HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Bachelorarbeit

CoachAssist

Aufzeichnung von Fussballspielen

Hochschule für Technik Rapperswil
Abteilung Informatik
Frühjahrssemester 2012
15. Juni 2012

Autoren	Florian Ziltener & Mike Osmand Loydd
Betreuer	Prof. Dr. Peter Heinzmann
Projektpartner	Philippe Eberli, cnlab AG
Experte	Dr. Th. Siegenthaler, CSI Consulting AG
Gegenleser	Prof. Dr. Markus Stolze

Arbeitsperiode	20.02.2012 - 15.06.2012
Arbeitsumfang	360 Arbeitsstunden bzw. 12 ECTS pro Student

Abstract

Die im Rahmen der Bachelorarbeit realisierte Android Applikation *CoachAssist* dient zur Protokollierung spezieller Events (z.B. Tore, Freistösse, Verwarnungen, Auswechslungen) bei Fussballspielen. Solche Daten sind für Coaches, Sportwissenschaftler und Fans von Interesse. Die Firma cnlab nutzt diese Daten für die GPS-Fussballspieler-Tracking-Applikation zur Visualisierung und Analyse der Laufwege der Spieler.

CoachAssist kann sowohl Standalone als auch in Verbindung mit dem cnlab Fussballspieler-Tracking Server betrieben werden. Für die Icon-basierte Eventerfassung stand die einfache Bedienbarkeit im Vordergrund. Spiel- und Spielerinformationen können lokal oder über die cnlab Webseite erfasst und verwaltet werden. *CoachAssist* legt alle Event-, Spieler- und Matchinformationen auf einer lokalen SQLite-Datenbank ab und synchronisiert die Daten, sobald mit dem cnlab Server verbunden wird.

CoachAssist basiert auf Android 4.0. Für die Datenbankkommunikation wurde ORMLite, eine Object Relational Mapping Framework verwendet. Der Datenaustausch mit dem cnlab Server erfolgt mit JSON. Zum Parsing wird Gson eingesetzt.

Die nun vorliegende *CoachAssist* Prerelease Version wird gegenwärtig von ausgewählten Fussballcoaches und Fans getestet. Nach dieser Testphase wird *CoachAssist* bei cnlab weiter optimiert. Eine Veröffentlichung von *CoachAssist* auf dem Android Market ist im Herbst 2012 geplant.

Aufgabenstellung

Tabelle 1: Aufgabenstellung - Die wichtigsten Angaben

Abteilung:	Informatik
Name der Studierenden:	Mike Hoskin Osmand Loydd, Florian Ziltener
Studienjahr:	Frühjahrsemester 2012 (19.02.2012-15.06.2012)
Titel der Bachelorarbeit:	CoachAssist
Gegenleser:	Prof. Dr. M. Stolze
Examinator:	Prof. Dr. P. Heinzmann
Experte / Koreferent:	Dr. Th. Siegenthaler, CSI Consulting AG, Zürich
Themengebiet:	Internet-Technologien und -Anwendungen
Projektpartner:	cnlab AG Rapperswil, P. Eberli

Ausgangslage

Der Industriepartner cnlab bietet ein Fussballspieler-Tracking-System an zur GPS-basierten Erfassung der Bewegungen von Fussballspielern (`http://www.cnlab.ch/fussball/`). Die Visualisierung und Analyse der Spiel- und Spielerdaten wird vereinfacht, wenn genaue Informationen über Spielevents (z.B. Auswechslungen von Spielern, Tore, Anspielzeiten, Spielendzeiten) vorliegen. Solche Informationen sind auch für Spielbetreuer wichtig. Viele Coaches notieren sich spezielle Spielevents auf Papier. Während der Coachausbildung muss zu jedem Spiel ein Logbücher geführt werden. Es gibt diverse Mobile und Tablet Apps, welche Coaches bei der Aufzeichnung von Spielevents unterstützen. Im Rahmen einer Semesterarbeit wurden verschiedene Apps analysiert und basierend auf jQuery Mobile wurde ein "*CoachAssist*-Prototyp" mit realisiert.

Ziel

Im Rahmen dieser Bachelorarbeit soll nun eine "*CoachAssist* native App" für Tablets mit erweiterter Funktionalität realisiert werden. *CoachAssist* soll in das cnlab Fussballspieler-Tracking-System integriert sein und alle nötigen Funktionen zur Erstellung von Spiel-Logbüchern anbieten. Am Ende der Bachelorarbeit soll die Anwendung bereit sein für "Friendly User Tests", d.h. erste Usability Tests sollen bereits abgeschlossen sein.

Teilaufgaben

1. Requirement Engineering
   a Aktualisierung der Marktanalyse zu verschiedenen Fussball-Apps
   b Festlegung der Kommunikation mit dem Fussballspieler-Tracking System von cnlab
   c Festlegung aller *CoachAssist* Funktionen (Spezifikation), z.B.
      - Erfassung der Spielranddaten (erweitertes Matchblatt)
        - Ort, Zeit
        - Wetterdaten
        - Spielfeldbezeichnung, Spielfeldeigenschaften
          * Grösse (Länge, Breite, Koordinaten der Eckpunkte, Höhe über Meer)
          * Bodenart (Rasen, Hartplatz, Sandplatz, Kunstrasen Typ A, B, C, ?)
          * Bodenbeschaffenheit (tiefer Boden, nass, trocken, ?)
        - Spieltyp
          * Meisterschaft, Freundschaft, Trainingsspiel
          * Liga
        - Beleuchtung (Tageslicht, Flutlicht, gemischt)
        - Heimmannschaft, Auswärtsmannschaft
        - Spieler
          * Name, Vorname, Passnummer, Geburtsdatum
          * Position (Torhüter, Verteidiger, Mittelfeldspieler, Stürmer)
        - Erfassung der Mannschaftsaufstellung
          * Spielsystem
          * Besetzung der Positionen
            · Spielername und Nummer (Trikotnummer)
            · Dauer in dieser Position (berücksichtigt Auswechslungen)
        - Spielzeiterfassung
          * Beginn/Ende gemäss Schiedsrichter An-/Abpfiff
          * Messmöglichkeit für die effektive Spielzeit

        - Eventerfassung
            * An-/Abpfiff
            * Eckball
            * Freistoss direkt/indirekt
            * Strafstoss (Elfmeter)
            * Einwurf
            * Verletzung
            * Verwarnung (gelbe Karte)
            * Ausschluss (rote Karte)
        - Notizen
        - Synchronisation mit cnlab Fussballspieler-Tracking System
        - Mehrsprachigkeit (dt, en, fr, it)
    - Geschickte Aufteilung von konfigurierten Daten (sind in Konfigurationseinstellungen festgelegt, können verändert werden) und einzugebenden Daten (müssen vor jedem Spiel eingegeben werden, werden aber teilweise von vorangegangenen Angaben übernommen)
    - Abruf und Visualisierung der Positionsauswertungen vorangegangener Spiele
        - Distanz (gesamtes Spiel, erste Halbzeit, zweite Halbzeit; bei Teileinsätzen in Klammern umgerechnet auf die gesamte Spielzeit)
            * Aufgeschlüsselt nach Spielern, Positionen, Mannschft
            * Entwicklung über mehrere Spiele
            * Anzeige von Minimum, Durschnitt, Maximum (bei der Zusammenfassung von mehreren Messwerten)
            * Visualisierung, Animation (vgl. Runtastic)
    - Taktikdarstellungsanwendung
        - Spieler anzeigen, verschieben
        - Handzeichnungen (Pfeile)
    - Audioaufnahme (Notizen)
    - Bildaufnahme (Spielfeld, Spieler, Situationen, evtl. Spielerfotos)
    - Videoaufnahme (z.B. Penalty, Eckballsituation)
    - GPS-Aufnahme (geografische Daten, Spielfeld ablaufen)
    - Verwendung zum Einlesen der GPS-Tags, Importer RJ (C)

2. Design

3. Realisierung

4. Testing

Referenzen

1. Administratives zu Studien- und Bachelorarbeiten, Merkpunkte für die Durchführung der Arbeit, `http://www.cnlab.ch/kurse/SABA/`
2. cnlab Outdoor-Object-Tracking Systems, 2011, `http://www.cnlab.ch/fussball/`.
3. Fussballspielen mit GPS-Navigation, SF DRS, Einstein, 5.5.2011, http://www.videoportal.sf.tv
4. Kick-Analyse mit GPS, 3sat, 4.6.2011 `http://www.3sat.de/page/?source=/nano/technik/155189/index.html`
5. Beispiele Coach/Fussball-Apps

   a iPad (bereits auf Demo iPad installiert)
   - iCoach
   - Mourinho Tactical Board

   b Android:
   - Fußball Strategie Board, Cowbell Software, Diagramm Fußball Strategien direkt auf Ihrem Handy mit Fußball Strategie Board.
   - Soccer clipboard lite, J Plus Corporation, Draw unlimited number of plays with your fingers and animate them.

Unterschrift des Aufgabenstellers:

Ort, Datum:	Name, Unterschrift:
	Prof. Dr. P. Heinzmann

Erklärung zur Urheberschaft[1]

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

Tabelle 2: Leistungsübersicht

Gegenstand, Leistung	**Person**	**Funktion**
Implementation, Dokumentation	Florian Ziltener	Autor der Arbeit
Implementation, Dokumentation	Mike Osmand Loydd	Autor der Arbeit
Entwerfen der Icons	Thomas Heinzmann	Designer
Korrektur der Dokumentation	Ruth Ziltener	Korrekturleser
Korrektur der Dokumentation	Tobias Ziltener	Korrekturleser
Korrektur der Dokumentation	Daniel Stucki	Korrekturleser
Idee, Aufgabenstellung, allgemeines Pflichtenheft, Betreuung während der Arbeit	Prof. Dr. Peter Heinzmann	Verantwortlicher Professor
Entwicklung der Serverseite und zur Verfügungstellung der Server-API, Betreuung während der Arbeit	Philippe Eberli, cnlab AG	Industriepartner

1. Diese Erklärung basiert auf der Muster-Erklärung in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 16. Februar 2009.

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.

Aus der Semesterarbeit wurde die Funktionalität der Anwendung und das User Interface Design übernommen. Die Funktionalität wurde bei der Bachelorarbeit erweitert und neue Use Cases wurden definiert.

Ort, Datum:	Name, Unterschrift:
	Mike Osmand Loydd
Ort, Datum:	Name, Unterschrift:
	Florian Ziltener

Vereinbarung zur Verwendung und Weiterentwicklung der Arbeit[1]

Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der *Bachelorarbeit "CoachAssist"* von Mike Osmand Loydd und Florian Ziltener unter der Betreuung von Prof. Dr. P. Heinzmann (für die Arbeit verantwortlicher Professor) geregelt.

Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student 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 weiter entwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht, aber nicht Pflicht.

Unterschriften sind auf der nachstehenden Seite zu finden.

1. Diese Vereinbarung basiert auf den Muster-Vereinbarungen in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 16. Februar 2009.

Ort, Datum:	Name, Unterschrift:
	Mike Osmand Loydd
Ort, Datum:	Name, Unterschrift:
	Florian Ziltener
Ort, Datum:	Name, Unterschrift:
	Prof. Dr. P. Heinzmann
Ort, Datum:	Name, Unterschrift:
	Industriepartner

Broschürentext

Ausgangslage

Der Industriepartner cnlab hat ein Tracking System entwickelt, mit welchem die Laufwege von Fussballspielern aufgezeichnet werden können. Ziel dieser Bachelorarbeit war eine Tabletanwendung *CoachAssist* für die Aufzeichnung von Events bei Fussballspielen zu entwickeln. *CoachAssist* bietet folgende Möglichkeiten:

- Spielevents und Mannschaftsdaten sollen vom cnlab Tracking System übernommen werden können. Die Anwendung soll aber auch ohne den cnlab Server betrieben werden können.
- Coaches sollen die Anwendung zur Teamverwaltung und Matchaufzeichnung nutzen können.
- Fussballfans sowie Sportwissenschafter sollen Matchevents und effektive Spielzeiten erfassen können.

Abbildung 1: Matchaufnahme: Intuitive Eingabe von Spielevents zur eigenen und gegnerischen Mannschaft. Alle Event werden im Liveticker dargestellt

Vorgehen / Technologien

Die *CoachAssist*-Applikation ist mit Android 4.0 realisiert. Die Entwicklung erfolgte gemäss der Android Developers Code- und Design-Guidelines. Zur Erstellung und Verwendung der SQLite Datenbank wurde das ORM Framework ORMLite verwendet. Die Kommunikation mit dem cnlab Server erfolgt in JSON-Format. Für die Serialisierung und Deserialisierung von Java Objekten und JSON wurde Gson eingesetzt.

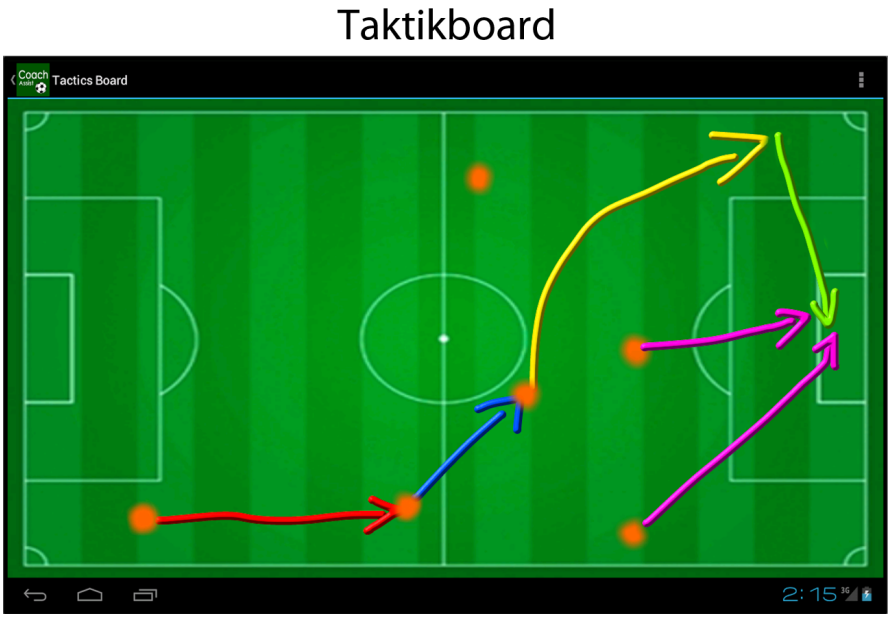


Abbildung 2: Taktikboard: Taktik-Szenen können gezeichnet und mit der Mannschaft besprochen werden.

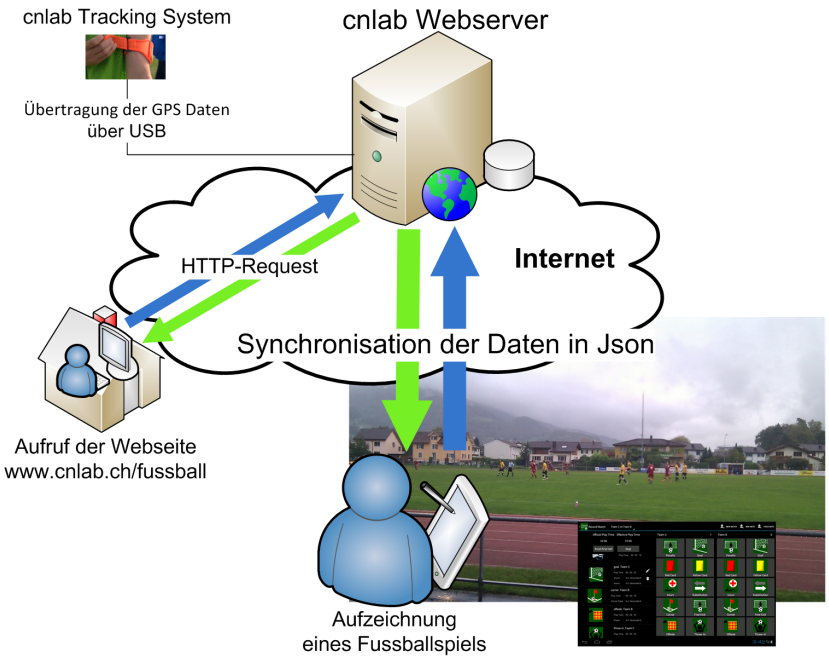
Ergebnisse

Die realisierte Android-Anwendung *CoachAssist* umfasst die folgenden Hauptfunktionen:

- Kommunikation mit dem cnlab Server (Abbildung 3)
- Matchaufzeichnung (Spielevents bzw. Liveticker, effektive Spielzeit, Spielerauswahl) (Abbildung 1)
- Verwaltung (Teams, Spieler und Matchs)
- Taktikboard (Abbildung 2)

Die Software wird gegenwärtig vom Industriepartner getestet. Nach allfälligen Anpassungen soll sie auch von Coaches getestet werden. Der offizielle Release der *CoachAssist* Anwendung ist auf Herbst 2012 geplant.

Abbildung 3: Systemübersicht: Tablet-Anwendung für Coaches, cnlab Tracking System und Fan zur Eventaufzeichnung, Datenlieferung zum cnlab Server.



Management Summary

Ausgangslage

Die Bachelorarbeit ist eine Fortsetzung der Semesterarbeit. In der Semesterarbeit wurde die Applikation mit Webtechnologien (JQuery Mobile und PhoneGap) umgesetzt. Aus Performancegründen wurde die Applikation verworfen. In dieser Bachelorarbeit wurde das Ganze neu aufgerollt. Als Einziges wurde das Grobdesign des User Interfaces und ein Teil des Funktionsumfangs in diese Arbeit übernommen. Der Funktionsumfang wurde erweitert. Folgende Punkte stehen für die Applikation im Vordergrund:

- Erfassung von Spielinformationen für das Fussballspieler-Tracking System von cnlab.
- Coaches sollen die Applikation zur Teamverwaltung und Matchverwaltung nutzen können.
- Fussballfans sowie Sportwissenschafter sollen Matchevents und effektive Spielzeiten erfassen können

Das Letztere ist erst seit dem Start der Bachelorarbeit eine wichtige Funktionalität. Dies hat zur Folge, dass ein Hauptziel der Bachelorarbeit eine Standalone Applikation ist. In der Semesterarbeit war die Applikation wesentlich stärker auf den cnlab Tracking System angewiesen.

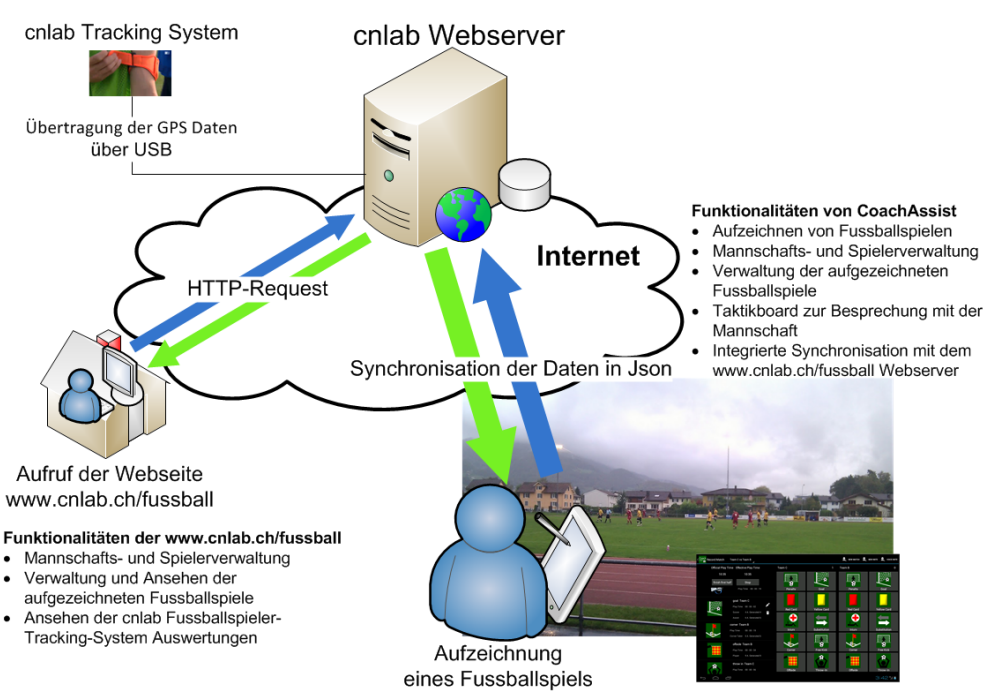
Vorgehen, Technologie

In dieser Arbeit wurde nach Rational Unified Process (RUP) vorgegangen. Folgende Iterationen wurden durchgeführt:

- Eine Inception Phase
- Zwei Eleborations Phasen
- Vier Construction Phasen
- Eine Transition Phase

Jede Iteration dauerte zwei Wochen. Nach jeder Iteration wurde ein Meilenstein gesetzt mit daraus resultierenden Dokumenten oder Programmversionen.
Die Software ist mit Android 4.0 realisiert. Für die Entwicklung sind die Code- und Design-Guidelines von Android Developers (`http://developer.android.com/index.html`[6]) miteinbezogen.
Für die Datenbankverwaltung wurde das Object Relational Mapping (ORM) Framework ORMLite verwendet, welches eine effiziente Verwaltung der SQLite Datenbank ermöglicht.
Für die Kommunikation 4 mit dem cnlab Server wird JavaScript Object Notation (JSON) verwendet. Für das Parsing wird die Klassenbibliothek Gson verwendet. Gson ermöglicht die Serialisierung und Deserialisierung von Java Objekte und JSON.

Abbildung 4: CoachAssist Kontextdiagramm



Ergebnisse

Die Software ist lauffähig und kann für Testzwecke verwendet werden. Die Funktionen mit der höchsten Priorität wurden komplett umgesetzt. Ein Teil der zweitrangigen Funktionen wurde ebenfalls umgesetzt.

Umgesetzte Hauptfunktionen sind:

- Team- und Spielerverwaltung
- Matchverwaltung mit Festlegung der Startaufstellung inkl. Formation
- Matchaufzeichnung inkl. Liveticker 5
- Taktikboard
- Kommunikation mit cnlab Server

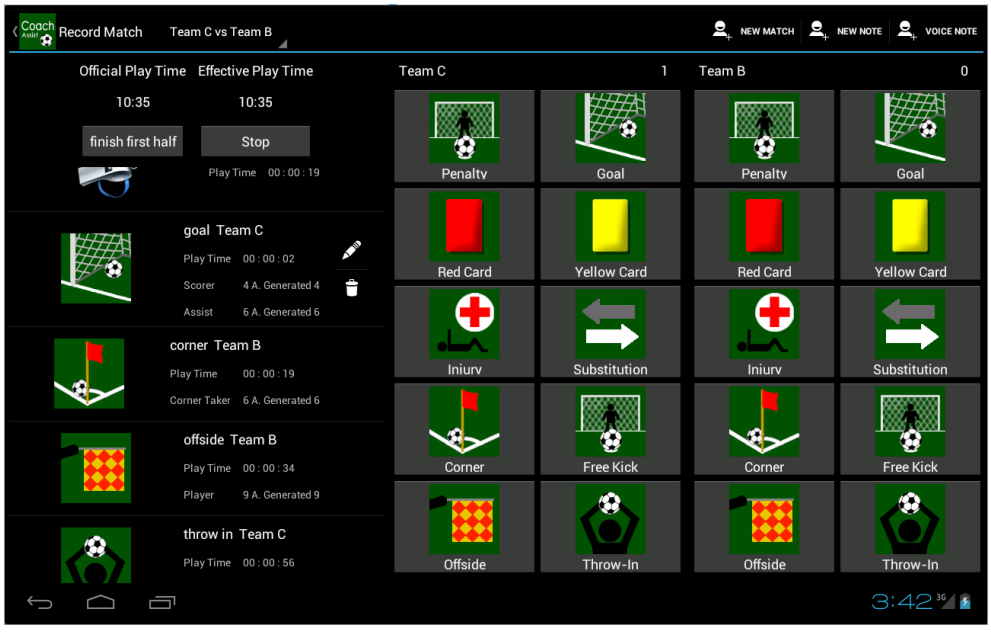


Abbildung 5: CoachAssist Match Record

Die Hauptziele dieser Arbeit sind erfüllt. Die Unterstützung des cnlab Tracking Systems ist durch die Aufzeichnung von Fussballspielevents gewährleistet. Daten können zum Server hochgeladen und für das System verwendet werden. Die Team- und Matchverwaltung für die Coaches wurde ebenfalls realisiert. Der letzte Punkt, eine Standaloneversion, wurde umgesetzt. Die dafür notwendigen Funktionen sind alle in der Applikation enthalten.
Die Funktionalität wurde mit Usability Tests überprüft. Durch die letzten Anpassungen ist eine Prerelease-Version entstanden. Die Application Programming Interface (API) zum cnlab Server wurde von P.Eberli entwickelt. Die Icons wurden vom Designer T. Heinzmann erstellt.

Ausblick

Die weitere Entwicklung wird an den Industriepartner cnlab übergeben. Die Applikation wird von cnlab weiter optimiert und mit zusätzlichen Funktionalitäten erweitert. Die Durchführung von weiteren Usability Tests ist ein wichtiger Punkt, damit eine Release Version zustande kommt.

Inhaltsverzeichnis

Teil I

Applikationsübersicht

1 Einleitung

In diesem Kapitel wird eine Übersicht über die Bachelorarbeit und die Applikation gegeben. Es wird auf die Systemübersicht eingegangen. Diese Übersicht zeigt, wie die Applikation im gesamten cnlab System eingebettet ist.

