Entwurfsbeschreibung Modellierung

Projektnummer: swp10-7
Projekttitel: jQuery Plugin – Erweiterte Autovervollständigung auf der Basis von SPARQL Endpunkten

Abgabe: 28.06.2010

„Es ist nicht die Aufgabe des Auftraggebers, den Systemanalytiker zu verstehen, sondern der Systemanalytiker wird dafür bezahlt, sich dem Auftraggeber verständlich zu machen.“

Verantwortlicher für Dokumentation und Qualitätssicherung: Marina Mitjagin
Projektleiter: Marcus Nitzschke

1. Allgemeines

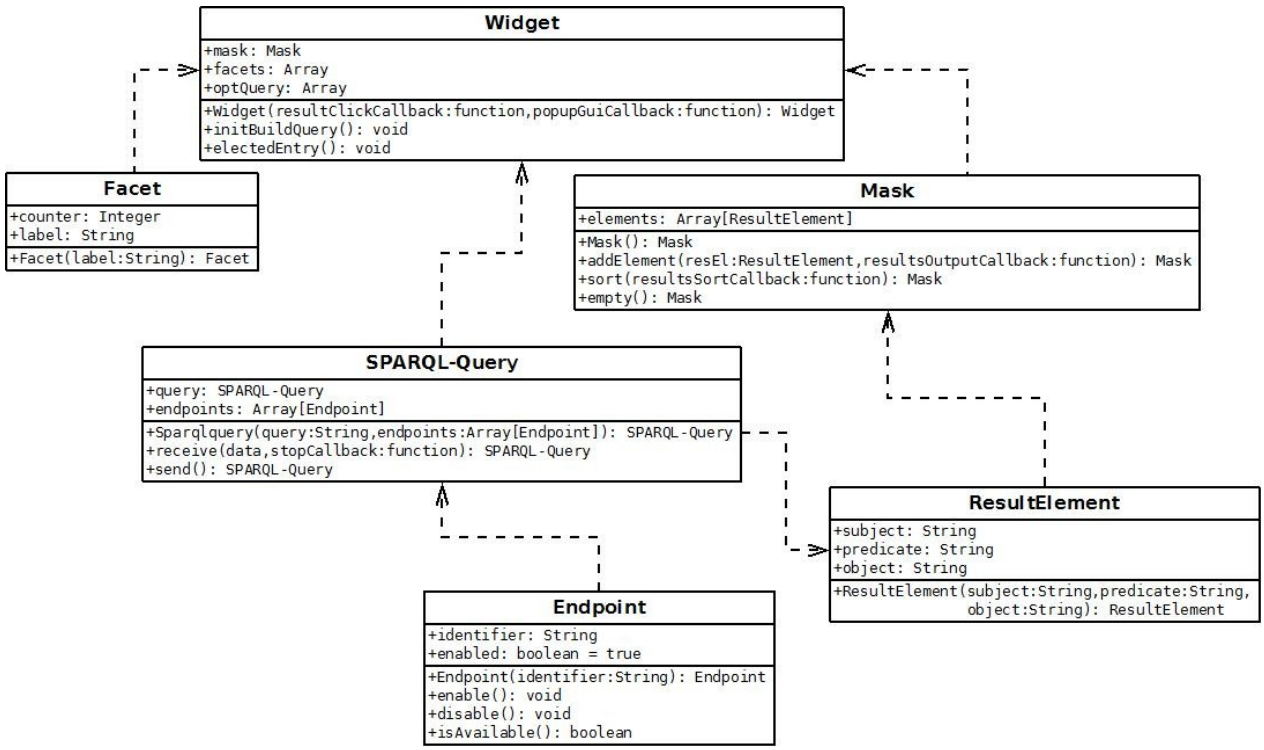
Im Rahmen unserer Semesteraufgabe gilt es ein Wigdet für das OntoWiki zu entwickeln. Dieses soll dem Benutzer ermöglichen über ein Eingabefeld Ressourcen zu definieren, zu editieren oder ggf. auch zu löschen.

