Fachhochschule
Kaiserslautern
University of Applied Sciences

Informatik und
Mikrosystemtechnik
Zweibrücken

Studiengang Informatik (Master)

Mobile Anwendungen mit Android

Sommersemester 2013

Sport-Ergebnis-App:
Spot Dat Sports Data

von

Markus Henn
Matrikel-Nummer: 861014

5. August 2013

Inhaltsverzeichnis

1 Ziele

1.1 Projektbeschreibung

Ziel ist die Entwicklung einer Anwendung zur Ansicht von Sportergebnissen, insbesondere der Sportart Fußball.

1.2 Anwendungsanforderungen

Die im Folgenden genannten Anforderungen haben sinkende Priorität in der Reihenfolge der Nennung.

1. Daten legal aus einer Quelle im Internet beziehen
2. Anzeige und Auswahl zur Verfügung stehender Wettbewerbe
3. Anzeige und Auswahl der Spieltage/Runden eines Wettbewerbs
4. Anzeige der Spielbegegnungen eines Spieltages/einer Runde mit Ergebnis, geplanter Anstoßzeit bzw. aktuellem Zwischenstand
5. Anzeige des Veranstaltungsortes einer Spielbegegnung
6. Anzeige von Ereignissen innerhalb einer Spielbegegnung (z. B. Tore/Karten mit Minute und Name des Torschützen bzw. der bestraften Person)
7. Anlegen und Anzeige von favorisierten Wettbewerben/Mannschaften und deren Spielbegegnungen
8. Kalender-Export einzelner Begegnungen
9. Geo-Export des Veranstaltungsortes
10. Anzeige einer aktuellen Tabelle des Wettbewerbs
11. Benachrichtigung bei Ereignissen innerhalb einer Spielbegegnung favorisierter Wettbewerbe/Mannschaften
12. Startseite der App konfigurierbar
13. Eingeschränkte Nutzung der App bei Offline-Betrieb ermöglichen
14. Anlegen neuer Daten (Wettbewerbe, Spieltage, Spielbegegnungen, Ergebnisse, usw.)
15. Editieren vorhandener Daten

2 Herangehensweise

Nach Definition der Ziele wurde eine Datenquelle als Grundlage der Anwendung gesucht und ausgewählt (Kapitel 2.1). Im Folgenden wurde die Planung (Kapitel 2.2) konkretisiert und eine erste Oberfläche, die die von der Datenquelle bereitgestellten Wettbewerbe auflistet (Kapitel 2.3), erzeugt. Danach wurde die Anwendung über Spieltage (Kapitel 2.4), Spielbegegnungen (Kapitel 2.5) und Spielbegegnungsdetails (Kapitel 2.6) verfeinert.

2.1 Datenquelle und -anbindung

Grundpfeiler der geplanten Anwendung sind die Daten von Sportveranstaltungen. In der vorbereitenden Recherche wurden zwei Datenbanken, die solche Daten zur Verfügung stellen und einen kostenlosen Zugriff erlauben, gefunden:

1. **openfooty**[1]. Openfooty bietet Daten zu Spielern, Teams und Spielbegegnungen verschiedener Fußball-Wettbewerbe. Zur nicht kommerziellen Nutzung kann ein kostenloser API-Schlüssel angefordert werden.
2. **OpenLigaDB**[2]. OpenLigaDB bietet "communitybasierte Sport-Ergebnisse". Auch dieser Anbieter hat grundlegende Daten zu Spielbegegnungen verschiedener Wettbewerbe aus der Welt des Fußballs, aber auch anderer Sportarten, im Angebot.

Bei der Recherche zeigte sich, dass einige Entwickler, die bei openfooty einen API-Schlüssel beantragten, keine Antwort und somit auch keinen Zugang zu den Daten erhielten. Die Wahl fiel deshalb auf die OpenLigaDB, wo zur Afrage der Daten kein API-Schlüssel benötigt wird.

OpenLigaDB stellt eine SOAP-Schnittstelle zur Verfügung. Für die Android-Plattform gibt es eine spezielle Variante der Java-Bibliothek „kSOAP 2“[3]: ksoap2-android[4]. Diese wird zur Anbindung der OpenLigaDB in diesem Projekt verwendet. SoapUI[5] wurde bei der Anbindung an die SOAP-Schnittstelle unterstützend eingesetzt. Beispielszenarien konnten damit einfach nachvollzogen und Fehlerfälle analysiert werden.

Die Verwendung von OpenLigaDB brachte aber auch einige Probleme mit sich. Die Schnittstellen-Definition nach WSDL (http://www.openligadb.de/Webservices/Sportsdata.asmx?WSDL) weist einige falsche Angaben auf. So sind z. B. angeblich optionale Request-Parameter in Wirklichkeit Pflicht. Andere Randbedingungen sind der WSDL oder der OpenLigaDB-Website nicht zu entnehmen. Wie sind Spieler hinterlegt, bei denen das Ergebnis noch nicht bekannt ist? Wie sind Spielereignisse, die noch keinem bestimmten Spieler zugeordnet wurden, gespeichert? Diese Unsicherheiten sind bei beim Entwurf besonders zu beachten.

2.2 Systemanalyse

Die Anwendung setzt sich hauptsächlich aus den drei Komponenten Datenquelle, Datenmodell und User Interface/Adapter zusammen (siehe Abbildung 1). Die Datenquelle wird durch die OpenLigaDB verwirklicht. Zur Erhaltung der Funktionalität bei Offline-Betrieb und zur Schonung der Datentransfer-Kapazitäten könnte in einem solchen Fall stattdessen ein zuvor gefüllter Cache zumindest einen Teil der Daten bereitstellen. Das Klassendiagramm (Analyse) der Modell-Komponente ist in Abbildung 2 zu sehen.

Da im Haupt-Thread der Activity keine längeren blockierenden Aufgaben erfüllt werden dürfen, ist für die über das Internet stattfindende Datenbeschaffung eine asynchrone Verarbeitung in einem separatem Thread notwendig. Die Activity-Klassen implementieren verschiedene Receiver-Schnittstellen und registrieren sich beim Daten-Provider als Daten-Receiver (siehe Abbildung 3). Im Falle von Problemen bei der Beantwortung der Anfragen, müssen vom Daten-Provider geworfene Exceptions abgefangen bzw. aufgerufene Fehlermethoden implementiert werden.