1.1 Ausgangslage

In dieser Bachelorarbeit wurde eine Tablet-Applikation für Fussballcoaches, basierend auf Android 4.0, entwickelt. Diese Applikation soll in erster Linie den Fussballcoaches helfen die Spiele seiner Mannschaft aufzuzeichnen. Die weiteren Funktionalitäten sind die Verwaltung der aufzuzeichnenden bzw. aufgezeichneten Spiele und die Verwaltung der Mannschaften inkl. den Spielern. Zusätzlich bietet die Applikation dem Benutzer des cnlab Fussballspieler-Tracking System die Möglichkeit, Daten mit dem cnlab Server zu synchronisieren. Durch die Synchronisation ist cnlab in der Lage weitere Auswertungen aus den Daten zu erstellen.

Diese Bachelorarbeit ist von der Semesterarbeit “Outdoor Object Tracing“ hergeleitet. Im Rahmen der Semesterarbeit wurde ein Prototyp, basierend auf jQuery Mobile und PhoneGap für Tablets, erstellt. Aufgrund schlechter Performance wurde die Weiterentwicklung des Prototyps gestoppt. Der Grund für die schlechte Performance ist auf PhoneGap und jQuery Mobile zurückzuführen. JQuery Mobile befand sich zu dem Zeitpunkt noch in der RC1 Version und hatte fehlende und nicht optimierte Funktionalitäten. Der zweite und entscheidende Faktor für die schlechte Performance war das Framework PhoneGap. Dieses öffnet im Hintergrund den Browser und stellt die Webseite dar. Dadurch entstand ein Delay von ca. 0.5 Sekunden. Dies verwirrte den Benutzer, da auf seine Interaktion keine direkte Reaktion folgte.

Die Anforderungen sind alle nötigen Funktionalitäten zur Erstellung von Spiellogbüchern umzusetzen. Der Funktionsumfang vom Prototyp sollte übernommen werden. Die Applikation sollte der Öffentlichkeit über den Android Market zur Verfügung gestellt werden. Dafür müssten vorher weitere Usability Tests durchgeführt werden. Dies konnte leider nicht umgesetzt werden.

1.2 Vision

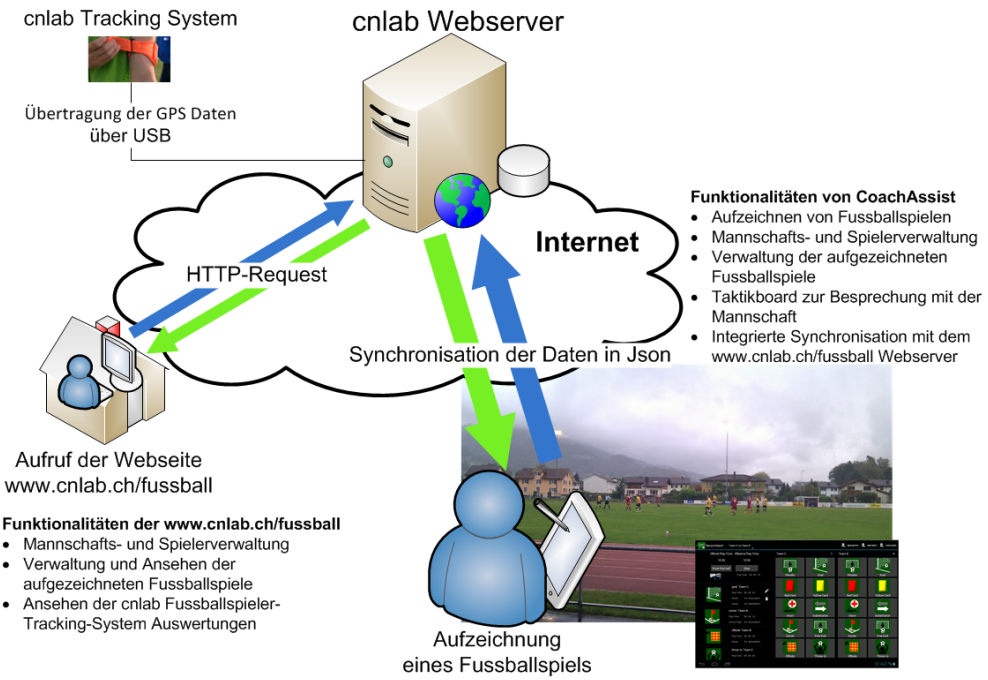
Die Vision dieser Bachelorarbeit ist: “Entwickeln einer Match-Aufzeichnungs Applikation, welche ein einfaches Erfassen von Fussballevent zur Unterstützung des cnlab Tracking-Systems bietet.“ Daraus resultierende Ziele sind:

- Eventerfassung mit Liveticker zur Darstellung
- Team- und Spielerverwaltung
- Matchverwaltung
- Kommunikation mit dem cnlab Server

1.3 Systemübericht

Abbildung 1.1 wird das Zusammenspiel aller Systemkomponenten dargestellt.

Abbildung 1.1: CoachAssist Kontextdiagramm



Wie im Kontextdiagramm ersichtlich, können die Mannschaften, die Spieler und die Spiele verwaltet werden. Trotz der Möglichkeit der Verwaltung auf der Webseite, wurde diese ebenfalls in *CoachAssist* implementiert.
Die Kommunikation läuft über den cnlab Webserver. Die Tablet-Applikation und die cnlab Webseite kommunizieren direkt mit dem Server. Die Daten der GPS Tracking Tags werden im cnlab eingelesen und auf dem Server abgelegt.

2 Applikation

In diesem Kapitel wird die Applikation im Detail beschrieben. Es wird die Graphical User Interface (GUI) betrachtet und die Funktionsliste erläutert.

2.1 Beschreibung

Beim Starten der Applikation erscheint das Menüfenster. Dies ist in der Abbildung 2.1 ersichtlich. Die Idee hinter dem Menü ist, dass der Benutzer schnell und effizient seine Absichten erledigen kann.

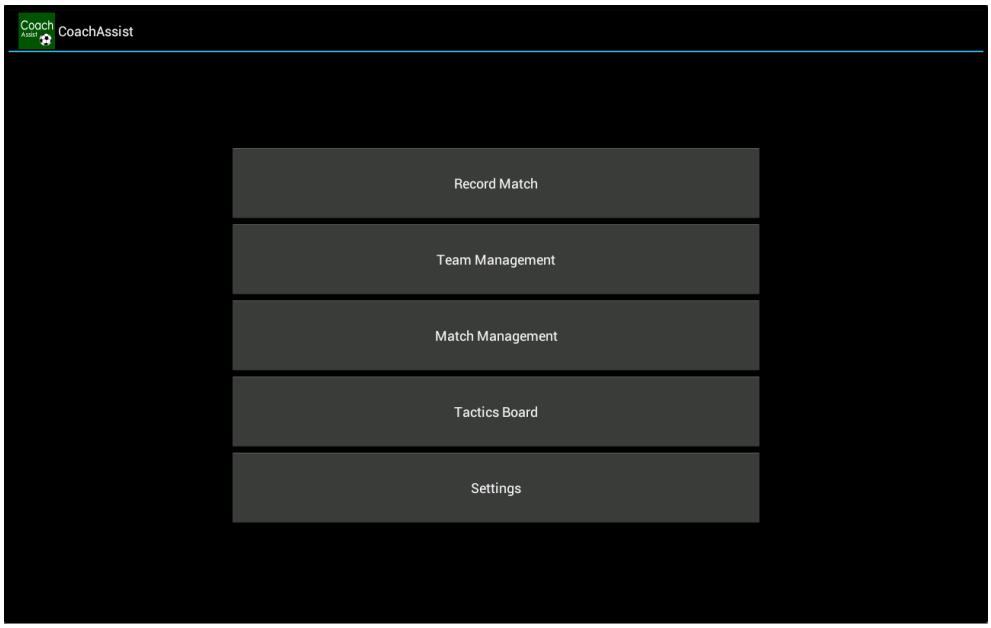
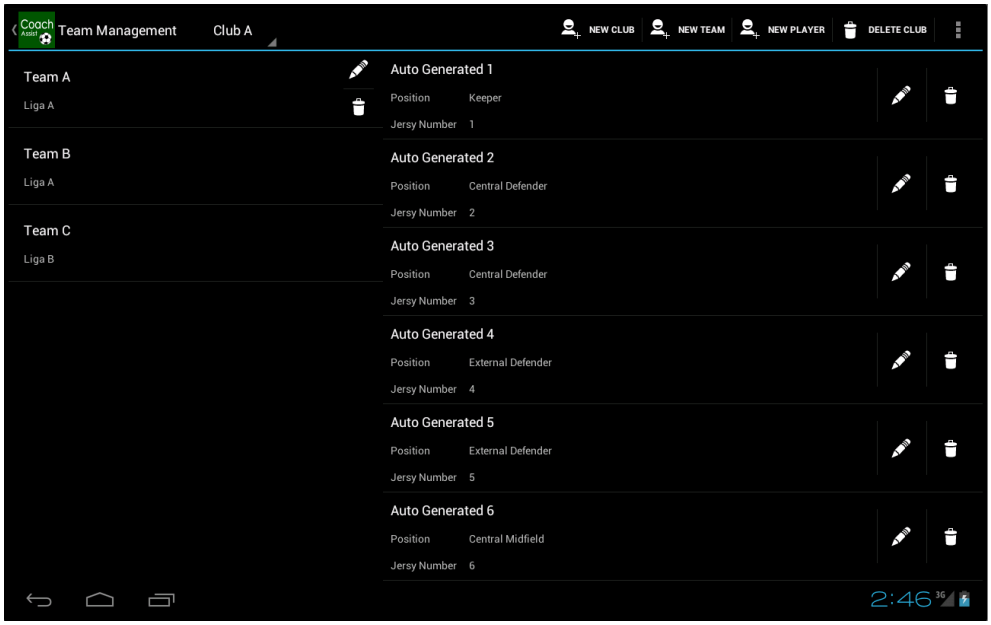


Abbildung 2.1: CoachAssist Main Screen

Im Team Management werden die Klubs, Mannschaften und die Spieler verwaltet. Das dazugehörige User Interface ist in Abbildung 2.2 zu finden. In der Navigationsleiste kann zwischen den erstellten Klubs gewechselt werden. Die Navigationsleiste befindet sich am oberen Rand des Fensters. Auf der linken Seite werden die Mannschaften des Klubs und rechts die Spieler der ausgewählten Mannschaft angezeigt. Bei der selektierten Mannschaft erscheint der Edit und Delete Button. Bei den Mannschaften und Spielern steht die Mehrfachselektion zur Verfügung. Durch die Mehrfachselektion können mehrere

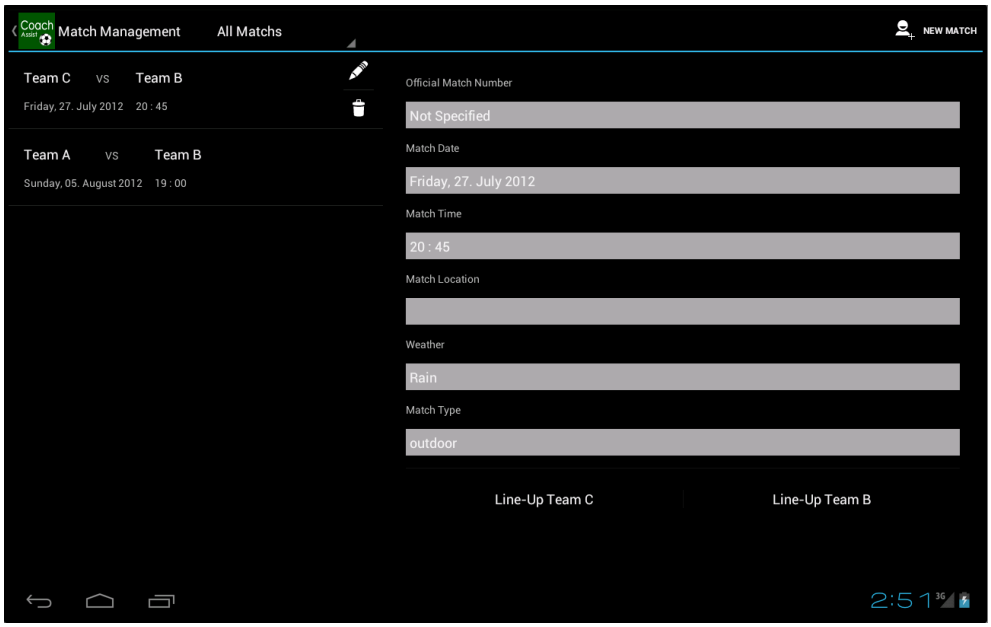
Mannschaften auf einmal gelöscht werden. Bei den Spielern können mehrere Spieler gelöscht oder zu einer anderen Mannschaft transferiert werden.

Abbildung 2.2: CoachAssist Team Management



Im Match Management werden die erstellten Spiele verwaltet. Auf der Abbildung 2.3 ist dieses User Interface zu betrachten. In der Navigationsleiste kann zwischen “Alle Spiele“, “Ausstehende Spiele“ und “Aufgezeichnete Spiele“ gewechselt werden. Auf der linken Seite sind die Spiele, sortiert nach Datum und Zeit, aufgelistet. Beim Anwählen eines Spiels erscheinen rechts die Spielinformationen. Des weiteren besteht die Möglichkeit die Formationen der beiden Mannschaften festzulegen. Bei den Spielen auf der linken Seite steht die Mehrfachselektion auch zur Verfügung. Durch die Mehrfachselektion können mehrere Spiele gelöscht werden.

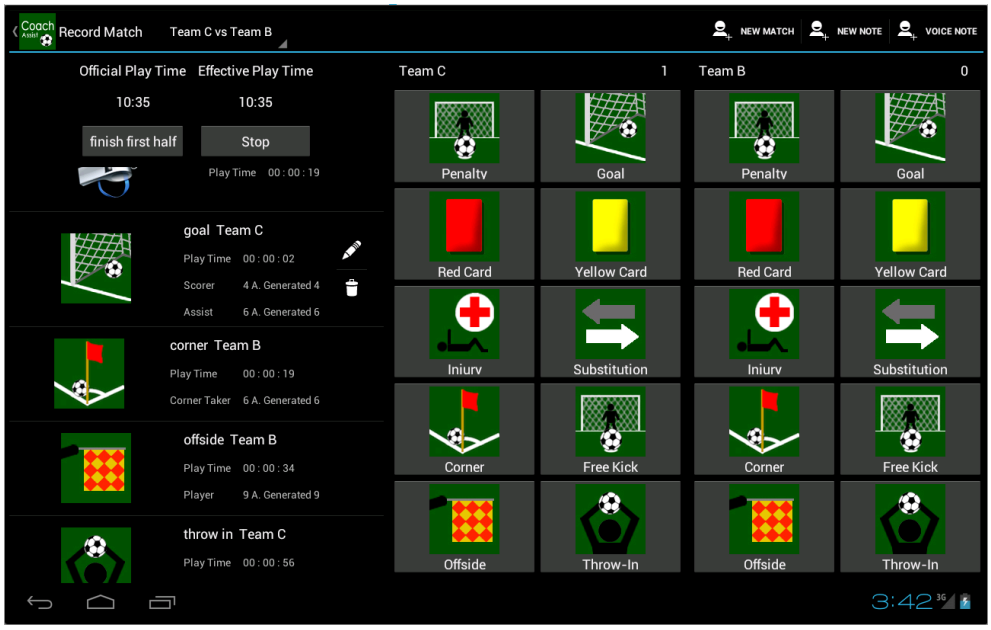
Abbildung 2.3: CoachAssist Match Management



Im Record Match werden die Spiele mit Event Buttons aufgezeichnet. Dies ist auf der Abbildung 2.4 zu sehen. Zusätzlich bietet die Applikation die Mög-

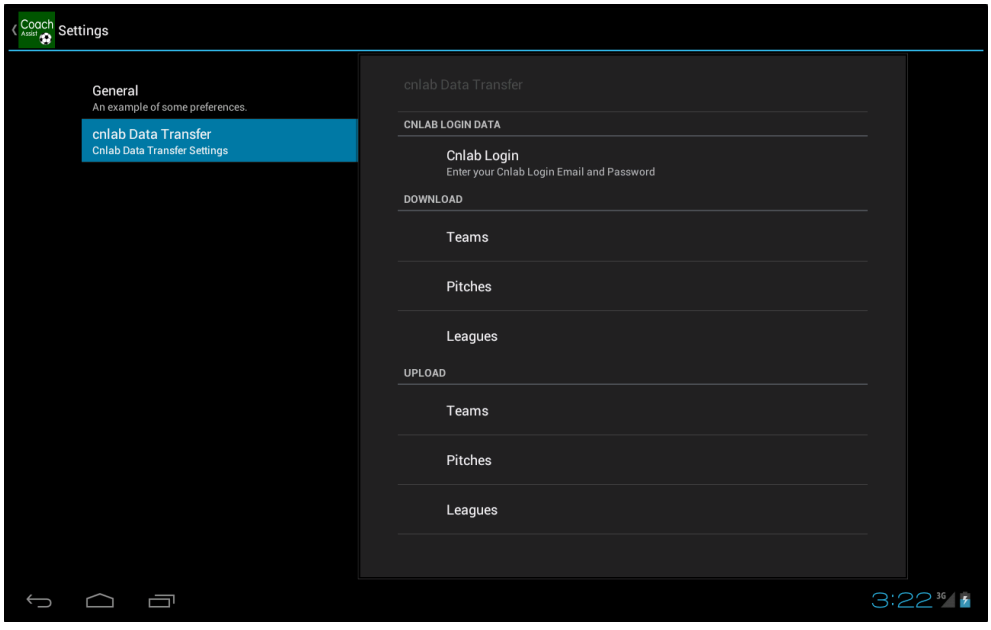
lichkeit, die effektive Spielzeit zu erfassen und Notizen zu erstellen. Das erfasste Event wird im Liveticker hinzugefügt und mit den wichtigsten Details angezeigt. Die Events im Liveticker können nach dem Selektieren, bearbeitet oder gelöscht werden. Beim Erfassen eines Torevent wird der Spielstand entsprechend angepasst.

Abbildung 2.4: CoachAssist Record Match



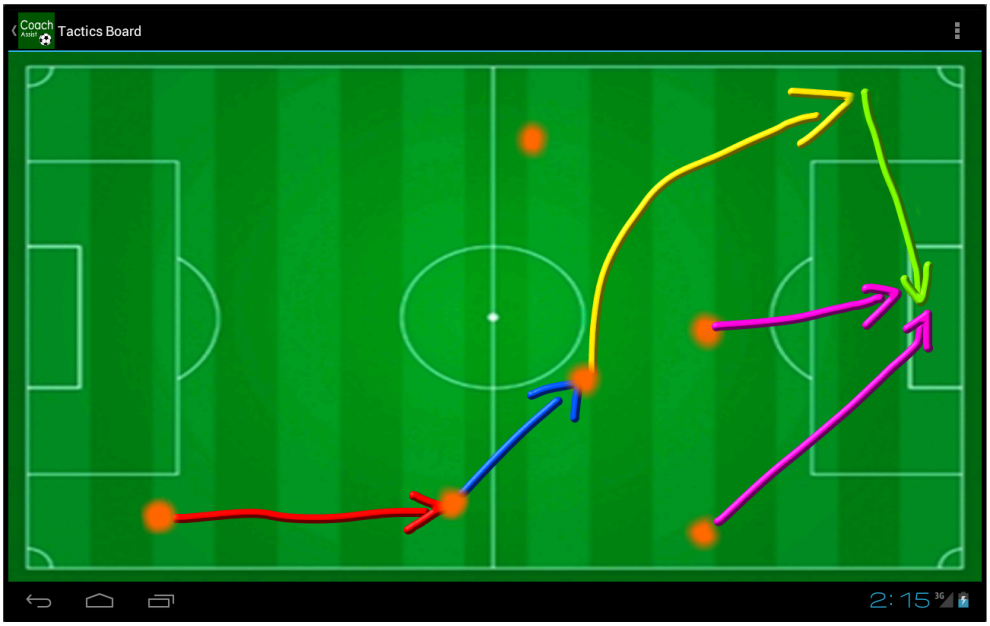
In den Einstellungen (Settings) können die Daten mit dem cnlab Webserver synchronisiert werden. Dafür muss der Benutzer sich einmalig auf der Webseite `htt://www.cnlab.ch/fussball` registrieren. Für die nachfolgenden Schritte muss der Benutzer registriert sein. Anschliessend werden die Login Daten im cnlab-Login eingegeben. Durch dies können Daten hoch- und heruntergeladen werden. Die Spiele, Mannschaften und die Spieler können über den Desktop bzw. dem Browser verwaltet werden. In der Abbildung 2.5 ist diese User Interface dargestellt.

Abbildung 2.5: CoachAssist cnlab Synchronisation



Für die Coaches bietet *CoachAssist* ein Taktikboard zur Besprechung von Taktiken, Spieltechniken oder Standardsituationen. Das Taktikboard ist auf der Abbildung 2.6 ersichtlich. Dieses Feature ermöglicht dem Coach die Besprechung direkt auf dem Fussballplatz abzuhalten.

Abbildung 2.6: CoachAssist Taktikboard



2.2 Funktionsübersicht

Zu Beginn des Projektes wurden alles gewünschten Funktionalitäten notiert und in Prioritäten aufgeteilt. Die folgende Auflistung zeigt die komplette Funktionsliste. Die im *CoachAssist* implementierten Funktionen sind mit einem Haken gekennzeichnet. Die Funktionen basieren auf der Semesterarbeit und der Analyse von vergleichbaren Produkten.

Priorität A Funktionen (muss realisiert werden):

- ☑ Eventerfassung pro Mannschaft
- ☑ Spielerauswahl bei Eventerfassung
- ☑ Erfasste Events, während dem Spiel editierbar
- ☑ Notizen erfassen (Handeingabe)
- ☑ Effektive und offizielle Spielzeit erfassen
- ☑ Anzeige der Spielinformationen (Mannschaftsnamen, Resultat, Torschützen mit Timestamp) während dem Spiel
- ☑ Fortlaufende Liveticker Aktualisierung für notierte Spiel-Events
- ☑ Erfassung der Spielranddaten (Gegner, Spielort, Anpfiffszeit, Spieltyp, Wetter)

- ☑ Erfasste Events (Liveticker) auch zu einem späteren Zeitpunkt abrufbar und editierbar
- ☑ Übersicht über alle erfassten bzw. noch auf dem Tablet vorhandenen Spiele
- ☑ Kommunikation mit dem cnlab Server (Upload, Download und Update)
- ☑ Mehrsprachigkeit (enthalten: DE, EN; mit Hilfe cnlab: IT, FR)
- ☑ Mannschaftsaufstellung, Ersatzbank und Formation festlegen
- ☑ Mannschaft- und Spielerverwaltung auf dem Tablet (aufgrund der Standalone Lösung)
- ☑ Abbildung der cnlab Server Datenbank auf dem Tablet

Priorität B Funktionen (soll realisiert werden, falls Zeit vorhanden):

- ☑ Taktikboard (Zeichnung)
- ☑ Mannschaftsformation selbst definieren
- ☐ Notizen erfassen (Voice-Eingabe)
- ☐ Detail Ansicht der einzelnen Spiele (Spiel-Statistik). Bsp. Anzahl Eckbälle pro Team, Torschützen, Anzahl Freistösse pro Team, etc.
- ☐ Detail Ansicht der einzelnen Spieler (Spieler-Statistik). Bsp. Anzahl Tore über alle Spiele (Saison) hinweg, etc.

Priorität C Funktionen (kann realisiert werden, falls Zeit vorhanden):

- ☐ GPS-Aufnahme des Spielfeldumfangs
- ☐ Bildaufnahmen
- ☐ Videoaufnahmen
- ☐ Kommunikation mit dem cnlab Fussballspieler-Tracking System Server.
- ☐ Darstellung Spielerpositionen und Statistiken pro Spieler vom cnlab Fussballspieler-Tracking System
- ☐ Einlesen von Daten von GPS-Tracks.

2.3 Technologien

Für die Applikation wurden zwei Libraries importiert. ORMLite wurde von Gray Watson entwickelt und Gson von Google. Diese sind Open Source und können frei verwendet werden. Das ORM-Framework wird stark in der Applikation genutzt, da ständig mit der Datenbank kommuniziert wird. Gson wir ausschliesslich bei der Kommunikation zum Server genutzt.

2.3.1 ORMLite

ORMLite ist ein Open Source Framework, welches simples Mappen der Java Klassen auf die SQL Datenbank ermöglicht. Neben der Android SQLite Datenbank wird eine Vielzahl von anderen Datenbanken unterstützt. Das Mapping wird mit Annotationen definiert. Eine detaillierte Erklärung zu den Annotationen ist im Anhang im Kapitel B.2.1 zu finden. Die Grundoperationen für Datenbankabfragen werden vom Framework bereit gestellt. Mit Grundoperationen werden CRUD-Operationen bezeichnet. Einfache Query-Abfragen für den Wert einer Spalte sind ebenfalls vorhanden. Komplexere SQL-Abfragen müssen selbst definiert werden. Dafür steht ein Querybuilder zur Verfügung. Das Erzeugen und Löschen der Datenbanktabelle wird ebenfalls vom Framework übernommen.

