Modellierungsbeschreibung

Diese Modellierungsbeschreibung stellt die Software zum derzeitigen Planungsstandpunkt vom 07.04.2016 dar und soll als Erweiterung der Entwurfsbeschreibung verstanden werden. Deshalb werden redundante Teile, die in der Entwurfsbeschreibung bereits hinreichend erläutert wurden, nicht noch einmal detailliert beschrieben.

1 Allgemeines

Nachdem wir, wie in vorangegangenen Arbeitsschritten geplant, im Vorprojekt die allgemeine Durchführbarkeit des Projektes erfolgreich nachweisen konnten, ist nun das Ziel die geschaffene Programmbasis zu vervollständigen. Die Vervollständigung orientiert sich dabei an den im Arbeitsplan festgehaltenen und vereinbarten Funktionalitäten. Diese sollen iterativ integriert werden.
Zudem konnten wir uns mit den Funktionsweisen der Frameworks *Jena* und *Vaadin* vertraut machen und einarbeiten. Da diese weiterhin essentiell für die Bearbeitung des Gesamtprojekts sein werden, ist der nun sichere Umgang ein erheblicher Gewinn zum erfolgreichen Abschluss.
Grundsätzlich haben wir bislang keine elementaren Probleme während der Bearbeitung feststellen können, sodass wir davon ausgehen, in der Lage zu sein, die Projektvision mit allen Muss-Funktionalitäten fristgerecht fertigzustellen.

2 Produktübersicht

Die im Vorprojekt implementierten Funktionalitäten wurden bereits in der Entwurfsbeschreibung dargelegt und müssen noch wie folgt erweitert werden.
Zunächst haben wir die interaktive Komponente, also das Manipulieren von Datensätzen noch nicht implementiert. Konkret fehlen noch die Funktionen
-das Ergänzen von Datensätzen und allgemein die Anzeige von Datenlücken in Datensätzen
-das Erstellen von neuen Datensätzen
-das Melden von Fehlern in Datensätzen
Damit wir diese integrieren können, muss das Backend entsprechend erweitert werden. Darüber hinaus sind diese Funktionen nur registrierten Nutzern bzw. einem Administrator-Account möglich. Die komplette Account Verwaltung wird mit dem Shiro Framework realisiert.
Weiterhin fehlt die Möglichkeit Suchergebnisse mit bestimmten Filtern einzuschränken. Beispielsweise soll es dem Nutzer der Webapplikation möglich sein sich nur Lexeme anzeigen zu lassen. Diese Funktion kann auf Ebene der Oberflächenstruktur implementiert werden.
Weitere Kann-Funktionalitäten, welche im Abschnitt 3.2.2 Kann-Funktionalitäten des Arbeitsplans nachzulesen sind, werden erst einmal in der Bearbeitung zurückgestellt und nur bearbeitet, falls die zeitlichen Ressourcen es zulassen.

Zuletzt muss ein Web-Application Server eingerichtet werden, wofür wir *Tomcat* verwenden werden.

3 Grundsätzliche Struktur- und Entwurfsprinzipien

Die 3-Schichten Architektur bleibt auch in den folgenden Releases erhalten. Detaillierte Beschreibungen in wie fern Klassen ergänzt werden, sind im Abschnitt 4 Struktur- und Entwurfsprinzipien einzelner Pakete nachzulesen.
Mit Einrichtung des *Tomcat* Servers muss automatisch die Entscheidung folgen, welche Architektur zur Verteilung der physischen Schichten gewählt wird. Da wir mit *Vaadin* arbeiten, ist die Architektur jedoch bereits vorgeschrieben, bzw. wird wie in folgender Abbildung dargestellt empfohlen.