1 openfooty, http://www.footytube.com/openfooty/
2 OpenLigaDB, http://www.openligadb.de/
3 kSOAP 2, http://ksoap2.sourceforge.net/
4 ksoap2-android, MIT-Lizenz, https://code.google.com/p/ksoap2-android/
5 SoapUI, Freie Software, http://www.soapui.org/

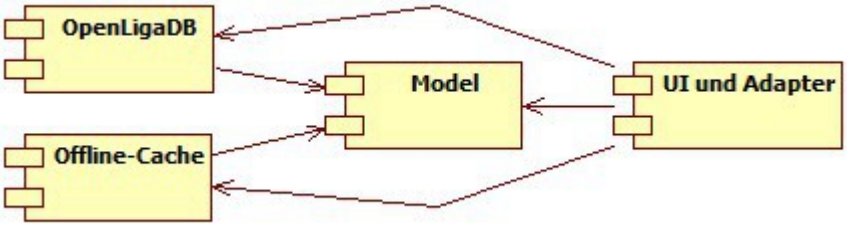


Abbildung 1: Komponentenmodell

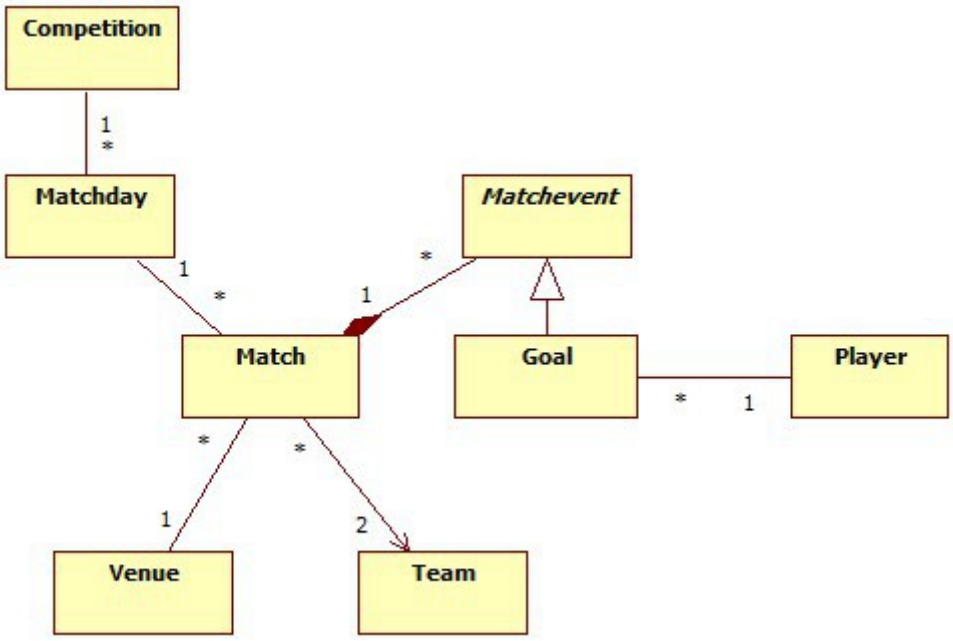


Abbildung 2: Klassenmodell der Model-Komponente

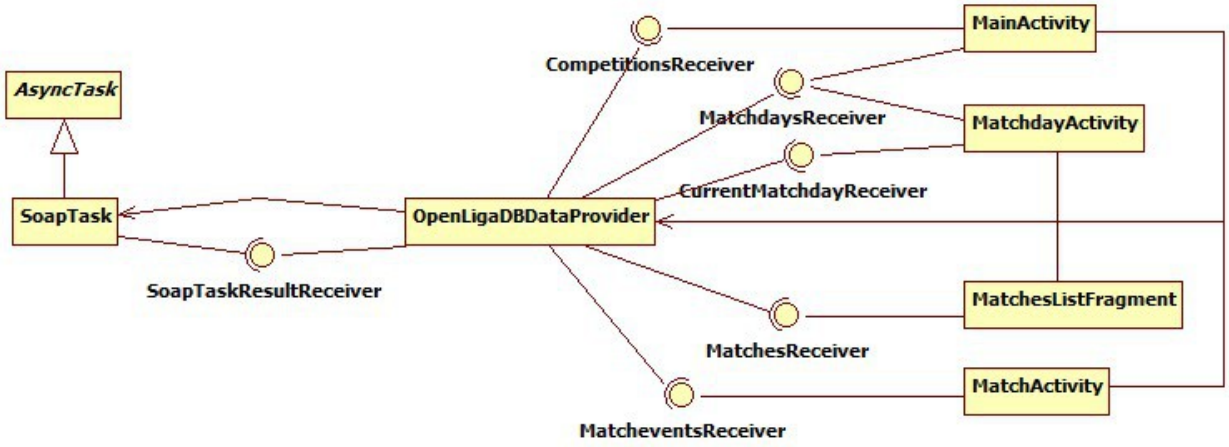


Abbildung 3: Schnittstellen

2.3 Verfügbare Wettbewerbe

Die Implementierung wurde mit der Auslese und Anzeige verfügbarer Wettbewerbe begonnen. Die dazugehörige `MainActivity` stellt den Eingangpunkt der App dar (siehe Abbildung 4). Sie implementiert die Schnittstelle `CompetitionsReceiver`. Das Layout dieser Activity besteht ausschließlich aus der Liste der Wettbewerbe. Ein Klick auf einen Wettbewerb bringt den Benutzer weiter zur Wettbewerbs-Ansicht.

Abbildung 4: MainActivity mit der Liste verfügbarer Wettbewerbe

2.4 Spieltage

Die Wettbewerbs-Ansicht entsprach anfangs einer Activity mit der Auflistung der Spieltage des per Intent-Extra übergebenen Wettbewerbs. Beim Klick auf einen Spieltag fand ein Wechsel zur Spieltags-Ansicht statt. Doch bei der Nutzung dieser Konstellation wurde festgestellt, dass es sehr umständlich war, von der Ansicht eines Spieltages zu der eines anderen, z. B. des folgenden Spieltages, zu wechseln. Wenn dieser Wechsel erleichtert werden könnte, wäre auch ein Einstieg direkt beim aktuellen Spieltag denkbar. OpenLigaDB stellt eine Methode zur Identifikation des aktuellen Spieltags zur Verfügung.

