Entwurfsbeschreibung Gesamtprojekt

Verantwortlicher: Hans Dieter Pogrzeba

Daniel Alexander, Jakob Kusnick, Richard Khulusi, Christopher Schott, Tobias Turke, Nikolas Oberhäuser, Hans Dieter Pogrzeba, Felix Conrads

Inhaltsverzeichnis

1. Allgemeines

Schach gehört zu den ältesten Spielen der Welt. Es wurden mittlerweile Millionen von Schachpartien zu tausenden von Spielern dokumentiert, jedoch sind die Daten über Schach auf unterschiedlichen Webseiten verteilt und semantisch heterogen. Dadurch können zur Zeit Fragen wie „Welche Spieler aus Deutschland mit einer Elo über 2500 haben im Jahr 2007 in China gespielt?“ nur sehr schwer beantwortet werden.

Um die Beantwortung solcher Fragen zu ermöglichen, soll durch dieses Projekt eine Architektur für die kontinuierliche Integration von Schachdaten umgesetzt werden. Ein Crawler soll Schachdaten in Form von PGN-Dateien aus dem Web sammeln und speichern. Ein Konverter wird dann diese Daten entsprechend vorgegebener Ontologien nach RDF transformieren. Neben gesammelten Partien werden außerdem Informationen zu den beteiligten Spielern, sowie den Events, zu welchen die Partie gehört, gesucht und gespeichert.
Durch Linking und Data-Fusion wird die Qualität des Datenbestands gesichert.

2. Produktübersicht

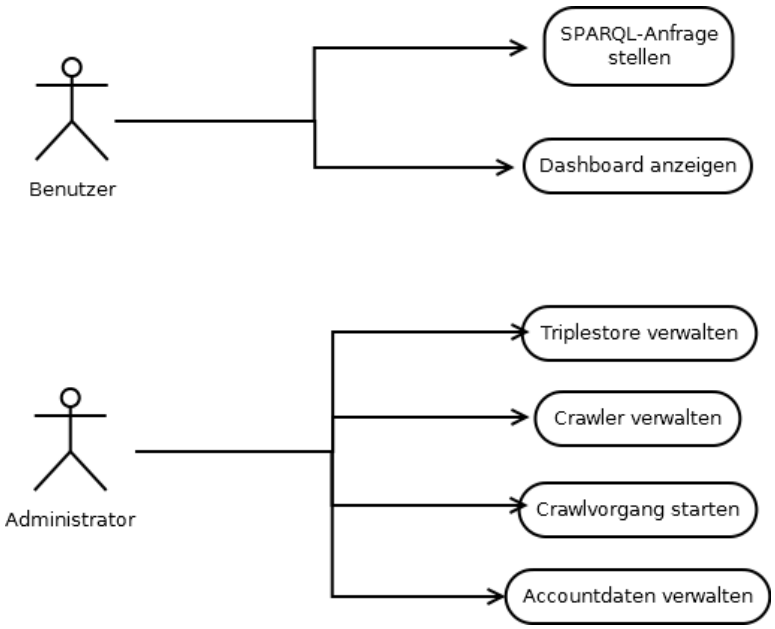
Das Resultat des Projektes ist eine Webanwendung, die von Usern und Administratoren benutzt wird.

2.1 User

Dem User wird ein Eingabefeld zur Abfrage mittels SPARQL-Query zur Verfügung gestellt. Außerdem bietet ein Dashboard nützliche Informationen.