Gerade in größeren Projekten, an denen mehrere Personen arbeiten, ist es sinnvoll, sich in der Gruppe vor der Programmierung ein eindeutiges Bild über die Funktionen aus Sicht des Programmierers zu verschaffen.

Dies geschieht mit den Ziel, die anstehende Implementierung zu erleichtern.

2. Produktübersicht

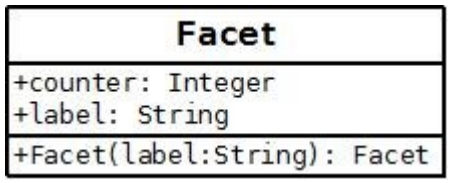
Die Ziele der Objektorientierten Analyse sind einerseits, die Anforderungen und Wünsche eines Auftraggebers, an das System zu ermitteln und zu beschreiben, andererseits aber auch ein Konzept zu erstellen, welches konsistent, vollständig, eindeutig und realisierbar ist.
Ein Teil der objektorientierten Analyse, ist das statische Modell. Dieses beschreibt, die Klassen des Konzeptes, die Beziehungen zwischen den Klassen, deren Vererbung und die Attribute der Klassen.
Hier sei besonders erwähnt, dass alle Attribute öffentlich(public) sind, da JavaScript keine Implementierung für private-Attribute vorsieht.



Klassendiagramm zum Plugin

Unser Konzept besteht aus den 6 Klassen:
Widget, Facet, Mask, SPARQL-Query, ResultElement und Endpoint.
Zur Realisierung der Methodenverkettung geben alle Methoden ein Objekt ihrer Klasse wieder.

2.1 Die Facetten



Die Facetten dienen der Sortierung und der Filterung der Suchergebnisse. Somit realisiert diese Klasse einen Großteil der Benutzerfreundlichkeit.

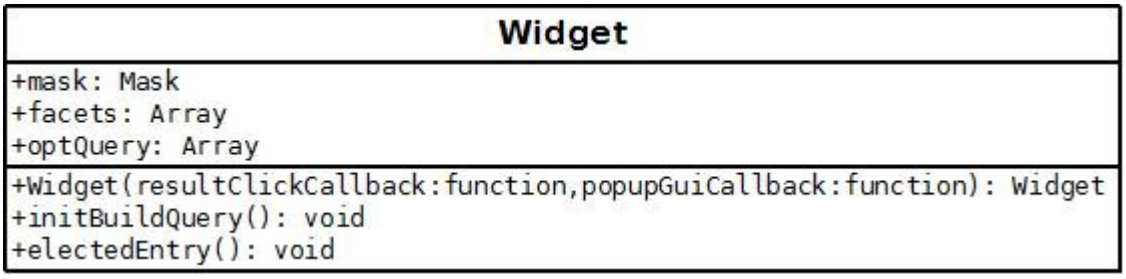
Die Attribute:

+counter: Integer → Diese Zahl gibt die Priorität einer Facette an
+label: String → Der Name der Facette

Die Methoden:

+Facet(label:String): Facet → der Konstruktor zur Erzeugung eines Objekts

2.2 Das Widget

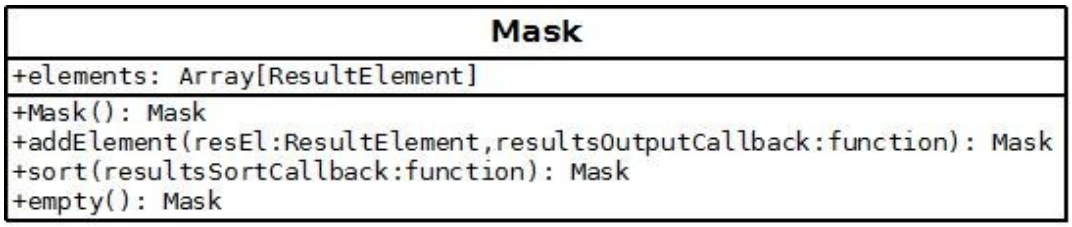


Widget bildet die Hauptklasse des Systemes. In ihr werden direkt oder indirekt Objekte aller anderen Klassen erzeugt und verwendet.