Abbildung 5: Auswahl des anzuzeigenden Spieltags über ein Kontextmenü

Hierzu wurde eine passende Lösung gefunden. Somit wird nach der Auswahl des Wettbewerbs in der `MainActivity` direkt eine `MatchdayActivity` geladen, die die Daten des aktuellen Spieltages des Wettbewerbs angezeigt. Der nächste bzw. vorherige Spieltag können direkt in der `MatchdayActivity` angesteuert und angezeigt werden. Genauer wird dies im nächsten Kapitel (2.5) beschrieben. Die Bedienbarkeit ist somit aber immer noch nicht optimal. Will ein Benutzer z. B. im Moment, in dem der 1. Spieltag eines Wettbewerbs aktuell ist, wissen, wie die Begegnungen des 34. Spieltages lauten, so müsste er entsprechend 33 mal den nächsten Spieltag ansteuern. Erst dann würde er seine gewünschten Informationen erhalten. Um dies zu erleichtern wurde in der `MainActivity` ein Menü eingefügt, welches beim langen Klicken auf einen Wettbewerb erscheint. Dieses Kontextmenü lädt daraufhin die Spieltage des

gewählten Wettbewerbs und zeigt sie an (siehe Abbildung 5). Die `MainActivity` implementiert deswegen außer der `CompetitionsReceiver`-Schnittstelle auch die `MatchdaysReceiver`-Schnittstelle. Bei Auswahl eines Spieltages im Menü wird dieser per Extra-Objekt an das Start-Intent der `MatchdayActivity` angehägt, welche dann entsprechend direkt den übergebenen Spieltag anzeigt.

Abbildung 6: MatchdayActivity

2.5 Spielbegegnungen

Wie bereits erwähnt wird in der `MatchdayActivity` (siehe Abbildung 6) beim Start standardmäßig die Begegnungen des aktuellen Spieltages aufgelistet. Zur Bestimmung des aktuellen Spieltages ist eine Anfrage an die Datenquelle (OpenLigaDB) vonnöten, weshalb die Activity die `CurrentMatchdayReceiver`-Schnittstelle implementiert. Ist per Intent ein spezieller Spieltag übergeben worden, wird dieser Spieltag beim Start geladen. Das Empfangen der Spielbegegnungsdaten erfolgt über Implementierung der `MatchesReceiver`-Schnittstelle. Die Verwendung des `ViewPager`s ermöglicht den komfortablen Wechsel zum vorherigen bzw. nachfolgenden Spieltag per Swipe (zur Seite Schieben) nach rechts bzw. links. Die Anzeige des aktuellen bzw. vorherigen/folgenden Spieltages erfolgt in einem sogenannten `PagerTitleStrip`.

Damit auch in dieser Activity ein schneller Wechsel zu einem spezifischen Spieltag möglich ist wird im `OptionsMenu` der Zugriff zu einem Spieltag nach Wahl zur Verfügung gestellt. Hier erfolgt die Anzeige des ausgewählten Spieltages einfach durch Aufruf der `setCurrentItem`-Methode des `ViewPager`s.

Die Anzeige der Spieltagsbegegnungen werden im `ViewPager`-Konzept nicht von der Activity selbst, sondern von einem `Fragment` übernommen. Der `SectionPagerAdapter` sorgt dafür, dass zumindest die direkt ansteuerbaren `Section`s, also in diesem Fall die Spieltag-`Fragment`e des vorherigen, aktuellen und folgenden Spieltages, geladen sind. So entfällt beim Wechsel meist auch die Wartezeit für die Datenbeschaffung. Damit ein `MatchesListFragment` die Spielbegegnungsdaten entgegennehmen kann implementiert es die `MatchesReceiver`-Schnittstelle.

Das `ListItem`-Layout einer Spielbegegnung wurde in der Datei `listem_match.xml` manuell definiert. Die durch den `MatchesArrayAdapter` überschriebene `getView`-Methode kümmert sich um die korrekte Zuordnung der Spielbegegnungsdaten in diesem Layout. Auch eine `ImageView` für die durch OpenLigaDB ebenfalls definierten Vereins-Emblems der Kontrahenten ist vorgesehen. Auf eine Anzeige eben dieser wird allerdings aus rechtlichen Gründen verzichtet. Einige Wettbewerbs-Veranstalter, wie z. B. die Deutsche Fußball Liga GmbH (DFL), halten die

Nutzungs- und Verwertungsrechte der Emblems der Wettbewerbsteilnehmer[6].

Bei langem Klick auf eine Spielbegegnung wird dem Benutzer per `Contextual Action Bar` (CAB) angeboten die Begegnung in seinen Kalender zu exportieren, falls die API des installierten Android-Betriebssystems diese Funktion anbietet. Die Möglichkeit Kalendereinträge per Intent zu erstellen existiert seit API 14 (Ice Cream Sandwich). Ist trotz API-Version größer gleich 14 keine Anwendung installiert, die das Intent behandeln kann, erfolgt statt der Ausführung der Aktion die Anzeige einer Fehlermeldung. Bei einem normalen Klick auf eine Spielbegegnung wird eine weitere Activity mit den Spielbegegnungsdetails geöffnet.

