SEARCH–Tag EEXCESS SECH–Browser

Programmier Leitfaden
Spezifikation, Konstruktion

Prof. Dr. Peter Stöhr Gottfried von Recum
Alexander Pöhlmann Lothar Mödl Burak Erol
Brian Mairhörmann Andreas Netsch Philipp Winterholler
Andreas Ziemer

Version 0.4.9

Inhaltsverzeichnis

Teil I.

Einleitung

1. Übersicht

1.1. Abschnitt

1.1.1. Teilabschnitt

1.1.1.1. Unterteilabschnitt

Paragraph

Unterparagraph

1. Eine
2. kleine
3. Aufzählung

2. User Interface

2.1. Abschnitt

2.1.1. Teilabschnitt

2.1.1.1. Unterteilabschnitt

Paragraph

Unterparagraph

1. Eine
2. kleine
3. Aufzählung

3. Programmlogik

3.1. Abschnitt

3.1.1. Teilabschnitt

3.1.1.1. Unterteilabschnitt

Paragraph

Unterparagraph

1. Eine
2. kleine
3. Aufzählung

Teil II.

Spezifikation

4. Browser

Im Folgenden werden die verschiedenen Anwendungsfälle des SECH-Browsers übersichtlich vorgestellt. Sie veranschaulichen die für den Benutzer verfügbaren Funktionalitäten des Browsers.

4.1. Menüführung

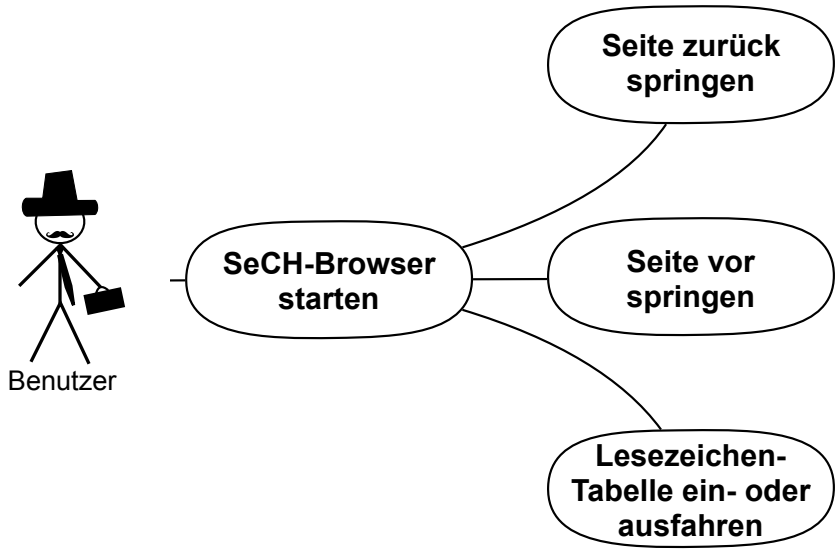

Abbildung 4.1.: Anwendungsfall „navigieren“

Dieser Anwendungsfall zeigt die wesentlichen Funktionen des Browsers an. Der Nutzer kann auf einer Homepage eine Seite zurück- bzw. wieder vorspringen und er hat die Möglichkeit die Lesezeichentabelle aus- und auch wieder einzufahren.

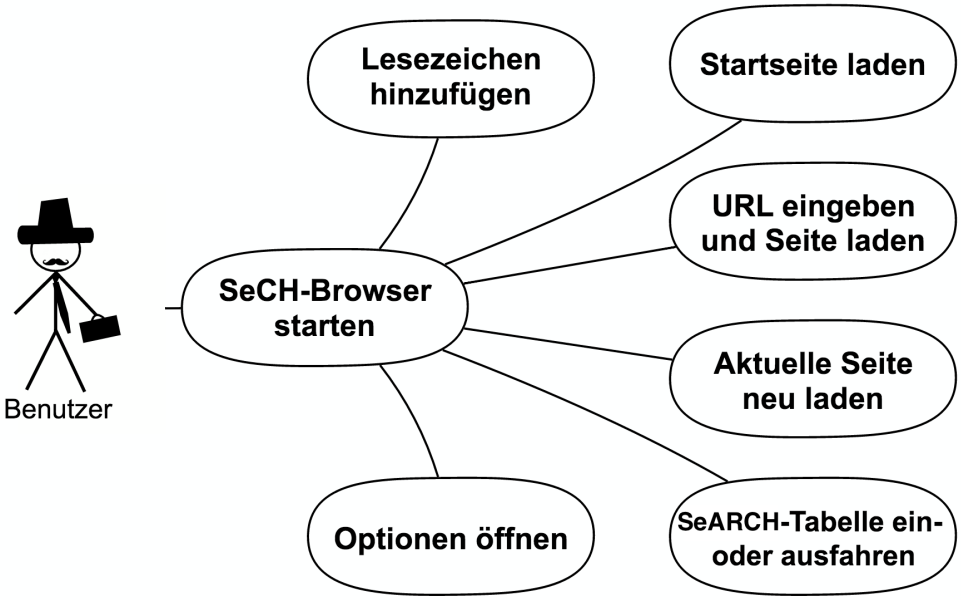

Abbildung 4.2.: Anwendungsfall „browsen“

In diesem Anwendungsfall werden weitere Navigationselemente des Browsers vorgestellt. Der Nutzer kann:

- eigene Lesezeichen hinzufügen
- die von ihm definierte Startseite laden
- eine Webadresse in die Adresszeile eingeben und diese dann aufrufen
- die aktuelle Seite neu laden
- die SEARCH-Tabelle aus- und einfahren
- das Fenster für Optionen öffnen.