Attribute:
+mask: Mask → die Maske in der die Ergebnisse angezeigt werden
+facets: Array → das Array der verfügbaren Facetten
+optQuery:Array → Array optionaler statements die zum Hauptquery hinzugefügt werden

Methoden:
+Widget(): Widget → der Konstruktor zur Erzeugung eines Objektes
+initBuildQuery(): void → Diese Funktion wird nach jeder Tipppause aufgerufen
+electedEntry(): void → Diese Funktion setzt die Werte des Inputfeldes

2.3 Die Maske

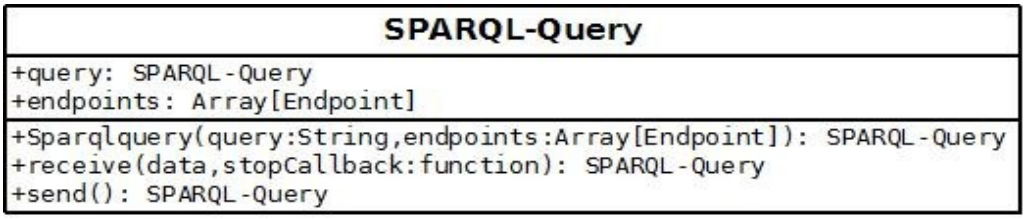


In der Maske werden die Ergebnisse angezeigt. Hier können sie auch mittels der Facetten gefiltert und sortiert werden.

Attribute:
+elements: Array[ResultElement] → Das Feld aller Ergebnisse

Methoden:
+Mask(): Mask → Der Konstruktor zur Erzeugung eines Objektes
+addElement(resELI:Resultelement, resultOutputCallback: function): Mask
→ Hinzufügen von Elementen
+sort(resultSortCallback:function): Mask
→ Hiermit können die Ergebnisse sortiert werden
+empty(): → leert die Maske

2.4 Die SPARQL-Query



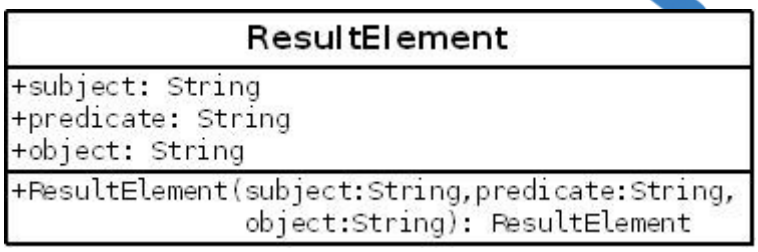
In dieser Klasse werden die Anfragen realisiert.

Attribute:
+query: String → Die SPARQL Anfrage
+endpoints: Array[Endpoint] → Ein Feld aller SPARQL-Endpunkte

Methoden:
+Sparqlquery(query:String,endpoints:Array[Endpoint]): SPARQL-Query
→ der Konstruktor zur Erzeugung eines Objektes
+receive(data, stopCallback:function): SPARQL->Query
→ die Funktion empfängt die Daten, parst diese und leitet weiterführende Aktionen ,wie die Ausgabe und Kalkulation der Facetten, ein
+send(): SPARQL-Query → Hiermit wird die Anfrage gesendet

2.5 Ergebniselement

Mit dieser Klasse werden die Ergebniselemente realisiert.

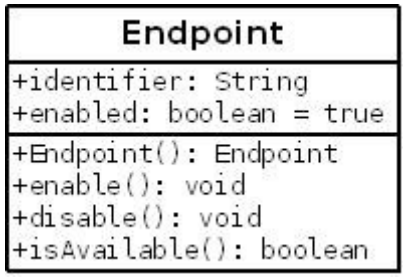


Attribute:
+subject: String → Bestandteil eines Ergebnistripels
+predicate: String → Bestandteil eines Ergebnistripels
+object: String → Bestandteil eines Ergebnistripels

Methoden:
ResultElement(subject: String,predicate: String,object: String): ResultElement
→ Der Konstruktor zur Erzeugung eines Objektes

2.6 Endpunkt

Mit dieser Klasse werden die SPARQL-Endpunkte realisiert.