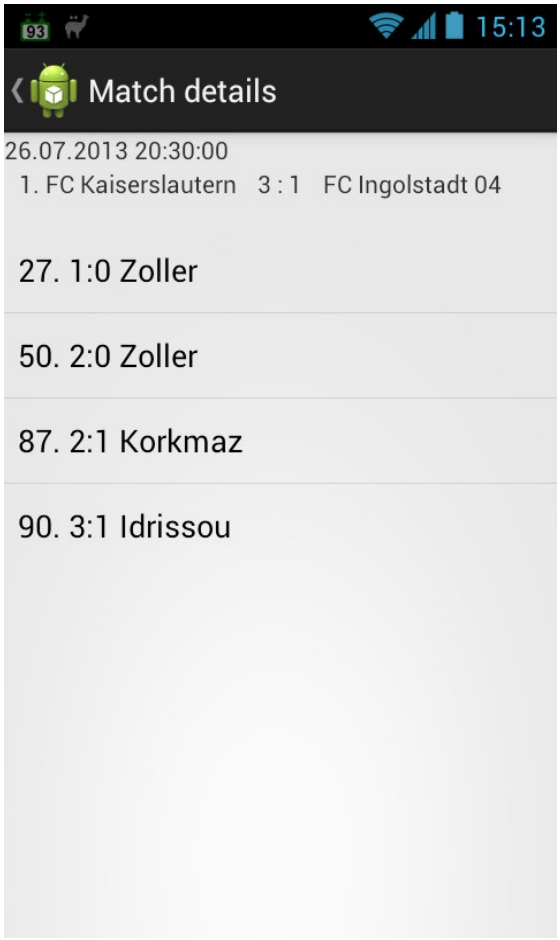


Abbildung 7: MatchActivity

2.6 Spielbegegnungsdetails

Die `MatchActivity` zeigt außer der aus der `MatchdayActivity` bekannten Daten der Begegnung Austragungszeitpunkt, Austragungsort und, falls vorhanden, Informationen über Reihenfolge, Zeitpunkt und evtl. Torschütze eines Tores an (siehe Abbildung 7). Zur Anzeige der aus der `MatchdayActivity` bekannten allgemeinen Daten wird das in `listitem_match.xml` definierte Layout wiederverwendet. Die Anstoß- bzw. Startzeit der Begegnung wird wie der Veranstaltungsort in einer einfachen `TextView`, die Tore in einer `ListView`, dargestellt. Zum Erhalt der Tordaten wird die `MatcheventsReceiver`-Schnittstelle implementiert. Definiert ist das ganze im Layout `activity_match.xml`. Hier ist des Weiteren noch eine `ListView` für eine evtl. später folgende Anzeige von vorhergehenden Begegnungen der beiden Kontrahenten vorgesehen.

3 Abschlussbetrachtung

3.1 Zielerfüllung

Die folgenden Ziele konnten nicht oder nicht komplett erfüllt werden:

4. "Anzeige der Spielbegegnungen eines Spieltages/einer Runde mit Ergebnis, geplanter Anstoßzeit bzw. aktuellem Zwischenstand" – Bei der Ansicht der Spielbegegnungen eines Spieltages wird keine geplante Anstoßzeit angezeigt. Es wäre sinnvoll die Spiele eines Spieltages, die zur gleichen Zeit angepfiffen werden, zu gruppieren. Die Anstoßzeit müsste in diesem Fall nur einmal genannt werden. Die `ListView` müsste dafür so etwas wie Sektionen und Sektionsüberschriften unterstützen. Eine mögliche Lösung stellt z. B. das Google-Code-Projekt android-amazing-listview[7]

6 Quelle u. a. http://geb.german-elite.net/blog.php?b=112

7 android-amazing-listview, Apache License 2.0, https://code.google.com/p/android-amazing-listview/

zur Verfügung. Aus Zeitgründen wurde davon abgesehen und die Anstoßzeit stattdessen nur in der Match-Detail-Ansicht angezeigt.

7. "Anlegen und Anzeige von favorisierten Wettbewerben/Mannschaften und deren Spielbegegnungen" – Diese Funktion wurde aus Zeitgründen nicht implementiert. Die Möglichkeit Mannschaften zu favorisieren wird außerdem noch durch die Nutzung der OpenLigaDB, die keine Methode anbietet, um Wettbewerbsunabhängige Informationen über ein Team zu erhalten, erschwert. So müssten z. B. zur Bestimmung aller Wettbewerbe, an denen ein Team teilnimmt, erst die Teilnehmer aller Wettbewerbe einzeln abgefragt werden. Außerdem ist nicht sichergestellt, dass die OpenLigaDB pro Team nur einen Datensatz enthält, der das Team repräsentiert.

9. "Geo-Export des Veranstaltungsortes" – Die OpenLigaDB bietet keine verlässliche Möglichkeit den Veranstaltungsort zu lokalisieren. Zur Umsetzung dieser Funktion müssten die Daten der Veranstaltungsorte also durch externe Daten ergänzt werden.

10. "Anzeige einer aktuellen Tabelle des Wettbewerbs" – Auch für diese Funktion bietet die OpenLigaDB derzeit keine Methode an.

11. "Benachrichtigung bei Ereignissen innerhalb einer Spielbegegnung favorisierter Wettbewerbe/Mannschaften" – Außer der Tatsache, dass die Favoriten-Funktion nicht verwirklicht wurde, steht auch die Community-basierte Eingabe der OpenLigaDB-Daten diesem Punkt im Weg. Daten, die in unbestimmter Zeit mit unterschiedlicher Informationsmenge eintreffen, sind nicht geeignet eine Live-Benachrichtigung über Spielereignisse anzubieten. Allenfalls eine Benachrichtigung über Spielergebnisse wäre somit theoretisch denkbar.

Die Punkte 12 "Startseite der App konfigurierbar", 13 "Eingeschränkte Nutzung der App bei Offline-Betrieb ermöglichen", 14 "Anlegen neuer Daten (Wettbewerbe, Spieltage, Spielbegegnungen, Ergebnisse, usw.)" und 15 "Editieren vorhandener Daten" wurden aus Zeitgründen nicht mehr umgesetzt.

3.2 Weitere Entwicklung

Es ist geplant die Anwendung nach der Projektabgabe der OpenSource-Community zur Verfügung zu stellen. Mit der Community könnten dann die durch Zeitmangel nicht in Angriff genommenen Funktionen und weitere Ideen umgesetzt werden.

Ein weiterer Punkt wäre z. B., dass die Anzeige einer Spielbegegnung den Benutzer momentan nicht erkennen lässt, ob es sich bei dem Ergebnis um ein Zwischenergebnis oder um das Endergebnis handelt. Eine unterschiedliche Darstellung, wie z. B. durch die Schrift- oder Hintergrundfarbe könnten eine Lösung sein. Die Spielbegegnungsansicht könnte außerdem einfach durch eine Liste vorheriger Begegnungen der beiden Kontrahenten erweitert werden, da die OpenLigaDB zur Abfrage dieser Daten eine Methode bereitstellt.