2.3.2 Gson

Gson ist eine Java Library, welche verwendet wird um ein Java Objekt in JSON zu konvertieren. Der umgekehrte Weg wird ebenfalls unterstützt. Um die Konvertierung in Java Objekte zu ermöglichen muss die gewünschte Klasse bereits bestehen. Eine detaillierte Beschreibung ist im Anhang im Kapitel B.2.2 zu finden.

2.4 Guidelines

Für die bisherige und weitere Entwicklung der Applikation wird sich nach den Android Guidelines gerichtet. Die Codeguidelines sind unter dem Link `http://source.android.com/source/code-style.html`[15] zu finden. Hinter diesen Guidelines steckt Google. Bei den Field Naming Conventions ist sich nicht komplett nach den Vorgaben zu richten. Die Vorbuchstaben für Variablenbezeichnung “s“ und “m“ werden nicht verwendet. Sonst wird sich komplett nach den Guidelines gerichtet. Eine 100% Übereinstimmung ist nicht verpflichtend. Spezielle Abweichungen werden toleriert.
Für die GUI-Guidelines werden die Informationen von `http://developer.android.com/design/index.html`[16] mit einbezogen. Für das Design werden diese Guidelines als Richtwert gesehen. Das Design wurde und muss nicht komplett nach den Guideline-Vorgaben umgesetzt werden.

3 Software Design

Dieses Kapitel beschreibt das aktuelle Software Design. Es ist speziell an zukünftige Entwickler dieser Applikation gerichtet. Für dieses Kapitel wird ein Grundverständnis von Software Entwicklung vorausgesetzt. Einzelne Teile der Applikation werden detailliert erklärt. Bei den Darstellungsformen handelt es sich vorwiegend um Ausschnitte aus dem Klassendiagramm.

3.1 Steuerung der Business Logik

Die Business Logik beschreibt die Kommunikation von der View mit den Modelklassen. Im *CoachAssist* wird das GUI Design Pattern Model-View-Presenter angewendet. MVP ist abgeleitet von MVC Pattern. Im MVP verarbeitet die View die GUI Events, nicht der Controller wie beim MVC. Im MVP sind View und Model komplett voneinander getrennt. Die Kommunikation findet lediglich über den Presenter statt. Die Abbildung 3.1 zeigt den Aufbau dieses Patterns.

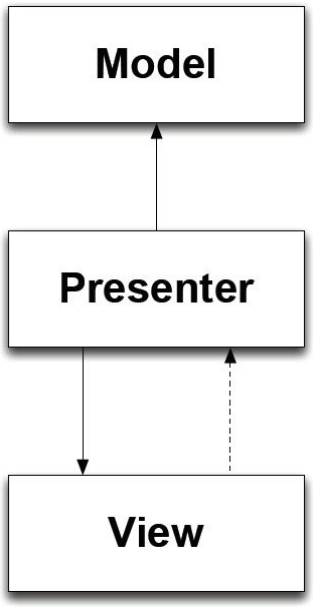


Abbildung 3.1: GUI Design Pattern Model-View-Presenter [20]

Die MVP ist im *CoachAssit* wie folgt umgesetzt: Die Activity, die Fragments

und Dialogs übernehmen die Rolle des Presenters und das XML-Layouts bildet die View. Der Model Layer beinhaltet die Modellklassen und die Value Objects.

Die Zugriff der View auf die Modelklasse verläuft über die Activity. Zwischen der Activity und den Modelklassen ist ein Model Controller implementiert, da gemäss Android Development Guide die Modellklassen Singleton sein müssen. Die Begründung kann im folgendem Link `http://developer.android.com/resources/faq/framework.html`[7] nachgelesen werden.

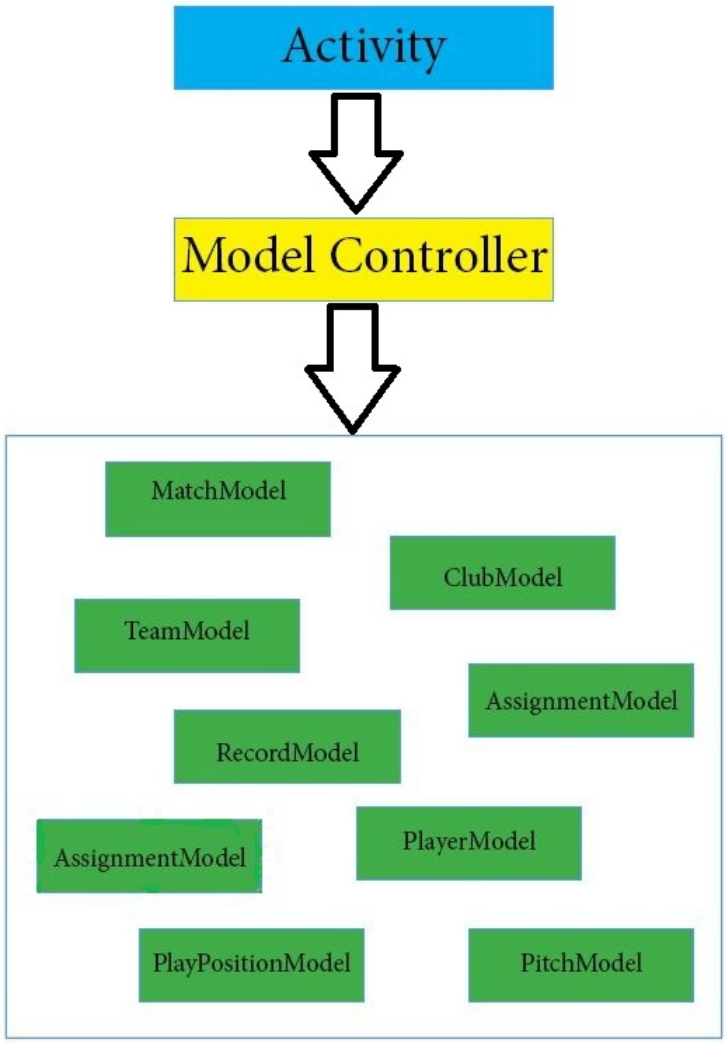


Abbildung 3.2: Model Controller als Kommunikationsschnittstelle zu allen Modelklassen

Durch das Einbinden vom Model-Controller müssen die Modellklassen nicht mehr Singleton sein und die Kopplung zwischen den Activities und den Modellklassen wird verringert. Der Model-Controller ist Singleton und verwaltet die Instanzen der Modelklassen. Die Activities erhalten vom Controller eine Referenz der benötigten Modellklasse. In der Abbildung 3.2 ist dieser Vorgang dargestellt.

3.2 Schichten Architektur

In diesem Kapitel wird aufgezeigt, welche Klassen sich in welcher Schicht in der Schichtenarchitektur befinden. Die Abbildung 3.3 soll einen Überblick über den Aufbau der Applikation verschaffen. Es wird eine Vier-Schichten-Architektur verwendet. In den einzelnen Schichten sind jeweils die darin befindenden Klassen-Typen aufgelistet.

Abbildung 3.3: CoachAssist Schichten Architektur

Die Präsentationsschicht beinhaltet Klassen, welche für die GUI zuständig sind. Im XML Layout sind die Benutzeransicht-Templates festgelegt. Die Applikationslogik übernehmen die Activitiy Klassen. Für den Datenzugriff erfolgt eine Abfrage von der Modelklassen aus. Dies mit Hilfe des OrmLite Framework. Die Persistentsschicht umfasst die *CoachAssist* Datenbank Tabellen.

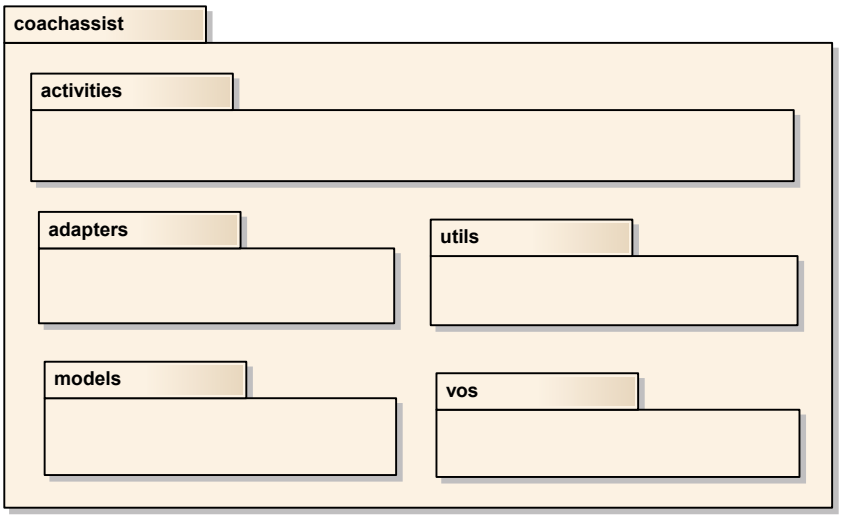
3.3 Logische Architektur

Als erstes wird die gesamte Package-Struktur erläutert. Daraufhin folgt eine Beschreibung der Haupt-Packages.

3.3.1 Package Diagramm

Abbildung 3.4 zeigt eine Übersicht über die Haupt-Packages, welche in der Applikation definiert sind. Im Activities Package sind Klassen definiert, welche für den Aufbau der View zuständig sind. Die Kommunikation mit den XML-Views erfolgt aus diesem Package. Hierbei handelt es sich um Logik, welche die Views updatet und welche auf die Interaktionen des Benutzers reagiert.

Abbildung 3.4: Package-Übersicht

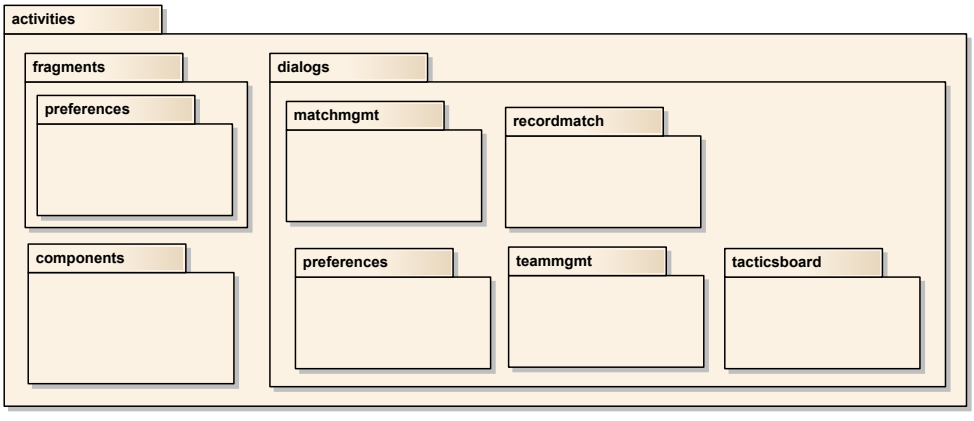


Das Package Adapters beinhaltet selbst erstellte Adapter, welche in den Activities verwendet werden. Diese Adapter werden benötigt um Elemente in ListViews darzustellen.
Das Models-Package beinhaltet Model Klassen, welche in den Activities benötigt werden um Daten aus der Datenbank auszulesen und Daten zu erzeugen. Das Package Value Object (VO) ist eng mit diesem verbunden. Im VO-Package sind die Objekte enthalten, welche in die Datenbank abgelegt werden. Diese Objekte werden im Model erzeugt, aktualisiert oder gelöscht.
Das letzte Package Utils beinhaltet Klassen, welche zum Beispiel für den Verbindungsaufbau zum Server oder zur Datenbank genutzt werden. Sie unterstützen das Model bei spezifischen Operationen.

3.3.2 Activities

Das Package Activities ist in mehrere Subpackages unterteilt, wie in der Abbildung 3.5 zu erkennen ist. Ein Fragment ist ein Teil eines User Interface in einer Activity. Die Activity ist in Teilbereiche aufgeteilt. Das Package Components beinhaltet selbst erstelle GUI-Elemente (Custom Views). Diese erben von View Elementen und sind mit spezifischem Code erweitert. Package Dialogs beinhaltet Dialoge. Dieses Package wurde in die fünf View Gruppen, aus der Menüsicht unterteilt.

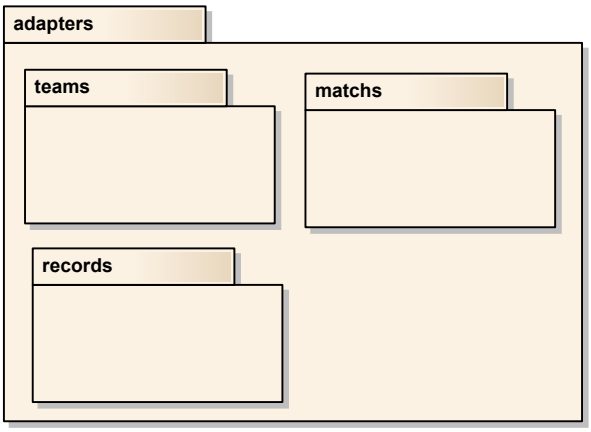
Abbildung 3.5: Package Activities



3.3.3 Adapters

Das Package Adapters beinhaltet Adapter-Klassen. Adapter sind in Android Darstellungen einer Listenansicht. Sie verwalten eine Liste von Elementen und sind für deren Darstellung verantwortlich. In der Abbildung 3.6 ist ersichtlich, dass das Package in drei Subpackages unterteilt ist. Diese Aufteilung ist wegen den drei Haupt-Activities zustande gekommen.

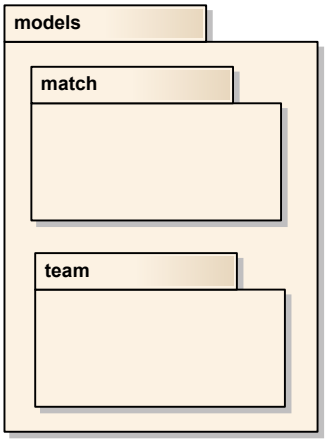
Abbildung 3.6: Package Adapters



3.3.4 Models

Das Model Package beinhaltet die Model-Klassen. Diese Klassen sind für die Kommunikation zwischen den Activities und der Datenbank vorgesehen. Eine Model-Klasse kann je nach dem Listen, Erstellungsmethoden oder Bearbeitungsmethoden von VOs beinhalten. Auf die Beziehung zwischen Model-Klassen, VOs und Datenbank wird im Kapitel 3.5 eingegangen. Die Abbildung 3.7 zeigt die Aufteilung in zwei Subpackages. Diese Aufteilung ist später auch beim Vos Package zu sehen. Das Subpackage Formations wurde bewusst weggelassen, da es nicht den gleichen Detaillierungsgrad hat, wie die anderen zwei Packages. Der Grund für eine ähnliche Aufteilung ist, dass eine Model-Klasse immer mit einem VOs zusammen arbeitet.

Abbildung 3.7: Package Models

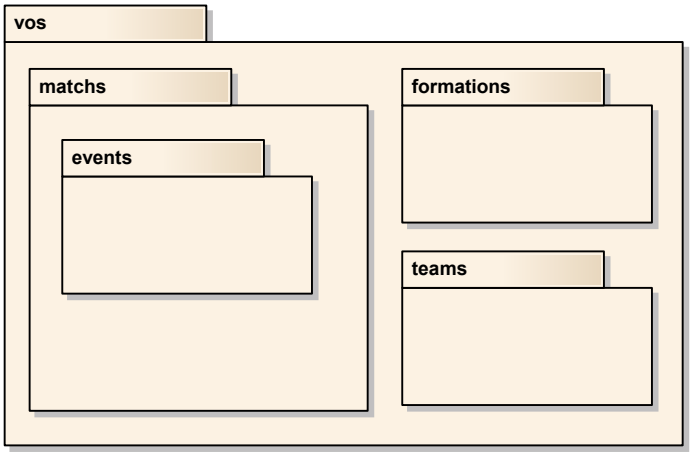


3.3.5 Values Objects (VOs)

Das VOs Package enthält Klassen, welche in die Datenbank gemapped werden. Für das Mapping wird ein ORM-Framework verwendet. Für dieses Packa-

ge hat sich die Aufteilung in drei Subpackages nach Zugehörigkeit ergeben. Diese Aufteilung ist in der Abbildung 3.8 zu betrachten.

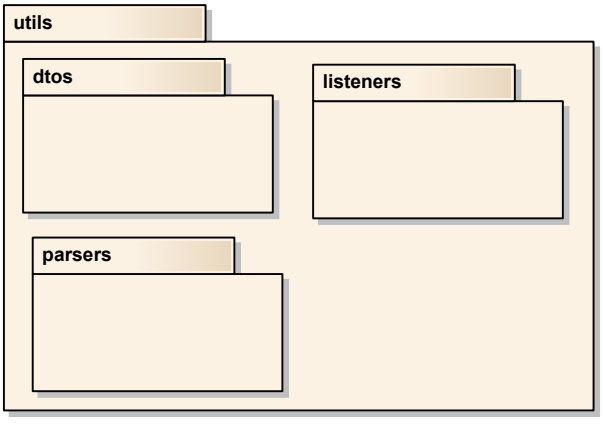
Abbildung 3.8: Package Vos



3.3.6 Utils

Das Utils Package beinhaltet Klassen, welche für spezifische Operationen benötigt werden. Eine wichtige Klasse ist der “DatabaseHelper“. Er ist für die Erzeugung der Datenbank und deren Verwaltung zuständig. Eine weitere wichtige Klasse ist der “TransferHelper“. Dieser sendet und empfängt die Request vom cnlab Server. Dieses Package enthält Helper und Utils, welche für die Applikation benötigt werden. In Abbildung 3.9 ist ersichtlich, dass sich drei Subpackages ergeben haben. Das erste Subpackage Data Transfer Object (DTO) beinhaltet Klassen, welche für das Wiederherstellen der empfangenen Daten im JSON-Format verantwortlich sind. Das zweite Package Parsers enthält zwei Parser. Diese sind für die Erstellung von VOs in DTOs und umgekehrt zuständig. Im letzte Package Listeners befinden sich selbst definierte Listener-Klassen.

Abbildung 3.9: Package Utils



3.4 Interaktion des User Interfaces

Dieses Kapitel erklärt die Interaktion der implementierten Benutzeroberflächen. Dabei wird nur auf die wichtigsten Widgets eingegangen.

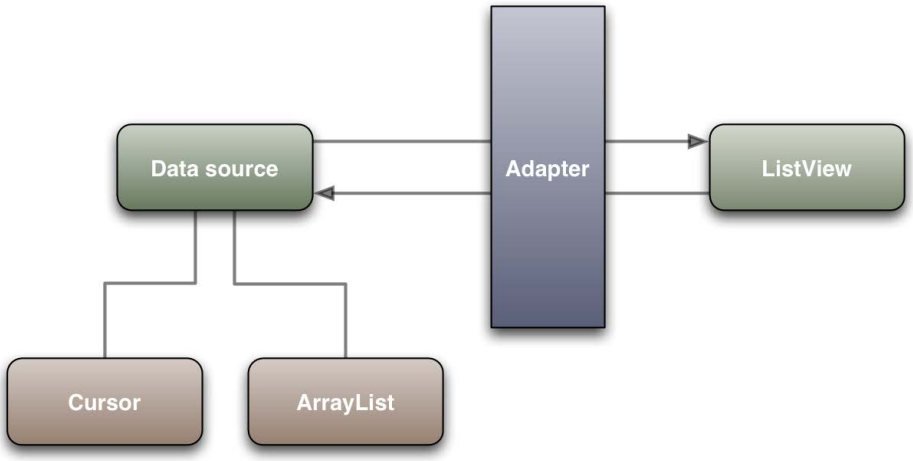
3.4.1 Korrekte Implementierung der ListView

Die ListView ist ein häufig genutztes Widget. Sie vereinfacht das Abfüllen und das Darstellen von Listen. In diesem Unterkapitel wird die korrekte Implementierung der ListView anhand eines konkreten Beispiels in *CoachAssit* erläutert. Durch die korrekte Implementierung der Listview kann die Performance optimiert werden. Zugleich wird der Speicherverbrauch auf das Minimum reduziert. Für detaillierte Informationen wird auf die folgenden zwei Links verwiesen: `http://www.google.com/events/io/2010/sessions/world-of-listview-android.html`[13] und `http://www.google.com/events/io/2009/sessions/TurboChargeUiAndroidFast.html`[14]. Die nachfolgenden Informationen bauen auf den beiden Links auf.

Funktionsweise des Adapters in Android

Die Abbildung 3.10 stellt die Implementierung der ListView in Android visuell dar. Zwischen der ListView und dem Model wird ein Adapter zwischengeschaltet. Dieser sorgt dafür, dass die ListView Items korrekt dargestellt werden.

Abbildung 3.10: UML Diagramm der Android ListView Implementierung [4]

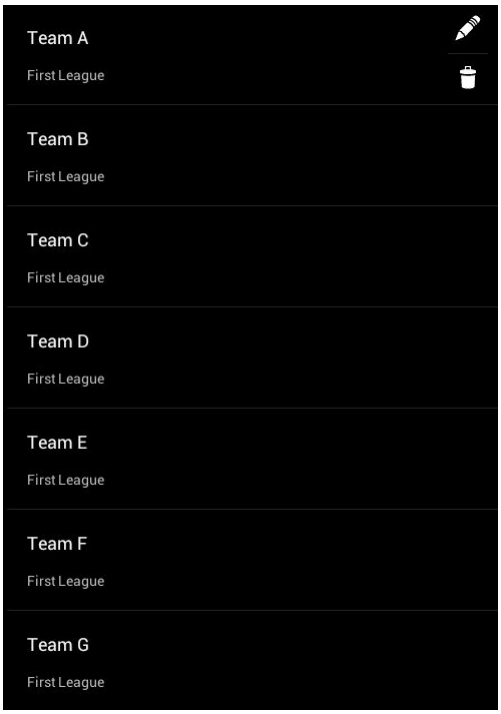


Implementierungsbeispiel des Adapters im *CoachAssist*

Als Beispiel dient die Implementierung der ListView im Team Management. Der “TeamListAdapter“ muss zwei Typen von Items managen. Beim ersten Typ werden nur Informationen angezeigt. In diesem Beispiel sind das der Mannschaftsname und die Liga. Beim zweiten Typen sind zusätzlich die Buttons “Edit“ und “Delete“ sichtbar. Dieser Typ wird nur beim Auswählen eines

Items gerendert. In der Abbildung 3.11 ist Team A ausgewählt dh. es ist vom Typ 2 und die restlichen Items sind vom Typ 1.

Abbildung 3.11: ListView mit mehreren Typen



Der Grund für den Aufwand dieser Typenimplementierung ist wie folgt: Angenommen die Typen sind nicht im Adapter definiert, dann werden die Buttons beim Scrollen bei Team H wieder sichtbar, obwohl sie nicht selektiert wurden. Durch Klicken auf den Delete Button beim Team H wird das Team A gelöscht, da die Buttons ursprünglich für Team A aktiviert wurden. Der Grund liegt darin, dass die Buttons standardmässig unsichtbar sind. Erst beim Selektieren werden sie auf sichtbar gesetzt. Der Adapter ändert, wie definiert, nur den Mannschaftsnamen und die Liga. Die Buttons werden nicht angerührt.

Die Implementierung der Methode “getView“ ist im Codeausschnitt 3.1 im “TeamListAdapter“ ersichtlich. Laut Empfehlung von Android Developers `http://www.android-developer.com`[6] wird für die Optimierung der Performance empfohlen, die “convertView“ zu verwenden, falls diese nicht “null“ ist. Zusätzlich wird das “ViewHolder“-Pattern verwendet. Dieses Pattern vermeidet, dass “findViewById“ bei jedem Rendern eines Item aufgerufen wird.

```
1  @Override
2    public View getView(int position, View convertView,
       ViewGroup parent) {
3
4    ViewHolder holder;
5
6    int type = getItemViewType(position);
7    if ( convertView == null ) {
8
9      holder = new ViewHolder();
10     convertView = inflater.inflate(R.layout.list_teams,
         null);
```

```
11
12      if ( type == VIEW_TYPE_SELECTED ) {
13        ViewStub stub = (ViewStub)convertView.findViewById(R
              .id.listTeams_viewStub_imageButtonLayout);
14        View view = stub.inflate();
15
16        holder.buttonEdit = //Definition des Edit Buttons
              und Zuweisung des Listeners
17        holder.buttonDelete = //Definition des Delete
              Buttons und Zuweisung des Listeners
18      }
19      holder.teamName = (TextView) convertView.findViewById(
            R.id.listTeams_textView_teamName);
20      holder.leagueName = (TextView) convertView.
            findViewById(R.id.listTeams_textView_league);
21      convertView.setTag(holder);
22    }
23    else {
24      holder = (ViewHolder) convertView.getTag();
25    }
26    Team team = teams.get(position);
27    if ( team != null ) {
28      //Setzen der Werte im Objekt Team
29    }
30
31    return convertView;
32  }
33
34  static class ViewHolder {
35    TextView teamName;
36    TextView leagueName;
37    ImageButton buttonEdit;
38    ImageButton buttonDelete;
39  }
```

Codebeispiel 3.1: Implementation der “getView“-Methode

Die Informationen für diese Umsetzung des Codes sind auf der Webseite `http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/view/List14.html`[6] einzusehen.