4.2. Lesezeichenverwaltung

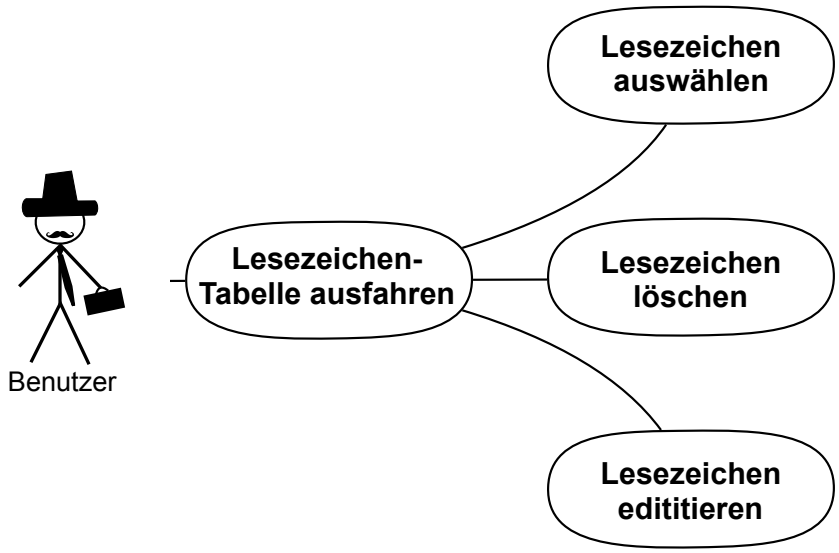

Abbildung 4.3.: Anwendungsfall „Lesezeichen“

Ist die Lesezeichentabelle ausgefahren, so kann der Benutzer ein persönlich angelegtes Lesezeichen auswählen und es wird die von ihm gewünschte Seite aufgerufen. Das Löschen und das Editieren ausgewählter Lesezeichen ist ebenfalls in der Tabelle möglich.

4.3. SEARCH-Tagverwaltung

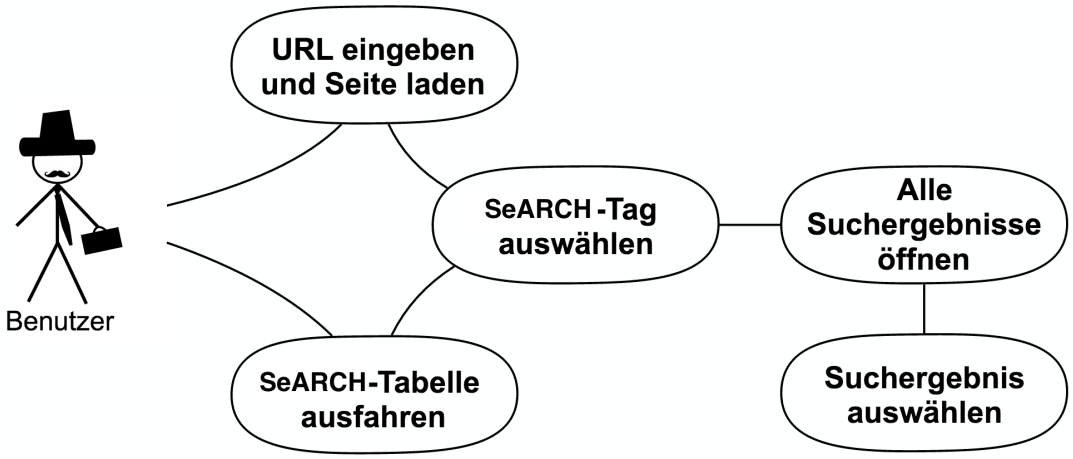

Abbildung 4.4.: Anwendungsfall „SEARCH-Tagverwaltung“

Der Nutzer kann sich auf zwei unterschiedlichen Wegen die Suchergebnisse für das gewünschte SEARCH-Tag anzeigen lassen. Zunächst muss der Nutzer durch Eingabe und Bestätigung einer Webadresse die Seite aufrufen. Die erste Variante ist das hervorgehobene SEARCH-Tag anzutippen. Alternativ kann der Nutzer in einer zweiten Variante durch Ausfahren der SEARCH-Tabelle ein gewünschtes SEARCH-Tag antippen. Beide Wege führen zur Anzeige des ersten Suchergebnisses des jeweiligen SEARCH-Tags. Anschließend kann der Nutzer sich alle weiteren Suchergebnisse zu einem SEARCH-Tag anzeigen lassen.

4.4. Einstellungen

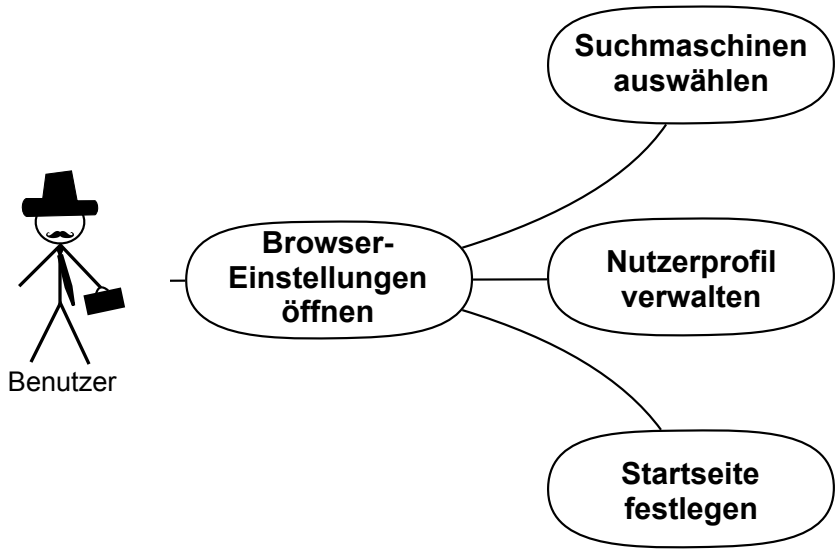

Abbildung 4.5.: Anwendungsfall „Einstellungen“

Befindet sich der Nutzer in den Browsereinstellungen, so hat er die Möglichkeit die Suchmaschinen die für die Suchergebnisse der SEARCH-Tags verwendet werden sollen auszuwählen. Möchte er bestimmte Suchmaschinen nicht verwenden kann er diese abwählen. Außerdem kann der Benutzer sein persönliches Nutzerprofil verwalten und seine Startseite festlegen, die direkt nach dem Öffnen der Anwendung als erste Seite geladen wird.

Teil III.

Konstruktion

5. Übersicht

Die Konstruktion gliedert sich in drei Abschnitte:

1. Übersicht
2. User Interface
3. Programmlogik

6. User Interface

Im Folgenden wird ein Überblick über die Klassenabhängigkeiten des User Interfaces gegeben, der Ablauf bei der Anzeige eines SEARCH-Links erklärt sowie die Packages innerhalb des User Interfaces erläutert.