2.2 Administrator

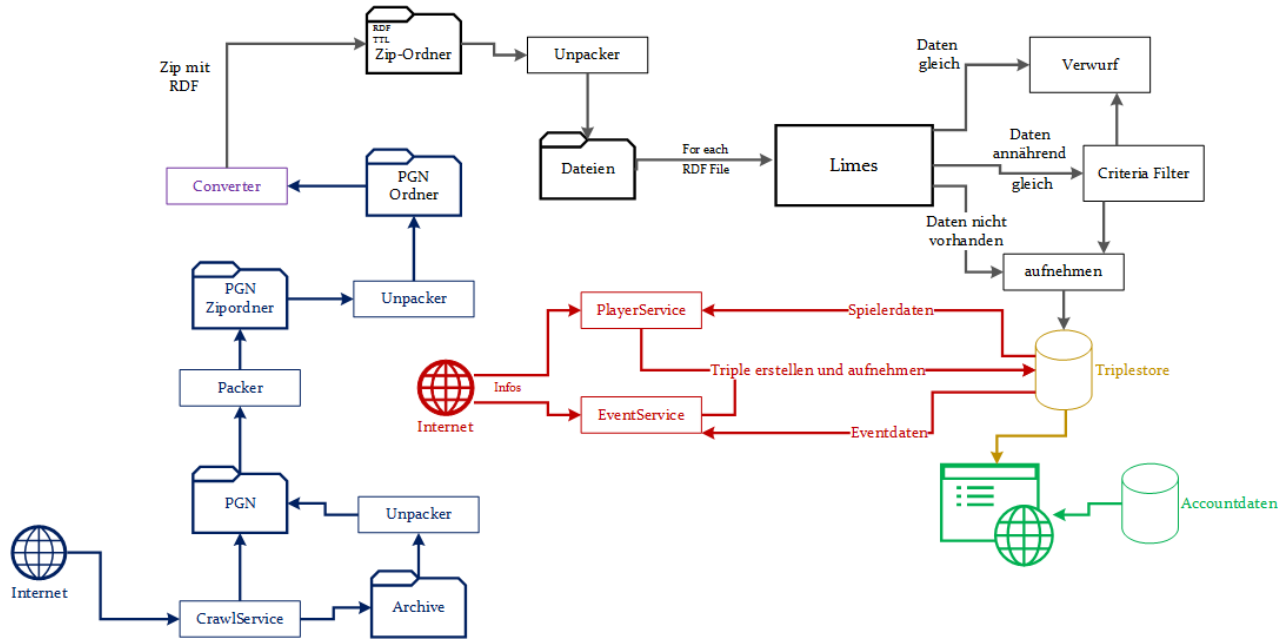
Ein Administrator übernimmt die Verwaltung der Funktionen. Er kann die Einstellungen des Crawlers und des Triplestores, sowie auch die Administratoraccountdaten ändern. Zusätzlich kann er den Crawlvorgang manuell starten.



3. Grundsätzliche Struktur- und Entwurfsprinzipien

Zur Projektverwaltung wird Git, zur Aufgaben- und Bugverwaltung Jira verwendet.
Zur Speicherung der Triple wird der im Vorprojekt durch einen Benchmark ausgewählte Triplestore verwendet. Zudem kommt zur Speicherung der Accountdaten für Administratoren eine MySQL Datenbank zum Einsatz.
Die Webanwendung wird als Java EE Projekt umgesetzt. Die entstehende WAR-Datei wird auf einem Jboss Application Server gestartet. Um diesen Prozess so einfach wie möglich zu halten, wird Maven zur Build-Konfiguration verwendet.

3.1 Übersicht der Gesamtstruktur



4. Struktur- und Entwurfsprinzipien einzelner Pakete

4.1 Schnittstelle zu Triplestores

Die Klasse Connection vereinfacht die Verwaltung des Triplestores. Die nötigen Einstellungen werden in einer XML Datei gespeichert. Die Connection kann so konfiguriert werden, dass Dateien im RDF Format schnell hochgeladen werden können. Hierfür wird cURL verwendet, welches auf dem Server zur Verfügung stehen muss.

4.2 Crawler

4.2.1 PGN-Crawler

Der PGN-Crawler durchsucht mit Hilfe des Frameworks Crawler4j das Internet nach PGN Dateien sowie Archivdateien, die evtl. PGNs enthalten könnten und speichert diese in entsprechende Ordner. Dabei übernimmt die Klasse CrawlService das Laden aller Einstellungen und die Verwaltung des Vorganges. Gleichzeitig werden durch die Klasse Unpacker die gefundenen Archive entpackt und deren Inhalt ggf. in den PGN-Ordner verschoben.
Von dort werden sie, um Speicherplatz zu sparen, in größeren Einheiten verpackt.