Attribute:
+identifier: String → Der Name des Endpunktes
+enabled: boolean → Ob der Endpunkt aktiviert ist

Methoden:
+Endpoint(): Endpoint → Der Konstruktor zur Erzeugung eines Objektes
+enable() → Aktivieren des Endpunktes
+disable() → Deaktivieren des Endpunktes
+isAvailable(): boolean → Überprüfung, ob der Endpunkt verfügbar ist

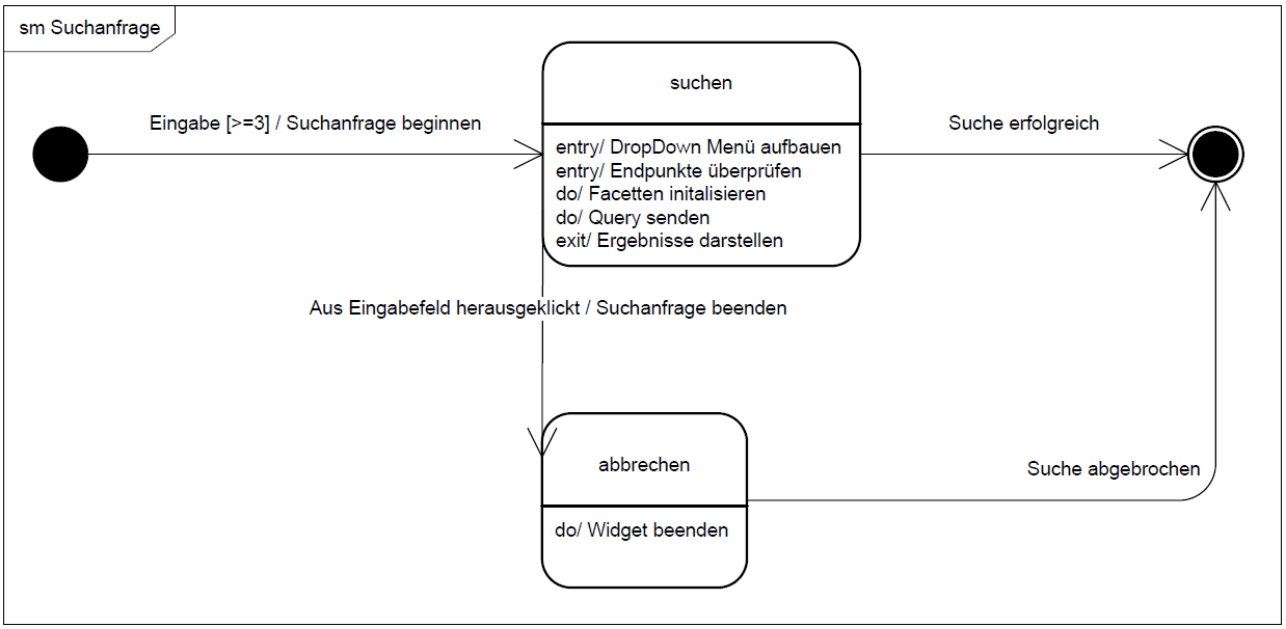
3. Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem

Die im Pflichtenheft erwähnten Funktionen werden nun ausgewählt, um sie näher, anhand von ausgewählten UML- Diagrammen, zu beschreiben.

3.1 Suchanfrage

In dem Zustandsdiagramm "Suchanfrage" ist der Vorgang einer Suchanfrage durch den Nutzer modelliert. Der Anfangszustand wechselt in den Zustand "suchen", wenn der Auslöser und die Einschränkung durch den Wächter erfüllt wurden, um mit der Suchanfrage zu beginnen. Damit eine Suchanfrage beginnt, muss der Nutzer mindestens 3 Zeichen eingegeben haben (F10). Vor dem Eintritt in den Zustand suchen wird das DropDown Menü aufgebaut, sowie alle durch den Nutzer ausgewählten Endpunkte auf Verfügbarkeit geprüft. Nachdem die Eintrittsfunktionen