Bei der Benutzung von verschiedenen Item Typen müssen die zwei Methoden “getItemViewType“ und “getItemViewType“ in der Adapterklasse überschrieben werden. Durch die “getItemViewType“-Methode wird überprüft, von welchem Typ das zu generierende Item ist. Die Unterscheidung der zwei Typen erfolgt über “VIEW_TYPE_UNSELECTED“ und “VIEW_TYPE _SELECTED“. Falls ein Item selektiert wird, wird über die Methode “setSelectedTeamPosition“ die Position des selektierten Items in der Adapterklasse abgespeichert. Beim Aufruf der “getItemViewType“-Methode wird der Parameter “Position“ mit der Variable “selectedTeamPosition“ verglichen und der entsprechende Typ wird zurückgeliefert.

```
1  private final int VIEW_TYPE_UNSELECTED = 0;
2  private final int VIEW_TYPE_SELECTED = 1;
3  private int selectedTeamPosition = 0;
4
5  @Override
```

```
6  public int getItemViewType(int position) {
7    if ( position == selectedTeamPosition ) {
8      return VIEW_TYPE_SELECTED;
9    }
10   return VIEW_TYPE_UNSELECTED;
11 }
12
13 public void setSelectedTeamPosition(int position) {
14   selectedTeamPosition = position;
15 }
```

Codebeispiel 3.2: Implementation der “getItemViewType“-Methode

Die Methode “getItemViewType“ dient der “getView“-Methode die Anzahl Typen zu ermitteln und den Recycler zu generieren. Da wie erwähnt im Team ListView zwischen zwei Typen unterschieden wird, gibt die “getItemViewType“-Methode zwei zurück.

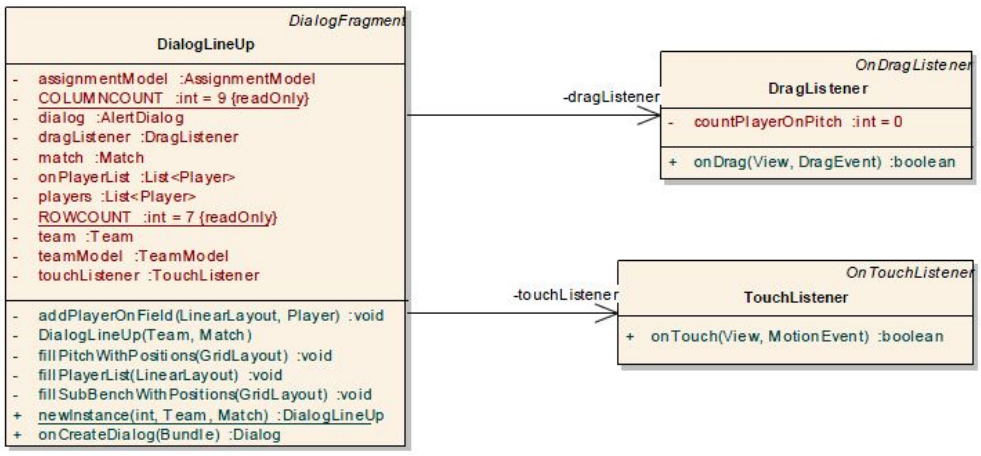
```
1  private final int VIEW_TYPE_COUNT = 2;
2
3  @Override
4  public int getItemViewType() {
5    return VIEW_TYPE_COUNT;
6  }
```

Codebeispiel 3.3: Implementation “getViewTypeCount“-Methode

3.4.2 Mannschaftsaufstellung mit Drag and Drop

Für Drag and Drop werden zwei Listener benötigt. Der erste ist ein “onDrag-Listener“ und der zweite ist ein “onTouchListener“. Alle Felder, auf denen das Drag-Objekt abgelegt werden kann, müssen einem “onDragListener“ zugewiesen sein. Alle Objekte, die per Drag and Drop verschoben werden können, müssen statt dessen einem “onTouchListener“ zugewiesen sein.

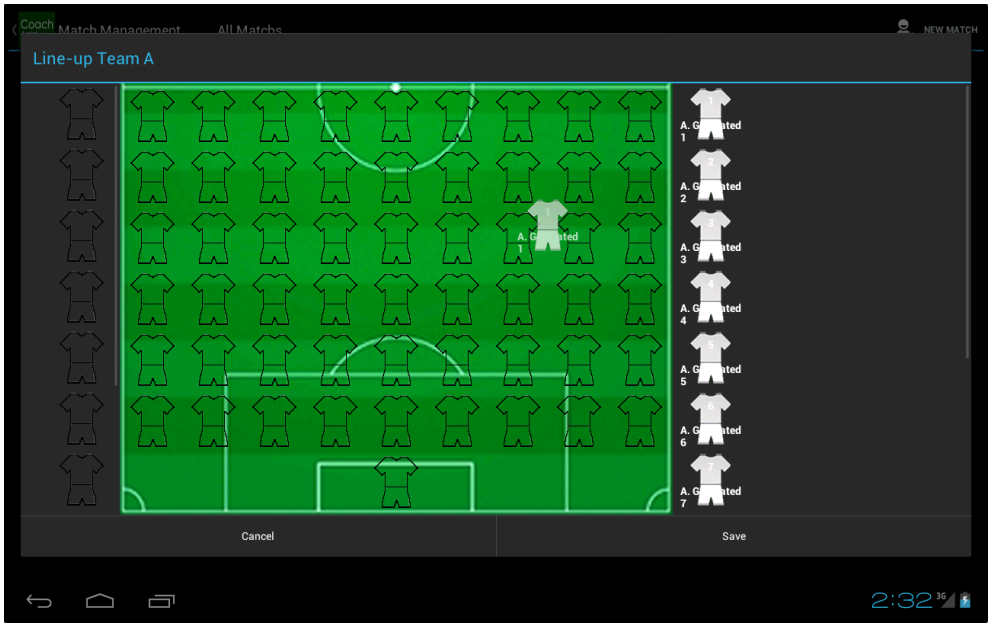
Abbildung 3.12: Klassendiagramm der Drag and Drop Implementierung



Es wurde eine Klasse “DragListener“ erstellt, die den “onDragListener“ implementiert. Ebenfalls wurde eine Klasse “TouchListener“ erstellt, die den “onTouchListener“ implementiert. Beide Klassen sind mit der Klasse “DialogLineUp“ verbunden. In dieser Klasse werden “DragListener“ und “TouchListener“ instanziiert und dem Objekt zugewiesen. Die Abbildung 3.12 zeigt, wie diese zwei Listener der Klasse zugewiesen sind.

Im “TouchListener“ ist die “onTouch“-Methode überschrieben. Diese Methode legt fest, was beim Berühren des Drag-Objekts geschieht. In der Abbildung 3.13 ist ersichtlich, wie die “onTouch“-Methode definiert ist. Diese bewirkt, dass ein Schatten des Drag-Objekts dem Finger folgen soll.

Abbildung 3.13: CoachAssist Drag and Drop Screenshot



Im “DragListener“ wird die “onDrag“-Methode überschrieben. Diese Methode legt fest, wie die “DragEvents“ abgehandelt werden. Im Folgenden werden die “DragEvents“ erklärt und auf ihre Implementierung im *CoachAssist* eingegangen.

- *DragEvent.ACTION_DRAG_STARTED*:
  Dieses Event legt das Geschehen beim Start des Events fest. In *CoachAssist* werden alle möglichen und freien Drop-Felder auf dem Spielfeld und der Ersatzbank eingeblendet.

- *DragEvent.ACTION_DRAG_ENTERED*:
  Beim Ausführen dieses Events wird definiert, was angezeigt werden soll, wenn das Drag-Objekt in ein Drop-Feld gezogen wird. Dieses Event wurde nicht in *CoachAssist* verwendet.

- *DragEvent.ACTION_DRAG_EXITED*:
  Dieses Event definiert, was angezeigt werden soll, wenn beim Ziehen eines Drag-Objekt, ein Drop-Feld verlassen wird. Dieses Event wurde nicht in *CoachAssist* verwendet.

- *DragEvent.ACTION_DROP*:
  Die Ausführung dieses Events beschreibt, was beim Ablegen des Drag-Objekt geschehen soll. Im *CoachAssist* wird die Anzahl Spieler auf dem Spielfeld überprüft. Solange die Anzahl Spieler auf dem Spielfeld kleiner als elf ist, können Spieler auf dem Spielfeld platziert werden. Sobald die Grenze erreicht ist, können keine weiteren Spieler hinzugefügt werden. Es muss zuerst ein Spieler vom Spielfeld entfernt werden, damit

ein anderer hinzugefügt werden kann. Die maximale Anzahl Ersatzbankspieler ist auf zehn begrenzt.

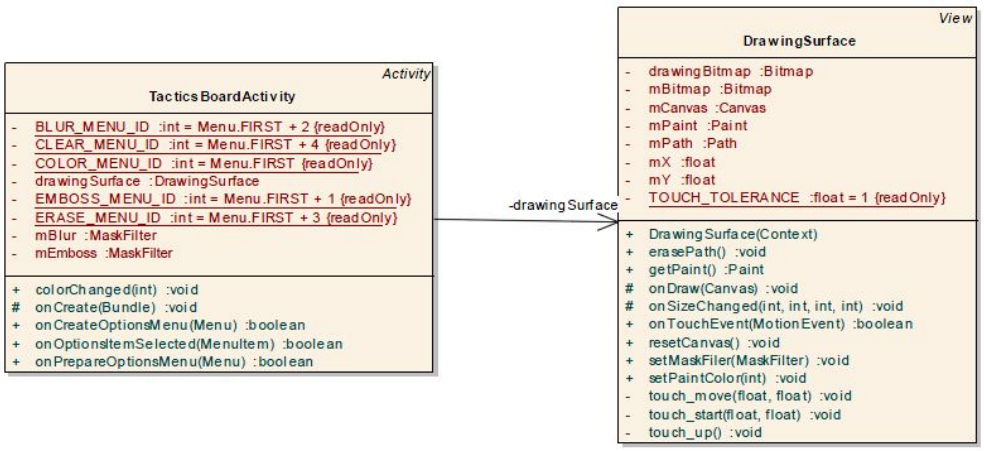
- *DragEvent.ACTION_DRAG_ENDED*:
  Dieses Event legt das Geschehen beim Beenden des Vorganges fest. In *CoachAssist* werden alle sichtbaren Drop-Felder ausgeblendet.

- *DragEvent.ACTION_DRAG_LOCATION*:
  Dieses Event definiert das Ablegen des Drag-Objektes. Ebenfalls wird festgelegt, was geschieht, wenn es sich nicht um Drop-Feld handelt. Im *CoachAssist* wird, falls man das Drag-Objekt auf ein Non-Drag-Feld ablegt, nicht verschoben.

3.4.3 Taktikboard mit Fingerpainting

Die Taktikboard View wird ausschliesslich im Code erzeugt. Dh. es wird nicht auf eine XML-Datei zurückgegriffen. Für das Zeichnen wird lediglich ein Canvas benötigt. Die View wird mit der Klasse “DrawingSurface“ gerendert. Die “DrawingSurface“ Klasse erbt von der Android Klasse “View“. Dafür müssen die Methoden “onDraw“ und “onTouchEvent“ überschrieben werden, welche für das Fingerpainting verantwortlich sind. In der Abbildung 3.14 wird diese Klasse dargestellt.

Abbildung 3.14: Klassendiagramm von Fingerpainting Implementierung



Die “onTouchEvent“-Methode definiert die Handhabung der Bewegung des Fingers. Es stehen drei “MotionEvents“ zur Verfügung. Im Folgendem werden die Events erklärt und auf ihre Implementierung in *CoachAssist* eingegangen.

- *MotionEvent.ACTION_DOWN*
  Dieses Event legt das Geschehen am Anfang der Touch-Bewegung fest bzw., wenn der Finger den Screen berührt. In *CoachAssist* wird die Pfadinstanz zurückgesetzt. Die Pfadinstanz wird zum Festhalten des Weges der Touch-Bewegung benötigt.

- *MotionEvent.ACTION_MOVE*
  Dieses Event definiert den Vorgang beim Bewegen des Fingers. In der Applikation wird die X- und Y-Position der Pfadinstanz hinzugefügt. Der Pfad wird virtuell sichtbar gemacht.

- *MotionEvent.ACTION_UP*
  Dieses Event definiert den Vorgang beim Beenden der Touch-Bewegung. In *CoachAssist* wird die Pfadinstanz dem Canvas übergeben und die Zeichnung auf dem Canvas persistiert.

Abbildung 3.15 zeigt den Vorgang des Zeichnens einer Linie.

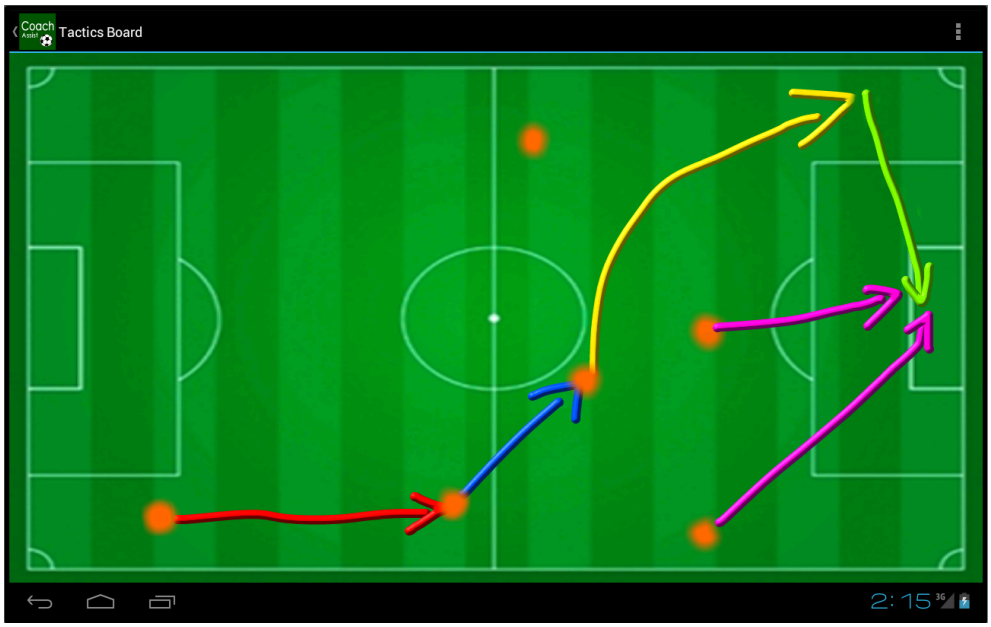


Abbildung 3.15: CoachAssist Taktikboard Screenshot

Um das Fingerpainting mit einem Fussballfeld als Hintergrund zu realisieren, musste eine zweite Canvas-Schicht für das Zeichnen implementiert werden. Ansonsten wäre das Feature radieren nicht möglich gewesen. Beim Event *TOUCH_UP* wird die Zeichnung auf dem Canvas persistent übernommen. Falls beim Canvas ein Hintergrundbild definiert ist, wird die Zeichnung direkt auf das Hintergrundbild übernommen. Deswegen wurde ein zweiter Canvas über dem Ersten hinzugefügt. Beim ersten Canvas wurde das Fussballfeld als Hintergrundbild definiert und beim Zweiten der Hintergrund transparent eingestellt.

3.5 Kommunikation der Datenbank

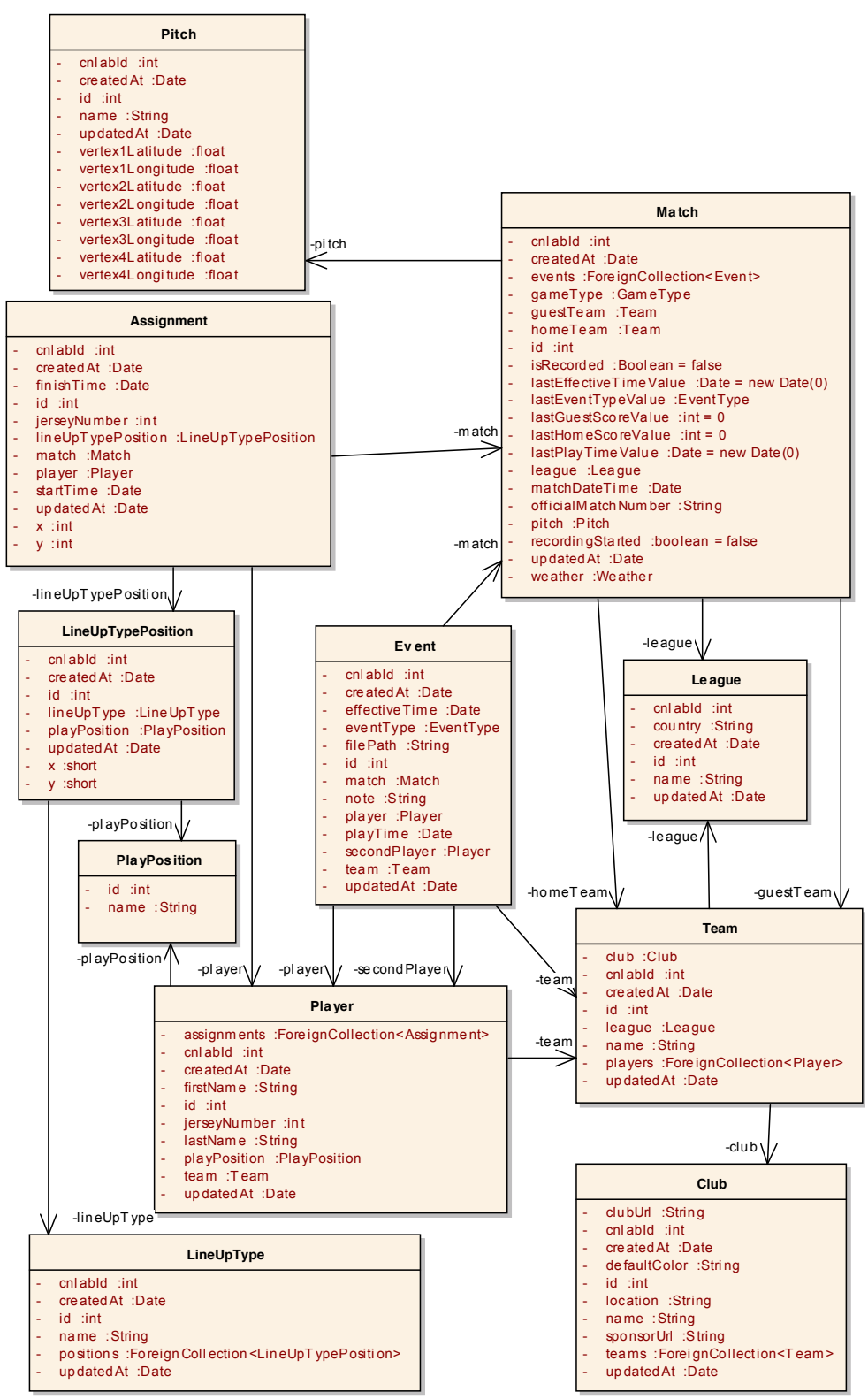
Bei der verwendeten Datenbank handelt es sich um eine SQLite Datenbank. Die Datenbank wird mit Hilfe eines ORM-Framework via Java Objekte erstellt. Das verwendete Framework nennt sich ORMLite. Das Mapping erfolgt mit Annotationen. Eine detaillierte Erklärung zu diesem Tool ist im Anhang im Kapitel B.2.1 zu finden. Die Objekte werden in der Applikation als VOs bezeichnet. Diese Objekte werden in den Modelklassen instanziiert und mit DTO in die Datenbank gespeichert, herausgelesen oder gelöscht.

3.5.1 Übersicht

Die Abbildung 3.16 zeigt ein Klassenmodel. Diese Klassen werden mit den passenden Annoationen in die Datenbank gemapped. Aus diesem Grund ist dieses Klassenmodel auch indirekt das Datenbankmodel. Im Allgemeinen ist

zu erwähnen, dass die Klassen, welche mit der cnlab Datenbank synchronisiert werden, drei zusätzliche Attribute haben. Diese Attribute heissen “cnlabId“, “createdAt“ und “updatedAt“. Diese Attribute werden benötigt um die Verknüpfung zur cnlab Datenbank zu gewährleisten.

Abbildung 3.16: Datenbankmodel



Match

Die Klasse Match enthält spielrelevante Attribute über den Stand des Spiels. Neben diesen sind zusätzliche Attribute definiert, welche eine Wiederherstellung eines Spiels ermöglichen. Zum Beispiel die Wiederherstellung eines Spiels, welches zum Teil aufgezeichnet, aber durch ein unerwartetes Ereignis unterbrochen wurde. Eine Aktualisierung dieser Wiederherstellungsdaten erfolgt, wenn ein Event im Spiel aufgezeichnet wird.

Pitch

Bei der Klasse Pitch wird das Attribut “name“ verwendet. Ein Pitch stellt das Spielfeld dar, auf dem das Match statt findet. Die acht Vertex Attribute sind für die Koordinaten. Diese werden mit der Implementierung des Features “Einlesen der Koordinaten“ verwendet.

League

Die League Klasse ermöglicht die Aufteilung von Fussballspielen oder Teams in die passenden Stufen. Die Leagues sind nach Ländern aufgeteilt. Das League-Feld ist beim Team aber auch beim Match ein optionales Feld.

Team und Club

Die Klasse Team ist einem Club zugewiesen. Ebenfalls enthält diese Klasse eine Liste von dazugehörigen Spielern. Club ist ein Zusammenschluss von Teams.

Player

Die Klasse Player enthält alle Informationen zu einem Spieler. Sie besitzt eine Standardspielposition und eine Standardtrikotnummer. Diese zwei Attribute sind aber nicht bei jedem Spiel verpflichtend. Sie können auch von der Assignment-Zuweisung überschrieben werden.

Event

Die Klasse Event beinhaltet je nach EventTyp spezielle Attribute. Die einzigen Pflichtattribute sind Match, EventTyp und PlayTime. Die anderen zu definierenden Attribute werden durch den EventTyp bestimmt.

Assignment

Das Assignment wird für die Teilnahme eines Spielers an einem Spiel benötigt. Ein Assignment hat eine Start- und Endzeit. Bei einem Spielerwechsel werden vier Assignments verändert. Zum einen die aktuellen zwei, welche

gerade am Laufen sind und zwei Neue. Die aktuellen werden mit der Endzeit versehen und es die zwei Neuen mit der Startzeit. Folgende Events können ebenfalls eine Assignment-Änderung nach sich ziehen:

- Rote Karte
- Zwei Gelbe Karten
- Spielerwechsel

PlayPosition

Die Klasse PlayPosition beinhaltet Spielerpositionen. Diese Positionen können einem Player oder einer LineUpTypePosition zugeteilt werden. Die LineUpTypePosition überschreibt die Player-Zuweisung während dem Match selbst.

LineUpType und LineUpTypePosition

Diese zwei Klassen definieren eine Formation. LineUpType besitzt elf LineUpTypePositions. Diese Aufstellungsmöglichkeit ist nicht komplett implementiert. Die Klasse wurde bereits dafür vorbereitet.

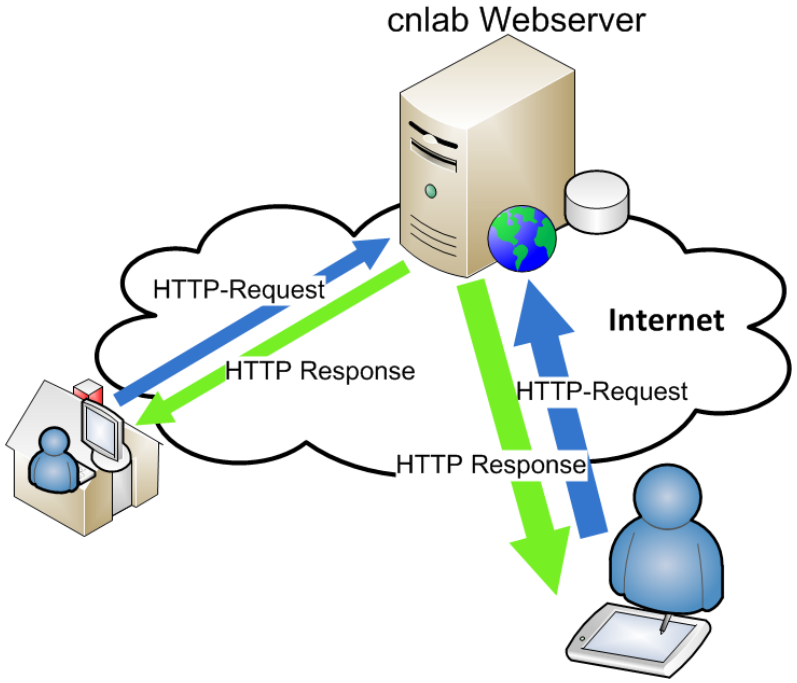
3.6 Kommunikation Client und Server

In diesem Kapitel wird die Kommunikation zwischen der Tablet-Applikation *CoachAssist* und dem cnlab Server behandelt. Ein weiterer Teil dieses Kapitel behandelt ebenfalls die Kommunikation zwischen dem cnlab Server und der cnlab Web-Anwendung.

3.6.1 Übersicht

Abbildung 3.17 zeigt, wie die Kommunikation im Allgemeinen abläuft.

Abbildung 3.17: Server Kommunikation



Die Tablet Applikation kommuniziert direkt mit dem cnlab Server. *CoachAssist* sendet, bei Abfrage der Serverdaten, einen Http-GET-Request an den Server. Daraufhin antwortet der Server mit einer Response. In dieser Response ist ein Konstrukt im JSON-Format enthalten. Dieses Konstrukt wird mit der Library Gson in ein Java Objekt gespeichert. Auf diesen Vorgang kommen wir im Unterkapitel 3.6.3 zu sprechen. Nach der Erzeugung können die Daten normal in einer Java-Applikation verwendet werden.

