JRMDS

SWT14W01

Pflichtenheft

Softwaretechnologieprojekt 2014
TU Dresden
Betreuer: Hendrik Sollich

Zusammenfassung

Pflichtenheft der Gruppe 1 „JRMDS“ für das externe Softwareprojekt jQAssistant
der Firma Buschmais GbR , durchgeführt vom Softwaretechnologielehrstuhl der TU Dresden.

Historie

Version	Status	Datum	Erläuterungen
0.1	Grobentwurf	29.10.14	Erster Entwurf

Inhaltsverzeichnis

1. Vorangegangene Entwicklung und Zielsetzung

Die buschmais GbR bietet mit dem Open-Source-Werkzeug "jQAssistant" eine flexible Lösung an, um bei der Implementierung von Softwareprojekten die Definition von Regeln, u.a. einheitliche Namenskonventionen und Strukturierung von Modulen/Programmteilen, festzulegen und deren Einhaltung während der Umsetzung kontinuierlich zu überwachen. Dafür werden die Strukturen eines gesamten Softwareprojektes eingelesen und in einer Graphendatenbank (Neo4j) abgespeichert, um diese dann mithilfe der Abfragesprache "Cypher" zu analysieren und auszuwerten. Die dafür nötigen Regeln sind derzeit als Cypher-Queries in XML-Dateien gespeichert und werden zurzeit manuell gepflegt.
Um die Bearbeitung und Administration der Regeln zu vereinfachen, soll eine zentrale Verwaltung über Repositories ermöglicht werden.

2. Anforderungen

2.1. Muss-Kriterien

MK01	Server mit grafischer Weboberfläche
MK01.1	CRUD – Erstellen, Anzeigen, Bearbeiten, Löschen
MK01.2	Suche nach ID und Beschreibung
MK02	Zuordnung von Regelsätzen zu Projekten
MK03	Anlegen/Bearbeiten/Löschen von Regeln/Gruppen/Templates über die Oberfläche
MK04	Validierung von Regeln, z.B. Prüfung auf Zyklen und verletzte Abhängigkeiten
MK05	Abbildung der Beziehungen/Referenzen zwischen Objekten (Auswahlmenü, ggf. Drag&Drop)
MK06	Jedes Projekt kann weitere Regeln importieren
MK06.1	lokaler Server: um Regeln aus einem anderen Projekt zu übernehmen
MK06.2	entfernter Server: Import über XML-Schnittstelle
MK07	Lokale Regeln können entfernte Regeln überschreiben
MK07.1	Überschreiben muss in der UI sichtbar sein

MK08	Bereitstellung von Regeln als REST-Ressource
MK08.1	XML: Build-Clients (z.B. Maven-Plugin), Repositories und IDEs (z.B. zukünftiges Eclipse-Plugin)
MK08.2	JSON: UI-Client (Browser)
MK09	Speicherung der Regelsätze in einer Neo4j Datenbank

2.2. Kann-Kriterien

KK01	Import von Regeln aus bestehenden XML-Dateien
KK02	Abgleich von Regeln zwischen Repositories
KK02.1	Online vs. Offline
KK03	Benutzerverwaltung
KK03.1	Login und Rechteverwaltung auf Projektbasis
KK03.2	Rechte für Projekte: für alle lesbar (auch Gäste), bearbeiten nur für registrierte Benutzer
KK04	um Sortierung und Übersicht zu wahren, kann jede Regel/Gruppe/Template usw. mit mehreren Tags versehen werden

3. Fachlicher Überblick

Umzusetzen ist eine Serverlösung mit GUI zum Verwalten der Regeln bzw. Filter

3.1. Technische Umgebung

Die Anforderungen sollen mit einem auf Java basierenden Server umgesetzt werden, der mit einer Webanwendung kommuniziert. Diese wird mit HTML, JavaScript und JQuery umgesetzt, es wird ein Browser benötigt. Die Graphendatenbank neo4j wird verwendet, um die Strukturen der Software und die Regeln zu speichern. Über ein REST-Interface werden Anfragen und Herausgeben der Regeln bearbeitet, wenn nötig werden sie über DocumentBuilder geparst. Die Übertragung der Daten zwischen Client und Server werden mit JSON in Verbindung mit Ajax realisiert.
Die Architektur lehnt sich an das MVC an mit Spring Web MVC.

3.2. Zielgruppe

Die Zielgruppe umfasst Softwareentwickler, die ihren Code auf Wartungsfähigkeit, Performance und Validität prüfen möchten.