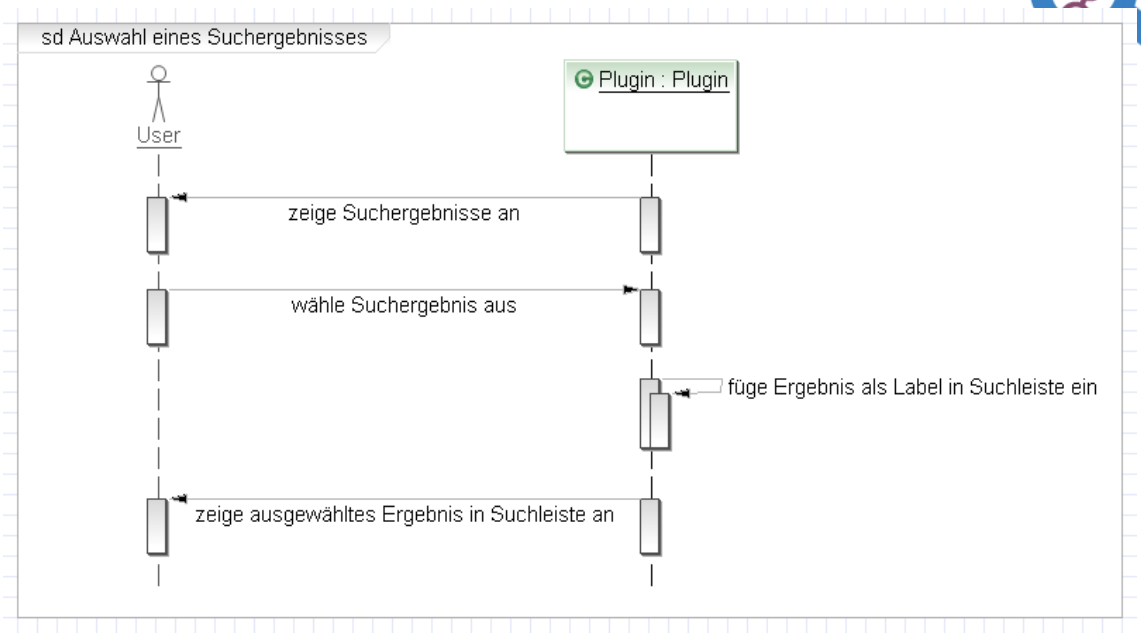
ausgeführt wurden, werden die Query gesendet und die Facetten initialisiert. Die Austrittsaktion stellt schließlich die Ergebnismenge dar und geht in den terminierenden Zustand über. Während des Zustands "suchen" ist es jedoch möglich, dass der Nutzer aus dem Eingabefeld heraus klickt. Dies bewirkt ein Abbrechen des Suchvorgangs (F20) und das Widget beendet seine Aufgabe. Es wird danach der Terminalzustand erreicht.