Das Heraufladen erfolgt ähnlich wie das Anfordern von Daten. Der Unterschied ist, dass ein POST-Request an den Server gesendet wird. Dem POST-Request wird das Objekt im JSON-Format mitgegeben. Der Vorgang zur Erzeugung des Objektes funktioniert analog zum vorher beschriebenen Prozess, nur auf dem umgekehrten Weg.

Die Kommunikation mit dem Server kann ebenfalls über einen Computer erfolgen. Dafür wird auf dem Computer der Browser verwendet. Im Browser wird die Seite `http://www.cnlab.ch/fussballapi` aufgerufen. Auf dieser Seite können die Daten auf dem cnlab Server verwaltet werden. Einträge können ebenfalls über diese Seite erstellt werden. Diese können mit der Tablet-Applikation heruntergeladen werden.

3.6.2 Http-Request/-Response

Für das Senden und Empfangen von Daten wird jeweils ein Request gesendet und eine Response empfangen. Das Aussehen dieser Request / Response-Beziehung wird im folgenden Unterkapitel behandelt. Die API vom Server, welche für die Requests und Responses benötigt wird, ist im Unterkapitel 3.6.6 zu finden.

Empfangen von Daten

Für das Empfangen von Daten wird ein GET-Request an den Server gesendet. Diesem Request wird der URL der Daten mitgegeben. Für die Authentifizierung wird ein Token verwendet, der am Ende der URLs angehängt wird. Der URL ist im folgenden Codeausschnitt zu betrachten.

```
1 String requestUrl = "http://www.cnlab.ch/fussballapi/clubs
    .json?auth_token=fqLHyqwrBeSEZ7CJ3Ndr";
```

Codebeispiel 3.4: URL für das Anfordern der Clubs

Das Login und die Token-Funktionalität ist im Unterkapitel 3.6.5 detaillierter erklärt.
Der GET-Request wird mit der URL erzeugt. Um den Request abzusenden, wird ein HttpClient erstellt. Dieser sendet ihn mit der Methode “execute“ ab.

```
1 HttpGet httpGet = new HttpGet(requestUrl);
2 httpGet.setHeader("Accept", "application/json");
3 httpGet.setHeader("Content-type", "application/json");
4 DefaultHttpClient client = new DefaultHttpClient();
5 HttpResponse response = client.execute(httpGet);
```

Codebeispiel 3.5: GET-Request Beispiel

Die Response wird als return-Wert zurückgegeben und kann weiter verarbeitet werden. Auf die Verarbeitung wird im Kapitel 3.6.3 detaillierter eingegangen.

Senden von Daten

Das Senden erfolgt ähnlich wie das Empfangen. Statt des GET-Requests wird ein POST-Request verwendet. Der Request URL für das Erzeugen sieht gleich aus. Ausser, es handelt sich um Daten, welche eine Zugehörigkeit haben. Ein Beispiel dafür ist ein Team, welches Memberships zugewiesen hat. Beim Hochladen muss der URL folgendermassen aussehen.

```
1 String requestUrl = "http://www.cnlab.ch/fussballapi/clubs
    /1/memberships.json?auth_token=fqLHyqwrBeSEZ7CJ3Ndr";
```

Codebeispiel 3.6: URL zur Erzeugung einer Membership

Beim Request Aufbau wird zusätzlich der Body mit der Methode “setEntity“ gesetzt. Das Senden erfolgt gleich wie beim Abrufen der Daten. In der Response ist das gesendete Objekt enthalten. Dieses Objekt enthält jetzt eine cnlab-Id. Diese Id wird benötigt, um das Objekt auf dem Server zu identifizieren und gegebenenfalls Aktualisierungen auf dem Objekt vorzunehmen. Diese Id wird in der Tablet Datenbank gespeichert. Das Handling ist im folgenden Codeausschnitt einzusehen.

```
1 HttpPost httpPost = new HttpPost(requestUrl);
2 StringEntity se = new StringEntity(jsonString);
3 httpPost.setEntity(se);
4 httpPost.setHeader("Accept", "application/json");
5 httpPost.setHeader("Content-type", "application/json");
6 DefaultHttpClient client = new DefaultHttpClient();
7 HttpResponse response = client.execute(httpPost);
```

Codebeispiel 3.7: POST-Request Beispiel

3.6.3 Umwandlung von JSON zu Java

Das empfangene JSON-Konstrukt wird aus der Response gelesen. Die empfangenen Daten werden der Gson Library übergeben. Der zweite Parameter, welcher der Library mitgegeben wird, ist die DTO-Klasse. Diese zwei Eigenschaften genügen um aus dem JSON ein Java-Objekt zu erzeugen und umgekehrt.
Die DTO-Klassen wurden automatisch mit Hilfe der Seite `http://jsongen.byingtondesign.com/`[2] erstellt. Das JSON-Konstrukt wird der Seite eingegeben und daraus wird die Java-Klasse erstellt. Grund für die zusätzlichen Objekte ist die Trennung von den Datenbankobjekten (VOs) und Servertransferobjekten (DTOs). Beide Eigenschaften in einer Klasse hätten diese überladen. Um ein einfaches Umwandeln zu garantieren wurden zwei Parser erstellt. Die Parser erledigen das Transferieren in das jeweilige Pendant. Mehr zu Gson ist im Anhang im Kapitel B.2.2 zufinden.

3.6.4 Parser

Nachdem das Objekt mit Gson aus dem JSON-Konstrukt generiert wurde, muss das Objekt auf die Datenbank gemapped werden. Dafür gibt es einen Parser. Dieser Parser nimmt das Gson erstellte Objekt (auch DTO genannt) entgegen und wandelt es in ein VO um. Dieses Objekt kann mit dem ORM-Lite Framework in die Datenbank abgelegt werden. Der Parser hat für jedes DTO eine Methode, welche das Parsen in das entsprechende VO ermöglicht. Dieser Parser ist in der Applikation als “DtoParser“ bekannt.

Für den umgekehrten Weg wird ebenfalls ein Parser verwendet. Dieser nennt sich “VoParser“. Dieser Parser wandelt die Objekt aus der Datenbank in die entsprechenden DTO um. Diese werden dann einfach mit Gson in JSON umgewandelt. Darauf folgt, wie in der Übersicht bereits erwähnt, der POST-Request an den Server.

3.6.5 Login

Bei der ersten Anmeldung gibt der Benutzer die registrierte E-Mail-Adresse und sein Passwort ein. Das Login wird für die Serverauthentifizierung benötigt. Mit dem unten stehenden POST-Request 3.8 wird die eingegebene E-Mail-Adresse und das Passwort zum cnlab Server geschickt.

```
1 POST    /fussballapi/users/token
```

Codebeispiel 3.8: API - cnlab Token anfordern

Das Hinzufügen der E-Mail-Adresse und des Passwortes zum POST-Request wird im Codeausschnitt 3.9 gezeigt.

```
1 DefaultHttpClient httpClient = new DefaultHttpClient();
2 HttpPost httpPost = new HttpPost(URI_TOKEN);
3 JSONObject jsonObject = new JSONObject();
4 jsonObject.put("email", email);
5 jsonObject.put("password", password);
```

```
6 String json = jsonObject.toString();
7 httpPost.setEntity(new StringEntity(json, "UTF8"));
8 httpPost.setHeader("Content-type", "application/json");
```

Codebeispiel 3.9: POST-Request mit Email und Passwort

Der cnlab Server sendet daraufhin ein Token. Dieses wird für die weiteren Kommunikationen mit dem Server benötigt. Bei jedem weiteren Request wird das Token anstelle der E-Mail-Adresse und Passwort mitgesendet. Das erhaltene Token wird mit einem SHA verschlüsselt in der Datenbank abgespeichert.

3.6.6 Server-API

Für die Requests an den Server ist folgende API definiert. In diesem Unterkapitel wird nur auf einzelne Teile der Schnittstelle eingegangen. Die komplette API ist auf der CD zu finden.
Bei den beiliegenden Ausschnitten definiert der erste Teil die Request-Art und der zweite Teil die URL.

Liste aller Clubs:

```
1 GET    /fussballapi/clubs
```

Codebeispiel 3.10: API - Alle Clubs anfordern

Club Erstellung:

```
1 POST   /fussballapi/clubs
```

Codebeispiel 3.11: API - Erstellen eines Clubs

Spezifischer Club durch Id:

```
1 GET    /fussballapi/clubs/:id
```

Codebeispiel 3.12: API - Club anfordern

Update eines spezifischen Clubs mittels Id:

```
1 PUT    /fussballapi/clubs/:id
```

Codebeispiel 3.13: API - Updaten eines Clubs

Löschen eines spezifischen Clubs mittels Id:

```
1 DELETE /fussballapi/clubs/:id
```

Codebeispiel 3.14: API - Löschen eines Clubs

Liste von Memberships eines spezifischen Clubs:

```
1 GET    /fussballapi/clubs/:club_id/memberships
```

Codebeispiel 3.15: API - Memberships anfordern

Erstellen einer Membership für einen spezifischen Club:

```
1 POST   /fussballapi/clubs/:club_id/memberships
```

Codebeispiel 3.16: API - Erstellen einer Membership

Teil II

Software Engineering

4 Requirements

In diesem Kapitel werden die Requirements beschrieben. Diese wurden zu Beginn des Projektes erstellt und während dem Projekt geringfügig angepasst. Dieses Kapitel behandelt vorwiegend die Stakeholders, Personas, funktionale und nicht-funktionale Anforderungen.

4.1 Allgemeine Beschreibung

Ursprünglich war *CoachAssist* in erster Linie für Anwender des cnlab Fussballspieler-Tracking-Systems gedacht. Im Verlauf der Semesterarbeit hat sich gezeigt, dass CoachAssist auch unabhängig vom cnlab Fussballspieler-Tracking-System zum Einsatz kommen kann. Mögliche Anwender sind generell Personen, welche Fussballspiele dokumentieren wollen (z.B. Coaches, Fans oder Sportwissenschafter).

4.1.1 Benutzercharakeristik und Stakeholders

Die Tabelle 4.1 zeigt die Stakeholders mit einer kurzen Beschreibung und einer Rollendefinition. Im externen Dokument Vision ist eine ausführlichere Beschreibung der Stakeholders definiert. Dieses Dokument befindet sich auf der CD.

Tabelle 4.1: Stakeholders

Name	Beschreibung	Rolle
Cnlab Management	Dieser Stakeholder spiegelt die Interessen der Firma cnlab an dem Produkt wieder.	Verwendung und Verkauf der Applikation
Cnlab Entwickler (Support, Weiterentwicklung und Wartung)	Dieser Stakeholder repräsentiert die Softwareentwickler der Firma cnlab, die diese Applikation warten und supporten müssen.	Wartung, Support, Weiterentwicklung der Applikation
Applikationsentwickler	Dieser Stakeholder ist das Entwicklerteam der Bachelorarbeit.	Entwicklung der Applikation

Tabelle 4.2: Stakeholders Fortsetzung

Name	Beschreibung	Rolle
Fussballteam/ Verein	Dieser Stakeholder repräsentiert den Nutzer der Applikation. Speziell zu erwähnen ist, dass das Design und Handling der Applikation seinen Bedürfnissen entsprechen soll.	Verwendung der Applikation
Fan	Dieser Stakeholder definiert sich durch die gleichen Interessen wie der Stakeholder Fussballteam/-Verein. Nur die Bedürfnisse für das Design und Handling sind nicht so stark gewichtet.	Verwendung der Applikation
Sportwissenschaftler	Dieser Stakeholder will die Fussballspiele zu wissenschaftlichen Zwecken aufzeichnen. Eine nützliche Funktion dafür ist das Messen der effektiven Spielzeit.	Verwendung der Applikation
Hochschule Rapperswil	Dieser Stakeholder dient die Bachelorarbeit und das daraus resultierende Produkt evtl. zu möglichen Werbezwecken.	Werbezwecke

4.1.2 Personas

Nebst den Stakeholdern sind zwei Personas definiert. Diese Personas sollen ein Bild von den Endbenutzern zeigen. Die erste Persona ist aus der Semesterarbeit übernommen. Hierbei handelt es sich um Carlos Coach. Er spiegelt die Nutzung durch das Team bzw. durch den Coach wieder. Bei der zweiten Personas handelt es sich um Franz Fan. Franz ist ein begeisterter Fussballfan, der alles über seine Lieblingsmannschaft protokollieren will.

Persona Coach

- Carlos Coach
- 48 Jahre
- Fussballcoach des FC Sargans
- Softwareentwickler bei K-Soft
- Verheiratet, Vater von zwei Kindern
- Technikfan

Carlos ist Coach der Fünft-Ligamannschaft des FC Sargans. Er macht dies schon fast 10 Jahre lang. Er ist 48 Jahre alt. Fussballcoach ist er nebenberuflich. Sein Hauptberuf ist Softwareentwickler. In diesem Job arbeitet er zu 70%.

Das Fussballtraining findet meistens Abends statt. Jedoch muss er auch manches Wochenende opfern um an Turnierspielen teilzunehmen. Während des Spiels erstellt er öfters Notizen, um dann im Training auf das Spiel einzugehen.
Softwareentwickler ist er in einer kleinen Firma namens K-Soft. Dort entwickelt er Webseiten, aber auch mobile Applikationen.
Neben diesen zwei Berufen ist Carlos ein guter Familienvater. Er hat zwei Kinder, ein Mädchen und einen Jungen. Er nimmt seine zwei Kinder auch öfters zu den Fussballspielen mit.
Sein Sohn spielt selbst sehr gerne Fussball und möchte später in der Mannschaft seines Vaters spielen.
Weil Carlos Informatiker ist und "kleine Spielsachen" liebt, hat er vom PC bis zum iPhone jedes Gerät. Er ist ebenfalls stolzer Besitzer eines Galaxy Pads, welches er fast immer dabei hat. Er verwendet es für seinen Informatikberuf wie auch in der Freizeit. So nutzt er es auch um mit dem Standard Notiz App Notizen zu den Fussballspielen zu machen.

Persona Fan

- Franz Fan
- 29 Jahre
- Begeisterter Fan des FC Rapperswil
- Supporter bei Swisscom
- Single
- Fussballfanatiker

Franz ist ein begeisterter Fan des Fussballvereins FC Rapperswil. Dies ist er schon seit klein auf. Er hat bisher fast kein Spiel verpasst. Er ist 29 Jahre alt. Nebst seiner Leidenschaft für den FC Rapperswil ist er gelernter Supporter bei der Firma Swisscom. Er arbeitet bereits sieben Jahre bei dieser Firma.
Leider ist Franz immer noch Single. Er wendet zu viel Zeit für seine Lieblingsmannschaft auf. Er verfolgt jedes Fussballspiel. Er notiert sich die Tore, die gelben und roten Karten, sowie jede Auswechslung, die während einem Spiel stattfindet. Dies macht er jedoch im Moment mit Stift und Papier. Er besitzt zwar ein Archos Tablet, aber ist bisher noch nicht auf die Idee gekommen, eine Applikation für das Aufzeichnen zu verwenden.

4.1.3 Abhängigkeiten

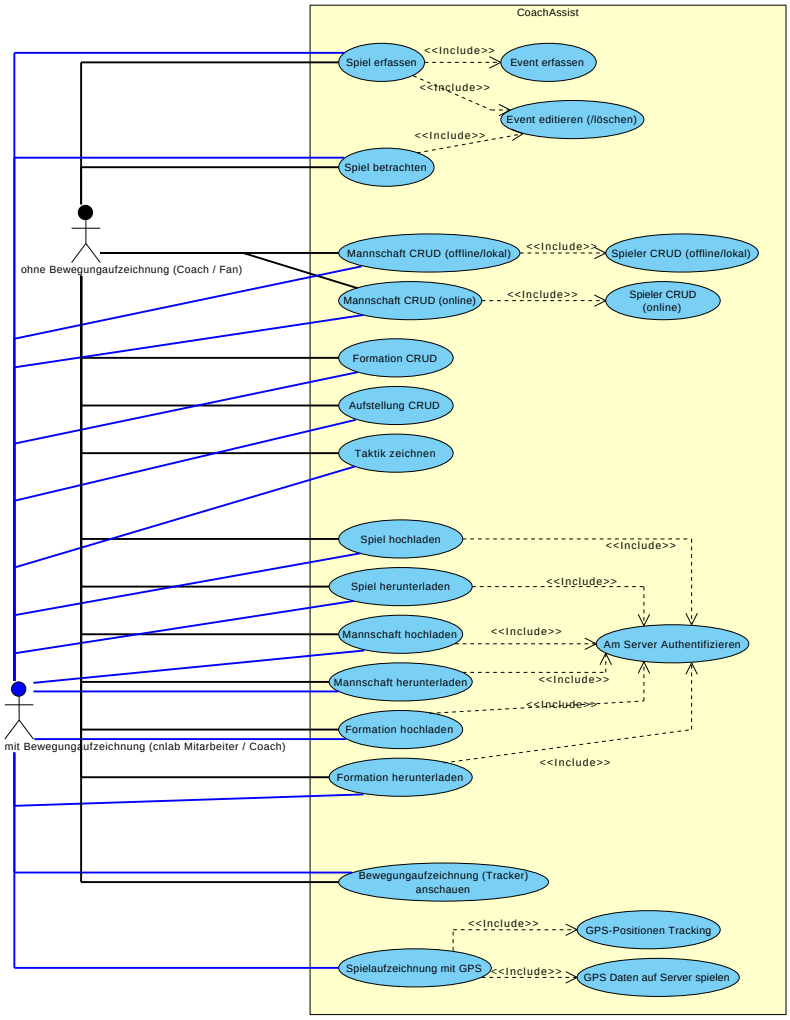
Die Applikation wird auf Android 4.0 (Ice Cream Sandwich) entwickelt. Die älteren Versionen werden nicht unterstützt, da Android nicht abwärtskompatibel ist. Für gewisse Feature sind folgende Hardware Requirements zu beachten:

- GPS (Nur nötig, falls die Funktion GPS-Aufnahmen Spielfeld genutzt wird.)
- Bluetooth- oder USB-Schnittstelle für die Datenübertragung von den GPS-Tracks (Nur nötig, falls die Funktion Einlesen von GPS-Tracks genutzt wird.)
- Guter Display, der auch bei starkem Sonnenschein die Applikation klar darstellt.

4.2 Funktionale Anforderungen

Die Abbildung 4.1 zeigt die Use Cases, welche die Applikation betreffen. Die Use Cases beschränken sich nicht nur auf die Applikation selbst. Es werden alles Use Cases dargestellt, welche mit dem Outdoor Tracking System zusammenhängen. Dies betrifft Aufnahmen mit dem GPS-Tracker, aber ebenso die Webseite `http://www.cnlab.ch/fussball`. Weitere Use Cases, die definiert wurden betreffen die Webseite zur Erfassung von Mannschaften und Spieler `http://www.cnlab.ch/fussballapi`. In der Abbildung 4.1 wurde bewusst nur zwischen zwei Akteure unterschieden. Die Stakeholder lassen sich in diese zwei Arten von Akteure gruppieren. Der Unterschied liegt in der Verwendung des cnlab GPS Tracking Systems.

Abbildung 4.1: UseCase Diagramm



Die Use Cases sind im Anhang im Kapitel A.1 in Briefly-Form aufgelistet.

4.3 Nichtfunktionale Anforderungen

Die nicht funktionalen Anforderungen wurden nach fünf Hauptkriterien des ISO 9126 Schema aufgebaut und dokumentiert.

4.3.1 Zuverlässigkeit

Die Zuverlässigkeit ist wichtig, da die Daten live aufgezeichnet werden und es dabei keine Datenverluste geben darf. Der Benutzer muss sich darauf verlassen können, dass die Eingaben, die er tätigt, auch gespeichert werden.

4.3.2 Benutzbarkeit

Die Software soll auch von Laienanwendern benutzt werden können. Deshalb ist die Bedienbarkeit wichtig. Der Benutzer soll sich möglichst einfach zu recht finden. Zur Verbreitung wird in einer späteren Phase des Entwicklungsprozesses der Android-Store verwendet.

4.3.3 Effizienz

Hierbei steht die Performance im Mittelpunkt. In der Semesterarbeit war die Reaktionszeit der Applikation nicht zufriedenstellend. Der Delay betrug teilweise eine halbe Sekunde, welches nicht vertretbar war. Mit der nativen Version der Applikation soll dieses Problem gelöst werden.

4.3.4 Änderbarkeit

Hier liegt speziell der Punkt “Wartbarkeit“ im Zentrum. Der Code soll möglichst einfach erweitert werden können. Deshalb ist es notwendig, dass der bestehende Code verständlich ist. Dies betrifft vor allem den Stakeholder “cnlab Mitarbeiter“, der mit dem Code weiterarbeiten muss.

4.3.5 Übertragbarkeit

Im Vergleich zur Semesterarbeit mussten bei diesem Punkt die Bedürfnisse heruntergestuft werden. Die Applikation ist nur noch auf Android Geräten verwendbar.

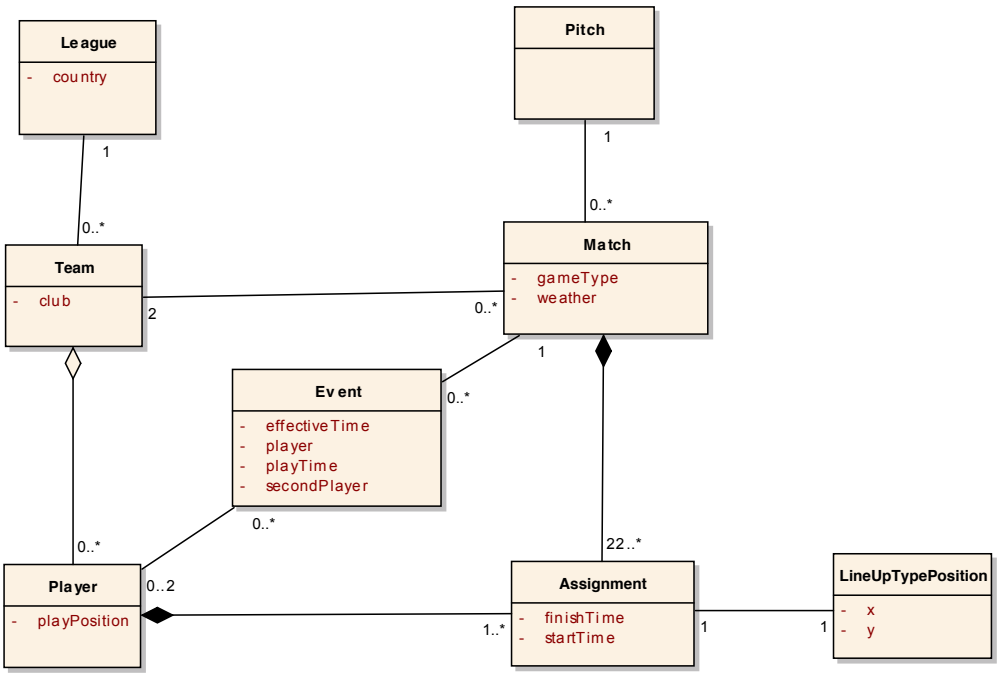
5 Software Analysis

In diesem Kapitel wird die Softwareanalyse besprochen. Dieses Kapitel zeigt, wie die Entwicklung der Applikation begonnen hat. Es soll dem Entwickler helfen, Gedankengänge nachvollziehen zu können.

5.1 Domainmodel

Die Abbildung 5.1 zeigt das Domainmodel. In dieser Abbildung sind nur die wichtigsten Klassen dargestellt.