6.1. Klassenabhängigkeiten

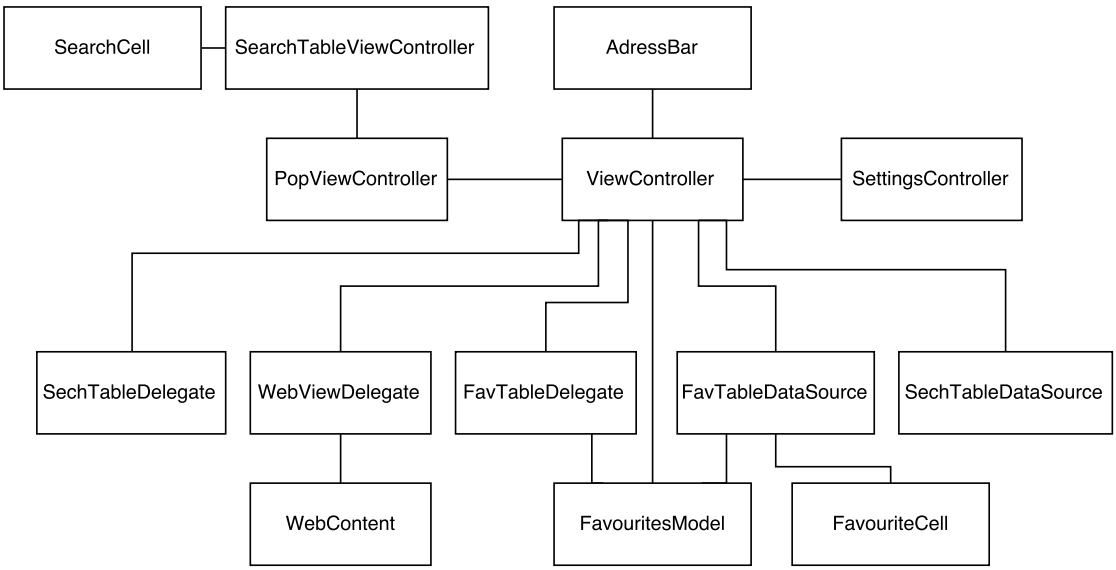

Abbildung 6.1.: Klassenabhängigkeiten im UI

Die obige Darstellung zeigt die Verbindungen der Klassen des User Interfaces. Hierbei steht der `ViewController` im Mittelpunkt, da von ihm die meiste Funktionalität ausgeht.

6.2. Programmablauf

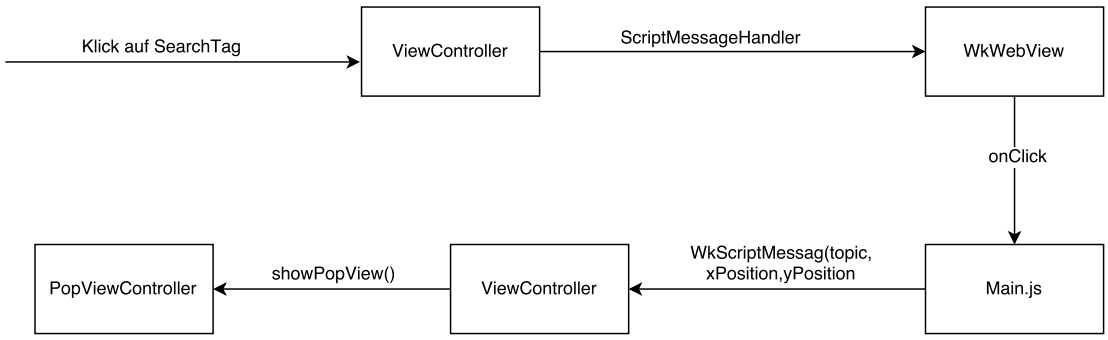

Abbildung 6.2.: SEARCH-Tag Anzeige

Obige Grafik zeigt den Programmablauf nach dem Tippen auf einen SEARCH-Tag. Die Position des Tipps wird über JavaScript ausgelesen und an den WkWebview weitergeleitet. Dieser erzeugt an der übertragenen Position einen `PopView` und zeigt diesen an.

6.3. Packages

6.3.1. ViewController

Der `ViewController` dient zur Steuerung der Funktionen welche direkt mit dem Interface zusammen gehören. Dazu gehören IBActions welche von Buttons oder anderen Interaktionsflächen angestoßen werden können, sowie Funktionen welche direkt mit diesen zusammenhängen, beziehungsweise aus diesen resultieren. Der `ViewController` ist die zweite Schicht nach dem User Interface. Im Folgenden werden die einzelnen `ViewController` mit ihren jeweiligen Funktionen beschrieben. Der `ViewController` ist der Hauptviewcontroller für die normale Browseransicht mit der Adresszeile und dem WkWebView. Er verwaltet die Kopfzeile mit Adresszeile sowie den Buttons für Vorwärts, Zurück, Lesezeichen, Lesezeichen hinzufügen, neu laden, SECH-Tabelle sowie die Einstellungen.

Die Datei `Main.js` welche vom `ViewController` hinzugefügt und dann vom WkWebView ausgeführt wird, markiert alle gefundenen SEARCH-Tags. Außerdem wird im Falle eines Tipps die Position sowie die Id des SEARCH-Tags übermittelt. Die Datei `readHead.js` enthält den Javascript Code zur Auslesen des Kopfbereiches einer HTML-Seite.

Der `PopViewController` ist zuständig für die Verwaltung eines neuen `PopViews`, beim tippen auf einen SEARCH-Tag. Er verwaltet die Anzeige des ausgewählten SEARCH-Links in einem neuen WkWebView sowie den Button für die Auswahl der verschiedenen Suchergebnisse zu einem SEARCH-Tag. Der `SearchTableViewController` verwaltet die einzelnen Suchergebnisse zu einem SEARCH-Tag. Alle Ergebnisse werden sortiert in einer Tabelle mit Bild, Titel und Link angezeigt. Der `SettingsController` ist für die Verwaltung der Browsereinstellungen zuständig.