Zustandsautomat zur Beschreibung einer einfachen Suchanfrage

3.2 Auswahl eines Suchergebnisses

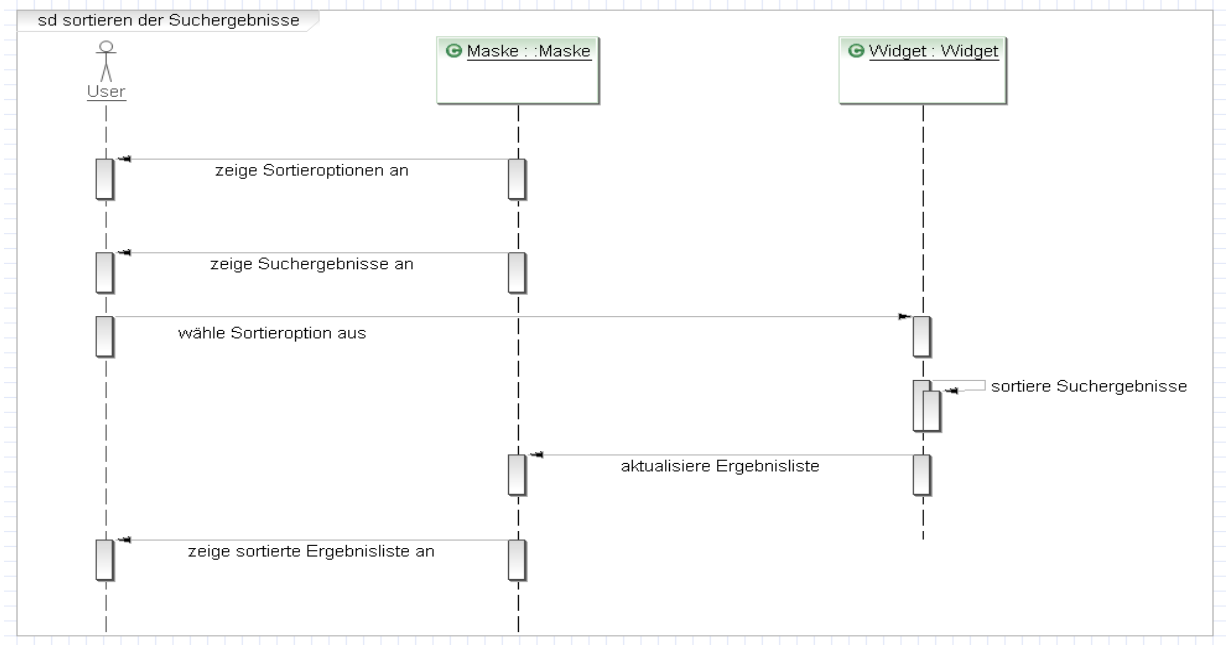
Die Suchergebnisse werden in einem Dropdown Menü angezeigt. Der Nutzer wählt ein Ergebnis durch Anklicken aus, welches dann als Label in die Suchleiste eingefügt wird.



Sequenzdiagramm zur Auswahl eines Suchergebnisses

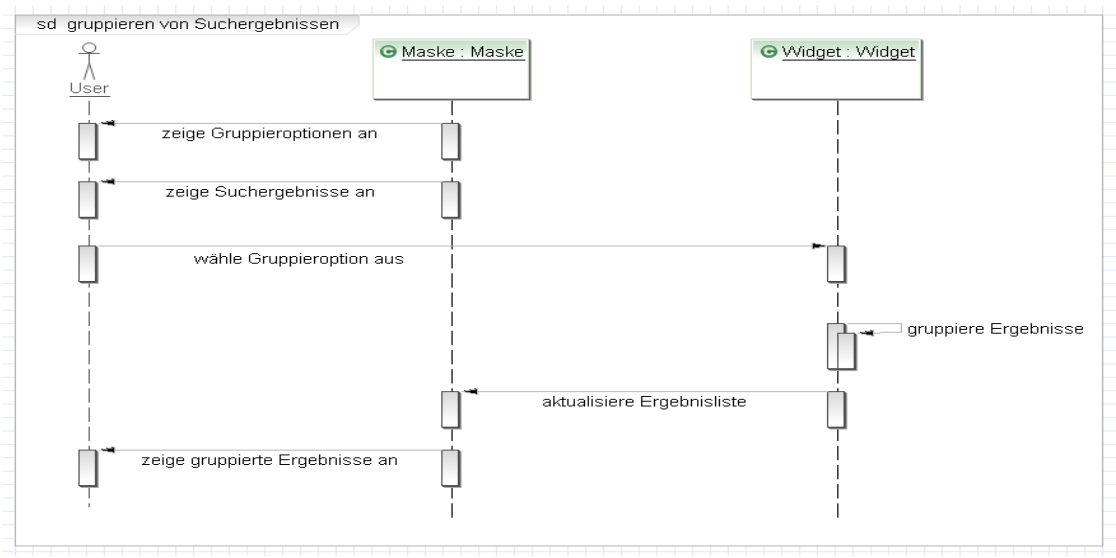
3.3 Sortieren der Suchergebnisse

Die Suchergebnisse und Sortieroptionen werden dem Nutzer in einer Maske angezeigt. Der Nutzer kann dann eine Sortieroption auswählen, die an das Widget weitergeleitet wird. Das Widget sortiert die Ergebnisliste entsprechend der Option und aktualisiert die DOM-Struktur. Die sortierte Ergebnisliste wird dann in der Maske dem Nutzer angezeigt.