Abbildung 5.1: Domainmodel



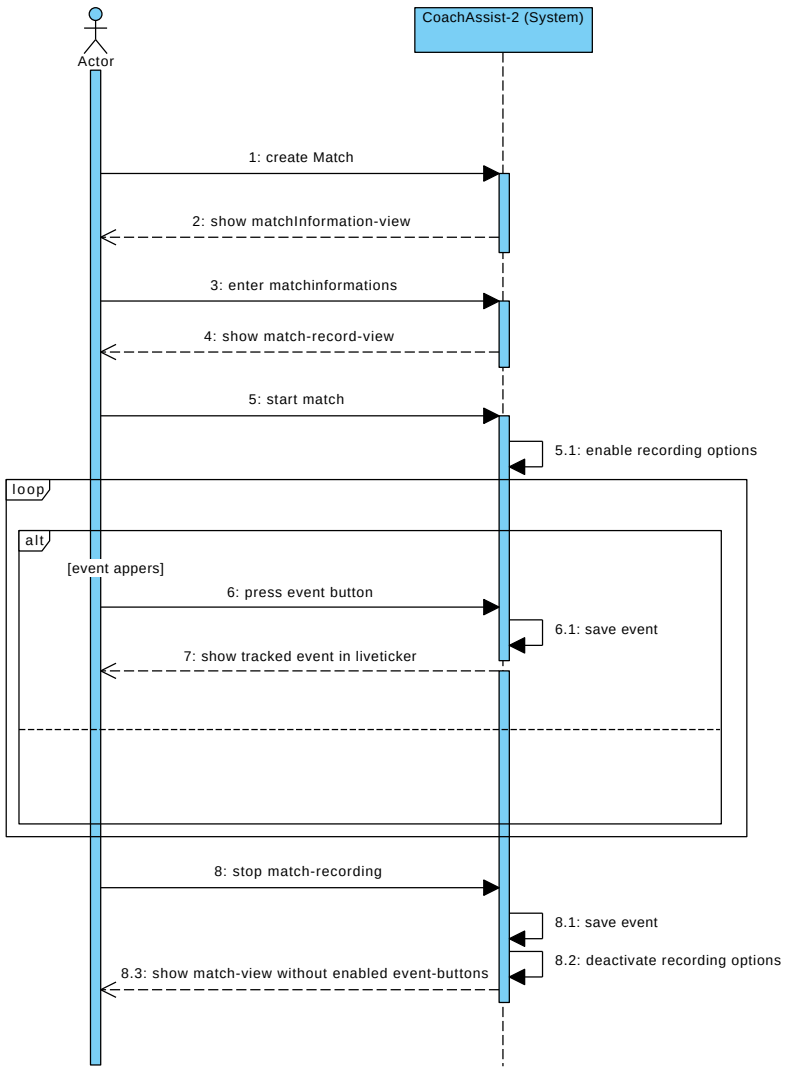
Im Kern ist die Klasse Match. Zu jedem Match gibt es Assignmets bzw. Zuteilungen. Bei diesen Zuteilungen wird ein Spieler einem Match zugeordnet. Dafür wird die Start- und Endzeit benötigt. Neben dieser Klasse sind einem Match auch mehrere Events zugeteilt. Es werden hierbei beliebig viele Events erfasst. Diese können je nach Eventtyp unterschiedliche Attribute enthalten. Zu Beginn der Bachelorarbeit waren die Events in einer Vererbungshierarchie

dargestellt. Dies musste wegen ORMLite verworfen werden. Das Framework unterstützt keine optimale Vererbung. Deshalb wurden alle Events in eine Klasse zusammen genommen. Sie werden mit dem Eventtypen unterschieden. Dieses Diagramm zeigt einen kleinen Abriss zur Domainproblemstellung. Im Kapitel Softwaredesign wird auf die Themen ausführlicher eingegangen.

5.2 Systemsequenzdiagramme (SSD)

Die Abbildung 5.2 zeigt den geplanten Ablauf einer Matchaufzeichnung. Die SSD soll dem Programmierer zeigen, wie die Idee des Ablaufs entstanden ist. Zuerst werden die Matchinformationen eingetragen. Danach wir das Match gestartet und die Events werden erfasst. Am Schluss wird die Aufzeichnung beendet und die Daten abgespeichert. Zusätzlich werden die Aufnahmeoptionen deaktiviert.

Abbildung 5.2: SSD: Matchaufnahme



5.3 Technologie Entscheide

Folgende zwei Technologien wurden geprüft. Der Grund für die Entscheidung wird ebenfalls angesprochen.

ORMLite

ORMLite wurde wegen des einfachen Handlings der Android Datenbank in das Projekt mit einbezogen. Es ermöglicht ein einfaches Erzeugen der Tabelle durch eine Java-Klasse, welche mit Annotationen versehen wurde. Ebenfalls sind die vordefinierten CRUD-Operationen praktisch. Hauptgrund für die Nutzung ist das sparen von Zeit, welches durch die vordefinierten Operationen entsteht. Im Anhang ist im Kapitel B mehr zu diesem Tool erwähnt.

Gson

Gson wurde aus einem ähnlichen Grund wie das ORM-Framework eingebaut. Es ermöglicht ein einfaches Parsen eines JSON-Konstrut in ein Java-Objekt. Dies spart wieder eine Menge Zeit, da dieses Konstrukt nicht von Hand ausgelesen werden muss.

6 Testing

In diesem Kapitel wird auf das Testen der Applikation eingegangen. Es wurden Unit-, System- und Gerätetest durchgeführt. Ebenfalls wurde ein Usability-Test durchgeführt. Die Applikation wurde bereits vom Industriepartner getestet. Dieser Test ist im nachstehen Abschnitt erklärt. Für die Unit- und System-Tests wurde ein externes Testprojekt erstellt. Es wurde sich an die Angaben von Android Developers gehalten. Der Link dazu lautet: `http://developer.android.com/guide/topics/testing/index.html`[12] .

6.1 Unit Tests

Die Unit Tests sind mit dem Android Test Framework erstellt. Um normale Unit Tests in Android zu nutzen muss die Testklasse von AndroidTestCase erben. AndroidTestCase beinhaltet die JUnit Funktionalität, welche für die Unit Tests benötigt werden.
Die geplante Testabdeckung der Model- und VOs-Klassen wurde zu Beginn auf 100% gesetzt. Aus Zeitgründen und der verschiedenen Testmöglichkeiten für eine Android Anwendung nicht möglich.
Effektive getestet wurden folgende Klassen:

- Model-Klassen
  - Es wurden vier der neun Model-Klassen getestet.
- VO-Klassen
  - Es wurden nur zwei der 16 Klassen getestet.
- Util-Klassen
  - AeSimpleSHA wurde getestet.
  - Alle DTO-Klassen wurden getestet.

Die VO-Klassen wurden deshalb vernachlässigt, da sie nur aus “getter“ und “setter“-Methoden bestehen. Für die Model Klasse wurde der “Databasehelper“ manipuliert, so dass es manipulierte Data Access Objects zurück gibt.

Diese wurden so verändert, dass die verwendeten CRUD-Operationen auf eine ArrayListe innerhalb der Klasse zurück greifen. So können die Model Klassen unabhängig von der Datenbank getestet werden.

6.2 Usability Tests

Gegen Ende der Bachelorarbeit wurden Usability Tests durchgeführt. Dafür wurden zwei Szenarien entworfen. Das erste betrifft das Hinzufügen und Editieren von Fussballspielern und und das zweite, das Aufzeichnen eines Spieles. Das erste Szenario brachte folgende Mängel zum Vorschein:

- Die automatisch generierten Spielers beim erzeugen des Teams haben den Benutzer verwirrt.
- Nach dem Erstellen eines Spieler war der Benutzer verwirrt. Der Grund war, dass er die Spieler nicht direkt gesehen hat. Erst nach erstmaligem runterscrollen hat er bemerkt, dass die Spieler nach Alphabet sortiert sind. Die generierten Spieler beginnen alle mit A und waren somit alle zu oberst.

Daraus wurden folgende Massnahmen definiert:

- Im Dialog zur Teamerstellung eine Checkbox einbauen. Diese Chechbox definiert, ob die Spieler generiert werden sollen oder nicht.
- Sortierung der Spieler ändern.

Das zweite Szenario brachte folgende Mängel zum Vorschein:

- Zu Beginn war dem Benutzer nicht klar, welche Buttons zu welcher Mannschaft gehören.
- Der Benutzer versucht Events aufzuzeichnen, welche nicht existierten. Er blickte mehrmals bei einem Event auf das Tablet, um zu überprüfen, ob er das Event erfassen kann.
- Die Applikation reagiert ein wenig langsam auf die Klicks. Aufgrund der Probleme mit der Semesterarbeit ist dies ein sehr wichtiges Kriterium.
- Dem Benutzer war nicht klar das er die Events wieder löschen kann. Es ist deutlich zu sehen, dass es sich verklickt und da er nicht weiter weiss lässt er es einfach stehen.

Daraus wurden folgende Massnahmen definiert:

- Unterscheidung der Mannschaften klarer darstellen. Mannschaftsnamen grösser und übersichtlicher darstellen.
- Folgende Events könnten erweitert werden:
  - Torschuss
  - Ballabnahme
  - Faul
- Optimieren der Reaktionszeit auf Klicks.
- Einblenden des Löschen- und Editierbuttons für das letzte erfasste Event.

Die Änderungen konnten aus Zeitgründen nicht mehr in die Applikation übernommen werden. Diese werden vom Industriepartner umgesetzt.
Die Szenarien und die Aufnahmen des Usability-Tests sind auf der CD zu finden.

6.3 Systemtests

Die Systemtests sind mit dem Testframework Robotium erstellt. Robotium ermöglicht ein effektives Erzeugen von Black-Box Test-Cases für Android Applikationen. Robotium simuliert die Benutzereingaben und vereinfacht somit das Testen von Use Cases. Die vordefinierte Testabdeckung für Activities wurde hinfällig, da die Activities implizit durch Robotium getestet werden.

Folgende Test Cases wurden erstellt:

1. Team Management
   a Erstellen eines Clubs
   b Erstellen eines Teams
   c Erstellen eines Spielers
   d Löschen eines Spielers
   e Löschen eines Teams
   f Löschen eines Clubs
2. Match Management
   a Erstellen eines Spiels

Die Implementation der Robotium Test Cases sind sehr auffändig, da die Fragments nicht vollumfänglich unterstützt sind bzw. Android 4.0. Die Systemtest basieren auf Robotium Solo 3.3.

Aufgrund der Zeitmangel konnten keine weiteren Test Cases geschrieben werden. Folgende Use Cases Tests fehlen:

1. Use Case UC01 - Spiel erfassen
2. Use Case UC04 - Spiel hochladen
3. Use Case UC05 - Spiel herunterladen
4. Use Case UC09 - Mannschaft herunterladen
5. Use Case UC13 - Aufstellung CRUD
6. Use Case UC14 - Serverauthenthifizierung
7. Use Case UC15 - Taktik zeichnen

6.4 Monkey Tests

Monkey ist ein Stresstest-Tool. Dieses Tool für zufällige Touch-Gesten auf dem Tablet aus. Dieses Tool ist im Anhang im Kapitel B beschrieben.
Die Applikation wurde mit dem Android Tool Monkey getestet. Folgende Tests wurden durchgeführt und erfolgreich abgeschlossen:

- 1000 Klicks
- 10000 Klicks
- 100000 Klicks

6.5 Geräte Tests

Das Tablet “Archos 80 G9“ der Bachelorarbeit wurde unter die Lupe genommen. Das Ziel dieses Test war es, herauszufinden, wie geeignet es für die Aufzeichnung von Fussballspielen ist.
Folgende Kritikpunkte wurden dadurch zum Vorschein gebracht:

- Bei Sonnenlicht ist der Display nur schwer zu erkennen.
- Der Bildschirm spiegelt stark.
- Touch Screen reagiert teilweise schlecht.
- Bildschirm ist von links und oben unter wesentlich flacherem Sichtwinkel noch erkennbar. Von der rechten und unteren Seite ist das Bild wesentlich schlechter sichtbar.

Einige Kritikpunkte stammen auch vom Feedback des Industriepartners.
Für das Tablet sprechen folgende Punkte:

- Das Gewicht von 465 Gramm ist für ein Fussballspiel praktisch. Das Tablet muss lange gehalten werden.
- Der Preis von ca. 300.- CHF ist auch für einen Fussballfan gerechtfertigt.
- GPS Funktionalität für das Feature Fussballfeld messen ist vorhanden.
- Grösse des Tablets (226mm x 155.3mm x 11.7mm) ist für die Nutzung auf dem Spielfeld gut geeignet.
- Für Video- und Bildaufnahmen bietet das Tablet eine gut Auflösung von 720p für die Frontkamera.

Produktinformationen wurden von der Webseite `http://www.archos.com/store/psearch.html?country=de&lang=de&prod_id=archos80g9`[1] bezogen. Ebenfalls sind Informationen aus dem Testbericht der Webseite `http://www.netzwelt.de`[19] mit eingeflossen.

Fazit für die Nutzung des Tablet ist schlecht. Dies ist zu Begründen mit den Problemen mit dem Display bei Sonneneinstrahlung und das schlechte reagieren des Touch Screens.

6.6 Provisorischer Abnahmetest durch cnlab

Der Industriepartner hat die Applikation getestet. Es wurden Fehler, Verbesserungen und Verschönerungen notiert. Im Rahmen der Bachelorarbeit werden nicht alle Optimierungen umgesetzt. Das ausführliche Testprotokoll ist auf der CD enthalten.

Teil III

Schlussfolgerung

7 Ergebnis

Das Hauptziel zur verbesserten Performance wurde nur teilweise erreicht. Die Performance hat sich gegenüber dem Prototyp aus der Semesterarbeit verbessert. Die Reaktionen der Event Buttons liegt jedoch noch nicht im akzeptablen Bereich. Das Speichern des Events muss weiter optimiert werden.
Alle gewünschten Features mit der Priorität A wurden komplett implementiert. Alle Ziele der Bachelorarbeit wurden erreicht. Leider konnten nicht soviele Usability Tests durchgeführt werden wie gewünscht.

Folgende Funktionen wurden vollständig umgesetzt:

- Eventerfassung pro Mannschaft
- Spielerauswahl bei Eventerfassung
- Erfasste Events während dem Spiel editierbar
- Notizen erfassen (Handeingabe)
- Effektive und offizielle Spielzeit erfassen
- Anzeige der Spielinformationen (Mannschaftsnamen, Resultat, Torschützen mit Timestamp) während dem Spiel
- Fortlaufende Liveticker Aktualisierung für notierte Spiel-Events
- Erfasste Events (Liveticker) auch zu einem späteren Zeitpunkt abrufbar und editierbar
- Übersicht über alle erfassten bzw. noch auf dem Tablet vorhandenen Spiele
- Mannschaft- und Spielerverwaltung auf dem Tablet (aufgrund der Standalone Lösung)
- Abbildung der cnlab Server Datenbank auf dem Tablet

Folgende Funktionen wurden nur teilweise umgesetzt und hätten Erweiterungen nötig:

- Mannschaftsaufstellung, Ersatzbank und Formation festlegen
  - Mannschaftsaufstellung kann frei definiert werden. Spieler kann auf beliebige Positionen des Spielfeldes gezogen werden.
- Kommunikation mit cnlab Server (Upload, Download und Update)
  - Upload und Download von Teams, Pitches und Leagues möglich
- Taktikboard (Zeichnung)
  - Es kann frei auf dem Fussballfeld gezeichnet werden.
- Notizen erfassen (Voice-Eingabe)
  - Mit dem Code wurde bereits im Hintergrund begonnen.
- Detail Ansicht der einzelnen Spiele (Spiel-Statistik). Bsp. Anzahl Eckbälle pro Team, Torschützen, Anzahl Freistösse pro Team, etc.
  - Daten werden bereits erfasst.
- Mehrsprachigkeit (enthalten: DE, EN; mit Hilfe cnlab: IT, FR)
  - EN ist Standardsprache.
  - DE wurde bereits übersetzt.
- Erfassung der Spielranddaten (Gegner, Spielort, Anpfiffzeit, Spieltyp, Wetter)
  - Gegner, Wetter, Spieltyp, Spielfeld, Anpfiffzeiten, Abpfiffzeiten sind vorhanden.

Die Kommunikation mit dem cnlab Server konnte nur begrenzt eingebaut werden. Aus Zeitmangel und zusätzlichen Problemen mit dem Server konnte diese Feature nur teilweise umgesetzt werden.

7.1 Vergleich zu Mitbewerberprodukten

Für den Vergleich wird eine Übersicht mit anderen Applikationen betrachtet und wie diese zu *CoachAssist* stehen. Die Übersicht ist in der Tabelle 7.1 ersichtlich. Die Informationen stammen aus dem Android Market[9].

Tabelle 7.1: Applikationsvergleich Übersicht

	CoachAssist	Soccer Stats w/ Timer	Soccer Stats for Parents
Preis	gratis	1.55 CHF	gratis
Eventerfassung inkl. Spielerauswahl	Ja	Ja	Nur ein paar Events
Spielzeit	Effektive und Offizielle Spielzeit	Ja	Ja
Liveticker	Ja	Nur Zähler	Nur Zähler
Globale Datenspeicherung	Ja	Nein	Nein
Mehrsprachigkeit	EN, DE (später FR, IT)	Nein	Nein
Mannschaftsaufstellung	Ja	Nur eigene Mannschaft	Nein
Taktikboard	Ja	Nein	Nein
Aufgezeichnete Spiele betrachten	Spätere Version	Ja	Ja
Spiel und Spieler Statistik	Spätere Version	Ja	Ja
Twitter/Mail	Nein	Ja	Nein
Hilfe Funktion	Nein	Nein	Ja

CoachAssist kennzeichnet sich durch die Globale Datenspeicherung, Mehrsprachigkeit und durch das Taktikboard aus. Der ausführliche Liveticker für Events ist ein weiteres wichtiges Merkmal von *CoachAssist*. Dies ist im Vergleich zu den anderen zwei Tools ein klarer Vorteil. Soccer Stats /w Timer hat im Gegensatz eine Twitter und Mail Funktion. Die ermöglicht einfaches publizieren der aufgenommenen Daten. Soccer Stats for Parents unterstützt den Nutzer mit einer Hilfe Funktion.

8 Ausblick

Die vorliegende Prerelease Version von *CoachAssist* wird cnlab übergeben. Sie führen weitere Usability Tests und nehmen weitere Optimierungen vor. Gemäss Planung soll im Herbst die erste Version von *CoachAssist* auf den Android Market kommen.
Weitere Funktionen der cnlab Fusballspieler-Tracking System werden in *CoachAssist* integriert.

Folgende Funktionen wurden nur teilweise umgesetzt und bedürften Erweiterungen.

- Mannschaftsaufstellung, Ersatzbank und Formation festlegen
  - Dem Benutzer ermöglichen selbst vordefinierte Formationen zu erstellen und zu verwenden.
- Kommunikation mit cnlab Server (Upload, Download und Update)
  - Match hochladen einbauen
  - Synchronizer einbauen, um herunter- oder hochgeladene Daten zu synchronisieren
- Taktikboard (Zeichnung)
  - Spieler einer Mannschaft
- Notizen erfassen (Voice-Eingabe)
  - Grundlagen wurden bereits eingebaut
- Detail Ansicht der einzelnen Spiele (Spiel-Statistik). Bsp. Anzahl Eckbälle pro Team, Torschützen, Anzahl Freistösse pro Team, etc.
  - Daten werden erfasst, müssten jedoch noch ausgewertet werden. Dazu müsste eine zusätzliche GUI eingebaut werden.

- Detail Ansicht der einzelnen Spieler (Spieler-Statistik). Bsp. Anzahl Tore über alle Spiele (Saison) hinweg, etc.
  - Daten werden erfasst, müssten jedoch noch ausgewertet werden. Dazu müsste auch eine zusätzliche GUI eingebaut werden.
- Mehrsprachigkeit (enthalten: DE, EN; mit Hilfe cnlab: IT, FR)
  - Übersetzung der Sprachen IT, FR durch cnlab
- Erfassung der Spielranddaten (Gegner, Spielort, Anpfiffszeit, Spieltyp, Wetter)
  - Spielranddaten könnte noch mit weiteren Attributen erweitert werden.

Die nachstehenden Funktionen sind zur kompletten Implementierung:

- GPS-Aufnahme des Spielfeldumfangs
- Bildaufnahmen
- Videoaufnahmen
- Kommunikation mit dem cnlab Fussballspieler-Tracking System Server.
- Darstellung Spielerpositionen und Statistiken pro Spieler vom cnlab Fussballspieler-Tracking System
- Einlesen von Daten von GPS-Tags.

Teil IV

Verzeichnisse

Literaturverzeichnis

[1] Archos. Archos produktbeschreib. http://www.archos.com/store/psearch.html?country=de&lang=de&prod_id=archos80g9, Juni 2012.

[2] Rick Byington. Json generater. http://jsongen.byingtondesign.com/, Juni 2012.

[3] cnlab. cnlab fussball. http://www.cnlab.ch/fussball/, Juni 2012.

[4] Romain Guy, editor. *Turbo-charge your UI*. Google Inc., Google I/O, May 2009.

[5] Google Inc. Android: Avd und sdk guide. http://developer.android.com/guide, Juni 2012.

[6] Google Inc. Android developer. http://developer.android.com/index.html, Juni 2012.

[7] Google Inc. Android developer: Framework. http://developer.android.com/resources/faq/framework.html, Juni 2012.

[8] Google Inc. Android: Lint. http://tools.android.com/tips/lint, Juni 2012.

[9] Google Inc. Android market. https://market.android.com/?hl=de, Juni 2012.

[10] Google Inc. Android: Monkey. http://developer.android.com/guide/developing/tools/money.html, Juni 2012.

[11] Google Inc. Android: Robotium. http://code.google.com/p/robotium, Juni 2012.

[12] Google Inc. Android: Testing. http://developer.android.com/guide/topics/testing/index.html, Juni 2012.

[13] Google Inc. Android: The world of listview. http://www.google.com/events/io/2010/sessions/world-of-listview-android.html, Juni 2012.

[14] Google Inc. Android: Turbochargeuiandroidfast. http://www.google.com/events/io/2009/sessions/TurboChargeUiAndroidFast.html, Juni 2012.

[15] Google Inc. Code guidelines. http://developer.android.com/source/code-style.html, Juni 2012.

[16] Google Inc. Design guidelines. http://developer.android.com/design/index.html, Juni 2012.

[17] Google Inc. Gson. http://code.google.com/p/google-gson, Juni 2012.

[18] Uni Magdeburg. Latex vorlage. http://wdok.cs.uni-magdeburg.de/studium-und-lehre/hinweise-und-vorlagen/wdok-diplom.tar.gz/view, Oktober 2011.

[19] netzwelt GmbH. Archos testbericht. http://www.netzwelt.de/news/91609-archos-80-g9-turbo-ics-test-android-tablet-standfuss.html, Juni 2012.

[20] Ray Ryan, editor. *Architecting GWT applications for production at Google*. Google Inc., Google I/O, May 2010.

[21] Gray Watson. Ormlite. http://ormlite.com/, Juni 2012.

[22] Wikipedia. Wikipedia. http://en.wikipedia.org/wiki/Jquery, Juni 2012.

Glossar

Android
Ein Smartphone und Tablett Betriebssystem von Google basierend auf Java[22]. 18

API
Eine Application Programming Interface ist eine Programmierschnittstelle, welche eine Anbindung an das System zur Verfügung stellt[22]. 19, 51, 55

cnlab Fussballspieler-Tracking System
GPS-basierte Erfassung der Bewegungen von Fussballspielern (`www.cnlab.ch/fussball/`). 27, 33, 82

CRUD
CRUD beschreibt folgende Datenbankoperationen: C (create): Daten erstellen, R (read): Daten lesen, U (update): Daten aktualisieren, D (delete): Daten löschen[22]. 34, 67, 70, 104

DTO
Ein Data Transfer Object bündelt Daten in einem Objekt. In dieser Arbeit dienen sie zur Übertragung von Daten zum cnlab Server und zurück[22]. 40, 47, 53, 69, 113

Framework
Ein Framework ein ein Programmiergerüst. Es ist kein fertiges Programm. Es stellt einen Rahmen für den Programmierer zur Verfügung[22]. 53

GET-Request
Mit einem Get-Request wird eine Ressource von einem Server angefordert. Der Request benötigt dafür den URI, der die Resource identifiziert[22]. 51, 52, 95

goldene Schnitt
Der goldene Schnitt beschreibt ein Teilungsverhältnis einer Strecke oder anderen Grösse[22]. 110

Gson
Gson (auch bekannt als Google Gson) ist eine open-source-Java library für die Sereialisierung und Deserialisierung von Java Objekten in JSON[22]. 3, 18, 33, 34, 51, 53, 67, 113, 123

GUI
Mit Graphical User Interface ist eine grafische Benutzeroberfläche gemeint[22]. 29, 34, 35, 38, 81, 82, 93, 107, 111

JSON
JavaScript Object Notation ist ein Datenformat, welches für Mensch und Maschine einfach lesbar ist. Es ist eine Textform zum Zweck des Datenaustauschs zwischen Anwendungen[22]. 18, 51, 53, 95, 105, 113

Library
Eine Library ist eine Zusammenstellung von Ressourcen. Sie wird für die Entwicklung einer Software benötigt, welche diese Library einbindet[22]. 51, 53

ORM
Object Relational Mapping ist eine Abbildung von Objekten mittels Annotationen in die Datenbank[22]. 18, 33, 47, 103

ORMLite
ORMLite ist ein ORM-Framework. Es dient dazu die VOs in die Datenbank zu mappen[22]. 3, 18, 33, 47, 53, 66, 67, 103, 119, 123

PhoneGap
PhoneGap ermöglicht den Applikationen, die auf Webtechnologien entwickelt wurden, pseudo-native auf Mobile Geräten zu laufen[22]. 27

POST-Request
Mit einem Post-Request können Daten an den Server gesendet werden[22]. 51–53

Response
Eine Response ist die Antwort des Servers auf eine Anfrage eines Clients[22]. 51, 53

RUP
Rational Unified Process ist ein Vorgehen zur Softwareentwicklung[22]. 115, 123

SQL
Structured Query Language ist eine Datenbanksprache[22]. 34

SQLite
SQLite ist ein relationales Datenbanksystem[22]. 3, 18, 47

VO

Ein Value object enthält einfache Daten, welche in einem Objekt abgespeichert wurden. In dieser Arbeit dienen sie zum Mapping in die Datenbank. Eine andere Bezeichnung für VO ist Entity[22]. 38–40, 47, 53, 69, 88, 95, 104

Widget

Ein Widget ist eine Komponente einer grafischen Benutzeroberfläche. Widgets sind immer in einer bestimmten Benutzeroberfläche eingebunden und nutzen diese zur Interaktion mit dem Anwender oder anderen Widgets der Benutzeroberfläche[22]. 41

XML

Die Extensible Markup Language (XML) ist eine Auszeichnungssprache zur Strukturierung von Daten[22]. 37

Tabellenverzeichnis

Abbildungsverzeichnis

Verzeichnis des Quellcodes

Teil V

Anhang

A Details zu Diagrammen

In diesem Kapitel werden Details zu den Diagrammen erläutert. Dabei handelt es sich um die detaillierte Beschreibung jedes Use Cases.