4. Diagramme

4.1. Anwendungsfälle gekoppelt mit Sequenzdiagrammen

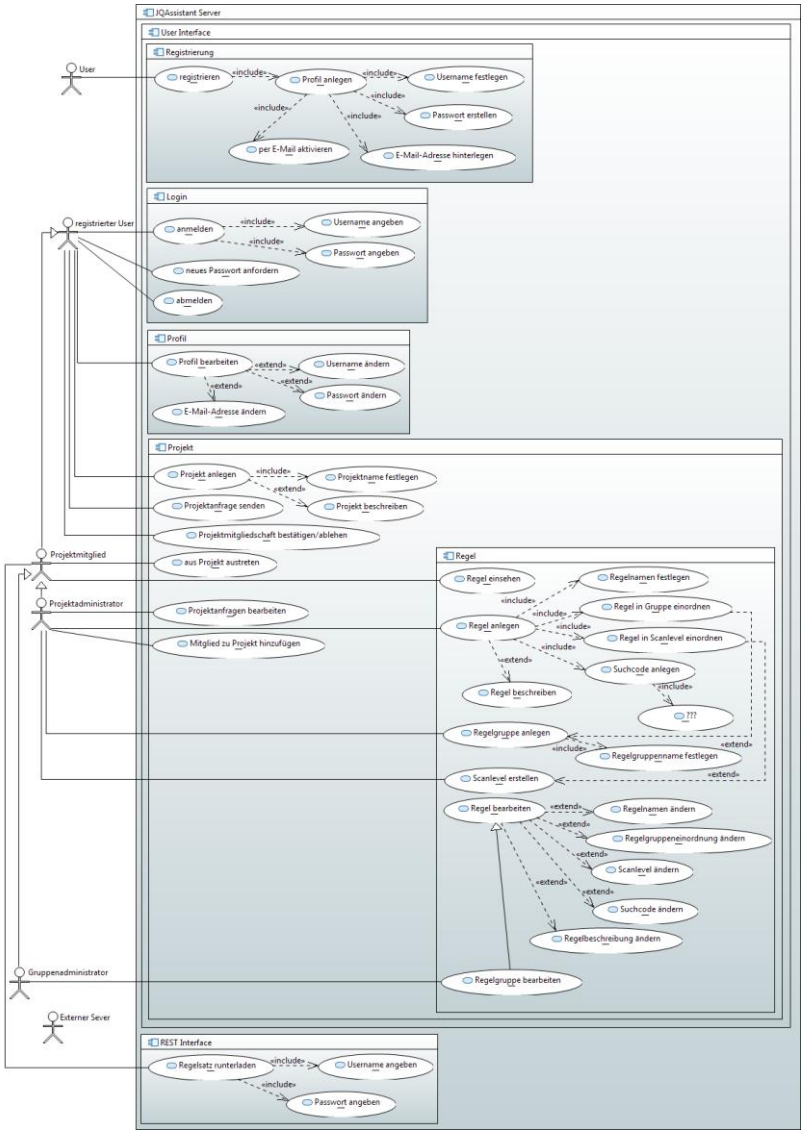


Bild in Normalgröße