4.2.2 Spieler- und Eventdatencrawler

In vielen PGN-Files werden Turniere und Schachspieler aufgeführt, zu denen bislang keine weiteren Daten außer denen aus dem jeweiligen PGN-File bekannt sind. Nachdem die PGNs also ausgelesen und in den Triplestore gespeichert wurden, müssen etwaige Metadaten gesammelt werden. Der SpielerService dient dazu, Spieler-Metadaten von der Fide-Homepage und der DBPedia herunter zu laden. Der EventService lädt die Daten über das Event von der Fide-Homepage herunter und außerdem die Daten des Crosstable zu jedem Turnier.

Nachdem der Crawler und das Data Fusion abgeschlossen sind, stellt der SpielerService eine SPARQL-Anfrage an den Triplestore, um Spielerdaten geliefert zu bekommen, die jeden Spieler identifizieren sollen. Dafür geeignet sind die Spieler-ID der Fide wie auch der Name der jeweiligen Spieler, falls keine Fide-ID bekannt ist. Mithilfe dieser Daten wird auf der Homepage der Fide und in der DBPedia nach zusätzlichen Informationen zu jedem Spieler gesucht, welche durch einen HTML-Parser von den Webseiten extrahiert werden sollen. Die so gewonnenen Daten werden anschließend in RDF-Files konvertiert und im Triplestore gespeichert.

Der EventService sucht per SPARQL-Anfrage nach Events im Triplestore. Nach diesen Events wird auf der Fide-Seite gesucht. Wird ein Ergebnis gefunden, so werden die Turnier-Daten mittels html-Parser heruntergeladen und formatiert. Danach wird (falls vorhanden) der Crosstable zum jeweiligen Turnier aufgerufen und die Informationen ebenfalls heruntergeladen und formatiert. Die gesammelten Daten werden dann wiederum in RDF-Triples konvertiert und im Triplestore gespeichert.

4.3 Sammeln / Data Fusion

Um die Datenbestände sinnvoll zu fusionieren, wird das Javaframework LIMES benutzt. LIMES vergleicht die Daten der Ausgangsquelle (vom Crawler gefundene Daten) mit einer Zielquelle (Daten im Triplestore).
Dazu werden einfache wie auch komplexere Metriken benutzt, mit deren Hilfe die einzelnen Ressourcen (aus Datei und Triplestore bzw. aus Triplestore und Triplestore) miteinander verglichen werden. Wenn diese komplett übereinstimmen ist davon auszugehen, dass die beiden Daten dieselben Spiele beschreiben und die neuen Daten verworfen werden können, um die Aufnahme von Duplikaten zu vermeiden.

Wenn die Daten sehr ähnlich sind, sich also nur in wenigen Komponenten und dort nur geringfügig unterscheiden, ist es sehr wahrscheinlich, dass diese Daten ebenfalls dieselben Spiele beschreiben. In diesem Fall muss überprüft werden, ob die neu gefundene Datei qualitativ hochwertigere Daten besitzt als die bereits vorhandenen. Dies wird abhängig von einzelnen Komponenten der PGNs bestimmt, da ein Name anders behandelt werden muss als zum Beispiel eine ELO.
Stellt sich nun also heraus, dass die neue Datei den vorhandenen Datensatz verbessert wird diese übernommen. Ist eine Verbesserung der Qualität nur in einzelnen Komponenten zu erkennen, so werden nur diese im Datenbestand geändert. Auf diese Art wird der gesammelte Datensatz immer hochwertiger.
Dieses Vorgehen wird letztlich in einer LimesConfig.xml festgehalten.

4.4 Oberfläche (User und Administrator)

Die Oberfläche bietet dem User die Möglichkeit, sich über ein Dashboard interessante Informationen abzurufen oder über ein Suchfeld selbst SPARQL-Anfragen zu stellen.
Administratoren können sich mit einem Passwort anmelden und die Einstellungen anpassen.
Die Benutzerverwaltung wird hierbei mit Hilfe einer MySQL Datenbank realisiert.
Screenshots zur Oberfläche befinden sich im Anhang.