A.1 Use Cases

Use Case UC01 - Spiel erfassen

Der Benutzer wählt in der Menüansicht den Menüpunkt “Match aufzeichnen“. Er gelangt somit in die Matchaufzeichnungsansicht. Er erzeugt ein Spiel mit den benötigten Informationen und startet das Spiel mit dem Start-Button. Mit dem Start des Matches werden die Event-Buttons aktiv. Für das Erfassen von Events siehe UC02. Zu den Spielzeiten drückt der Benutzer Halbzeitpause, Halbzeitfortsetzung oder Spiel Ende.

Use Case UC02 - Event erfassen

Der Benutzer klickt auf den von ihm gewählten Menübutton. Es öffnet sich ein Dialog. Der Dialog verlangt vom Benutzer Informationen zum Event. Nach dem die Informationen erfasst wurden, klickt der Benutzer auf “Ok“. Der Event wird nun im Liveticker angezeigt.

Use Case UC03 - Spiel betrachten

Der Benutzer wählt in der Menüansicht den Menüpunkt “Übersicht der erfassten Spiele“. Im Übersichtsfenster sind alle bisher erfassten Spiele aufgelistet. Die Spiele, die sich nicht mehr auf dem Tablet befinden, sind deaktiviert bzw. abgedunkelt. Der Benutzer wählt ein Spiel aus, um dessen Event-Liveticker zu betrachten. Falls sich ein Spiel nicht auf dem Tablet befindet, kann es heruntergeladen werden gemäss UC5.

Use Case UC04 - Spiel hochladen

Für diesen Use Case ist der UC14 Serverauthentifizierung Voraussetzung. Der Benutzer verbindet das Tablet mit dem Internet und öffnet das Fenster “Übersicht der erfassten Spiele“ gemäss UC03 Spiel betrachten. Der Benutzer markiert die Spiele, die er auf dem Server hochladen will. Nachdem alle hochzuladenden Spiele ausgewählt sind, drückt der Benutzer auf den Button “Ausgewählte Spiele hochladen“.

Use Case UC05 - Spiel herunterladen

Für diesen Use Case ist der UC14 Serverauthentifizierung Voraussetzung. Der Benutzer verbindet das Tablet mit dem Internet und öffnet das Fenster “Übersicht der erfassten Spiele“ gemäss UC03 Spiel betrachten. Der Benutzer markiert die Spiele, die er vom Server herunterladen will. Nachdem alle herunterzuladenden Spiele ausgewählt sind, drückt der Benutzer auf den Button “Ausgewählte Spiele herunterladen“.

Use Case UC06 - Mannschaft CRUD

Der Benutzer wählt in der Menüansicht den Punkt Teamverwaltung aus. Dort können Mannschaften erstellt, gelöscht, aktualisiert und betrachtet werden. Zu jeder Mannschaft können noch Spieler verwaltet werden gemäss UC07 Spieler CRUD.

Use Case UC07 - Spieler CRUD

Nachdem der Benutzer eine Mannschaft ausgewählt hat, kann er die Spieler erstellen, löschen, aktualisieren und betrachten.

Use Case UC08 - Mannschaft hochladen

Für diesen Use Case ist der UC14 Serverauthentifizierung Voraussetzung. Der Benutzer verbindet das Tablet mit dem Internet und wählt in der Menüansicht den Punkt “Einstellungen“ aus. Dort können die Mannschaften ausgewählt und hochgeladen werden.

Use Case UC09 - Mannschaft herunterladen

Für diesen Use Case ist der UC14 Serverauthentifizierung Voraussetzung. Der Benutzer verbindet das Tablet mit dem Internet und wählt in der Menüansicht den Punkt “Einstellungen“ aus. Dort können die Mannschaften ausgewählt und heruntergeladen werden.

Use Case UC10 - Formation CRUD

Der Benutzer wählt in der Menüansicht den Punkt “Formation“ aus. Der Benutzer kann die Formationen erstellen, betrachten, editieren und löschen.

Use Case UC11 - Formation hochladen

Für diesen Use Case ist der UC14 Serverauthentifizierung Voraussetzung. Der Benutzer verbindet das Tablet mit dem Internet und wählt in der Menüansicht den Punkt “Einstellungen“ aus. Durch diesen können die Formationen ausgewählt und hochgeladen werden.

Use Case UC12 - Formation herunterladen

Für diesen Use Case ist der UC14 Serverauthentifizierung Voraussetzung. Der Benutzer verbindet das Tablet mit dem Internet und wählt in der Menüansicht den Punkt “Einstellungen“ aus. Dort können die Formationen ausgewählt und heruntergeladen werden.

Use Case UC13 - Aufstellung CRUD

Der Benutzer wählt in der Menüansicht den Punkt “Aufstellung“ aus. Durch diesen können Aufstellungen erstellt, gelöscht, aktualisiert und betrachtet werden.

Use Case UC14 - Serverauthentifizierung

Der Benutzer wählt den Menüpunkt “Einstellungen“ aus. Unter der Kategorie “Serververbindung“ gibt er seinen Benutzernamen und das dazugehörige Passwort ein. Danach klickt er auf “Verbinden“, um die Verbindung mit dem Server herzustellen.

Use Case UC15 - Taktik zeichnen

Der Benutzer wählt im Menü den Punkt “Taktik“ aus. Er gelangt somit auf eine Spielfeldansicht. Auf der rechten Seite hat er eine Auflistung der Spieler, welche er per Drap & Drop in das Feld reinziehen kann. Unter dem Spielfeld hat er verschiedene Zeichnungsmöglichkeiten mit denen er eine Taktik auf der Abbildung des Spielfeldes definieren kann.

Use Case UC16 - Spielfeldgrösse mit GPS bestimmen

Der Benutzer hat die Möglichkeit das Spielfeld abzulaufen und die Eckpunkte mittels GPS zu speichern.

Use Case UC17 - Bewegungsaufzeichnung betrachten

Der Benutzer wählt den Menüpunkt “Bewegungsaufzeichnungen“ aus. Danach erreicht er eine View, in welcher er die Spiele mit GPS-Daten findet. Zuerst muss er ein aufgezeichnetes Spiel via DropDown-Feld auswählen. Nachdem er das gewünschte Spiel ausgewählt hat, kann er es sich anschauen.

B Werkzeuge und Tools

In diesem Kapitel werden Werkzeuge und Tools beschrieben, welche im Projekt genutzt wurden.

B.1 Entwicklungs-, Programmierumgebung

B.1.1 IDE

Als IDE dient Eclipse. Die Version lautet “Eclipse Classic 3.7.2“. Diese Applikation dient zur Entwicklung von *CoachAssist*. Die Applikation ist unter dem Link zu finden `http://www.eclipse.org/`.

B.1.2 Android, AVD und SDK

CoachAssist ist mit Android entwickelt. Die Version, für welche die Applikation entwickelt wurde, ist “4.0.3“. Um die Applikation ohne ein Testgeräte zu testen dient der AVD Manager (Android Virtual Device Manager). Der AVD Manager benötigt die Android SDK. Entwicklungsinformationen für Android sind unter dem Link `http://developer.android.com/guide`[5] zu finden. Auf dieser Webseite befindet sich ebenfalls die SDK (`http://developer.android.com/sdk/index.html` und Informationen zum AVD Manager (`http://developer.android.com/guide/developing/devices/index.html`). Die SDK Versionsnummer ist “r18“.

B.2 Libraries

B.2.1 ORMLite

ORMLite ist ein ORM-Framework. Der folgende Codeausschnitt zeigt eine annotierte Klasse. Diese Klasse wird vom Framework in die Datenbank gemapped.

```
1 @DatabaseTable(tableName = "team")
2 public class Team {
```

```
3
4   public static final String CLUB_ID_Name = "club_id";
5
6   @DatabaseField(generatedId = true)
7   private int id;
8   //Normal field which can't be null
9   @DatabaseField(canBeNull = false)
10  private String name;
11  //Foreign field to club with column name "club_id"
12  @DatabaseField(foreign = true, foreignAutoRefresh = true
        , canBeNull = false, columnName = CLUB_ID_NAME)
13  private Club club;
14  //Foreign collection for an 1:n relation
15  @ForeignCollectionField
16  private ForeignCollection<Player> players;
17
18  public Team() {
19    // ORMLite needs a no-arg constructor
20  }
21
22  public int getId() {
23    return id;
24  }
25
26  public void setName(String name) {
27    this.name = name;
28  }
29
30  public String getName() {
31    return name;
32  }
33
34  public void setClub(Club club) {
35    this.club = club;
36  }
37
38  public Club getClub() {
39    return club;
40  }
41
42  public setPlayers(ForeignCollection<Player> players) {
43    this.players = players:
44  }
45
46  public ForeignCollection<Player> getPlayers() {
47    return players;
48  }
49
50 }
```

Codebeispiel B.1: VO-Klasse Team mit Annotationen

Um CRUD-Operationen für diese Klasse verwenden zu können, wird ein Data Access Object benötigt. Mit diesem Objekt können die Query-Abfragen ausgeführt werden. Ein solches Beispiel ist im folgenden Codeausschnitt zu finden.

```
1 //Databasehelper wird für die Erzeugung des Daos benötigt
2 DatabaseHelper helper = DatabaseHelper.getHelper(context);
3 Dao<Team, Integer> daoTeam = helper.getDao(Team.class);
```

```
4  //Erstellen eines Teams
5  Team team = new Team("FC Beispiel");
6  daoTeam.create(team);
7
8  //Updaten eines Teams
9  team.setName("FC Beispiel 2");
10 daoTeam.update(team);
11
12 //Abfrage über die Spalte "club_id" in der Teamtabelle.
       Liefert alle Teams die zu diesem Club gehörten.
13 int id = 0;
14 List<Team> teams = daoTeam.queryForEq(Team.CLUB_ID_NAME,
       id);
15 //Besser Variante für den eben erwähnte Fall wäre club.
       getTeams();
16
17 //Herauslesen aller Teams in der Datenbank
18 List<Team> teams2 = daoTeam.queryForAll();
19
20 //Löschen des Teams aus der Datenbank
21 daoTeam.delete(team);
```

Codebeispiel B.2: Query-Abfragen

Informationen wurden von der Webseite vom ORMLite[21] bezogen. Eine detaillierte Anleitung ist unter dem Link `http://ormlite.com`[21] zu finden. Die verwendete Versionsnummer lautet “4.40“.

B.2.2 Gson

Gson ist eine Java Library, die Java Objects in JSON konvertiert und umgekehrt. Für die Konvertierung wird eine bestehende Java Klasse benötigt. Diese Klasse benötigt die Attribute, welche auch im JSON Konstrukt vorkommen. Dazu werden in dieser Klasse “getter“- und “setter“-Methoden für jedes Attribut benötigt.
Mit der bestehende Klasse Club sieht die Konvertierung nach JSON folgender Massen aus:

```
1 Club club = new Club("FC Beispiel");
2 Gson gson = new Gson();
3 String json = gson.toJson(club);
```

Codebeispiel B.3: Parsing in ein JSON-Konstrukt

Das Quellcodebeispiel B.3 zeigt die Konvertierung ins JSON Objekt. Der umgekehrte Weg wird im nächsten Codebeispiel B.4 beschrieben.

```
1 Gson gson = new Gson();
2 Club club = gson.fromJson(json, Club.class);
```

Codebeispiel B.4: Parsing in ein Java Objekt Club

Es können auch Listen verarbeitet werden. Ein Beispiel dafür ist im Codebeispiel B.5 einzusehen.

```
1 Gson gson = new Gson();
2 Type type = new TypeToken<List<Club>>() {
3 }.getType();
```

```
4 List<Club> clubs = gson.fromJson(json, type);
```

Codebeispiel B.5: Parsing in eine Java Liste von Clubs

Informationen wurden von der Webseite von Gson [17] bezogen. Mehr zu Gson ist unter dem Link `http://code.google.com/p/google-gson`[17] zu finden. Die verwendete Version lautet 2.2.1.

B.3 Analyse und Codedesign Tools

B.3.1 Lint

Lint ist ein Tool, welches das Android Projekt scannt und potentielle Fehlerquellen erkennt und meldet. Einige Beispielfehler, welche im Projekt aufgetreten sind:

- Fehlende Übersetzung
- Nicht genutzte Ressourcen
- Usability-Probleme
- Performance-Probleme

Mehr zu Lint ist unter dem Link `http://tools.android.com/tips/lint`[8] zu finden.

B.3.2 Checkstyle

Checkstyle durchsucht das Projekt und überprüft ob die Coding-Standards eingehalten wurden. Das Definitionsfile ist im Projekt unter Analysis zu finden. Mehr zu Checkstyle ist unter dem Link `http://checkstyle.sourceforge.net/` zu finden.

B.3.3 FindBugs

FindBugs analysiert das Projekt und sucht nach potentiellen Bugs. Mehr zu FindsBugs ist unter dem Link `http://findbugs.sourceforge.net/` zu finden.

B.3.4 Stan4J

Stan4J ist ein Strukturanalyse-Tool. Es überprüft den Code und zeigt Abhängigkeiten auf. Mehr zu Stan4J ist unter dem Link `http://stan4j.com/` zu finden.

B.3.5 JAutoDoc

JAutoDoc generiert automatisch die Javadoc und File headers für den Code. Die generierte Javadoc wurde jedoch manuell überarbeitet. Mehr zu JAutoDoc ist unter dem Link `http://jautodoc.sourceforge.net/` zu finden.

B.4 Testing Tools

B.4.1 Monkey

Monkey ist ein Android Testing-Tool. Es führt eine vordefinierte Anzahl Klicks auf der GUI aus. Der Klick-Ort ist zufällig und so kann ein grosser Teil der Applikation durchgeklickt werden. Mehr zu Monkey ist unter dem folgenden Link `http://developer.android.com/guide/developing/tools/monkey.html`[10] zu finden.

B.4.2 Robotium

Robotium ist ein Android Testing-Tool. Es ermöglicht die Ausführung von Systemtext. Es wird auf der GUI ausgeführt. In der Testklasse wird definiert, welche Buttons angeklickt werden sollen. In diesem Projekt wurden damit die Use Case Abläufte getestet. Mehr zu Robotium ist unter dem Link `http://code.google.com/p/robotium/`[11] zu finden.

B.5 Projektverwaltungs Tools

B.5.1 GitHub

Das Repository ist auf GitHub abgelegt. Es befindet sich unter dem Link `https://github.com/MVandort/CoachAssist`.
Das das dazugehörige Testprojekt befindet sich auf `https://github.com/flow666/CoachAssistTesting`.

B.5.2 Redmine

Für die Verwaltung der Zeitplans und der Arbeitspakete dient Redmine. Redmine ist eine webbasierte Projekt Management Applikation. Mehr dazu ist unter dem Link `http://www.redmine.org` zu finden.

B.5.3 TexMaker, Latex und Dokumentation

Für die Dokumentation wurde Latex benutzt. Als Editor wurde TexMaker verwendet. Die verwendete Version ist 3.3.4. Der Vorteil dieses Editors ist, dass er auf mehreren Plattformen läuft. Die unterstützten Plattformen sind Linux, Mac und Windows. Der Editor ist unter dem Link `www.xm1math.net/texmaker` zu finden. Für die Latex-Datei wurden Informationen von der Vorlage der Uni Magdeburg[18] bezogen. Informationen für die Dokumentation wurden von der cnlab Webseite[3] bezogen.

C User Interface

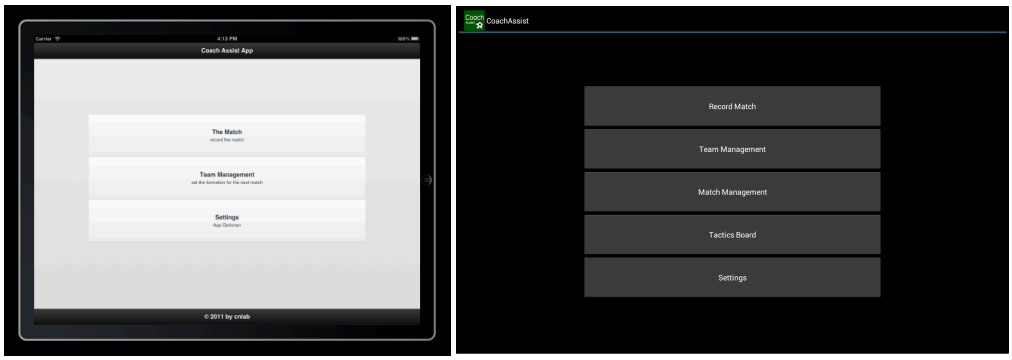
Das Design des User Interface wurde aus der Semesterarbeit übernommen. In der Semesterarbeit wurde zuerst ein Paper Prototyp erstellt und dieser wurde ausgiebig getestet. In diesem Kapitel wird abgehandelt, inwiefern sich das User Interface von der Semesterarbeit zur Bachelorarbeit unterscheidet. Des weiteren wird der Grund für die Änderung erläutert.

C.1 Vergleich User Interface SA zur BA

Menü

Das Menü ist vom Design-Aufbau her gleich geblieben. Im Bild C.1a ist die alte Version des User Interfaces zu sehen. Im Bild C.1b ist die neue Version zu betrachten.

Abbildung C.1: Vergleich User Interface Menü



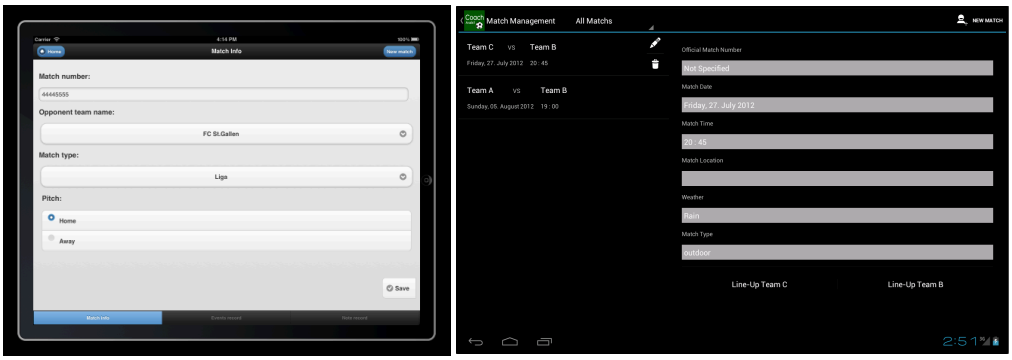
(a) Altes Menü (b) Neues Menü

Match Management

Beim User Interface, für das Einlesen von Matchinformationen, haben sich in den Grundzügen bereits Änderungen manifestiert. Das Match kann im Vorhinein über die Matchverwaltung aufgerufen werden, wie im Bild C.2b gezeigt wird. In der alten Version von *CoachAssist* wird es jeweils vor dem

Aufzeichnen eines Fussballspiels eingeblendet und muss somit gleich ausgefüllt werden. In der Abbildung C.2a ist dieses Phänomen ersichtlich. Deshalb hat sich der View-Aufbau geändert. Neben dem Formular ist in der neuen Version eine Liste der Spiele zu sehen. Dies hat die View wesentlich mehr eingeschränkt. Für das Verhältnis wurde, wie bei den anderen Views, der goldene Schnitt genommen.

Abbildung C.2: Vergleich User Interface Match Mgmt

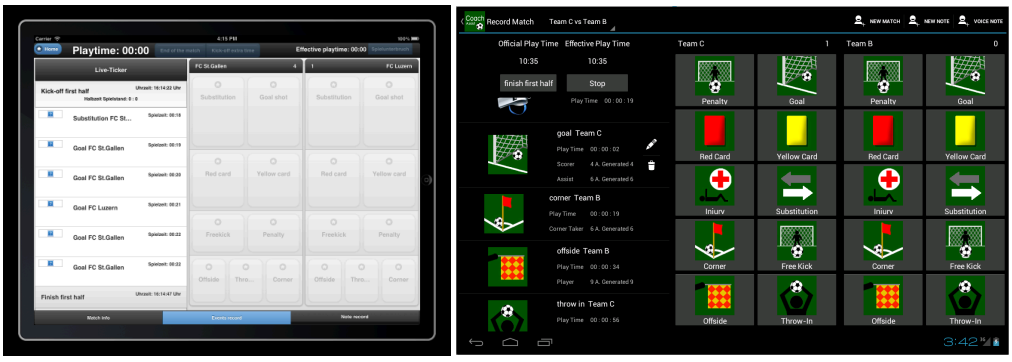


(a) Alte Match Mgmt View (b) Neue Match Mgmt View

Matchaufnahme

Die Struktur des User Interface für die Matchaufnahme ist gleich geblieben. Dies kann in den zwei Bildern C.3a und C.3b betrachtet werden. Geändert hat sich in der neuen Version zum einen die kleine Box über dem Liveticker. Der Status des Spieles und der Start-Button wurde zentraler dargestellt. Grund dafür war das Platzproblem in der oberen Leiste. Ein weiterer Punkt, der sich in der neuen Version verbessert hat, ist die Grösse der Buttons. Es wurde speziell vom Industriepartner gewünscht, dass alle Event-Buttons die gleiche Grösse haben. In der alten Version wurden die Event-Buttons nach Nutzungshäufigkeit entweder grösser oder kleiner dargestellt.

Abbildung C.3: Vergleich User Interface Match Aufnahme



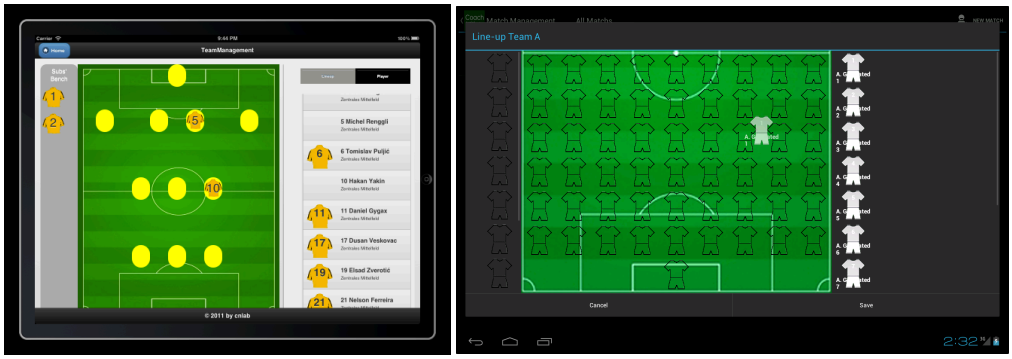
(a) Alte Match Record View (b) Neue Match Record View

Aufstellung

Die Struktur des Aufstellung User Interfaces hat sich nicht wesentlich verändert. In der alten Version C.4a und der neuen Version C.4b ist die Aufteilung der Ersatzbank, des Spielfeldes und der Spielerliste gleich geblieben.

Der Back Button ist in der neuen Version weggefallen. Dies hat den Grund, dass der Aufrufpunkt dieser View sich geändert hat.

Abbildung C.4: Vergleich User Interface Aufstellung

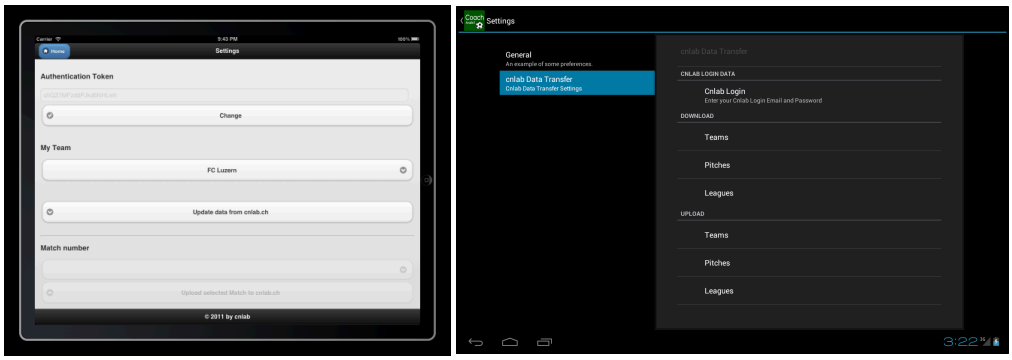


(a) Alte Formation View (b) Neue Formation View

Transfer

In der Transfer View haben ebenfalls Änderungen stattgefunden. In der alten Version C.5a erstrecken sich die Eingabefelder über die ganze Bildschirmbreite. Dies war in der neuen Version C.5b nicht mehr möglich. Die neue Transfer View ist in den Einstellungen eingebaut. Aufgrund der Android Richtlinien für die Einstellungen wurde die View neu aufgebaut. Die Eingabefelder mussten neben der Liste der Einstellungen platziert werden.

Abbildung C.5: Vergleich User Interface Transfer



(a) Alte Transfer View (b) Neue Transfer View

Allgemeine Informationen zur GUI

Designer Thomas Heinzmann erstellte die Icons. Jedoch konnten nicht alle bis Ende der Bachelorarbeit geliefert werden. Deshalb wurden in der Applikation Bilder verwendet, welche nicht veröffentlicht werden dürfen. Dies Betrifft die Bilder Anpfiff, Notiz und das Fussballfeld. Diese Bilder werden cnlab vor der Veröffentlichung ausgetauscht.

D Qualitätsmanagement

Im Kapitel 6 Testing sind die primären Qualitätsmassnahmen beschrieben. In diesem Kapitel sind zusätzliche Massnahmen zur Qualitätssicherung beschrieben.