6.3.2. Delegate

Die Delegates im SECH-Browser dienen zur Bereitstellung einer einheitlichen Struktur innerhalb des Programms. Von den einzelnen Delegate Klassen werden Funktionen zur Darstellung von Informationen oder interne Protokolle aufgerufen.

Im `WebViewDelegate` werden die Funktionen zur Validierung der Website mit Hilfe der JavaScript Dateien und die Extraktion der SEARCH-Tags angestoßen.

In der Klasse `FavTableDelegate` werden die Lesezeichen bereitgestellt. Der Benutzer hat die Möglichkeit ein ausgewähltes Lesezeichen auszuwählen und es sich im Browser anzeigen zulassen. Zusätzlich enthält die Klasse die Funktionen, um Lesezeichen bearbeiten und löschen zu können. Der `SechTableDelegate` enthält die Routine zum Darstellen von SEARCH-Links aus der SECH-Tabelle.

6.3.3. DataSource

Die DataSources im SECH-Browser dienen der Bereitstellung der Daten in den Tabellen. Von den einzelnen Klassen werden jeweils Funktionen zum Befüllen der einzelnen Zellen aufgerufen.

In der Klasse `FavTableDataSource` werden zunächst alle gespeicherten Lesezeichen geladen. Danach werden die einzelnen Zellen der Tabelle mit den Lesezeichen befüllt.

Im `SechTableDataSource` werden alle SEARCH-Tags die auf der Seite gefunden wurden in den Zellen der SECH-Tabelle dargestellt. Weiterhin befinden sich noch Funktionen zum Erweitern der SEARCH-Tags in der Klasse.

6.3.4. Components

In den Components werden die spezifischen Zellen für die Lesezeichentabelle und SEARCH-Tagtabelle modelliert, die in den jeweiligen Delegates verwendet werden. Weiterhin ist in den Components die Validierung für die Adresszeile definiert, welche die Eingaben in der Suchleiste überprüft und verwaltet.

6.3.5. Persistence

Im Persistence-Package befinden sich Models die für eine persistente Speicherung gedacht sind. Das `FavouritesModel` stellt einerseits das nötige Model, andererseits die Funktion für das Speichern von Lesezeichen zur Verfügung.

6.3.6. WebContent

Der `WebContent` stellt die Schnittstelle zwischen dem User Interface und der `SearchExtraction` dar. Er beinhaltet HTML-Code, in Form eines Strings, sowie die URL, von welcher der HTML-Code stammt. Der HTML-Code ist in den Head und den Body der Website aufgeteilt. Aus dem `WebContent` wird in der `SearchExtraction` ein `SearchModel` erzeugt.

7. Programmlogik

7.1. TaskCtrl

Der `TaskCtrl` ist eines der wichtigsten Elemente bei der Kommunikation zwischen der Oberfläche und dem Abrufen der Daten aus dem Internet. Er bestimmt mit Hilfe der vom Nutzer getroffenen Einstellungen an welche Suchmaschine(n) die Anfrage geschickt wird/werden. Die Einstellungen werden durch das `Settings.bundle` definiert und können durch die Klasse `SettingsManager` abgerufen werden.

Der `TaskCtrl` wird von der Klasse `SearchResults` initialisiert und aufgerufen. Zuerst werden unter Verwendung der Klasse `QueryCreationCtrl` Daten in einem allgemeingültigen Format für die Suchanfragen erstellt. Diese Daten werden daran anschließend von den jeweiligen `QueryBuildern` (`EEXCESSJSONBuilder`, `DuckDuckGoURLBuilder` und `FarooURLBuilder`) in das für die jeweilige Suchmaschine passende Datenformat umgewandelt. Die Suchanfragen werden asynchron durch den entsprechenden `ConnectionCtrl` (`JSONConnectionCtrl` und `URLConnectionCtrl`) versendet. Bei erfolgreicher Suche und nach erfolgreichem Parsen der Ergebnisse werden diese dann mit Aufruf der Methode `setRecommendation(status:String, message:String, result:SearchResult)` an den Aufrufer, die Klasse `SearchResults`, zurückgegeben. Diese speichert die Suchergebnisse in einem Cache zwischen. Der `TaskCtrl` wird nur aufgerufen, falls zu dem jeweiligen SEARCH-Tag noch keine Ergebnisse im Cache vorhanden sind.

7.2. SEARCHExtraction

Die `SEARCHExtraction` nimmt ein `WebContent`-Objekt entgegen und verarbeitet es zu einem `SEARCHModels`-Objekt. Hierbei wird der HTML-Code, der im `WebContent` enthalten ist, auf SEARCH-Tags abgesucht. Diese SEARCH-Tags werden dann zu einzelnen `SEARCHModel`-Objekten zusammengebaut und in einem gemeinsamen `SEARCHModels`-Objekt gesammelt.

Für den Ablauf der Analyse und die Erzeugung des `SEARCHModels`-Objektes ist der `SearchManager` zuständig. Die Methoden für die Analyse des HTML-Codes stellt die `RegexForSEARCH` zur Verfügung.

Die Darstellung beschreibt den Ablauf und die Schnittstellen zwischen dem `WebContent` und dem `SEARCHModels`-Objekt. Der `WebContent`, erzeugt vom `WebViewDelegate`, wird in Schritt 1 dem `SearchManager` übergeben. Der `SearchManager` erzeugt dann aus dem