4.2. Top-Level-Architektur

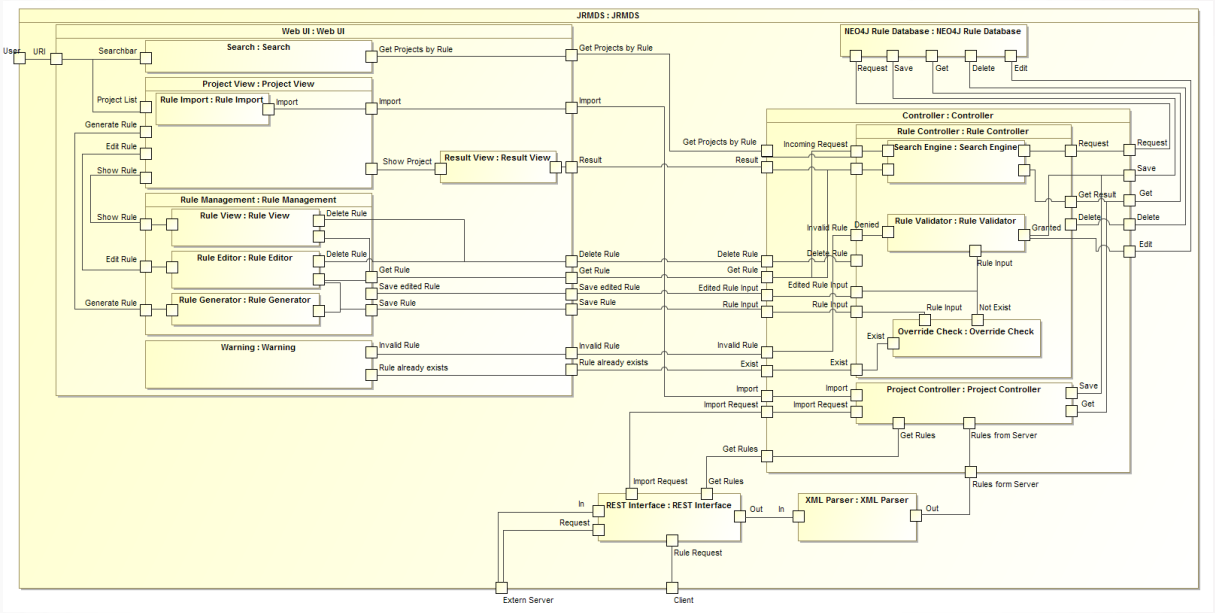
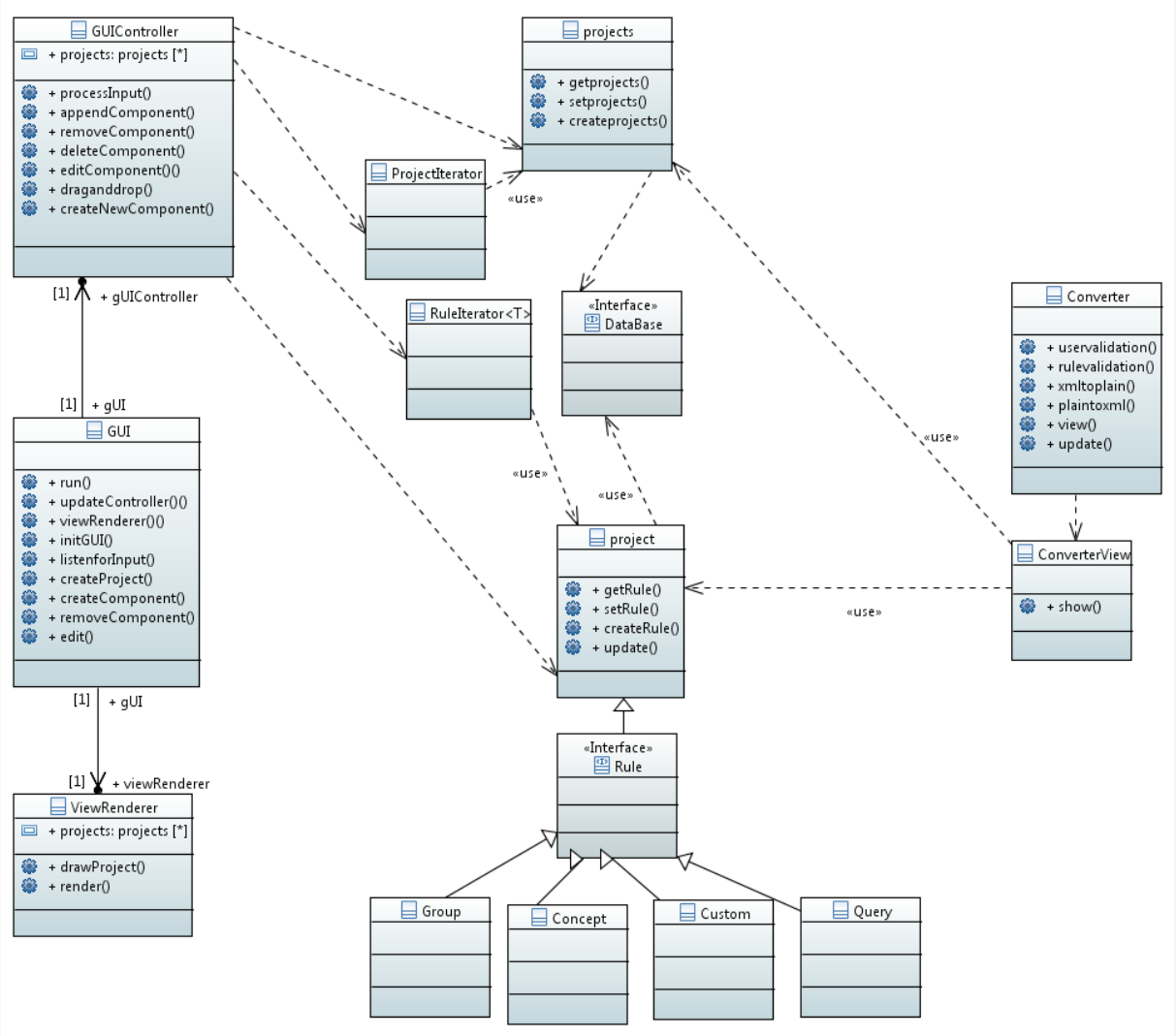