D.1 Reviews

Reviews wurden zu den RUP-Dokumente durchgeführt. Die Dokumente wurden jeweils zum Ende einer Iteration dem Betreuer abgegeben. Die Enden jeder Iteration wurden als Meilensteine definiert. Die RUP-Dokumente waren somit Lieferobjekte für die Meilensteine. Die Dokumente wurden vom Betreuer und vom Industriepartner durchgesehen und dazu wurde ein Feedback abgegeben. Diese wurde am Ende anhand der Bemerkungen überarbeitet.

D.2 Code

Die Code Qualität wurde mit Analyse-Tools verbessert. Wie im Kapitel 2.4 erwähnt, wurden Code-Guidelines miteinbezogen. Folgende Code-Tools wurden verwendet:

- Lint
- BugFinder
- CheckStyle

Für CheckStyle musste ein Definitionsdatei erstellt werden, welche den Android generierten Code ignoriert. Diese befindet sich im Projektworkspace im Ordner Analysis. Ebenfalls mussten die Naming Convetions für die DTOs-Klassen ignoriert werden. Diese können der Naming Convention nicht entsprechen, da sie von Gson in JSON umgewandelt werden. Dies hätte zur Folge, dass das JSON nicht mehr stimmen würde. Eine Beschreibung der erwähnten Tools ist im Kapitel B zu finden.

E Projekt-Management

In diesem Kapitel wird das Projekt-Management erläutert. Bestandteil dieses Kapitel ist das Vorgehen, Auswertung des Zeitplans und das Risikomanagement des Projektes.

E.1 Planung

E.1.1 RUP Itertionen

In dieser Bachelorarbeit wurden nach Rational Unified Process (RUP) vorgegangen, jedoch nicht strikt. Dies bedeutet, dass nur die RUP-Dokumente geschrieben wurden, welche nützlich für diese Bachelorarbeit waren. Die Architektur- und Qualitätsdokumente wurden direkt im Hauptbericht geschrieben. Der Projektplan, das Requirement- und das Analyse Dokument wurden erfasst. Diese sind in der CD enthalten.
Gemäss RUP wurden folgende Iterationen vorgängig definiert.

- Inception (Woche 1-2)
  - I1 (Woche 1-2)
    - In dieser Phase werden Projektantrag und Projektplan definiert.
    - Zusätzlich werden in dieser Phase die Funktionen definiert.
- Elaboration (Woche 3-6)
  - E1 (Woche 3-4)
    - In dieser Phase wir die Analyse durchgeführt. Das heisst, es wird ein Domainmodell erstellt. Zudem werden die Anforderungsspezifikationen definiert.
  - E2 (Woche 5-6)
    - Der Schwerpunkt dieser Phase liegt auf der Erstellung des Prototyps.

- Construction (Woche 7-14)
  - C1 (Woche 7-8)
    - In dieser Phase werden die definierten Funktionen bestimmt (siehe Funktionsliste zu Meilenstein 1)
  - C2 (Woche 9-10)
    - In dieser Phase werden die definierten Funktionen bestimmt (siehe Funktionsliste zu Meilenstein 6)
  - C3 (Woche 11-12)
    - In dieser Phase werden die definierten Funktionen bestimmt (siehe Funktionsliste zu Meilenstein 7)
  - C4 (Woche 13-14)
    - In dieser Phase werden die definierten Funktionen bestimmt (siehe Funktionsliste zu Meilenstein 8)
- Transition (Woche 15-16)
  - T1 (Woche 15-16)
    - In dieser Phase wird das Programm exportiert bzw. das Apk-File wird erstellt und es werden restliche Abschlussarbeiten erledigt.

Die Funktionsliste mit der Zuordnung der Meilensteine befindet sich auf der CD.

E.1.2 Meilensteine

Für das Projekt wurden folgende Meilensteine definiert:

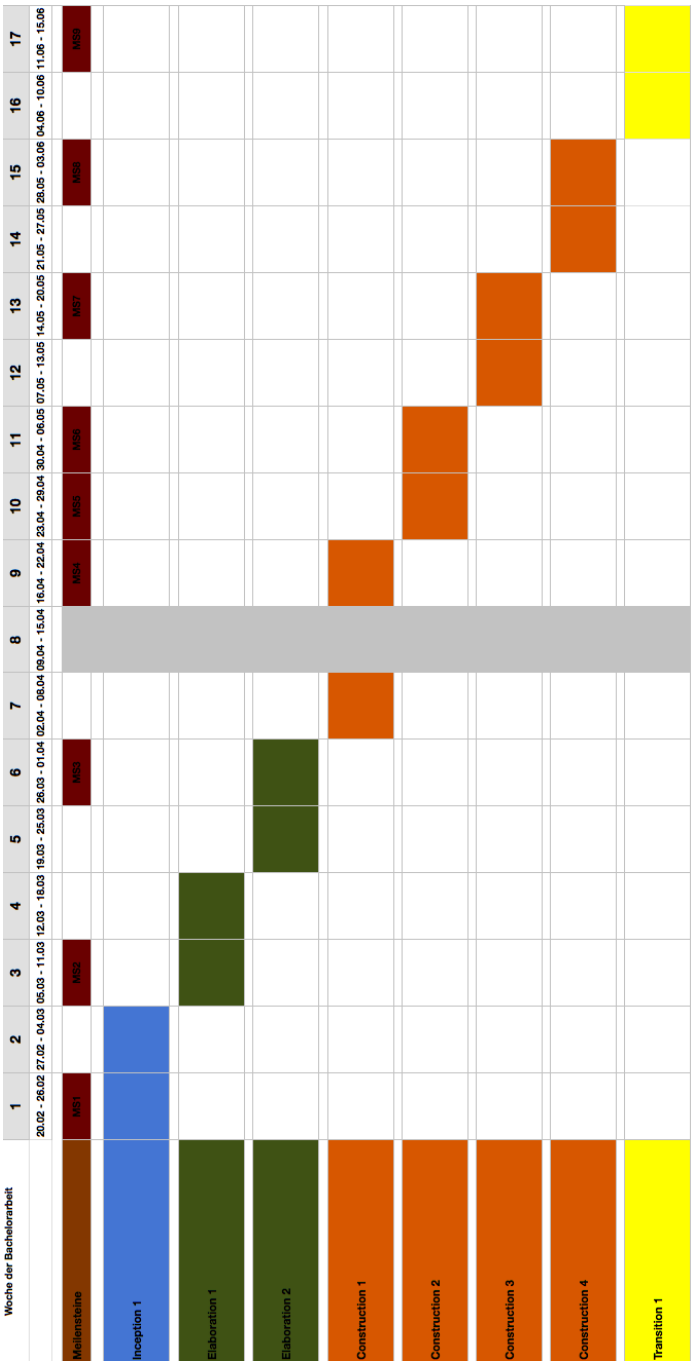
1. Projektantrag besprochen
2. Funktionsliste und Zeitplan erstellt und besprochen
3. Funktionierender Prototyp und Designdokumente
4. Softwareversion mit Features aus C1
5. Zwischenpräsentation
6. Softwareversion mit Features aus C2
7. Softwareversion mit Features aus C3
8. Softwareversion mit Features aus C4, Testprotokolle und Puffer. Aufgrund des Puffers sind Funktionen gekennzeichet, welche nur evtl. umgesetzt werden.
9. Abgabe des Berichtes und Schlusspräsentation

Die Meilensteine sind jeweils nach einer Iteration gesetzt. Die ersten zwei Meilensteine sind hierbei eine Ausnahme, diese wurden beide in der Inception Phase definiert.

E.2 Zeitplanung

Es stehen insgesamt 720 Stunden zur Verfügung. Pro Person ergibt das somit 360 Stunden pro Person. Diese Dauer ist auf 16 Wochen verteilt. Dies ergibt 22.5 Stunden pro Woche pro Person. Jedoch wurden schlussendlich 369 Stunden pro Student eingeplant. Das gibt eine gesamt Summe von 738 Stunden. Die Abbildung E.1 zeigt die Übersicht der Planung. Es werden die definierten Meilensteine und Iterationen aus dem Kapitel E.1 mit einbezogen.

Abbildung E.1: Zeitplan

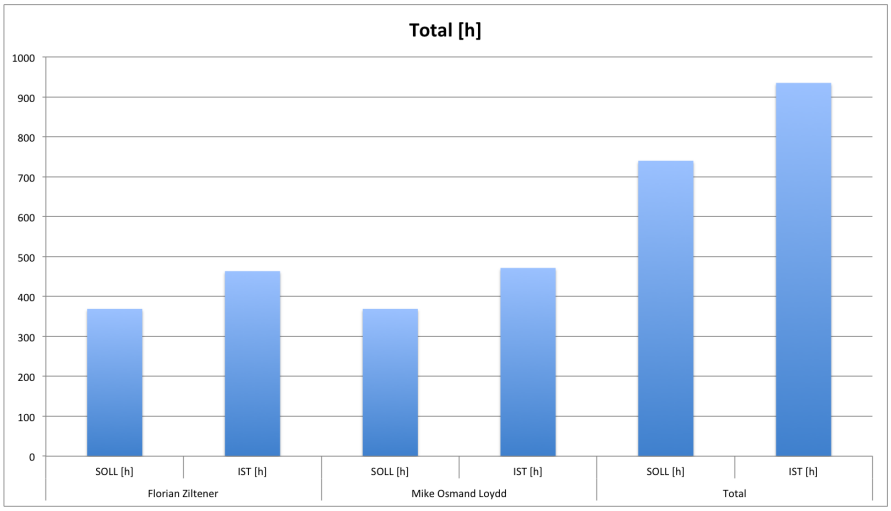


Eine Word und Excel-Version sind auf der CD vorhanden.

E.3 Auswertung des Zeitplans

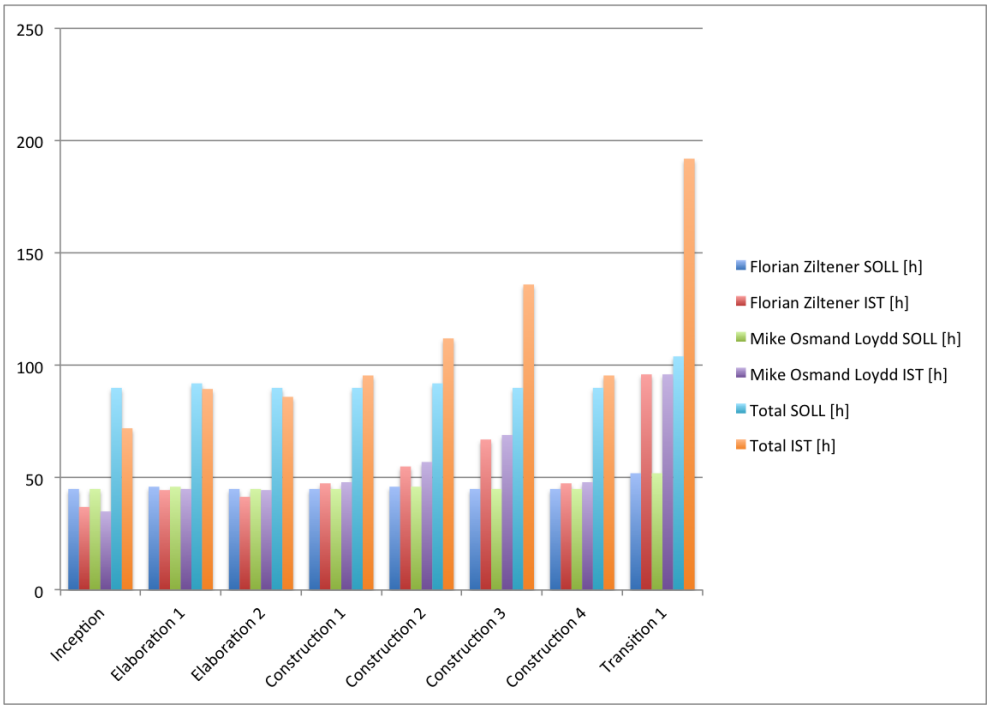
Die Abbildung E.2 zeigt die Total aufgewendeten Stunden pro Student und die gesamte Summe. Der Mehraufwand ist zu Begründen, dass nach der Zwischenpräsentation der Stand noch nicht erreicht wurde, der erwartet wurde. Ein weiterer Grund für den Mehraufwand war das Einarbeiten in Android 4.0 mit den Fragments.

Abbildung E.2: Zeitplanauswertung Total



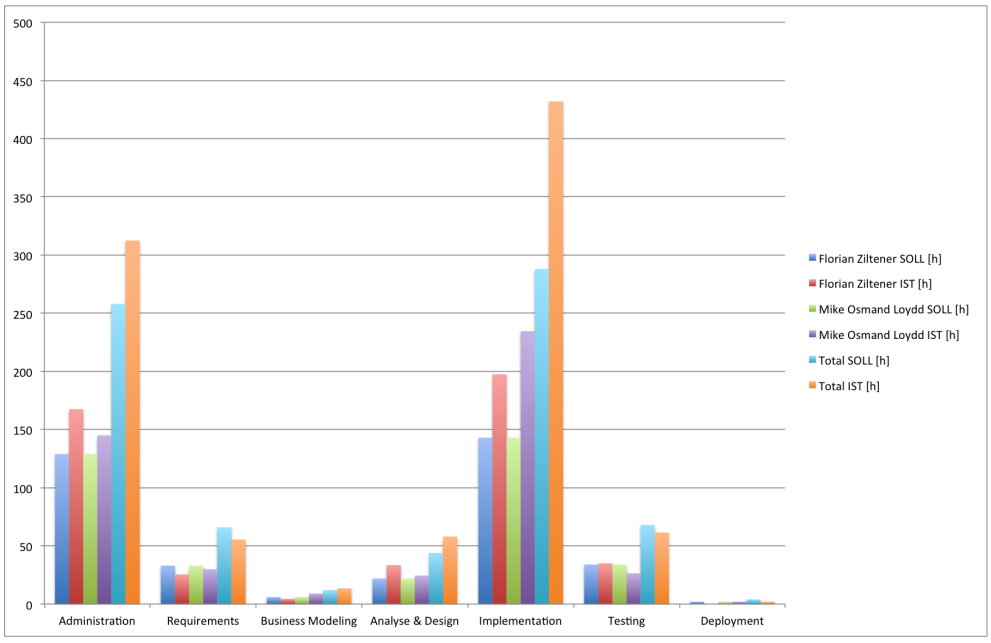
Die Abbildung E.3 zeigt die Auswertung nach den Iterationen. In dieser Abbildung ist ebenfalls zu erkennen, dass nach der Zwischenpräsentation bzw. Construction-Phase 2 der Aufwand zugenommen hat. In der letzten zwei Wochen wurden ebenfalls mehr Stunden geleistet. Dies lässt sich mit dem früheren Semesterende erklären. In den letzten zwei Wochen wurde jeden Tag an der Bachelorarbeit gearbeitet.

Abbildung E.3: Zeitplanauswertung Iteratinonen



Die Abbildung E.4 zeigt die Auswertung nach Arbeitspaketen. Im Arbeitspaket Administration ist die Dokumentation enthalten. Es wurde leider zu wenig Zeit für die Umsetzung geplant. Ebenfalls wurde mehr Zeit im Arbeitspaket Implementation aufgewendet. Dies hatte mehrere Gründe zur Folge. Den grössten zusätzlichen Aufwand bescherte die Implementation der Listview mit den Image Buttons beim scrollen. Ebenfalls Probleme bereitet die Kommunikation mit dem Server. Teilweise funktionierte der Server nicht mehr. Dies führte zu mehr Aufwand bei der überflüssigen Fehlersuche. Das Framework ORMLite machte mit dem automatischen Erneuern der Datenbankeinträge Probleme. Das beseitigen dieser Probleme kostete ca. zwei Tage. Ein weiterer Punkt den das Framework betrifft, ist das nicht unterstützen von Vererbung. Deshalb musste eine alternativ Lösung für die Umsetzung des Event-Speicherns gefunden werden.

Abbildung E.4: Zeitplanauswertung Arbeitspakete



E.4 Risikoanalyse

Abbildung E.5: Risikomanagement

Risikobeurteilung

Risk ID	Bereich	Risikobeschrieb	Auswirkung	(Vorbeugungs-) Massnahme	Kosten der Massnahme in Stunden	Max. Schaden in Stunden	Eintrittswahrscheinlichkeit während dem Projekt	Gewichteter Schaden in Stunden	Status
R01	Mensch / Personal	Krankheit / Ausfall eines Teammitglieds	Mehraufwand für das andere Teammitglied/ Zeitverlust / Kann zu R02 und R08 führen	Vorarbeiten oder Spareitime im Zeitplan einfügen	20	32	10%	3.20	eingetroffen
R02	Planung	vereinbarten Minimal-Anforderungen werden nicht erfüllt	Projektziel wird nicht erfüllt. Folge: Semesterarbeit nicht bestanden	Fortlaufend Feedback vom Betreuer/Dozent einholen. Allenfalls Anforderungen anpassen	10	720	5%	36.00	offen
R03	Daten	Datenverlust Dokumentation	Dokumente müssen neu geschrieben werden	Die Dokumentationen wöchentlich sichern mit Redmine. Alternativlösung wäre Dropbox.	5	80	5%	4.00	offen
R04	Daten	Datenverlust Code	Der Code muss nochmal generiert werden	Mit einer Repository (SVN, Git) arbeiten.	5	450	5%	22.50	offen
R05	Infrastruktur	Ausfall Arbeitsplatz (Computer)	Betroffene Person kann einige Zeit nicht weiterarbeiten / kann zu Risiko R08, R02, R03 oder R04 führen	Notfall-Arbeitsplatz bereit halten / einrichten	5	10	5%	0.50	offen
R06	Mensch / Personal	Meinungsverschiedenheit im Team	Kommunikation ist geschädigt / Arbeitsunterbruch / Projektabbruch	Teambildende Massnahmen durchführen	10	20	5%	1.00	eingetroffen
R07	Mensch / Personal	Missverständnise mit dem Kunden onlab	Kommunikation ist geschädigt / kann zu Projektabbruch führen	Den Kunden laufend über den Projektstatus informieren / Unklarheiten im Voraus oder sofort ausdiskutieren	20	80	10%	8.00	offen
R08	Implementation	Schwerwiegender Fehler beim Testen entdecken	Fehler aufspüren und beheben / kann zu R08 und R02 führen	Unit Tests vor der Implementation erstellen / Jede Funktion nach Implementation testen	60	120	15%	18.00	offen
R09	Implementation	Komplikationen mit Third-Party Applikationen	Zeitverlust / selbst implementieren	Alternativlösungen bereithalten / Manual durchlesen	5	20	15%	3.00	offen
R10	Knowledge	Nicht ausreichende Kenntnisse in der neuen Technologie	Unerwartete Fehlermeldungen bzw. Probleme / Kann sein das einige Features mit der Technologie nicht umgesetzt werden können	Vor dem Projektstart in die neue Technologie einlernen bzw. Überprüfen, ob die Minimal-Anforderungen implementiert werden können	20	40	10%	4.00	offen
		Total Kosten in Arbeitspaketen enthalten			160				
		Total Rückstellungen						100.20	

Eintrittswahrscheinlichkeit während dem Projekt:	15%	10%	5%
	häufig	gelegentlich	selten

F Persönlicher Berichte

F.1 Persönlicher Bericht Mike Osmand Loydd

F.1.1 Projektverlauf

Am Anfang hatte ich meine Bedenken, als wir entschieden den Prototyp aus der Semesterarbeit native in Android zu entwickeln. Da ich von schlechten Erfahrungen, welche mit Android 3.0 (Android Tablet Version) gemacht wurden, gehört habe. Ausserdem wollte ich eigentlich ein komplett anderes Themengebiet bearbeiten. Die Bedenken verschwand sehr schnell als der Hersteller des Entwicklungsgeräts während der Inception Phase das Update auf Android 4.0 herausbrachte.
Dank der Vorarbeit in der Semesterarbeit konnten wir früh mit der Implementation beginnen. Am Anfang der Implementation setzten wir uns intensiv mit Android 4.0 und den Android Design Guidelines auseinander. Die Implementation der ListView mit mehreren Typen und viele weitere kleine Probleme nahmen viel Zeit in Anspruch. Aufgrund dieser Probleme und der Zwischenpräsentation konnte wir nicht früher mit den Usability Tests anfangen. Das Testing wurde uns durch die Android Testing Tools vereinfacht. Viele verdeckte Bugs konnten entdeckt und behoben werden.
Durch viele Nachtschichten und Wochenendarbeiten konnten wir den Zeitplan mehrheitlich einhalten.

F.1.2 Was habe ich gelernt

Die Bachelorarbeit gab mir die Möglichkeit mich intensiver mit Android Development auseinander zu setzen. Obwohl ich schon vor dem *CoachAssist* eine kleine Applikation in Android 2.3.3 geschrieben hatte, konnte ich meine Fähigkeiten in Android Entwicklung durch die Bachelorarbeit vertiefen und erweitern. Die von Android zur Verfügung gestellten, neuen Testing-Tools sind sehr hilfreich, vorallem das Tool Monkey. Die Android 4.0 Entwicklung ist aus meiner Sicht viel einfacher und schöner als die Android Entwicklung mit den vorherigen Versionen.
Ein grosser Vorteil, den wir in der Bachelorarbeit im Vergleich zur Semester-

arbeit hatten, war die grosse Android Community. Für viele Probleme, leider nicht für alle, konnte ich in kürzester Zeit eine Lösung finden. Das einzige Problem, für welches ich länger brauchte, war die Implementation der ListView mit mehreren Typen. Besonders herausfordernd war die Implementation des Liveticker ListView.

F.1.3 Fazit

Ich bin sehr erfreut, dass wir die gewünschten Grundfunktionen implementieren konnten und, dass die Applikation stabil läuft. Die native Entwicklung hat uns viele Probleme, welche wir in der Semesterarbeit hatten, erspart. Und die Performance, wie gewünscht, verbessert. Das nächste Erfreuliche war, dass wir in der Bachelorarbeit, die Fehler aus der Semesterarbeit vermeiden konnten.
Im Grossen und Ganzen bin ich mit dem Projektverlauf und dem Endprodukt sehr zufrieden. Die Arbeit hat nicht nur viel Spass bereitet, sondern war auch sehr lehrreich. Ich danke cnlab für die sehr angenehme Zusammenarbeit und für die Möglichkeit *CoachAssist* in Android zu implementieren. Ich freue mich darauf Applikation im Market zu sehen und für den Eigennutzen herunterzuladen.

F.2 Persönlicher Bericht Florian Ziltener

F.2.1 Projektverlauf

Zu Beginn war ich ein wenig skeptisch dem Projekt gegenüber. Da wir es bereits in der Semesterarbeit mit einer anderen Technologie realisiert hatten. Ich hatte Befürchtungen, dass wir Altlasten aus der Semesterarbeit mitschleppen und diese in der Bachelorarbeit Probleme verursachen könnten. Dies hat sich, meiner Meinung nach, teilweise gezeigt. In der Zwischenpräsentation hätten wir, gemäss dem Gegenleser, schon viel weiter sein sollen. Dies hat die Stimmung nicht nur bei mir, sondern auch bei meinem Projektpartner getrübt. Jedoch haben wir dies auch als Chance wahrgenommen und uns nochmals Mühe gegeben. Zum Projektverlauf selbst bin ich der Ansicht, dass wir den Zeitplan gut eingehalten haben. Teilweise gab es auch Problem mit Android, da es sich hierbei um eine neue Technologie gehandelt hat. Dies hat uns auch schon während dem Semesterarbeit ein paar Nachtschichten beschert. Im Zeitplan hatten wir die ersten Wochen für die Einarbeitung in das Projekt und die Überarbeitung der Semesterarbeit genutzt. In den nächsten vier Wochen wurde ein Prototyp erstellt und das Softwaredesign festgelegt. Die RUP-Dokumente haben uns in den Phasen recht viel Zeit gekostet. Nach der Elaboration gings in die Construction Phase, in der wir die Featueres implementiert haben. Dies hat mir persönlich am meisten Spass gemacht. Nach dieser Phase kam noch die Transition-Phase, in der wir die Schlussdokumentation fertig gestellt habe.

F.2.2 Was habe ich gelernt

Das Wichtigste, welches ich persönlich aus dieser Arbeit mitnehme, ist das Wissen über die neue Technologie Android. Nebst der Android Programmierung konnte ich mich auch in die zwei Libraries Gson und ORMLite einarbeiten. Einen weiteren Punkt, den ich an Android sehr schätze, ist die vielfältige Möglichkeit zum Testen der Applikation.
Nebst den neuen Technolgien konnte ich ebenfalls mein Wissen in RUP steigern.

F.2.3 Fazit

Die wichtigsten Funktionen wurden realisiert, jedoch hätte ich gerne die eine oder andere Funktion mit einer geringeren Priorität aus persönlichem Interesse implementiert. Die Endversion ist nutzbar und sieht vom Design her ansprechend aus. Jedoch müssten noch ein weiterer Usability Test gemacht werden, um das User Interace zu optimieren.
Mit der aktuellen Version der Applikation sollten die Perfomance-Probleme aus der Semesterarbeit beseitigt sein. Der User erhält wesentlich schneller eine Antwort vom System wie zuvor. Meiner Meinung nach ist das Ziel der Bachelorarbeit erfüllt.
Abschliessend möchte ich cnlab und Prof. Dr. Heinzmann für die angenehme und zufriedenstellende Arbeit danken.