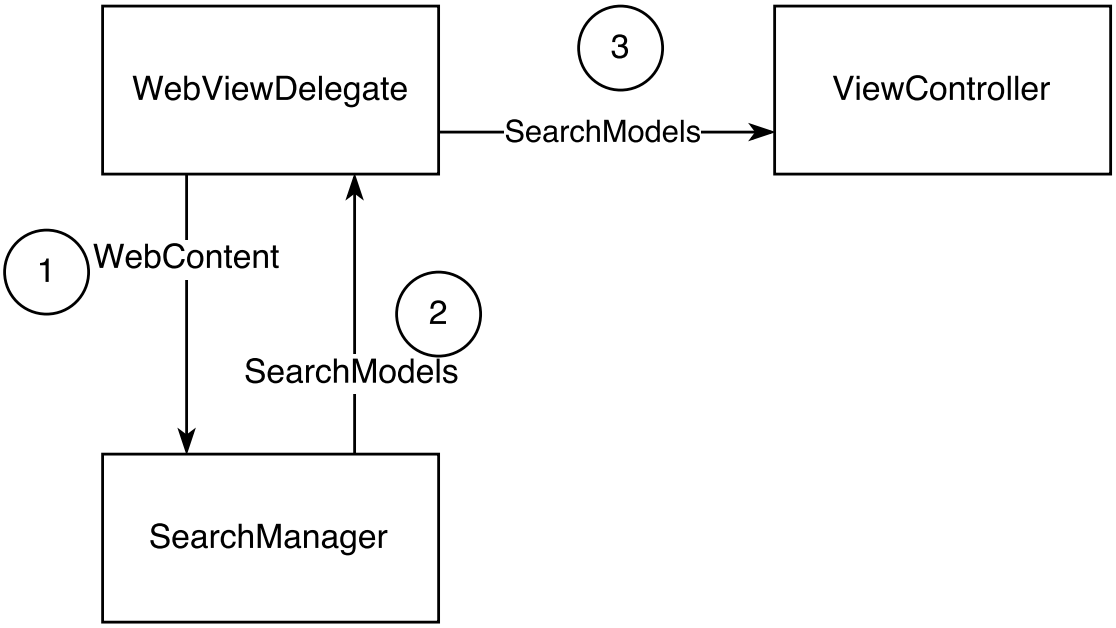

Abbildung 7.1.: Umwandeln eines WebContent-Objekts in ein SEARCHModels-Objekt

`WebContent` das `SEARCHModels`-Objekt und gibt dieses im folgenden Schritt 2 an den `WebViewDelegate` zurück. Der `WebViewDelegate` übergibt nun das `SEARCHModels`-Objekt an den `ViewController`, wo es schließlich für die anderen Packages zur Verfügung steht.

7.3. SEARCHModels

`SEARCHModels` stellt die Schnittstelle zwischen der `SEARCHExtraction` einerseits und dem `TaskController` andererseits dar. Von der `SEARCHExtraction` erzeugt, beinhaltet ein `SEARCHModels`-Objekt eine Sammlung von `SEARCHModel`-Objekten, erzeugt aus dem Inhalt einer Website. In einem `SEARCHModel`-Objekt werden SEARCH-Tag-Links mit der zugehörigen SEARCH-Tag-Section und/oder dem SEARCH-Teg-Head sowie möglichen Filtern gespeichert.

Zur globalen Identifikation wird in einem `SEARCHModel` die URL der Website von der es kommt gesetzt. Um ein `SEARCHModel` innerhalb einer Seite zu finden wird ein Index bezüglich der Position in der Seite gesetzt.

7.4. QueryCreation

`QueryCreation` generiert aus den Suchwörtern (`SearchModel`) und den Geräte- und App-Einstellungen ein allgemeines Anfrageformat. Dieses Anfrageformat wird anschließend in

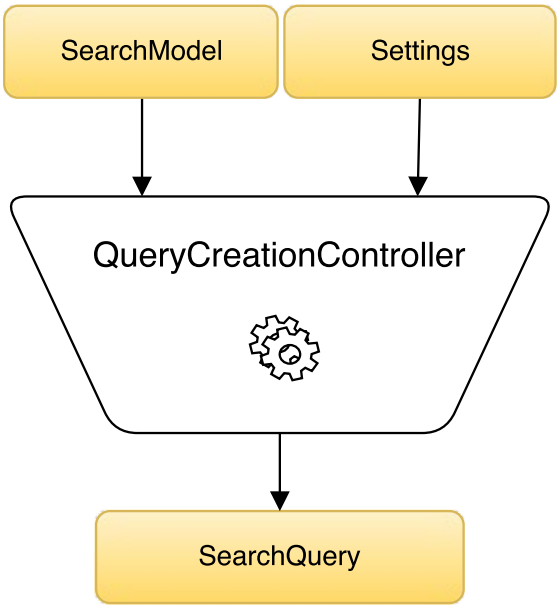

Abbildung 7.2.: Aufbau des Moduls `QueryCreation`

`QueryResolution` zu einem für jede Suchmaschine spezifischen Suchformat umgewandelt.