Sequenzdiagramm zum Sortieren der Suchergebnisse

3.4 Gruppieren der Suchergebnisse

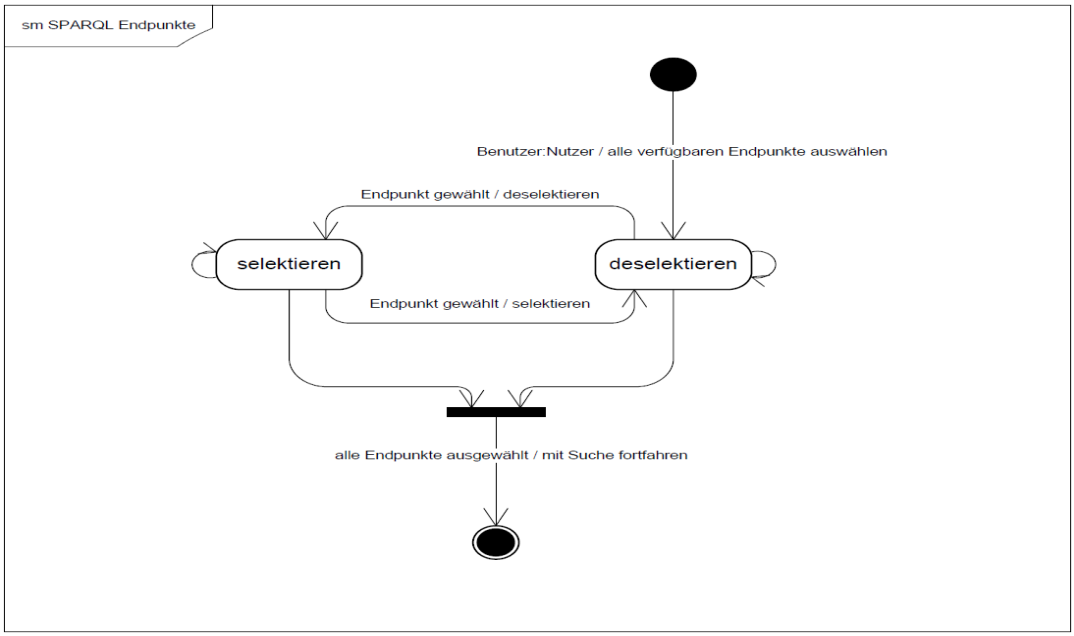


Sequenzdiagramm zum Gruppieren von Suchergebnissen

Die Suchergebnisse und Gruppieroptionen werden dem Nutzer in einer Maske angezeigt. Der Nutzer kann dann eine Gruppieroption auswählen, die an das Widget weitergeleitet wird. Das Widget gruppiert die Ergebnisliste entsprechend der ausgewählten Option und aktualisiert die DOM-Struktur. Die gruppierten Ergebnisse werden dann in der Maske dem Nutzer angezeigt.

3.5 SPARQL Endpunkte

Das Zustandsdiagramm "SPARQL Endpunkte" simuliert die beiden Muss-Funktionen "SPARQL Endpunkte aktivieren" (F31) und "SPARQL Endpunkte deaktivieren" (F32). Zu Beginn sind dem Nutzer alle verfügbaren, durch den Administrator definierte SPARQL Endpunkte (F30), ausgewählt. Nun wird dem Nutzer die Möglichkeit geboten, SPARQL Endpunkte zu deselektieren bzw. zu selektieren. Nachdem die gewünschten Endpunkte ausgewählt wurden, wird der Terminalzustand erreicht. Es kann mit der Suche fortgesetzt werden.