Außer der Möglichkeit Wettbewerbe zu favorisieren wäre es sinnvoll, wenn in der `MainActivity` bevorzugt Wettbewerbe der aktuellen Saison aufgelistet werden. Andere Wettbewerbe könnten entweder ausgefiltert oder zumindest nach den aktuellen Wettbewerben gelistet werden. Auch eine Suche bzw. Filterung der Wettbewerbe nach einem eingegebenen Begriff wäre hilfreich.

Der Umgang mit der Netzwerk-Kommunikation kann ebenfalls weiter verbessert werden. Während Wartezeiten sollte die Oberfläche der App in den meisten Fällen keine Eingaben annehmen und den Ladevorgang visuell darstellen. Netzwerkfehler wie eine nicht vorhandene Internetverbindung werden momentan nur zum Zeitpunkt des Requests erkannt. Fehler, die danach stattfinden und eine standardmäßig erwartete Fertigstellung der Anfrage verhindern, müssen noch erkannt

werden und die bereits definierte `setNetworkError`-Methode der `Receiver`-Schnittstellen aufrufen.

3.3 App-Daten Kurzfassung

3.3.1 Zielplattform

Die Anwendung wurde mit der aktuellen SDK-Version 17 kompiliert. Beim Anwender wird auf jeden Fall eine SDK ab Version 11, also Android 3.0 (Honeycomb), vorausgesetzt. Die bei Beginn der Entwicklung angestrebte Mindest-SDK-Version 10 wurde für die Usability der Anwendung aufgegeben. Grund dafür ist insbesondere die Verwendung des `ViewPager`-Konzepts, das die komfortable Anzeige eines Spieltages und den einfachen Wechsel zwischen aufeinanderfolgenden Spieltagen erlaubt. Ein anderer ist die `Contextual Action Bar` zur Aktionsauswahl einer markierten Spielbegegnung (Export). Der Einsatz der Android Support Library für die Unterstützung neuer Funktionen in älteren Versionen des Android-Betriebssystems wäre denkbar, wurde allerdings nicht geprüft.

Die Kalender-Export-Funktion ist erst ab SDK 14, also Android 4.0 (Ice Cream Sandwich), verfügbar.

3.3.2 Sprache

Sowohl das User Interface als auch der Code inklusive Kommentaren wurde komplett in englischer Sprache verfasst. Eine Ausnahme bilden Wörter oder Wortgruppen, die aus der OpenLigaDB ausgelesen werden, wie z. B. die Bezeichnung von Wettbewerben oder Spieltagen. Durch Verwendung der englischen Sprache wird eine mögliche Weiterentwicklung durch die OpenSource-Community erleichtert.

3.3.3 Bildschirm

Die Anwendung wurde insbesondere für typische Smartphone-Größen im Hochformat angepasst. Aber auch die Ansicht auf kleineren/größeren Endgeräten oder im Querformat sollte für die App kein Problem sein. Das Layout passt sich den gegebenen Bedingungen an.

3.3.4 Benötigte Rechte

- **android.permission.ACCESS_NETWORK_STATE** zur Überprüfung, ob eine Internet-Verbindung vorhanden ist.
- **android.permission.INTERNET** zur Nutzung der Internetverbindung (OpenLigaDB).