7.5. QueryResolution

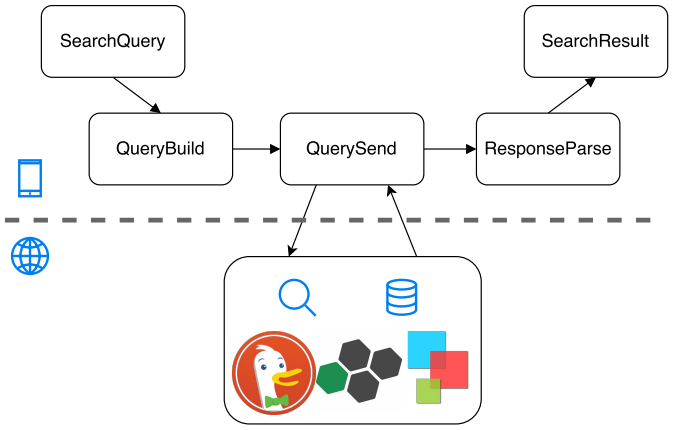

Abbildung 7.3.: Übersicht des Moduls QueryResolution

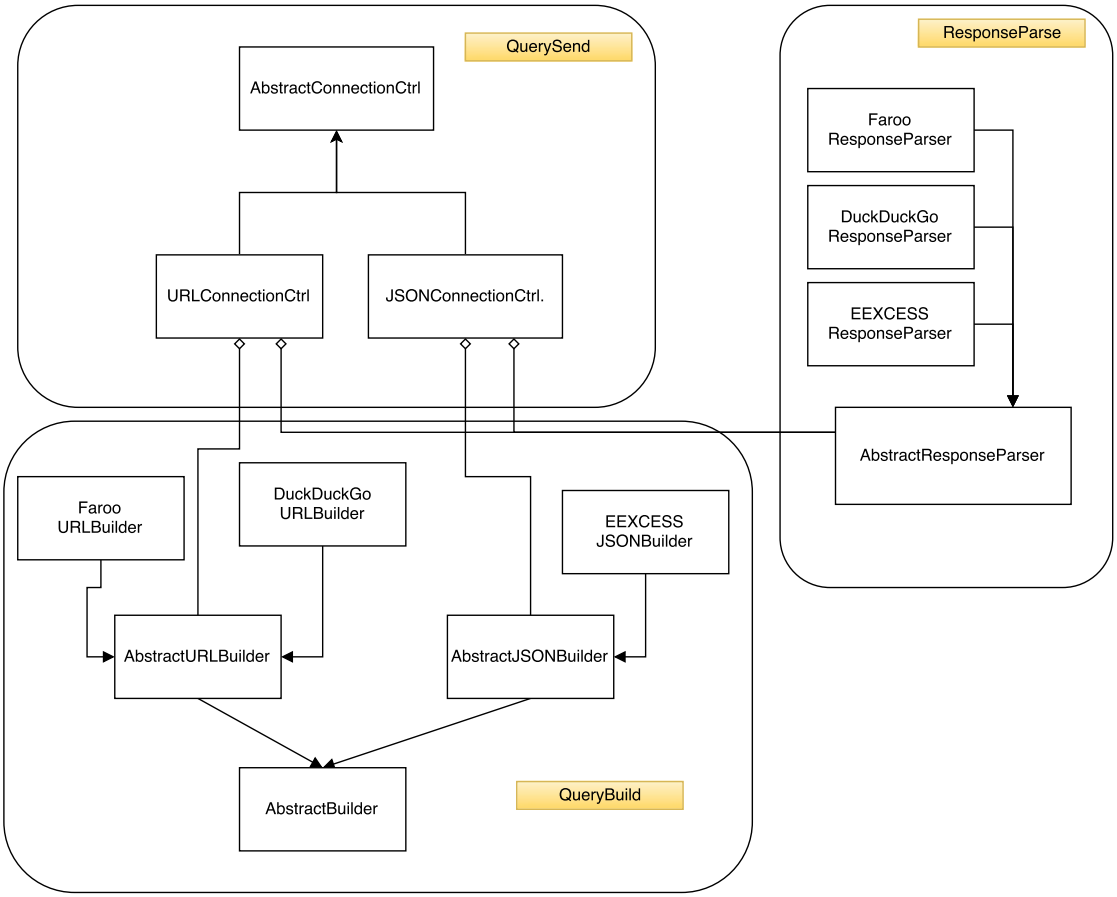

Abbildung 7.4.: Aufbau des Moduls QueryResolution

7.5.1. QueryBuild

Das Modul `QueryBuild` liefert Hauptintelligenz im Modul `QueryResolution`. Seine Aufgaben sind es, die `ConnectionController` in `QuerySend` zu konfigurieren, die Query in das suchmaschinenspezifische Anfrageformat umzuwandeln und anschließend dem `ConnectionController` den Parser für die Antwort zu geben.

Jede Suchmaschine hat eine Implementierung der Spezifikation `AbstractBuilder`, welche durch `URLAbstractBuilder` und `JSONAbstractBuilder` erweitert wird.

7.5.2. QuerySend

Im Modul `QuerySend` befinden sich die `ConnectionController`, die für das Versenden der Anfrage zuständig sind. Sie werden mithilfe von `QueryBuild` konfiguriert und erhalten den Parser (`ResponseParse`). Vor dem Weiterreichen an den `TaskController`, wird durch den Parser noch von NSData zu `SearchResult` umgewandelt.