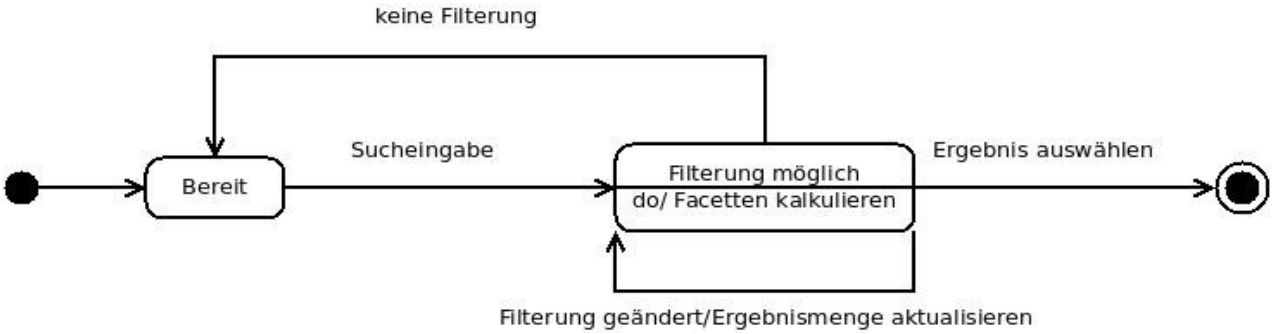


Zustandsautomat zu SPARQL-Endpunkten

3.6 Filterung durch Facetten

Der Ablauf für den Prozess /F40/ - Filterung durch Facetten gestaltet sich so,dass man nachdem 3 Zeichen in die Suche eingetippt wurden, die Möglichkeit hat aus den vorgeschlagenen Facetten, welche aus der aktuellen Ergebnismenge berechnet wurden, eine Filterung vorzunehmen. Wählt man nun eine Facette hinzu, wird die Ergebnismenge eingegrenzt und es werden ebenfalls die verfügbaren Facetten aktualisiert.

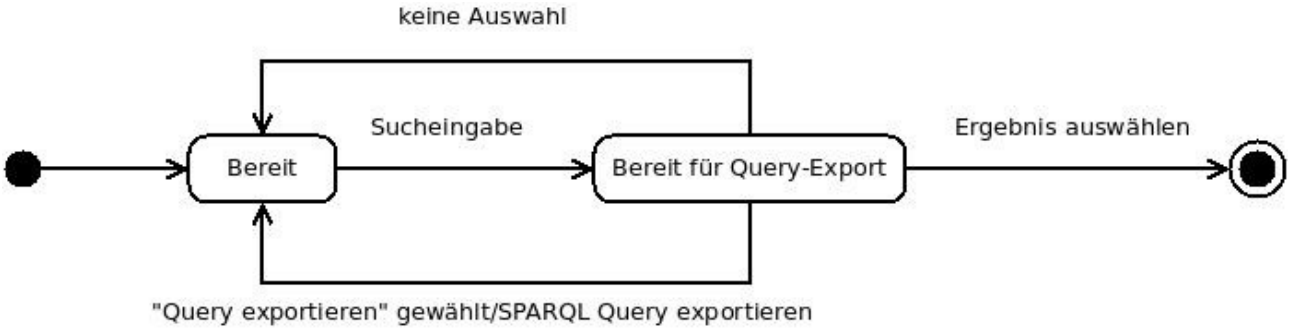
Diesen Zyklus kann man nun mehrmals durchlaufen, bis das passende Ergebnis gefunden und auswählt wurde. Es ist aber auch möglich ohne jegliche Filterung einfach durch weitere Sucheingaben zu einem Ergebnis zu kommen.



Zustandsautomat zur Filterung von Facetten

3.7 Query-Export

Funktion /F130/ - Query-Export gestaltet sich ziemlich trivial. Es besteht nach jedem Suchvorgang, bzw. präziser nach jeder minimalen Sucheingabe die Möglichkeit über einen Button den im Hintergrund angeforderten Query als Klartext zu exportieren. Nichtsdestotrotz ist natürlich auch ein Such-Zyklus ohne diese Funktion möglich.



Zustandsautomat zum Exportieren von Queries

4. Grundsätzliche Struktur- und Entwurfsprinzipien für die einzelnen Pakete

Es sind keine Pakete vorhanden.