5. Datenmodell

Siehe UML-Diagramm im Anhang

6. Testkonzept

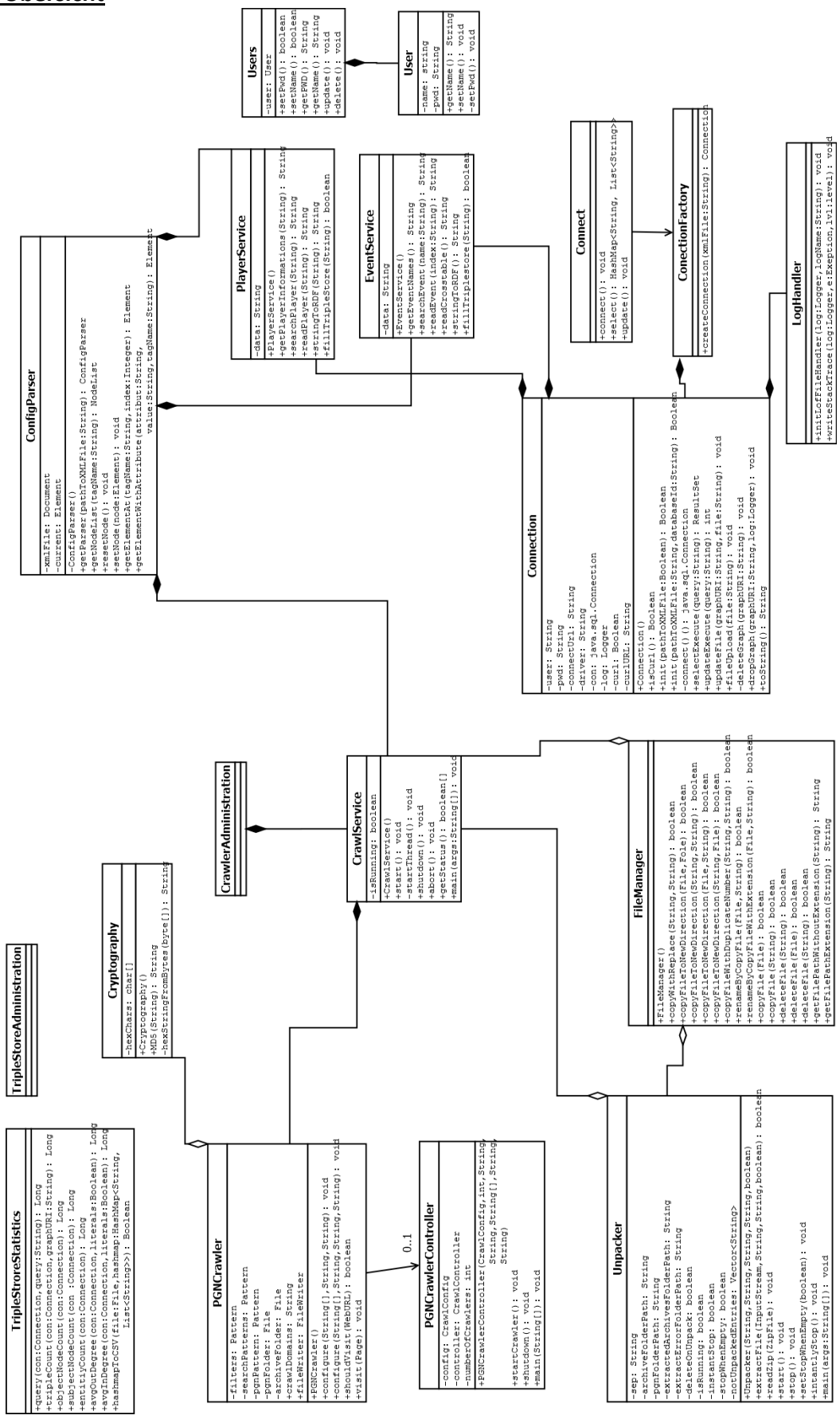
Wie im Dokument „Qualitätssicherungskonzept“ vereinbart, soll für automatisierte Tests das Framework JUnit verwendet werden. Dabei werden die erstellen Testklassen vom eigentlichen Code getrennt verwaltet.
Für manuelle Tests, insbesondere während des Programmierens, kann innerhalb der jeweiligen Klasse eine main-Funktion zum überprüfen des ggf. noch nicht funktionsfähigen Programmteils angelegt werden.

7. Glossar

Siehe externes Dokument

8. Anhang

8.1 UML Übersicht



8.2 Screenshots der Weboberfläche