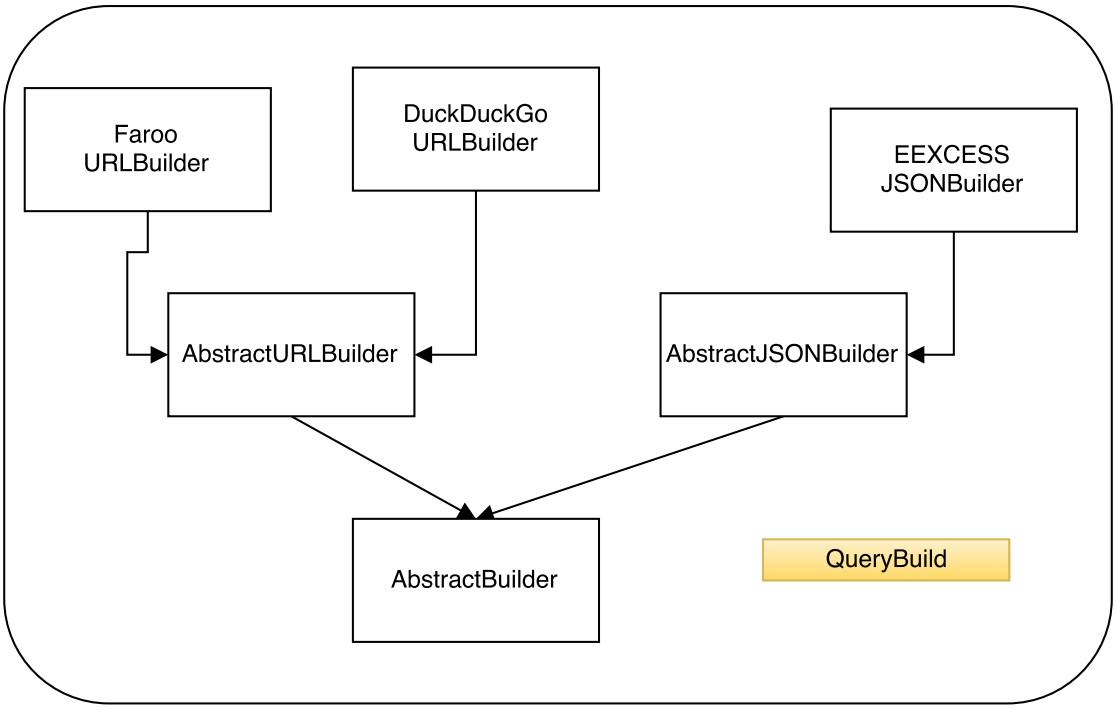

Abbildung 7.5.: Aufbau des Moduls `QueryBuild`

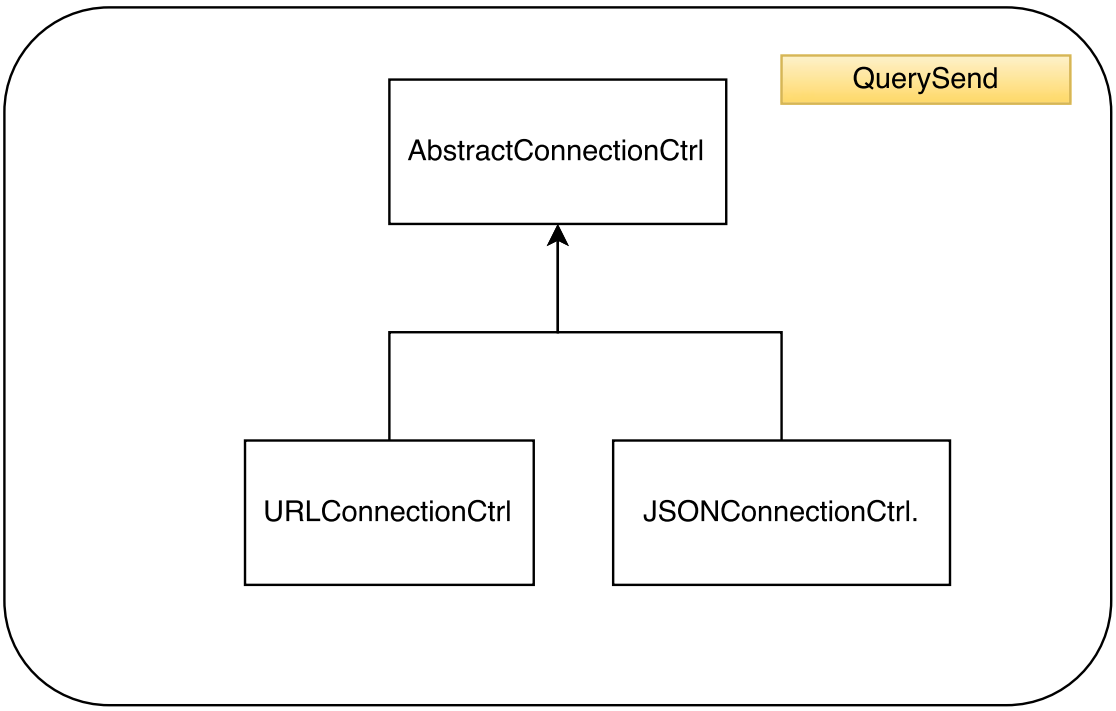

Abbildung 7.6.: Aufbau des Moduls `QuerySend`

7.5.3. ResponseParse

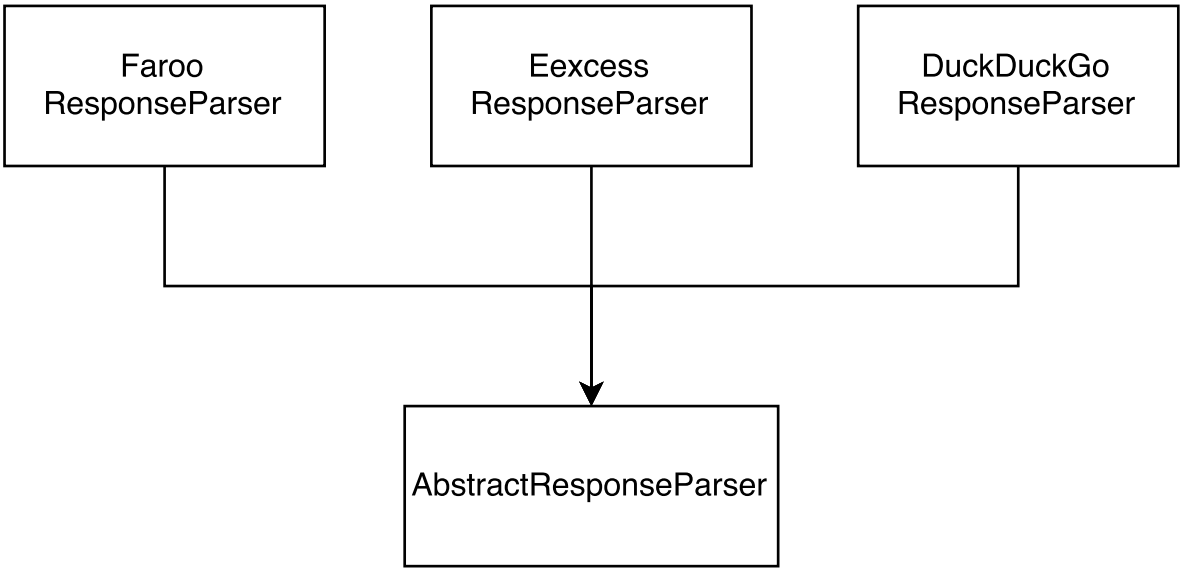

Abbildung 7.7.: Aufbau des Moduls ResponseParse

Die ResponeParser sind zuständig für das Parsen der Antworten der Suchmaschinen in ein allgemeines Format. Dabei wird in der aktuellen Version im `AbstractResponseParser` zwischen drei spezifischen Parsern unterschieden.

- `AbstractResponseParser` spezifiziert die untergeordneten Parser
  - `FarooResponseParser` verarbeitet die Antworten von Faroo
  - `DuckDuckGoResponseParser` verarbeitet die Antworten von DuckDuckGo
  - `EexcessResponseParser` verarbeitet die Antworten von EEXCESS

7.6. SearchResults

`SearchResults` ist ein Container, der für ein Suchwort dessen Suchergebnisse speichert. Der Container ist in der Lage, bei fehlenden Suchergebnissen Suchanfragen abzuschicken und informiert mit dem Observer, ob neue Suchergebnisse von der Suchmaschine zurückgegeben wurden. Der Observer wird im `PopViewController` registriert und ist somit in der Lage auf neue Suchergebnisse zu reagieren.