Bild in Normalgröße

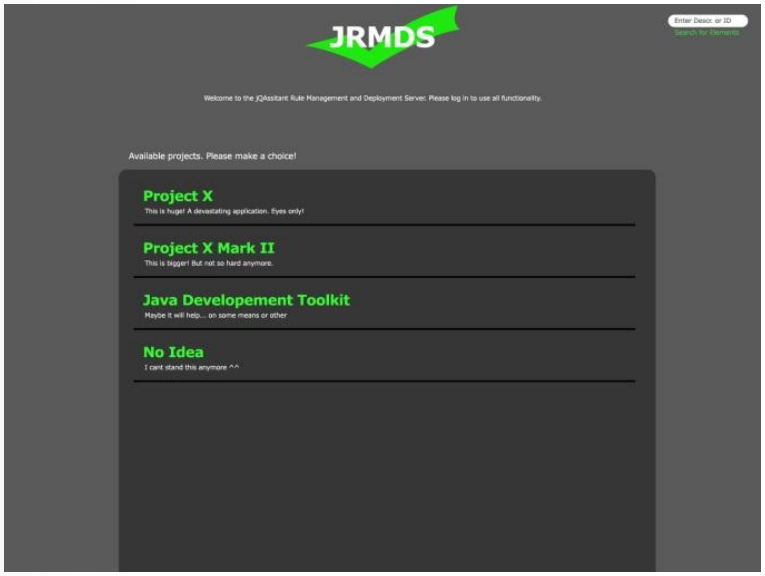
4.3. Analysediagramm



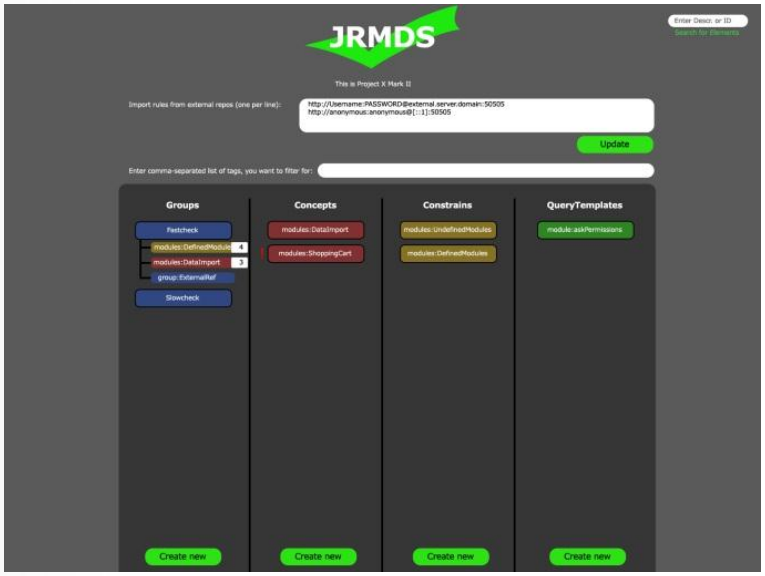
4.4. Komponentendiagramm

5. GUI Prototyp

Startbildschirm

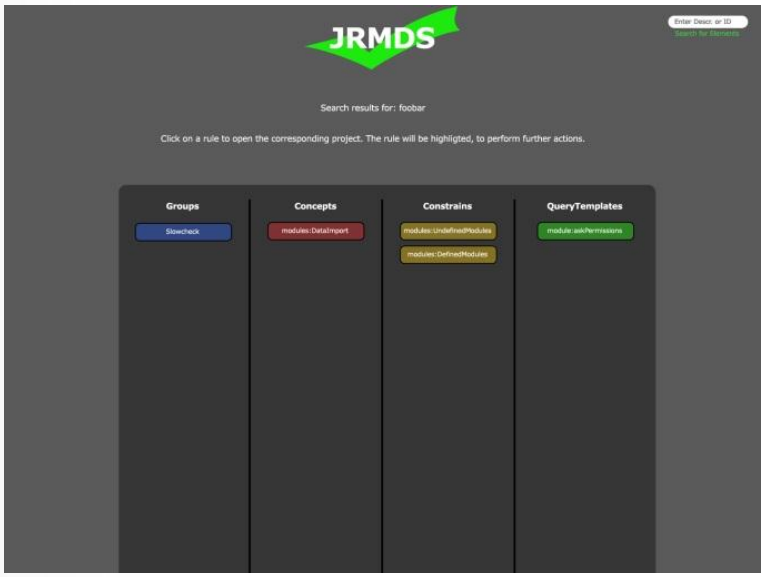


Projektverwaltung



Einstellungen editieren

Suche



6. Entwurfsdatenmodell

7. Testfälle