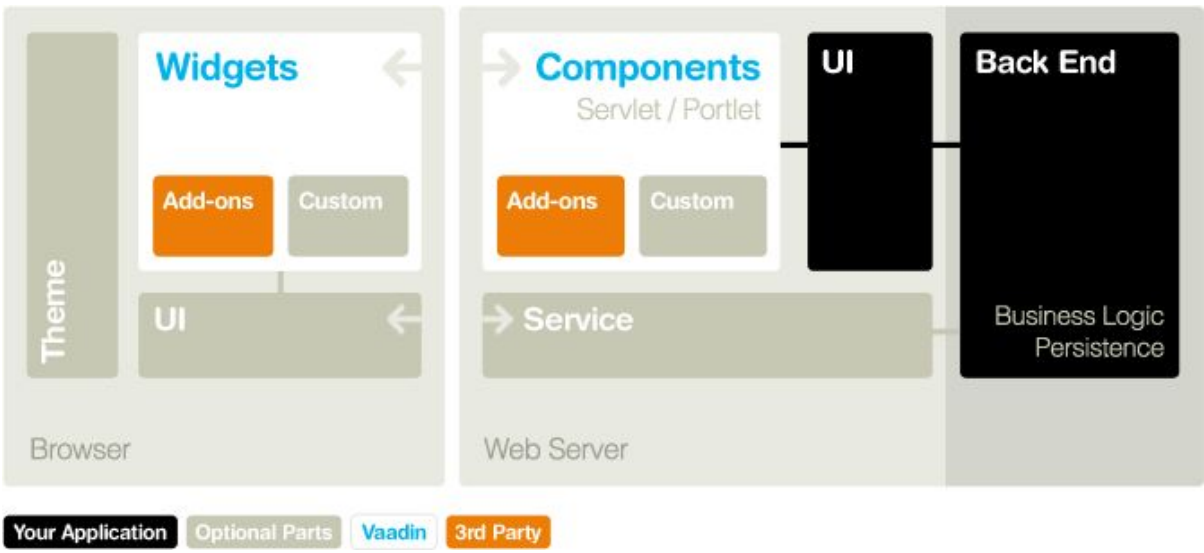

Abb. 1: Schichtenverteilung
(Quelle: https://vaadin.com/blog/-/blogs/vaadin-7-application-architecture)

An dieser Verteilung orientieren wir uns beim Projekt.
Auf dem Web Server befinden sich sowohl das UI mit all seinen grafischen Komponenten, sowie das komplette Backend. Vorzugsweise werden wir beide Einheiten auf dem gleichen Server laufen lassen. Dabei wird *Vaadin* als Servlet oder Portlet eingesetzt und regelt die Kommunikation zwischen Browser und Server, sowie Rendering und Event-Handling vollautomatisch.

4 Struktur- und Entwurfsprinzipien einzelner Pakete

Um neue Datensätze erstellen und bestehende verändern zu können, müssen entsprechende Funktionen in der Klasse **QueryDBSPARQL** ergänzt werden. Eine Modularisierung der Klasse ist nicht geplant, da das Backend als solches komplett bleiben soll und eine Aufteilung generell keine Vorteile mit sich bringen würde.

Um dem Nutzer eine einfache Möglichkeit zu bieten, Datensätze zu ergänzen, werden wir die Klasse **Overview** dahingehend erweitern, dass in der Detailansicht alle potentiellen Properties angezeigt werden und eventuelle Datenlücken deutlich erkennbar sind. Wie genau die fehlenden Einträge zu ergänzen sind, wird zur Implementierung getestet, damit wir eine möglichst intuitive Möglichkeit schaffen können. Vorstellbar wäre beispielsweise ein Button der ein standardisiertes Popup-Fenster öffnet, in dem der Editierungsvorschlag eingereicht werden kann.
Um die Filterfunktion zu integrieren bietet sich eine eigene Klasse an. Als Filter sollen Unterklassen der Klasse Word (vgl. Datenmodell) wählbar sein. Demnach muss die Filter Klasse die Liste der Suchergebnisse übergeben bekommen und diese dann entsprechend des gewählten Filters weiter selektieren. Die Kommunikation zwischen den Klassen erfolgt wie gehabt, an MVP orientierend, über den **SearchPresenter**. Sämtliche grafische Komponenten der Filter Klasse werden in der Klasse selbst implementiert. Diese werden wir ebenfalls erst in der Implementierungsphase genau festlegen und letztendlich in die **SearchUI** einbinden.
Sämtliche Aspekte, welche das Account Management betreffen, wurden bislang noch nicht hinreichend von uns studiert bzw. in der Gruppe verbindlich besprochen, sodass diese erst nach dem nächsten Scrum Meeting verbindlicher manifestiert werden können. Sobald diesbezüglich neue, geplante Vereinbarungen getroffen wurden, werden diese zeitnah dokumentiert und ergänzt. Selbiges gilt für Programmierentscheidungen, die aufgrund verbesserter Praxistauglichkeit getroffen werden und von der hier beschriebenen Vorgehensweise abweichen.

5 Datenmodell

Das Datenmodell bleibt im Vergleich zur Entwurfsbeschreibung erhalten. Für die finale Version der Webapplikation werden jedoch nicht die zum Testen zur Verfügung gestellten deutschen „Schema“ und „Inventory“ Dateien verwendet, sondern die hebräischen. Daraus resultieren jedoch keinerlei Nachteile in der Funktionalität, da diese ebenfalls als RDF Ontologien in der Turtle-Serialisierung vorliegen und die Arbeitsweise sich so nicht ändert.

6 Glossar