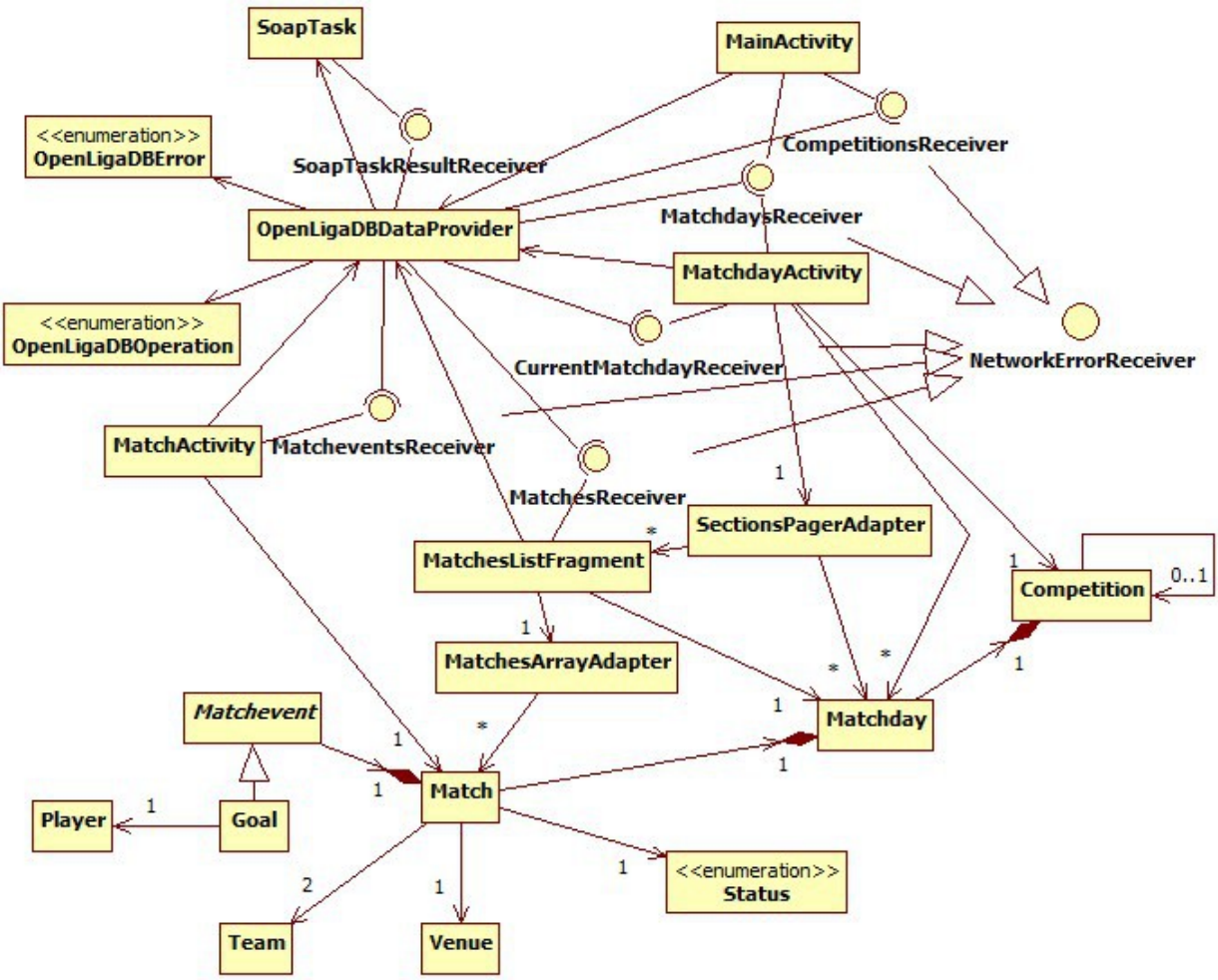
3.3.5 Verwendete Bibliotheken

- **ksoap2-android:** MIT-Lizenz; https://code.google.com/p/ksoap2-android/

Anhang

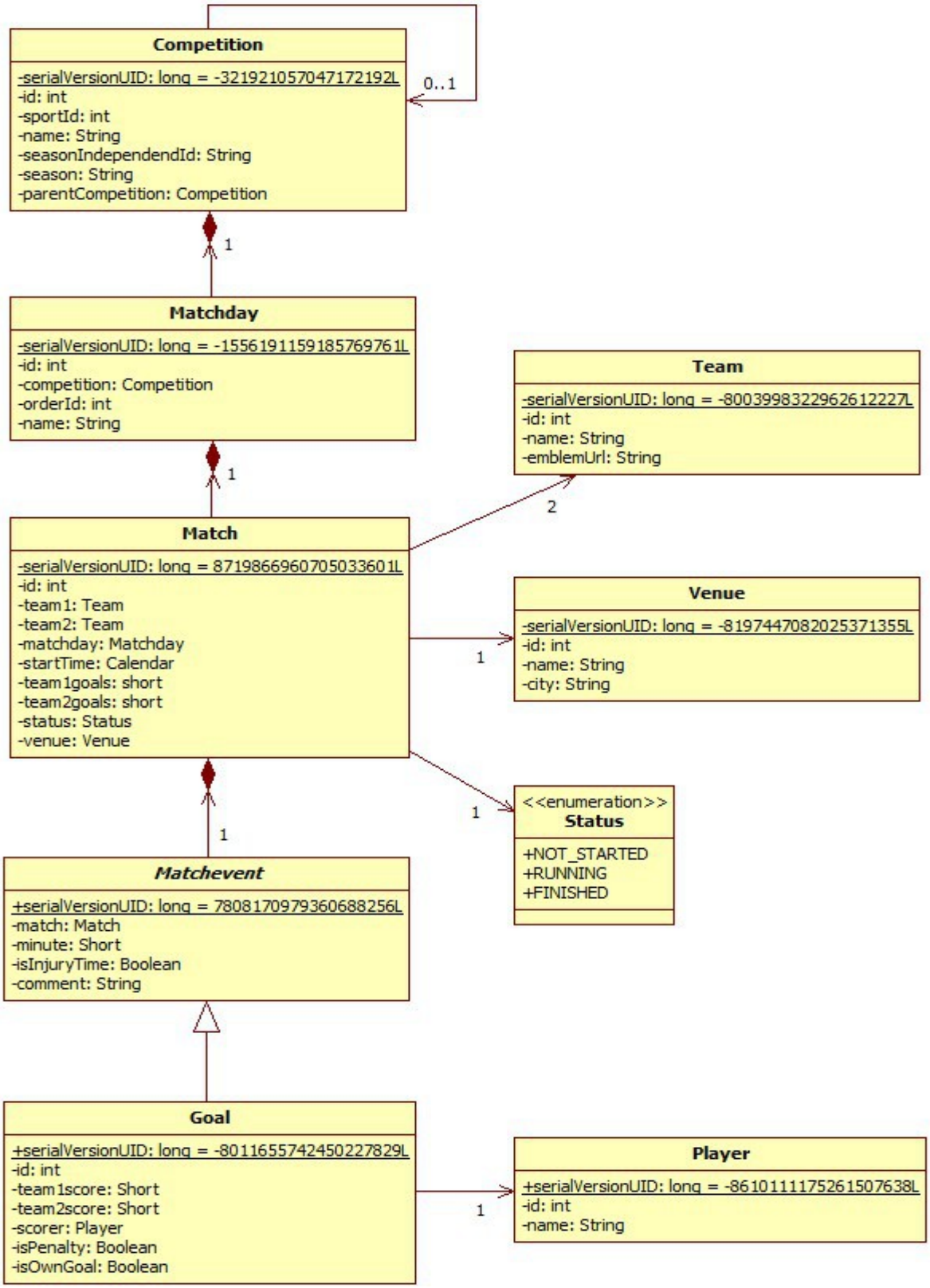
Alle Diagramme wurden mit WhiteStarUML[8] v.5.4.1.0 in „UML 2.0“-Notation erzeugt.

Klassendiagramm Entwurf:

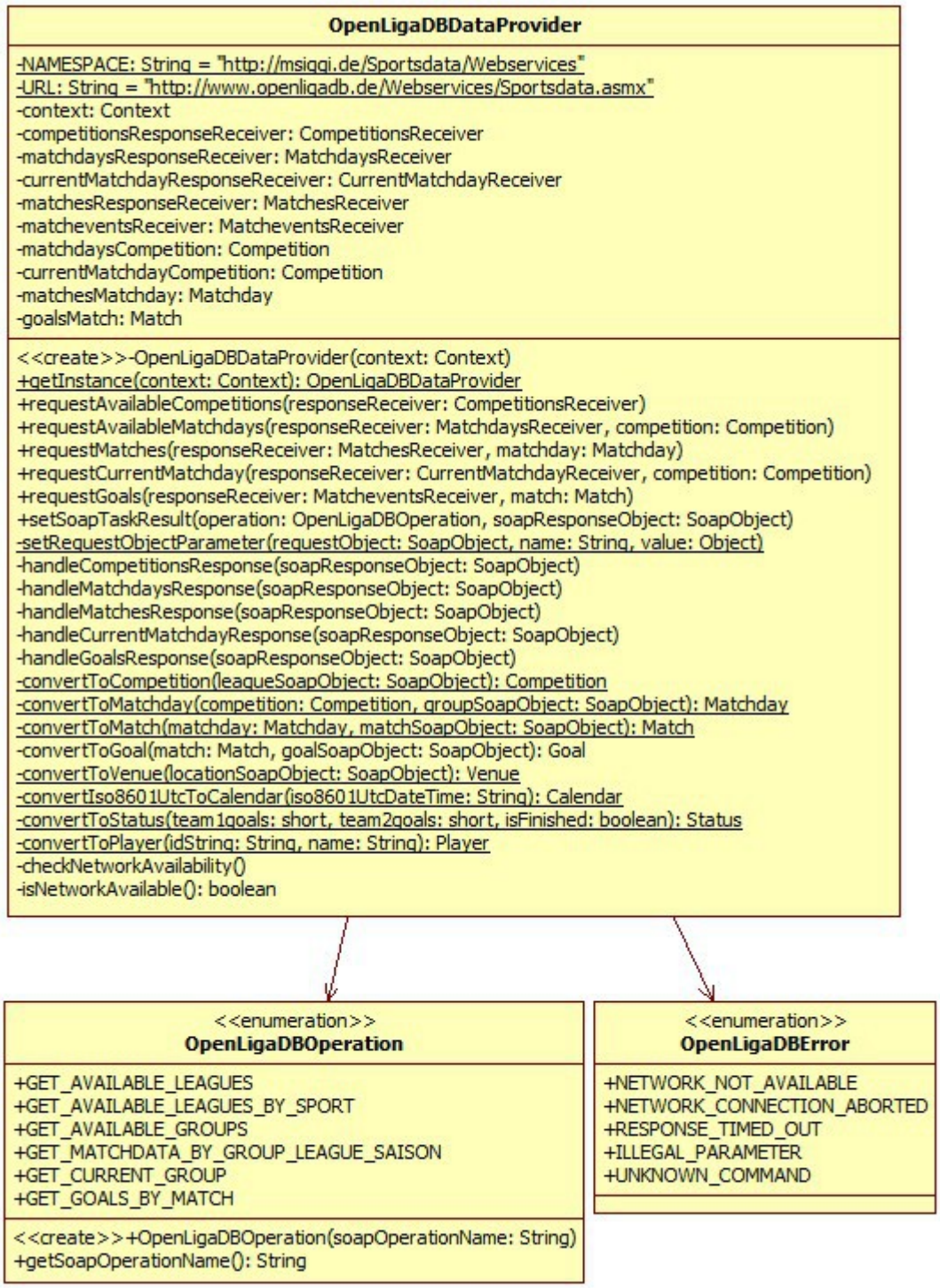


8 WhiteStarUML, GNU GPLv2, https://sourceforge.net/projects/whitestaruml/

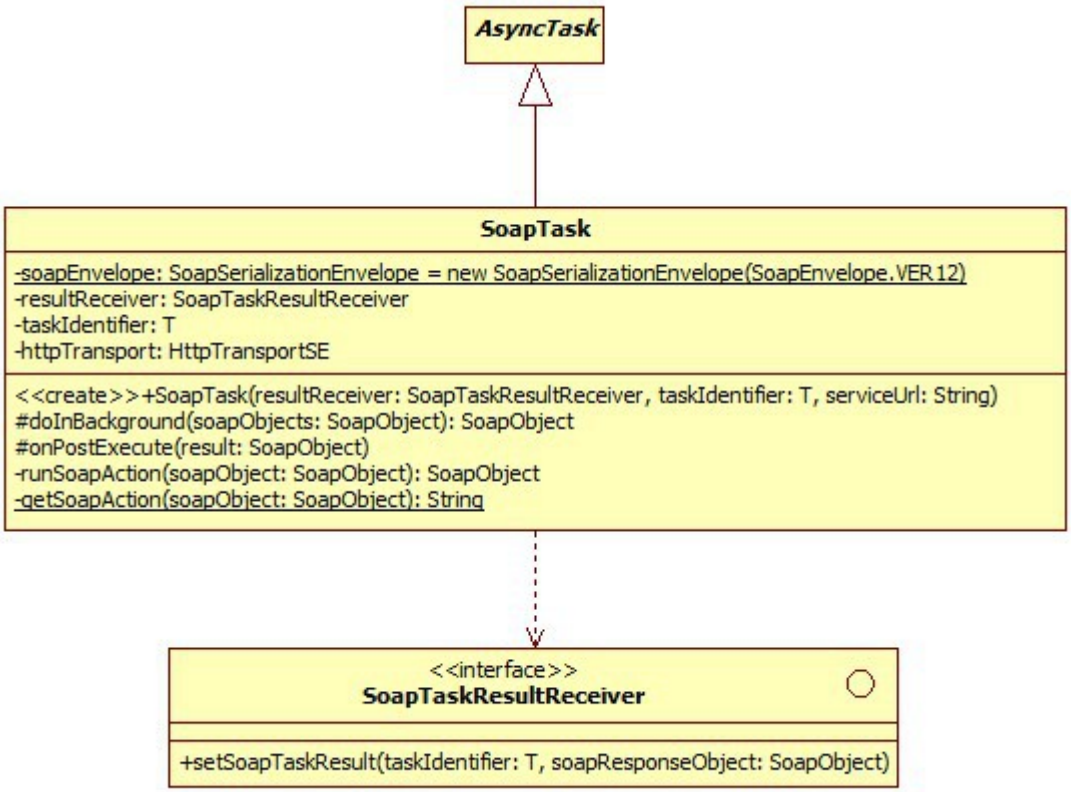
Klassendiagramm Entwurf Model-Komponente:



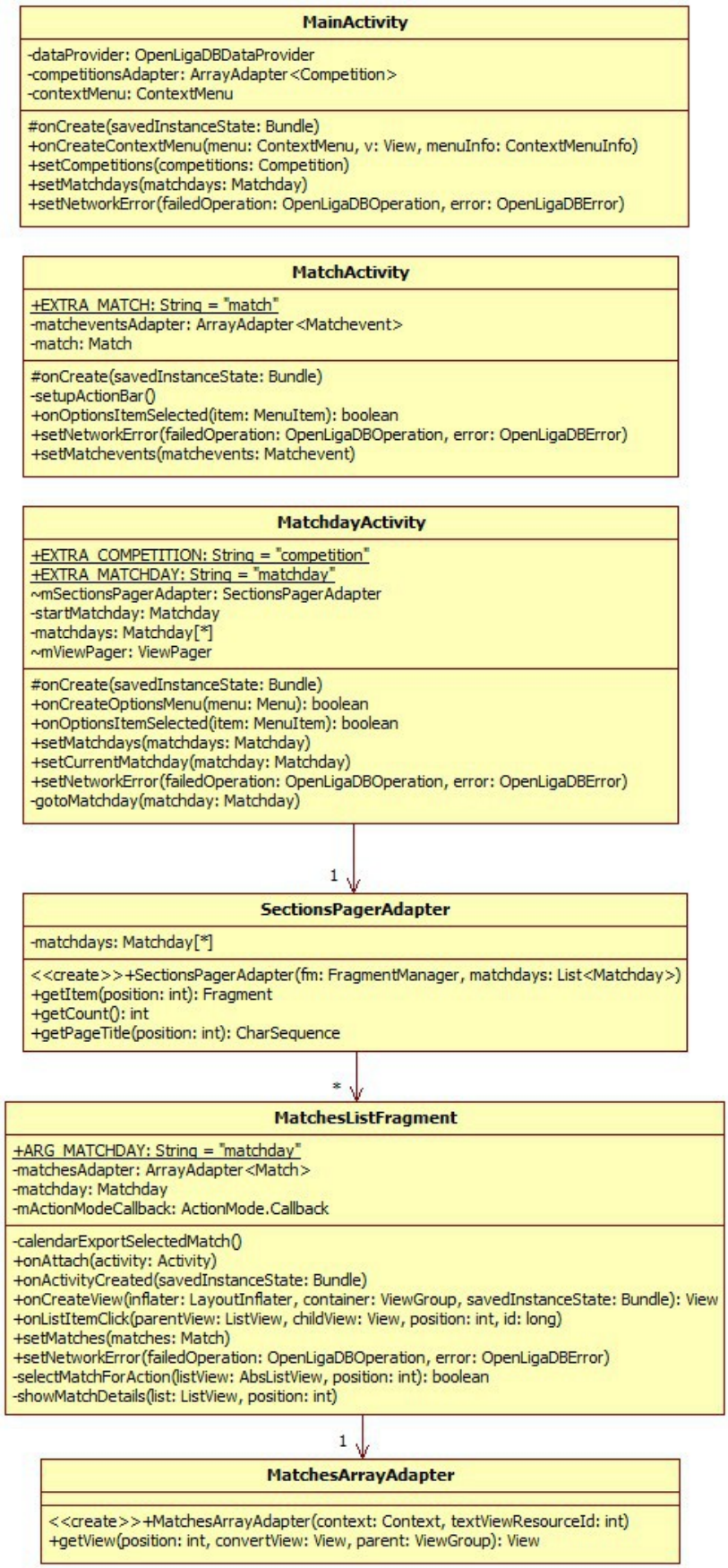
Klassendiagramm Entwurf OpenLigaDB-Komponente:



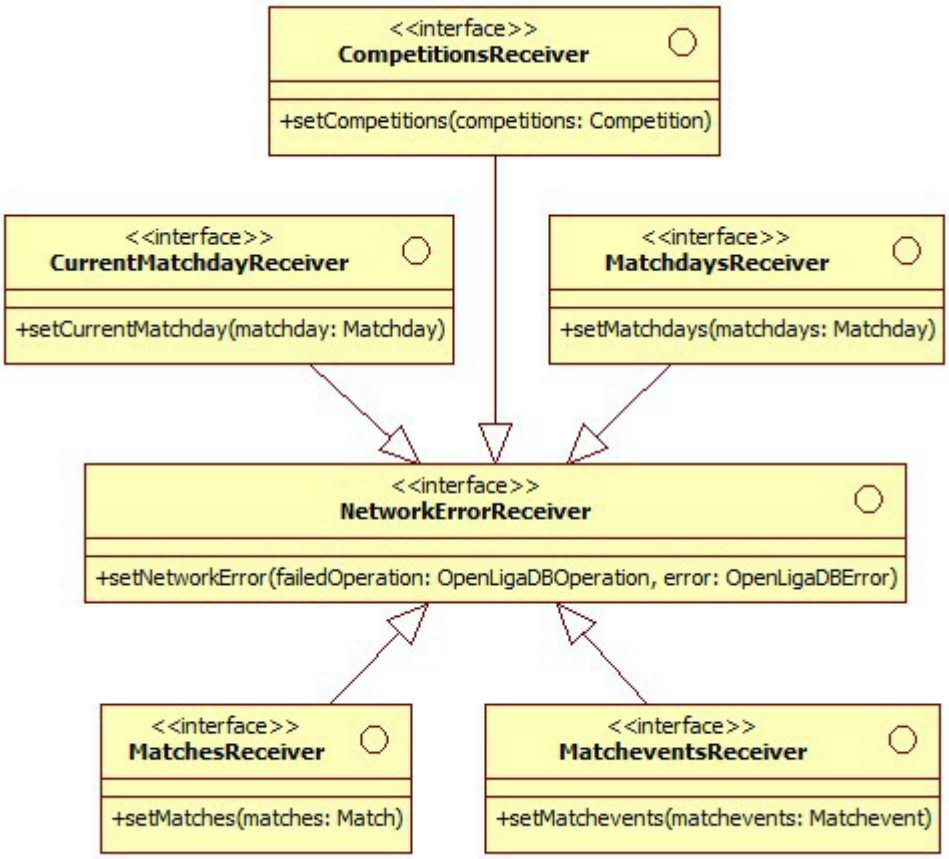
Klassendiagramm Entwurf SOAP-Komponente:



Klassendiagramm Entwurf UI/Adapter-Komponente:



Klassendiagramm Entwurf Receiver-Schnittstellen:



Sequenzdiagramm „MainActivity lädt Wettbewerbe“: