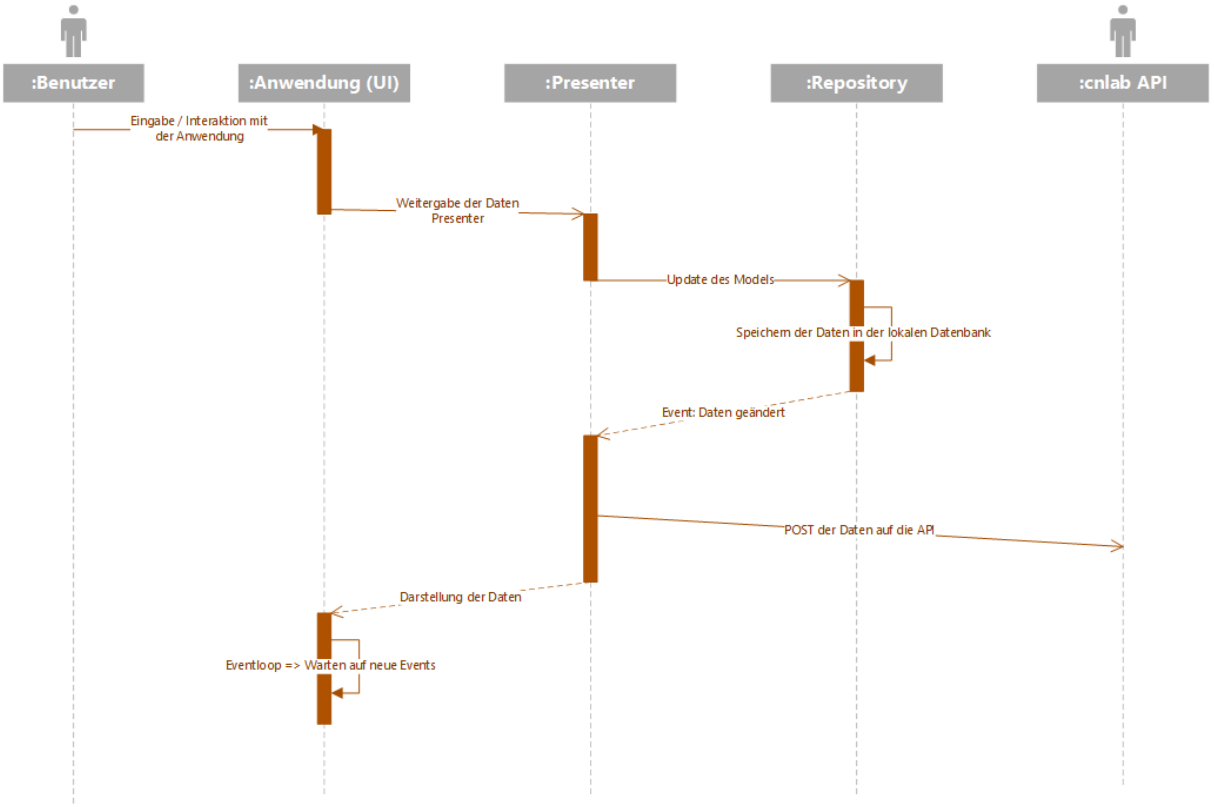


Abbildung 25: Sequenzdiagramm "Interaktion mit der Anwendung"

Der POST (Hochladen der Information zur API) wurde im Rahmen der Studienarbeit nicht implementiert. Die Funktion ist im lokalen API Client soweit vorbereitet.

4.2 Domainmodel

Die Domäne rund um die RadioTour Anwendung besteht aus den folgenden Entitäten.

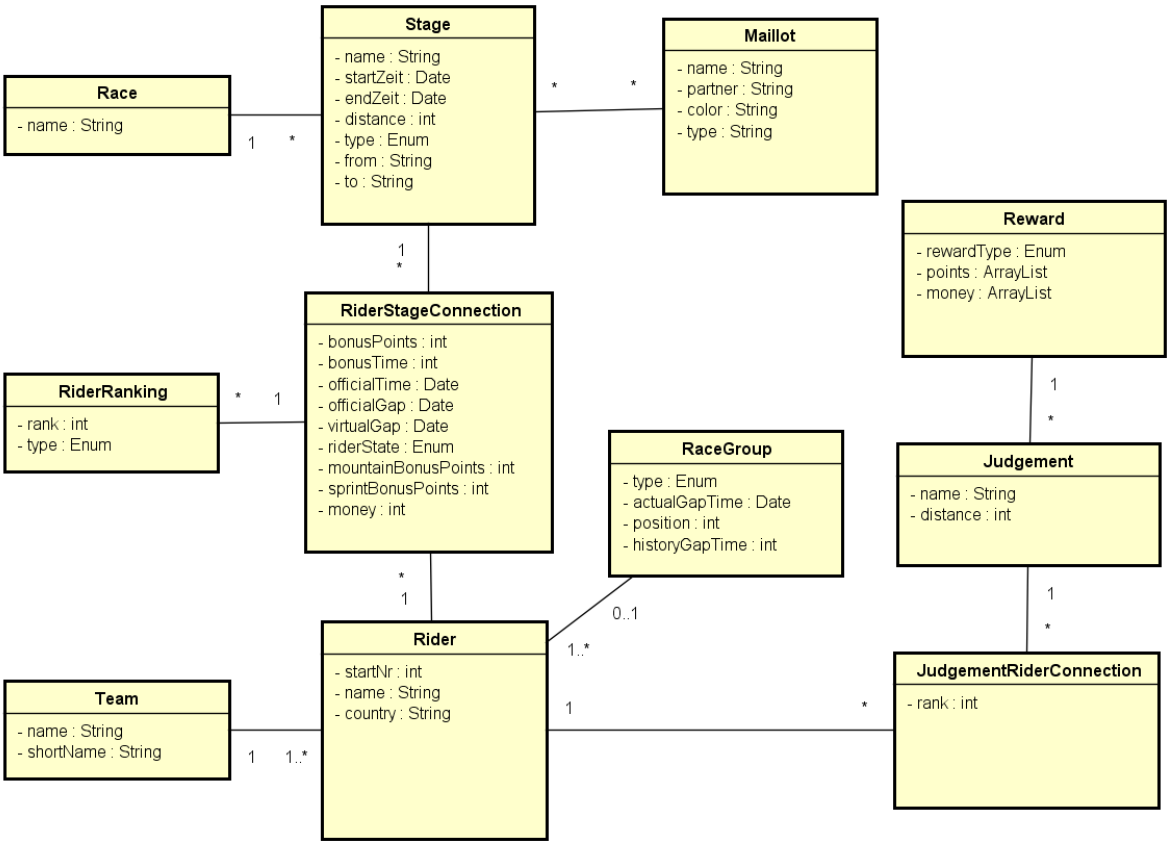


Abbildung 26: Domainmodell RadioTour

Die Entitäten des Domäne «RadioTour» im Detail.

Entitäten	Beschreibung
Race (Rennen /Tour)	Ein Race entspricht einem Rennevent (z.B. Tour de Suisse). Es weist nur einen Namen auf und dient zur Gruppierung der Etappen (Stages).
Stage (Etappe)	Eine Stage repräsentiert eine Etappe und ist an ein Rennen (Race) gebunden.

Maillot (Trikot)	Das Maillot wiederspielt ein Trikot, welches die Fahrer vor und während dem Rennen erreichen können. Ein Maillot kann in jeder Etappe einem anderen Fahrer zugewiesen sein und ist damit an die Stage (Etappe) gebunden.
RiderStageConnection	Die RiderStageConnection bildet das Bindeglied zwischen einem Fahrer und einer Etappe. Sie repräsentiert den Stand eines Fahrers während einer Etappe.
Rider (Fahrer)	Ein Rider repräsentiert einen Fahrer während eines Radrennens.
Team (Mannschaft)	Das Team (bestehend aus einem kurzen und langen Namen) repräsentiert alle Fahrer, welche für die gleiche Organisation unterwegs sind.
RaceGroup (Gruppe)	Die RaceGroup bildet die verschiedenen Fahrergruppen ab, welche während eines Radrennens entstehen. Unterschieden wird zwischen Feld, Spitzengruppe und normalen Gruppen.
Judgement	Ein Judgement ist eine Wertung, welche an bestimmen Positionen während eines Rennens aufgenommen werden.
Reward	Der Reward umschreibt die Punkte- und Zeitgutschriften, welche bei einer jeweiligen Wertung gutgeschrieben werden.
JudgementRiderConnection	JudgementRiderConnection ist eine Entität, welche die Beziehung eines Rider und eines Judgements (Wertung) genauer spezifiziert.
RiderRanking	Ein RiderRanking ist eine Rangierung innerhalb einer bestimmten Gesamtwertung (z.B. Sprint, Berg, ...)

Tabelle 8: Entitäten Domainmodell

4.3 Datenmodell

Das Datenmodell zeigt die Struktur, wie die Daten in unserer lokalen Datenbank (Realm) gespeichert werden. Das gesamte Datenmodell wurde so ausgelegt, dass in der Datenbank gleichzeitig mehrere Etappen und Rennen gespeichert werden können. Die aktuelle Implementierung der Anwendung erfordert dies nicht. Somit sind Schritte für einen Ausbau geebnet und erfordern grundsätzlich keine Anpassung des Datenmodells.

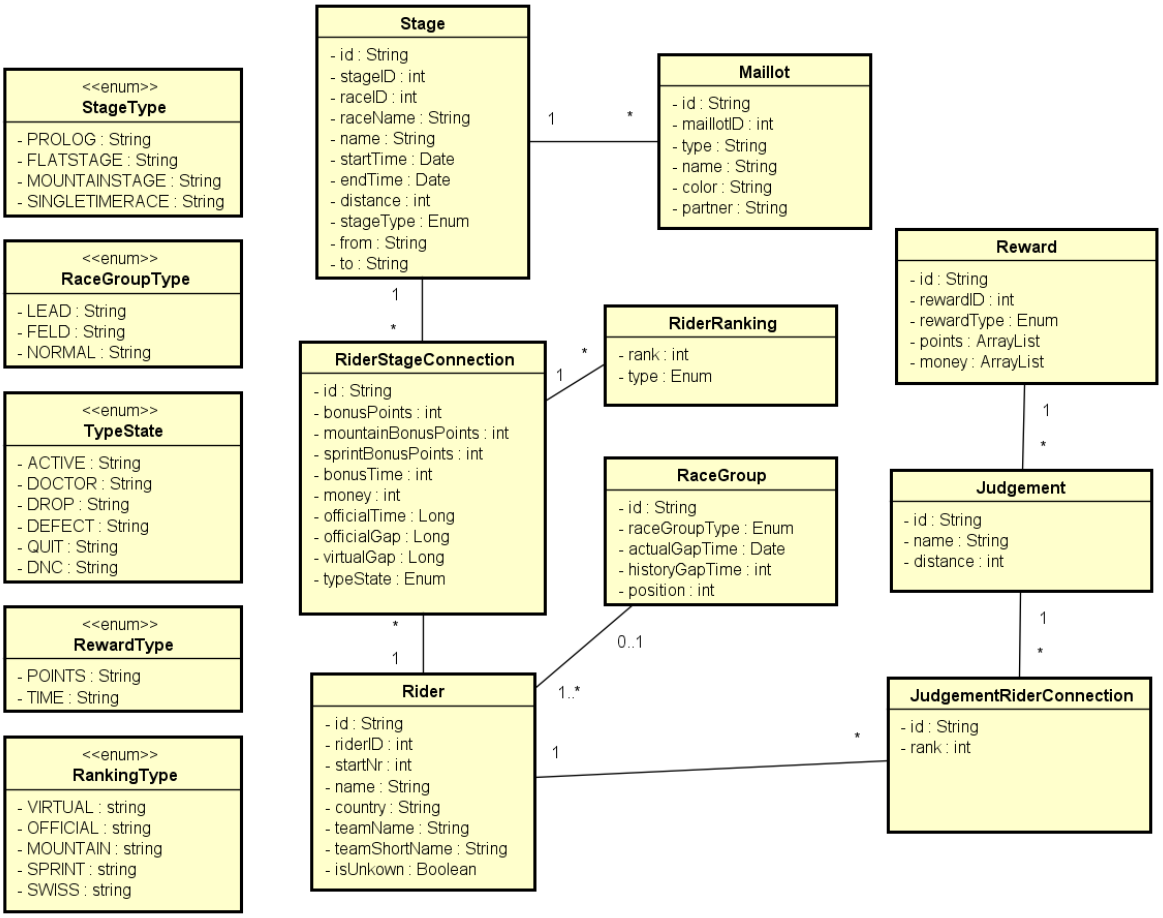


Abbildung 27: Datenmodell RadioTour

4.4 MVP Pattern [16]

Um die Interaktion zwischen dem Benutzer und den vorhandenen Daten zu gewährleisten, ist es notwendig ein automatischer Aktualisierungsmechanismus in die Applikation einzubauen.

Für die Umsetzung dieses Mechanismus haben wir uns dazu entschieden das «Model-View-Presenter» Konzept in unsere Applikation einzubinden.

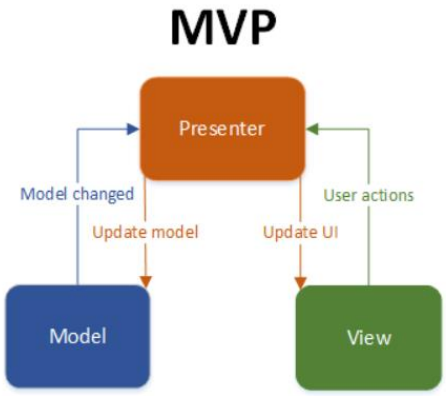


Abbildung 28: MVP Pattern

Grundlegend werden in diesem Pattern alle Änderungen am Datenmodel an die entsprechenden Anzeigen propagiert. Um dies zu gewährleisten wird eine Schnittstelle benötigt, welche die ganze Kommunikation verwaltet. Diese Schnittstelle wird als «Presenter» bezeichnet. Über den Presenter werden die Daten in der Datenbank verändert. Sobald eine Veränderung der Daten erfolgt wird mittels eines Callbacks [17] die erfolgreiche Manipulation an den Presenter propagiert. Der Presenter ist jetzt in der Lage die Anzeigeelemente zu aktualisieren und die neuen Daten darzustellen. Dies funktioniert auch in die entgegengesetzte Richtung, das heisst falls ein Anzeigeelement interaktionsfähig ist und dieses verwendet wird erhält der Presenter eine Meldung darüber. Auf Grund von dieser Meldung führt er die entsprechenden Aktionen auf dem Model in der Datenbank aus.

5 Realisierung

5.1 Realm

5.1.1 Datenbankklasse (Objekt)

Um die Objekte in der Realm Datenbank abzubilden, wurde das entsprechende Datenmodell genommen und im Code abgebildet. Um eine Klasse als Datenobjekt zu definieren, wird vom Typ «RealmObject» [18] geerbt. Dies erlaubt weitere Syntax-Annotationen um die Datenobjekte zu verwalten. So dient die «@PrimaryKey» Annotation z.B. dazu eine eindeutige, einzigartige ID für das Datenobjekt festzulegen. Mit der Annotation «@LinkingObjects» können Beziehungen zwischen Datenobjekten abgebildet werden.

```
public class Maillot extends RealmObject {
    @LinkingObjects("maillotConnections")
    private final RealmResults<Stage> stages = null;
    @PrimaryKey
    private String id;
    @Required
    private String type;

    public String getId() {
        return id;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

Codestück 1: Datenbankklasse "Maillot"

5.1.2 Repositories

Die Repositories dienen dazu die Datenobjekte in der Realm Datenbank zu verwalten. Dazu wird eine Realm-Instanz beim Applikationsstart definiert. Auf diese Instanz ist es während der Laufzeit möglich zuzugreifen. Realm basiert auf «Managed-Objects» [19]. Diese erlauben es eine direkte Manipulation in der Datenbank auf dem Objekt vorzunehmen. So ist es möglich, wie in diesem Beispiel, ein Objekt direkt auf der Instanz zu erzeugen und zu verwalten. Im untenstehenden Codestück wird ein neues Maillot in der Datenbank erfasst und persistiert.

```
@Override
   public void addMaillot(Maillot maillot, OnSaveMaillotCallback callback) {
      Realm realm = Realm.getInstance(RadioTourApplication.getInstance());

      realm.executeTransaction((Realm db) -> {
         Maillot realmMaillot = db.createObject(Maillot.class, UUID.rando-
mUUID().toString());
         realmMaillot.setColor(maillot.getColor());
         realmMaillot.setDbIDd(maillot.getDbIDd());
         realmMaillot.setName(maillot.getName());
         realmMaillot.setPartner(maillot.getPartner());
         realmMaillot.setType(maillot.getType());
      });

      if (callback != null)
         callback.onSuccess();
   }
```

Codestück 2: Maillot Repository | Add-Methode

5.2 MVP Pattern Implementation

Die Implementation des MVP Patterns wird anhand der Maillot Implementation aufgezeigt. Alle Presenter besitzen ein eigenes Interface von welchem geerbt wird. Dies bietet den Vorteil, dass ein gemeinsames Grundgerüst für alle Presenter besteht und die Verfügbaren Methoden, falls nötig, nach aussen propagiert werden können.

In einem ersten Schritt wird eine Methode zur Verfügung gestellt, bei denen die Möglichkeit besteht, dass sich interessierte Anzeigeelemente registrieren können.

```
public void addView(Fragment frag) {
      this.fragments.add(frag);
}
```

Codestück 3: Maillot Presenter | Registrierung für Anzeigelemente

In einem zweiten Schritt werden die Callback Methoden definiert. Wird ein solch spezifisches Callback (in diesem Beispiel «OnGetAllMaillotsCallback») ausgelöst, erfolgt der Methodenaufruf von «onSuccess». Bei diesem werden auf den registrierten Anzeigeelementen je nach Typ eine definierte Methode ausgelöst. In unserem Beispiel wird das «MaillotFragment» aktualisiert, in dem die Methode «showMaillots» zu finden ist.

```
@Override
public void subscribeCallbacks() {
    onGetAllMaillotsCallback = new IMaillotRepository.OnGetAllMaillotsCall-
back() {
        @Override
        public void onSuccess(RealmList<Maillot> maillots) {
            for (Fragment frag : fragments) {
                if (frag instanceof MaillotsFragment) {
                    MaillotsFragment maillotsFragment = (MaillotsFragment) frag;
                    maillotsFragment.showMaillots(maillots);
                }
            }
        }
    };
}
```

Codestück 4: Maillots | Implementierung der Callbacks

Um ein solches Callback überhaupt auszulösen muss der Methode dieses aktiv mitgegeben werden. In diesem Beispiel wird eine Abfrage auf die Datenbank gestartet, welche alle vorhandenen Maillots zurückgibt.

```
@Override
   public void getAllMaillots() {
      maillotRepository.getAllMaillots(onGetAllMaillotsCallback);
   }
```

Codestück 5: Maillot Presenter | Aufruf einer Datenbankabfrage

In unserem Repository, welches den Zugriff auf die Datenbank verwaltet, werden alle benötigten Informationen gesammelt. Wurde ein Callback beim Aufruf der Methode mitgegeben, wird dieses nun ausgelöst.

```
@Override
   public void getAllMaillots(OnGetAllMaillotsCallback callback) {
      Realm realm = Realm.getInstance(RadioTourApplication.getInstance());
      RealmResults<Maillot> results = realm.where(Maillot.class).findAll();
      RealmList<Maillot> res = new RealmList<>();
      res.addAll(results);

      if (callback != null) {
         callback.onSuccess(res);
      }
   }
```

Codestück 6: Maillot Repository | Maillots über Callback zurückgeben

In unserem Fragment wird die Methode «showMaillots» durch das Callback wie oben beschrieben ausgelöst. Dieses veranlasst das Anzeigeelement sich zu aktualisieren und neu zu zeichnen.

```
public void showMaillots(RealmList<Maillot> maillotRealmList) {
      this.maillots.clear();
      this.maillots.addAll(maillotRealmList);
      rvMaillots.swapAdapter(new MaillotsAdapter(maillots, mContext), true);
      rvMaillots.scrollBy(0, 0);
      this.adapter.notifyDataSetChanged();
}
```

Codestück 7: Maillots Fragment | Anzeige aktualisieren

5.3 Kommunikation zum Tour Live Server

Um eine Verbindung zur API aufzubauen wird ein «SyncHttpClient» [20] verwendet. Dieser bietet die Methode «get» bei der ein ResponseHandler als Parameter mitgegeben wird.

```
private static SyncHttpClient client = new SyncHttpClient();

private static void get(String url, RequestParams params, AsyncHttpResponse-
Handler responseHandler) {
    responseHandler.setUseSynchronousMode(true);
    client.get(getAbsoluteUrl(url), params, responseHandler);
}
```

Codestück 8: SyncHTTPClient

Die Kommunikation zum Tour Live Server wird über die Klasse «JsonHttpResponseHandler» [21] realisiert. Dabei wird das empfangene Resultat in unser spezifischen Datenobjekte im «Parser» [22] umgewandelt. Im Fehlerfall wird der auslösende Error aufgefangen und die Fehlermeldung an das UI [23] weitergeleitet. Hier wird eine dieser Anfragen anhand der Maillots als Beispiel dargestellt.

```
public static String getMaillots(String url, RequestParams params) {
    final String[] messages = {"success"};
    APIClient.get(url, null, new JsonHttpResponseHandler() {

        @Override
        public void onSuccess(int statusCode, Header[] headers, JSONObject
data) {
            // Not needed and therefore not implemented
        }

        @Override
        public void onSuccess(int statusCode, Header[] headers, JSONArray mail-
lots) {
            try
                Parser.parseMaillotsAndPersist(maillots);
                messages[0] = "success";
            } catch (Exception ex){
                messages[0] = ex.getMessage();
            }

        }
```

```
        @Override
        public void onFailure(int error, Header[] headers, Throwable throwable,
JSONObject riders){
            if(throwable.getMessage().equals(throwableType)){
                messages[0] = readTimeOutMessage + throwable.getMessage();
            } else {
                messages[0] = throwable.getMessage();
            }
        }
    });
    return messages[0];
}
```

Codestück 9: Konkrette Implementation eines API Aufrufs

6 Qualitätsmanagement

Um die Langlebigkeit und Stabilität der Applikation zu gewährleisten, wurden diverse Qualitätssicherungsmassnahmen getroffen. Dazu gehören neben funktionalen Tests auch Benutzerfreundlichkeitstests und Code Qualität Tests. Unter der Langlebigkeit ist einerseits das Nutzen über mehrere Jahren gemeint und andererseits das die Anwendung auch durch andere Personen weiterentwickelt werden kann (Verständlichkeit).

6.1 Unit Tests

Unit Tests dienen dazu die Funktionalität der Applikation zu gewährleisten. Diese prüfen Methoden auf ihre Korrektheit, sowie das richtige Verhalten im Fehlerfall. In unserer Anwendung sind Tests auf der Datenzugriffsebene durchgeführt worden und wir sind somit in der Lage zu gewährleisten, dass Schreib- und Leseoperationen auf unsere lokale Realm Datenbank funktionstüchtig sind. Um unsere Test zu realisieren ist es notwendig gewesen «Instrumented Tests» [24] einzusetzen, da wir auf den Kontext der Applikation während des Tests angewiesen sind.

6.2 Usability Tests (User Tests)

Benutzerfreundlichkeit ist ein ausschlaggebender Punkt in der Anwendungsentwicklung. Daher wurden regelmässig Tests mit dem Stakeholder durchgeführt. Durch das kontinuierliche Feedback ist es uns gelungen eine auf den Hauptnutzer optimierte Applikation zu entwerfen.

6.3 Code Style Guide

Für den Java Code wurde der Code Style Guide [25] von Google für Java verwendet.

Wenn immer möglich wurde der Styleguide in der Entwicklungsumgebung von jedem Teilnehmer aktiviert. Damit unterstützt der Styleguide das Schreiben von korrektem, wartbarem und schön aufgebautem Code.

Ausserdem wurden noch weitere Policies für Error Handling, Logging und Assertions definiert- Ausführlicheres ist in den entsprechenden Dokumenten zu finden, welche separaten zur Arbeit abgeben wurden. Eine Übersicht über alle Inhalte ist im Kapitel 9 Dokumentenverzeichnis zu finden.

6.4 Statische Code Analyse

Die Qualität des Codes wird kontinuierlich bei jedem Release von Sonarqube [26] überwacht. Sonarqube gibt dabei Aufschluss über Software Metriken, die Code Qualität und wie die anstehenden Probleme zu beheben sind.

Im Rahmen unseres Projektes wurde zum Teil bewusst auf eine Korrektur der «Code Smells[4]» verzichtet, da diese lediglich die Syntax berichtigen und keinen Einfluss auf die Funktions-Qualität haben.

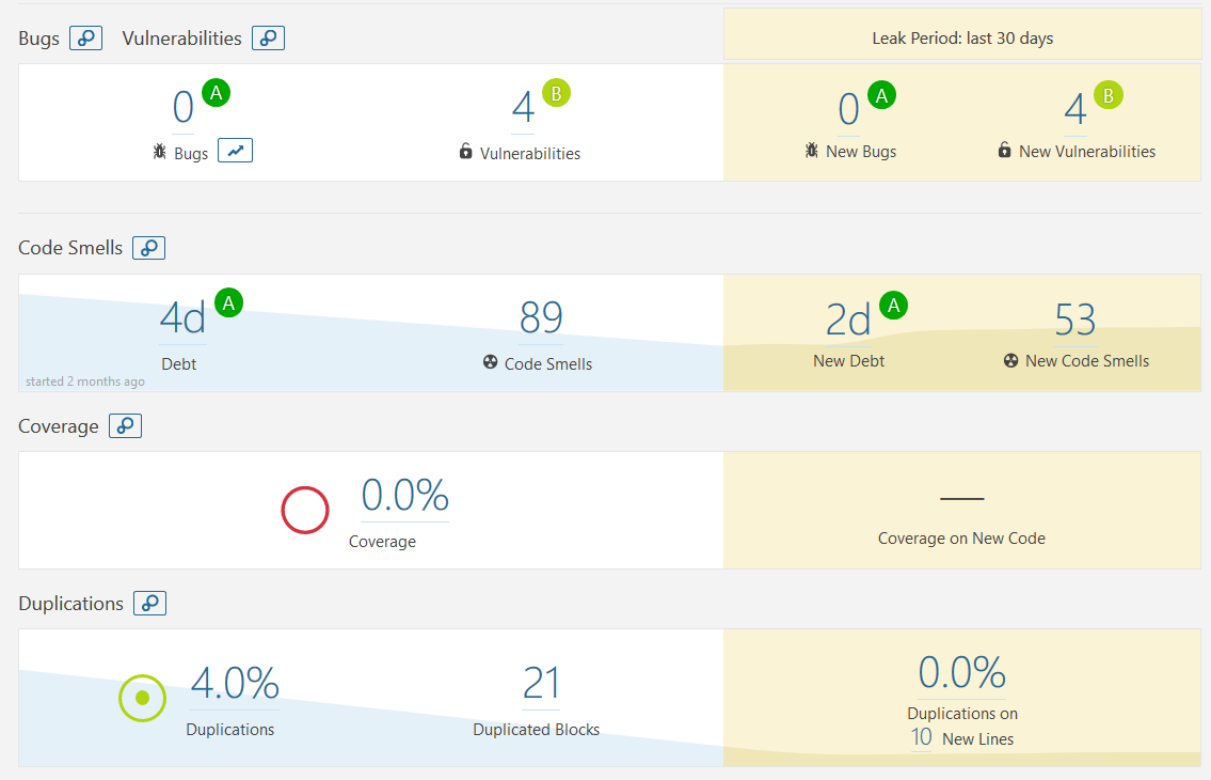


Abbildung 29: Sonarqube

[4] Codesmell = Funktionierender Programmcode, welcher schlecht strukturiert ist und überarbeitet werden sollte.

6.5 Continuous Integration [27]

Damit die Codequalität hochgehalten wird und nur eine funktionsfähige Software herausgegeben wird ist ein Integrationsablauf [28] definiert worden. Eine Funktionserweiterung erfordert eine Erstellung eines «Feature Branches», da eine Entwicklung auf dem «Master Branch» aktiv blockiert wird. Sobald die Implementation der Erweiterung abgeschlossen wird und der «Pull Request» stattfindet wird die Anwendung automatisch durch Travis, eine Continuous Integration Software, gebildet. Das Zusammenführen des «Branches» und des «Masters» erfordert zum einen das erfolgreiche Durchlaufen des Builds in Travis, zum anderen einen Code Review. Sobald diese beiden Bedingungen erfüllt sind, ist ein Zusammenführen möglich. Bei der Zusammenführung wird automatisch ein Release (eine APK) erzeugt und auf Github abgelegt. Auf Github kann jeder Benutzer die aktuellste APK herunterladen und selbst auf dem Tablet installieren. Eine Veröffentlichung der APK (der App) im Google Play Store ist aktuell nicht vorgesehen.

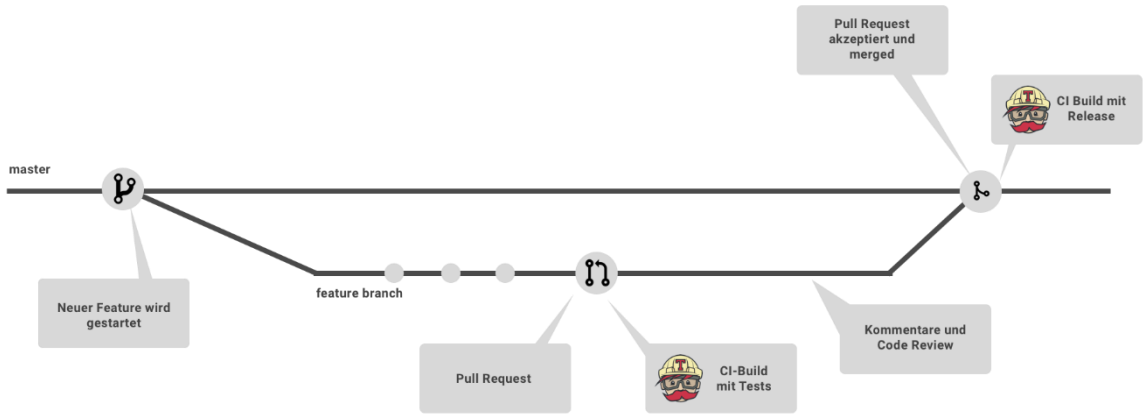


Abbildung 30: Continuous Integration Workflow

7 Ergebnisse und Schlussfolgerung

7.1 Endprodukt

Das Ziel dieser Arbeit war es eine aktualisierte Version der Anwendung Radio Tour für den Radio Tour Speaker bereit zu stellen. Die Anwendung wurde auf den neusten Stand der Technik gebracht und mit weiteren Funktionen ausgestattet. In einem agilen Prozess wurden stetig Änderungswünsche der Stakeholder aufgenommen und direkt umgesetzt. Das Endprodukt besteht aus einer Android Applikation «**RadioTour**» und der dazu passenden Hardware «Huawei Tablet MediaPad M3 Lite 10 32 GB LTE Schwarz». Die Lösung ist zum heutigen Stand bereit für erweiterte Tests durch den Radio Tour Speaker.

7.2 Offene Punkte

In diesem Abschnitt werden noch offene Umsetzungspunkte erläutert, welche in einer späteren Arbeit umgesetzt werden. Die Überschriften sind entsprechend der Auswahlreiter der Anwendung gegliedert.

7.2.1 Administration

- Globale Einstellungen beim Wechsel in Demomodus aktualisieren
- Verbindung zum Server inaktiv darstellen
- Status auf Demomodus wechseln, wenn dieser aktiviert wird

7.2.2 Maillots

- Reihenfolge umkehren, Leader offiziell in der ersten Zeile
- Leader offiziell hervorheben (fette Schrift)
- Gesamtrang jeweils in Klammern

7.2.3 Wertungen eingeben

- Darstellung der Fahrer immer gleich: Startnummer [Nationalität als Flagge] FAMILIENNAME Vorname(n), MANNSCHAFT (als Abkürzung), (Gesamtrang)
- Fehler bei der Eingabe: Selektierte Fahrer können nicht deselektiert werden, Fahrer können selektiert werden bevor ein Platz ausgewählt wurde

7.2.4 Fahrer auswählen

- AUFGABE und DNC können nicht korrigiert werden, nochmalige Selektion eines Fahrers und Selektion der Aktion sollten den Fahrer wieder auf nicht gewählt und nicht selektiert wechseln
- Fahrer können für mehrere Gruppen ausgewählt werden

7.2.5 Rennen

- Sortierung innerhalb einer Gruppe nach Startnummer
- Zeiteingaberaster an Dezimalsystem anpassen -> passen > 0…9, 10…19, 20…29, 30…39, 40…49, 50…59
- Vermeidung der Eingabe von zu kurzen Zeitabständen, Mechanismus für Minimalzeitberechnung überarbeiten

7.2.6 Performance

Aufgrund von zusätzlichen funktionalen Anforderungen (Darstellung von Daten) die erst spät in der Entwicklung entstanden sind, wurde kurzfristig die Berechnung für die aktuellen Leader in jeder Anzeige einzeln eingebunden. Dies wirkt sich negativ auf die Performance aus, die Berechnung muss ausgelagert werden und ins Datenmodell aufgenommen werden.

7.3 Ausblick

In einer weiterführenden Bachelorarbeit wird die Applikation nochmals verfeinert was die Benutzerhandhabung betrifft. Allfällige Fehler werden beseitigt, um einen reibungslosen Ablauf zu garantieren.

Ein Austausch der Daten zur API wird realisiert, so dass ein synchronisierter Zustand zwischen der Anwendung und dem Server besteht.

Um die Anwendung für alle öffentlich zu machen, ist eine Responsive Web App geplant die auch mit handelsüblichen Handys genutzt werden kann. In dieser soll die aktuelle Rennsituation für alle sichtbar sein, jedoch mit der Einschränkung des nur lesenden Zugriffs. Zum Einsatz kommen soll diese erstmals bei der Tour de Suisse 2018 vom 9. bis 17. Juni 2018.

8 Anhang

8.1 Persönliche Berichte

8.1.1 Dominik Good

Das Thema Radrennen verfolgte ich vor dieser Studienarbeit nicht aktiv. Deshalb gab es in der Anfangsphase des Projektes Unklarheiten was für Anforderungen an die Applikation gestellt werden. Erst nach einiger Einarbeitungszeit wurde mir bewusst, was das Projekt an Aufwand abwirft. Dennoch haben wir uns der Herausforderung gestellt und diese auch gemeistert. Durch eine reibungslose Zusammenarbeit ist es uns gelungen, das Projekt effizient und erfolgreich abzuschliessen. Ich danke meinem Partner, Urs Forrer, für die gemeinsamen Erfahrungen die wir sammeln durften und für die gelungene Zusammenarbeit. Während der Dauer dieses Projektes gab es für uns nie Probleme, sondern nur Lösungen. Gegenseitig konnten wir von den Erfahrungen des anderen profitieren und gemeinsam Schwierigkeiten meistern.

Nach einer sauberen Analysephase zu Beginn des Projektes wurde die Umsetzung umso einfacher, da die Anforderungen klar definiert wurden. Auf Grund dieser gründlichen Analyse wurde die Entwicklung auch erst nach vier Wochen gestartet. Aus Erfahrung von vorhergehenden Projekten haben wir von Beginn weg einen sauberen Workflow der Entwicklung definiert, und sparten somit immens Zeit ein was die Fehlerbehebung, Qualität und Konflikts Vermeidung betrifft.

Besonders gefallen hat mir die agile Entwicklung der Anwendung während der ganzen Projektdauer. Mit einer Sprintdauer von nur zwei Wochen waren wir stets in der Lage auf Änderungswünsche rasch zu reagieren und ermöglichten uns so dem Stakeholder laufend umgesetzte Anforderungen, die zum Teil neu entstanden sind, zu präsentieren.

Herausfordernd während der Entwicklung war vor allem die Performance der Anwendung. Da viele Anzeigen gleichzeitig dargestellt und aktualisiert werden müssen, ist dies ein wichtiger Aspekt für den Benutzer. Um die Performance hoch zu halten haben wir uns entschieden Caching einzusetzen, was in der Implementation allerdings nicht so einfach umzusetzen war. Auch die Implementierung des MVP Patterns hatte ich bis zu diesem Zeitpunkt noch nie ausgeführt. Durch diese Arbeit habe ich nun vollständig verstanden, wie ein solches Pattern eingesetzt werden kann. Der Einsatz einer neuen Datenbanktechnologie, welche noch am Entwicklungsanfang steht, war auch herausfordern. Dies auf Grund der dürftigen Dokumentation und den wenigen Beispielen. Dennoch haben wir diese Aufgabe erfolgreich gemeistert und sind froh uns für diesen Schritt entschieden zu haben.

Aus diesem Projekt nehme ich ein fundiertes Wissen in der Android Applikationsentwicklung mit. Zudem bin ich mir auf ein neues Mal bewusstgeworden, wie wichtig eine saubere Analysephase sowie ein sauberes Projektmanagement sind.

8.1.2 Urs Forrer

Die Studienarbeit stellte für mich das bisher grösste Softwareprojekt dar. Auf Grund meiner Lehre als Informatiker Fachrichtung Systemtechnik waren meine bisherigen Projekte (bis auf das Engineering-Projekt im letzten Semester) eher auf der Infrastrukturseite angesiedelt. Umso mehr habe ich mich auf die Studienarbeit gefreut. Da dies unser Wunschthema war, habe ich mich schon vor Semesterstart darauf gefreut. Als Antisport-Fan verfolge ich Sportanlässe sowie die Tour de Suisse im TV eigentlich nie. Durch entsprechende Unterlagen und Erklärungen seitens Herrn Heinzmann und Patrick Eichler ist es auch mir gelungen einen Einstieg und Gefallen daran zu finden.

Schon zu Beginn der vierzehn Wochen hat sich schnell herauskristallisiert, dass das Projekt doch etwas an Aufwand abwirft. Vor allem in Kombination mit einem gefüllten Semester und vielen Abgaben während dem Semester. Ich hatte immer das Ziel vor Augen und so waren die vierzehn Wochen im Vergleich zu anderen Semester sehr schnell vergangen. Die Zusammenarbeit mit Dominik Good hatte bestens funktioniert. Durch unser sehr lösungsorientiertes Arbeiten konnten wir erfolgreich alle möglichen Probleme und Stolpersteine meistern.

Persönlich empfinde ich die Analysephase von Software eher als etwas Mühsames. Man macht viel, sieht aber wenig. Das Projekt hat mir wieder einmal gezeigt, dass diese äusserst wichtig ist. Hätten wir dies nicht in diesem Rahmen gemacht, hätte uns die Tatsache später im Projekt wieder eingeholt. Somit konnten wir uns in der Entwicklung komplett auf die Umsetzung konzentrieren.

Durch die guten Vorbereitungen in der Analysephase gestaltete sich die Entwicklungsphase umso einfacher. Somit konnten wir bereits in der Hälfe der Zeit einen funktionierenden Prototyp abliefern. Die kurzgetakteten Feedbackloops (vor allem zum Ende hin) haben es uns ermöglich bei jedem Treffen wieder Erneuerungen und Bugfixes präsentieren zu können.

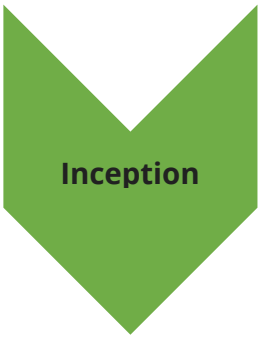
Persönlich hatte ich während dem Projekt mit der einen oder anderen Herausforderung zu kämpfen. Das bereits genannte MVP Pattern durfte ich in der Theorie bereits kennenlernen, eingesetzt hatte ich es aber noch nie. Umso froher war ich, dass Dominik hier einen guten Start mit der Implementierung hingelegt hat und mir damit den Einstieg etwas vereinfacht hat. Auch der Umgang mit dem Context im Android war sehr herausfordernd. Aufgrund der vielen UI Anpassungen an vielen Orten im Code war es nötig, auf diesen Context jeder Zeit zuzugreifen zu können. Eine nicht ganz einfache Tatsache wie sich herausstelle. Mit Unterstützung von Mikro Stocker (Mobile GUI Engineering Dozent an der HSR) ist es uns gelungen auch diese Herausforderung zu meistern. Wie bereits in vorherigen Projekten lagen wir bei den Zeiteinschätzungen der Arbeitspakete manchmal sehr stark daneben. Eine Tatsache die sich wohl nie ganz wegdenken lässt.

Persönlich nehme ich aus diesem Projekt ganz viele Eindrücke für die Zukunft mit. Zum einen konnte ich einen grossen Wissensschatz zu Android aufbauen, zum anderen habe ich wieder einmal gesehen wie wichtig die Elaborationsphase ist (auch wenn es etwas mühsam ist). Zusammengefasst, eine intensive aber sehr spannende und lehrreiche Zeit.

8.2 Projektmanagement

8.2.1 Phasen

Grober Überblick der Phasen und der entsprechenden Iterationen zu Beginn des Projekts



I1 - Iteration 1
Start am 18. September 2017
Ende am 26. September 2017

Beschreibung
Kickoff-Meeting, Projektplanung, Projektaufbau (Gemeinsame Datenablage, Projektmanagementtool)

E1 - Iteration 2
Start am 26. September 2017
Ende am 10. Oktober 2017

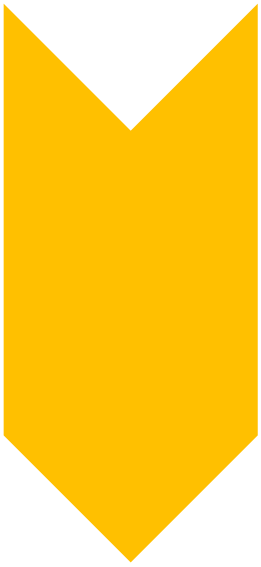
Beschreibung
Domain Model, Use Cases (im Brief-Format), Use Case Diagramm , nichtfunktionale Anforderungen, Logging und Error Policy, UI Design (UI Prototyp), Architektur und Einrichtung der Entwicklungsumgebung

C1 - Iteration 3
Start am 10. Oktober 2017
Ende am 24. Oktober 2017

Beschreibung
Evaluation lokale Datenbank und JSON Library, Einarbeitung in Technologien, Erster Prototyp, Vorbereitungen der Zwischenpräsentation, Einrichtung des Tablets

C2 - Iteration 4
Start am 24. Oktober 2017
Ende am 07. November 2017

Beschreibung
Zwischenpräsentation, Usability/User Tests, Implementierung des Imports, Erste Kapitel für den technischen Bericht



C3 - Iteration 5
Start am 07. November 2017
Ende am 21. November 2017

Beschreibung
Implementation Appbar, Performance Optimierungen

C4 - Iteration 6
Start am 21. November 2017
Ende am 05. Dezember 2017

Beschreibung
Demomode, Weitere Usertests, Splash Activity, Bugfixing

T1 - Iteration 8
Start am 05. Dezember 2017
Ende am 22. Dezember 2017

Beschreibung
End-To-End Test, Schlussbericht, Schlusspräsentation, Persönlicher Projektbericht

8.2.2 Projektplanung

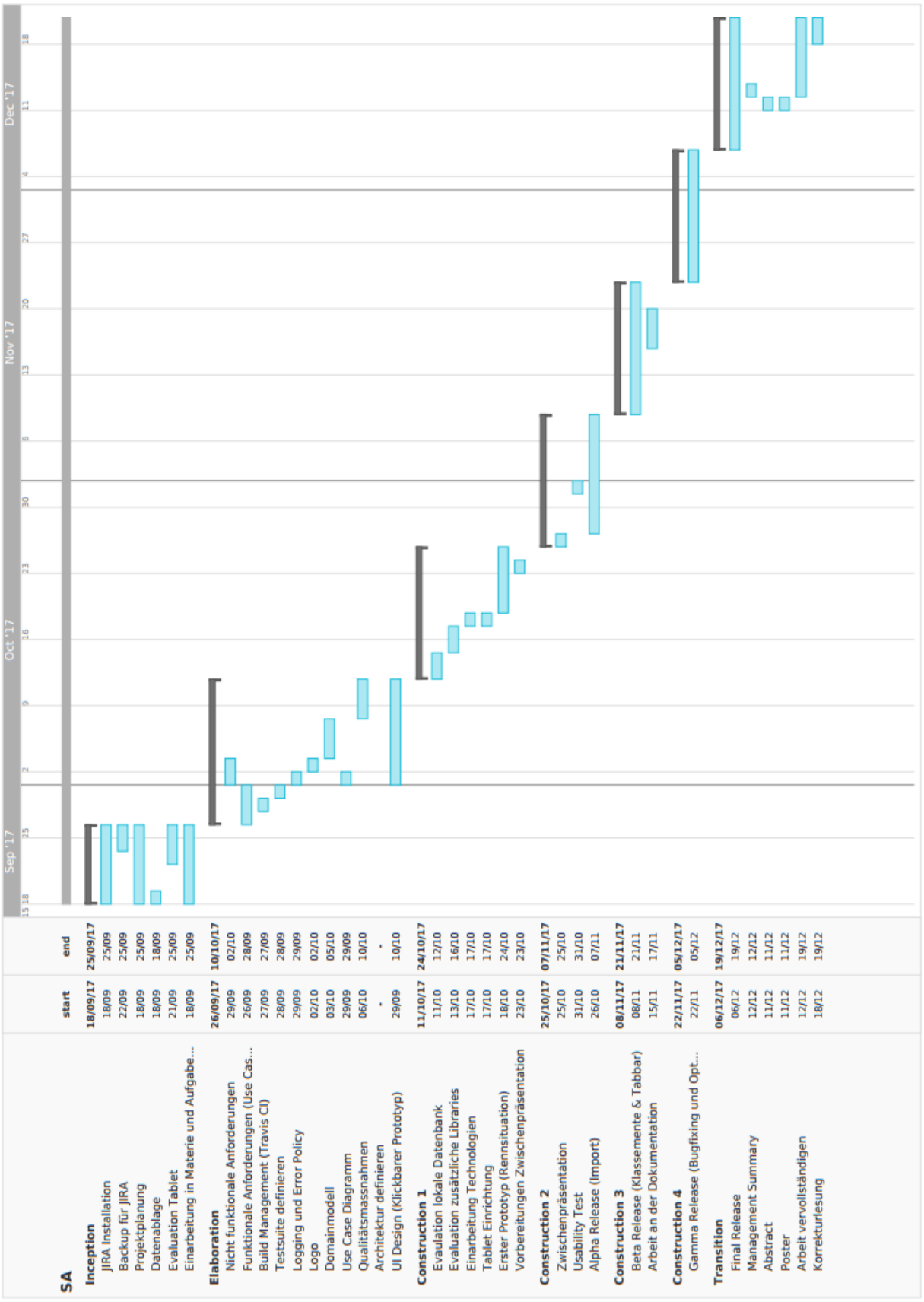


Abbildung 31: Grober Projektplan (Zu Beginn der Arbeit)

8.2.3 Releases

Hauptreleases, welche dem Product Owner zum Testen und zur Prüfung übergeben wurden. Durch Continuos Deployment wurden grundsätzlich mehrere Versionen pro Woche/Tag herausgegeben.

#	Name	Releasedatum
1	First Prototyp	24.10.2017
2	Alpha Release	07.11.2017
3	Beta Release	21.11.2017
4	Gamma Release	05.12.2017
5	Final Release	19.12.2017

Tabelle 9: Releases

8.2.4 Meilensteine

Nr.	Name	Beschreibung	Datum
MS1	Neues Tablet evaluiert	Neues Tablet für die Anwendung wurde evaluiert.	26.09.2017
MS2	Scope der Anwendung definiert	Requirements für die neue Anwendung sind definiert und wurden besprochen.	10.10.2017
MS3	Frist Prototyp	Ein Prototyp soll bestehen, welcher durch alle Architekturlayers geht.	24.10.2017
MS4	Feature Freeze	Ab diesem Zeitpunkt dürfen keine neuen Features mehr in die Software eingebaut werden. Bugfixing noch möglich.	05.12.2017
MS5	Code Freeze	Software liegt Bug frei vor und Code darf nicht mehr verändert werden.	12.12.2017
MS6	Abgabe der Arbeit	Alle Dokumente wurden abgegeben und das Projekt wird abgeschlossen.	22.12.2017

Tabelle 10: Meilensteine

8.3 Projekt Monitoring

Im diesem Kapitel sind diverse Auswertungen zum Projektverlauf zu finden.

8.3.1 Zeitaufwände pro Woche

Das Diagramm in der Abbildung 29 zeigt die Zeitverteilung über die vierzehn Kalenderwochen hinweg. Das zu erreichende Soll lag bei etwa **34 Stunden** pro Woche. Die Aufwände waren somit weitgehend konstant. Der Ausreiser zu Beginn der Arbeit ist auf das verspätete Kick-Off Meeting zurückzuführen.

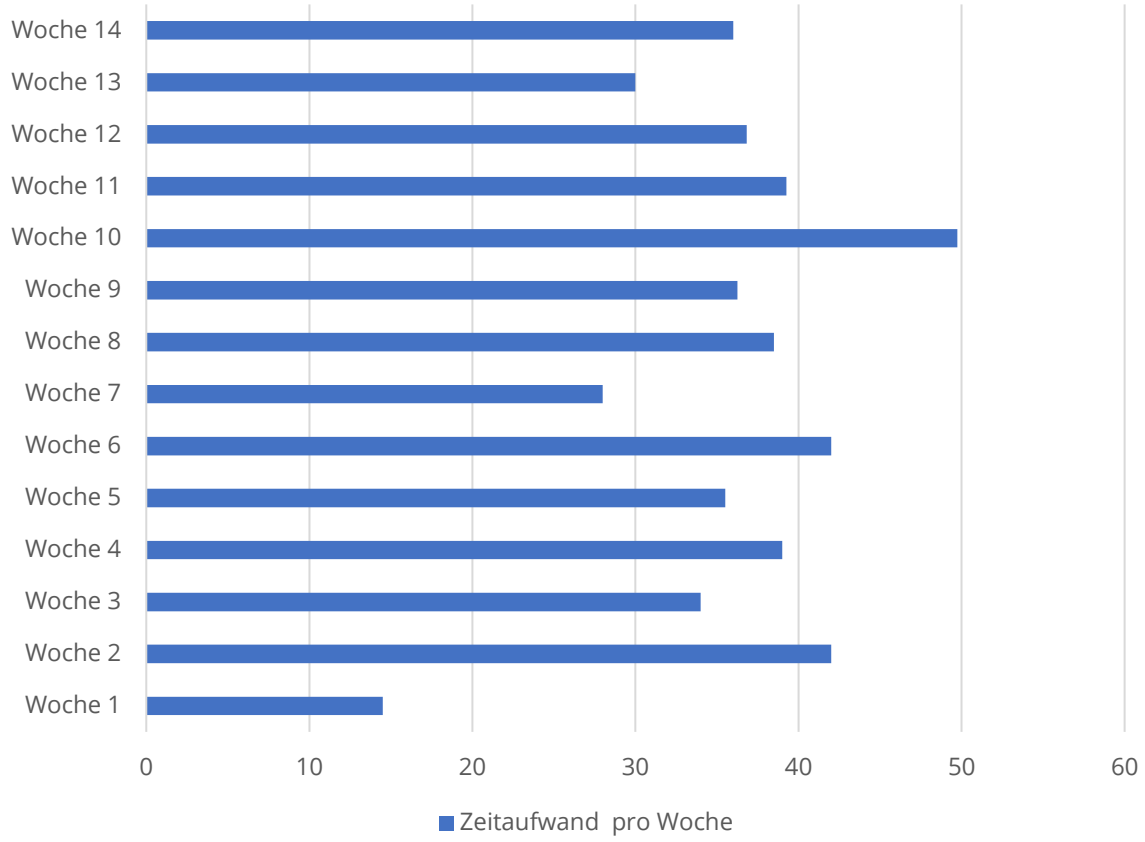


Abbildung 32: Zeitaufwände pro Woche

8.3.2 Zeitaufwände pro Phase

In der Abbildung 30 wird aufgeschlüsselt, welches Teammitglied in welcher Phase wie viel Zeit für die Studienarbeit aufgewendet hat. Bei den Angaben zu den Wochen handelt es sich um relative Angaben und nicht um die absoluten Kalenderwochen.

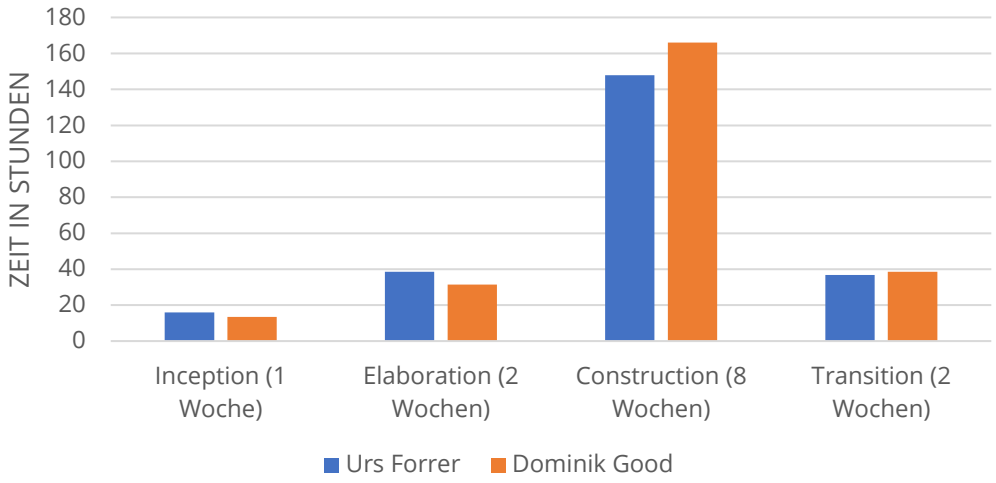


Abbildung 33: Zeitaufwände pro Phase pro Teammitglied

Das Kuchendiagramm in Abbildung 31 zeigt die Verteilung generell (nicht bezogen auf die Teammitglieder) pro Phase. Gut zwei Drittel der Zeit würde für die effektive Entwicklung der Anwendung verwendet. Diese hohen Aufwände in die Entwicklung waren nötig um das entsprechende Ergebnis zu erreichen.

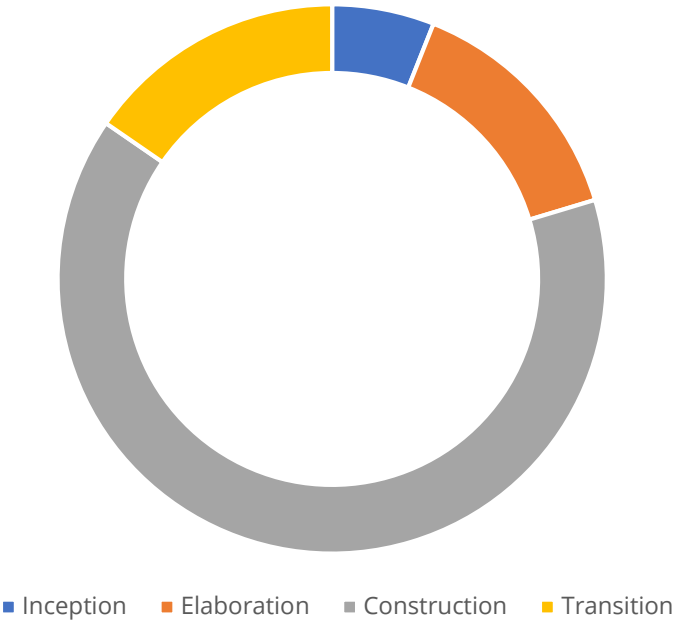


Abbildung 34: Verteilung pro Phasen

8.3.3 Verteilung im Team

Das letzte Diagramm, die Abbildung 32, zeigt die Verteilung der Zeit innerhalb des Teams. Der Unterschied von einer Stunde zeigt auf, dass die Arbeitsaufteilung sehr ausgeglichen war. Jeder konnte sich ausgeglichen einbringen.

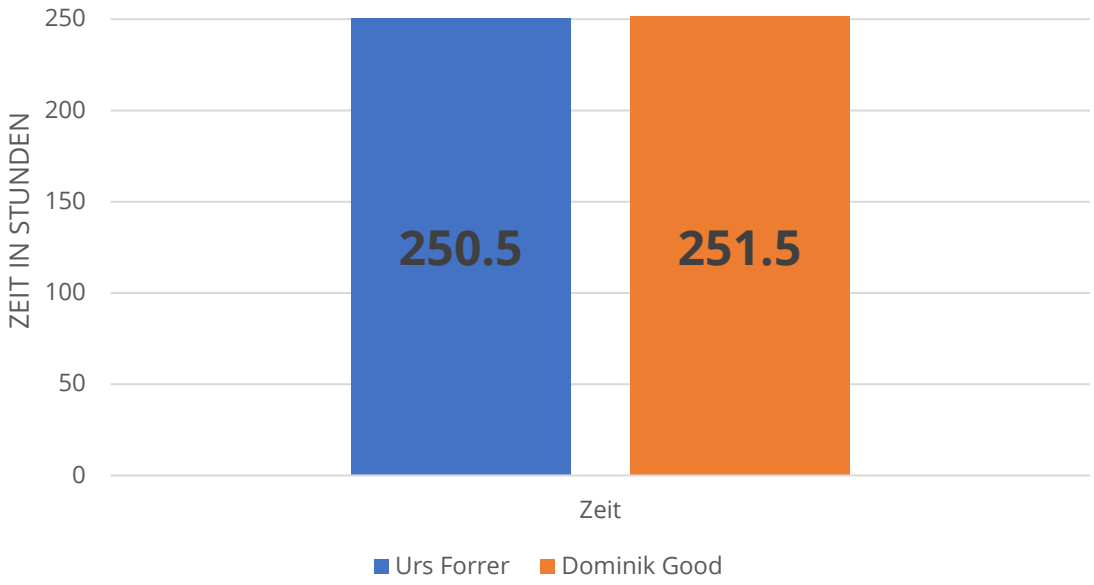


Abbildung 35: Zeitverteilung im Team

8.4 Codestatistiken

Die Tabelle 11: LOC Anwendung zeigt die Lines of Code (LOC) des gesamten Anwendungscodes. Dieser ist verteilt in Java und XML Code, wovon der XML Code hauptsächlich für das Layout zuständig ist. Im Vergleich zu unserem Engineering Projekt (HSRmarket oder Cocktail App) beträgt der Umfang dieser Arbeit ca. das Dreifache.

Sprache	LOC
Java	8196
XML	3049
Total	11’245

Tabelle 11: LOC Anwendung

Testcode wurde ebenfalls reichlich geschrieben. Im Verhältnis zum Anwendungscode beträgt dieser ca. 12%.

Sprache	LOC
Java	1468
Total	1468

Tabelle 12: LOC Testcode

Zusammengefasst betragen die Codes im Totalen 12'713 LOC.

Sprache	LOC
Java	9664
XML	3049
Total	12713

Tabelle 13: Total LOC

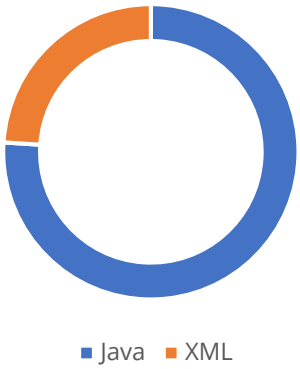


Abbildung 36: Diagramm Programmiersprache

9 Dokumentenverzeichnis

Dieses Kapitel umfasst eine Aufstellung der Dateien und Ordner, welche auf einem USB-Stick zusammen mit der Arbeit abgegeben wurden.

Verzeichnis	**Inhalt**
01_Mails	AW Radio Tour App JSON Daten.msg AW_ Studienarbeit Radiotour.eml Malliot API.msg SA Global Settings.msg SA Radiotour.eml Tablet für SA.eml WG_ Virtuelle Server für Studien-_Bachelorarbeiten.eml
02_Protokolle	20170922_Meeting.pdf 20170925_Meeting.pdf 20171002_Meeting.pdf 20171010_Meeting.pdf 20171024_Meeting.pdf 20171107_Meeting.pdf 20171121_Meeting.pdf 20171127_Meeting.pdf 20171205_Meeting.pdf 20171212_Meeting.pdf 20171219_Meeting.pdf
03_Coding_Guidelines	Coding_Policy.pdf
04_Anforderungen	NFA.pdf UC_Diagramm_RadioTour.asta Use_Cases. Pdf
05_Testing	Testing.pdf
06_UI_Design	SA_RadioTour_V2.pdf SA_RadioTour_V2.sketch Usability_Tests.pdf

07_TourLive_Unterlagen_cnlab	RadioTourBeispielmeldungen_Drehbuch_Testing.docx Radio-TourMarcheASuivre_V2.docx SpezialklassementeTdS2016.xlsx Startlist_TdS_2017.pdf TdS2015_Bergpreisklassement.pdf TdS2015_BesterSchweizer.pdf TdS2015_Gesamtklassement.pdf TdS2015_Punkteklassement.pdf TdS2017_Uebersicht_v1.1.pptx Technischer_Guide_D_web.pdf Teilnehmer_Participants_TdS_2017.pdf
08_Architetkur	Datenmodell.asta Domainmodell.asta Erweiterte_Architektur_Ansicht.png GesamtUebersicht_RadioTour.png SoftwareArchitektur. pdf Sequenzdiagramm_Eingabe_von_Daten.png
09_Zwischenpräsentation	SA_RadioTour_Zwischenpräsentation.pdf
10_Poster	Poster_SA_RadioTour.pdf
11_Sourcecode	Inhalt des Repos Link zum GitHub Repo

Tabelle 14: Dokumentenverzeichnis

10 Literaturverzeichnis

[1] “Drag and Drop.” [Online]. Available: https://de.wikipedia.org/wiki/Drag_and_Drop. [Accessed: 08-Dec-2017].

[2] “Samsung Galaxy Tab A (10.1", 16GB, 4G, Metallic Black) - digitec,” *Online*, 2017. [Online]. Available: https://www.digitec.ch/de/s1/product/samsung-galaxy-tab-a-101-16gb-4g-metallic-black-tablet-5780762. [Accessed: 23-Sep-2017].

[3] “Huawei MediaPad T2 Pro (10.10", 16GB, 4G, Charcoal Black) - digitec,” *Online*, 2017. [Online]. Available: https://www.digitec.ch/de/s1/product/huawei-mediapad-t2-pro-1010-16gb-4g-charcoal-black-tablet-5725026. [Accessed: 23-Sep-2017].

[4] “Lenovo Tab 4 (10.1", 16GB, 4G, Slate Black) - Tablet - digitec,” *Online*, 2017. [Online]. Available: https://www.digitec.ch/de/s1/product/lenovo-tab-4-101-16gb-4g-slate-black-tablet-6594690. [Accessed: 23-Sep-2017].

[5] Brack.ch, “Tablet Toughpad FZ-A2 (4G).” [Online]. Available: https://www.brack.ch/panasonic-tablet-toughpad-465418. [Accessed: 23-Sep-2017].

[6] Digitec, “Samsung Galaxy Tab S3 (9.70", 32GB, 4G, Schwarz).” [Online]. Available: https://www.digitec.ch/de/s1/product/samsung-galaxy-tab-s3-970-32gb-4g-schwarz-tablet-6156384. [Accessed: 23-Sep-2017].

[7] Brack.ch, “Tablet MediaPad M3 Lite 10 32 GB LTE Schwarz.” [Online]. Available: https://www.brack.ch/huawei-tablet-mediapad-m3-564226. [Accessed: 23-Sep-2017].

[8] E. Hardy, “Samsung Galaxy Tab A 10.1 Review: Mid-Range or Just Right?” [Online]. Available: http://www.tabletpcreview.com/tabletreview/samsung-galaxy-tab-10-1-review-2/. [Accessed: 23-Sep-2017].

[9] F. Wimmer, “Huawei MediaPad M3 Lite Tablet Review.” [Online]. Available: https://www.notebookcheck.net/Huawei-MediaPad-M3-Lite-Tablet-Review.226622.0.html. [Accessed: 23-Sep-2017].

[10] Nick Sutrich, “Samsung Galaxy Tab S3 Review.” [Online]. Available: https://www.androidheadlines.com/2017/03/samsung-galaxy-tab-s3-review.html. [Accessed: 23-Sep-2017].

[11] M. Moser, “Huawei MediaPad M2 Tablet Review.” [Online]. Available: https://www.notebookcheck.net/Huawei-MediaPad-M2-Tablet-Review.153396.0.html. [Accessed: 23-Sep-2017].

[12] Panasonic, “Panasonic Introduces Toughpad FZ-A2 Fully Rugged 10.1” Android Tablet.” [Online]. Available: http://shop.panasonic.com/about-us-

latest-news-press-releases/01122017-toughpad-android.html. [Accessed: 23-Sep-2017].

[13] Google, “Shared Preferences.” [Online]. Available: https://developer.android.com/reference/android/content/SharedPreferences.html. [Accessed: 06-Nov-2017].

[14] (Hwaci) Hipp, Wyrick & Company, Inc., “SQLite.” [Online]. Available: https://www.sqlite.org/. [Accessed: 06-Nov-2017].

[15] Realm, “Realm.” [Online]. Available: https://realm.io/. [Accessed: 06-Nov-2017].

[16] “Model View Presenter,” 2017. [Online]. Available: https://www.techyourchance.com/mvp-mvc-android-1/. [Accessed: 08-Dec-2017].

[17] “Callback.” [Online]. Available: https://de.wikipedia.org/wiki/Rückruffunktion. [Accessed: 08-Dec-2017].

[18] “Realm Object.” [Online]. Available: https://realm.io/docs/java/3.7.2/api/io/realm/RealmObject.html. [Accessed: 08-Dec-2017].

[19] “Realm Managed Objects.” [Online]. Available: https://medium.com/@ffvanderlaan/realm-auto-updated-objects-what-you-need-to-know-b2d769d12d76. [Accessed: 08-Dec-2017].

[20] “Sync Http Client.” [Online]. Available: https://loopj.com/android-async-http/doc/com/loopj/android/http/SyncHttpClient.html. [Accessed: 08-Dec-2017].

[21] “JsonHttpResponseHandler.” [Online]. Available: https://loopj.com/android-async-http/doc/com/loopj/android/http/JsonHttpResponseHandler.html. [Accessed: 08-Dec-2017].

[22] “Parser.” [Online]. Available: http://searchmicroservices.techtarget.com/definition/parser. [Accessed: 08-Dec-2017].

[23] “UI.” [Online]. Available: https://en.wikipedia.org/wiki/User_interface. [Accessed: 08-Dec-2017].

[24] “Instrumented Tests.” [Online]. Available: https://developer.android.com/training/testing/unit-testing/index.html. [Accessed: 10-Dec-2017].

[25] “Code Styleguide.” [Online]. Available: https://google.github.io/styleguide/javaguide.html. [Accessed: 12-Dec-2017].

[26] “Sonarqube.” [Online]. Available: https://www.sonarqube.org/. [Accessed: 12-Dec-2017].

[27] “CI.” [Online]. Available: https://automic.com/de/blog/was-ist-continuous-integration-und-was-ist-es-nicht. [Accessed: 12-Dec-2017].

[28] “Git.” [Online]. Available: https://git-scm.com/book/de/v1/Los-geht’s-Git-Grundlagen. [Accessed: 12-Dec-2017].

[29] “API.” [Online]. Available: https://de.wikipedia.org/wiki/Programmierschnittstelle. [Accessed: 19-Dec-2017].

[30] “JSON.” [Online]. Available: https://www.w3schools.com/js/js_json_intro.asp. [Accessed: 19-Dec-2017].

[31] Wikipedia, “OR Mapper.” [Online]. Available: https://de.wikipedia.org/wiki/Objektrelationale_Abbildung. [Accessed: 20-Dec-2017].

[32] “Android Local Unit Tests.” [Online]. Available: https://developer.android.com/training/testing/unit-testing/local-unit-tests.html#run.

11 Glossar

Begriff	Erläuterung
Activties	Eine Activity ist eine Komponente, welche einen Screen zur Verfügung stellt. Der Benutzer kann damit interagieren.
Android	Betriebssystem für Tablets
API [29]	Schnittstelle für den Datenaustausch
APK	Eine APK (Android Package) ist eine Datei zur Installierung einer Anwendung auf dem Android-Betriebssystem (wie eine .exe Datei unter Windows)
C#	Programmiersprache aus dem Hause Microsoft
CRUD	Akronym für die vier grundlegenden Operationen auf einem persistenten Speicher - **C**(reate) - **R**(ead) - **U**(pdate) - **D**(elete)
Fragments	Ein Fragment repräsentiert ein Teil (oder Behavior) der Benutzeroberfläche und lebt innerhalb einer Activity.
GPS	Global Positioning System zur Ermittlung des aktuellen Standorts
HTTP	Protokoll für Informationsaustausch im Web
INS	Institute for Networked Solutions an der Hochschule Rapperswil
Invision	Online Tool zur Erstellung von klickbaren UI-Prototypes
ISO9126	ISO-Norm zur Sicherstellung der Softwarequalität

Java	Programmiersprache, welcher innerhalb der JVM (Java Virtual Machine) läuft
JSON [30]	Spezifisches Datenformat
Kotlin	Neuartige Programmiersprache zur Entwicklung von Android Apps
LOC	Lines of Code, Anzahl Zeilen Code im Projekt
MVP [16]	Model-View-Presenter ➔Pattern für Interprozesskommunikation
NFA	Abkürzung für nicht-funktionale Anforderungen
Objective-C	Alte Programmiersprache auf der Apple Plattform
OR Mapper [31]	Technik zur Abbildung von Objekten einer Programmiersprache in einer Datenbank
Realm [15]	Datenbank basierend auf JSON
RUP	Rational Unified Process, ein Phasenmodell in der Software-Entwicklung
SA	Studienarbeit, Arbeit zur Vorbereitung auf die Bachelorarbeit
Shared Preferences [13]	Gemeinsame genutzte Datensätze verschiedener Teile einer Anwendung
Sonarqube [26]	Tool zur statischen Code Analyse
SRF	Abkürzung für den Schweizer Broadcaster «Schweizer Radio und Fernsehen»
Swift	Moderne Programmiersprache auf der Apple Plattform
UI [23]	Benutzeroberfläche für Interaktionen
Unit Test [32]	Test einer Funktion innerhalb einer Anwendung

Tabelle 15: Glossar