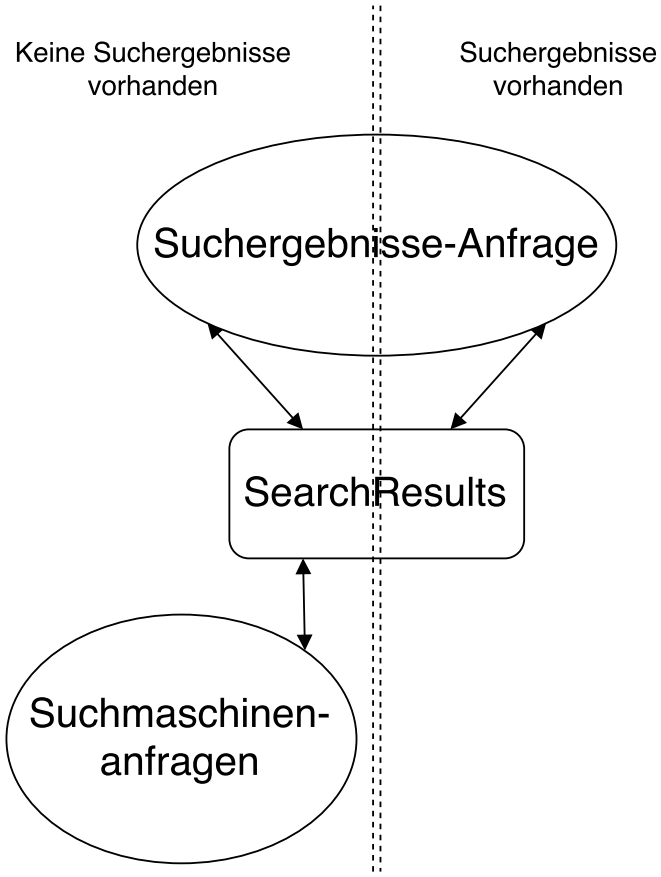

Abbildung 7.8.: Ablaufmöglichkeiten im `SearchResults`

7.7. Ranking

Das Ranking dient der Sortierung der Suchergebnisse nach den Vorlieben des Nutzers. Das Sortieren der Zusatzinformationen wird durchgeführt, bevor die Ergebnisse dem Nutzer präsentiert werden. Das bedeutet, das Sortieren wird in der Klasse `SearchResults` initialisiert und gestartet, nachdem die Zusatzinformationen aus dem Internet heruntergeladen wurden.

Die zur Zeit drei Persistency-Klassen `PersistenceController`, `RankingDataObject` und `RankingDataObjectPersistency` werden im aktuellen Stand des Projektes noch nicht verwendet. Im weiteren Verlauf sollen die Klassen für das Speichern von nutzerspezifischen Informationen verwendet werden, um so die Suchergebnisse noch besser auf die Interessen des Nutzers zuschneiden zu können.

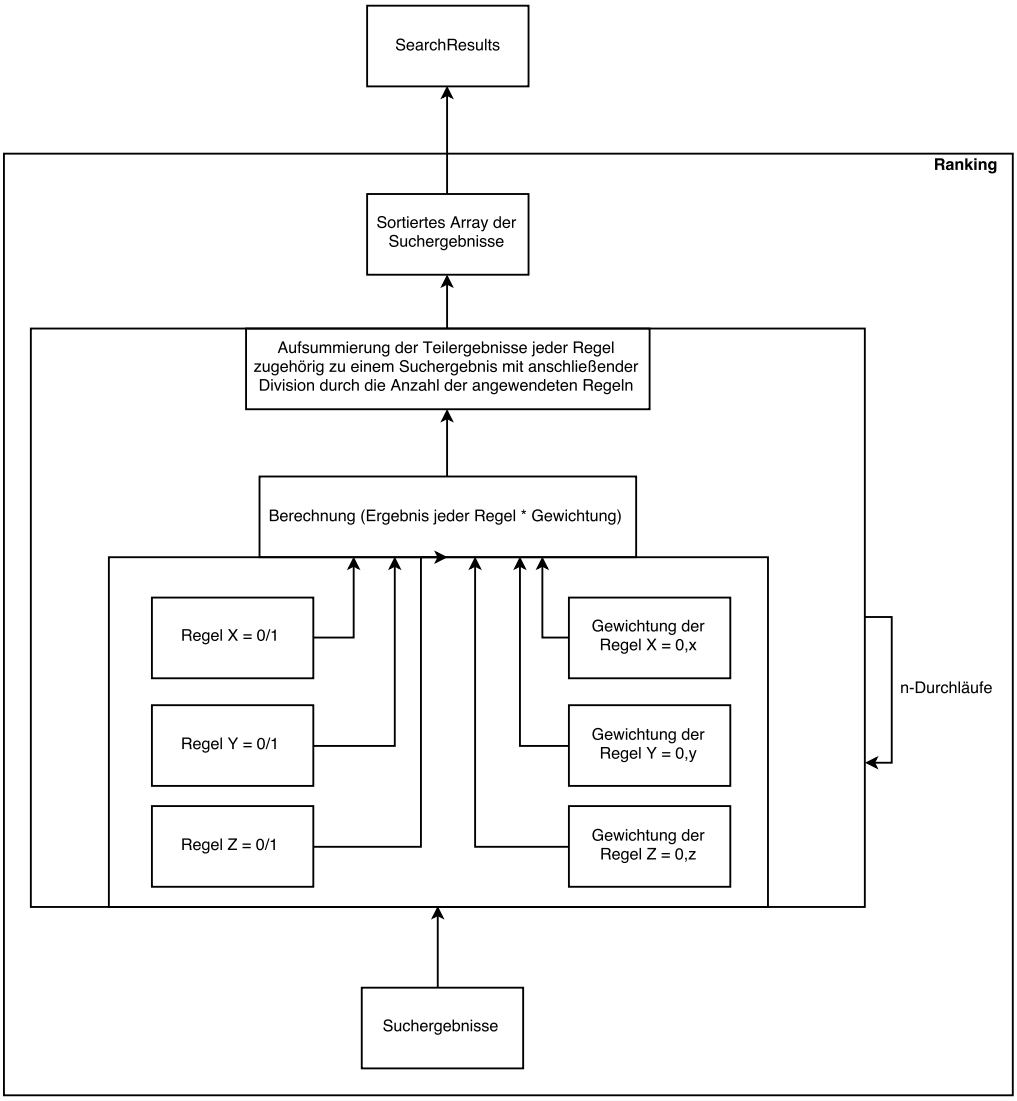

Abbildung 7.9.: Ablauf des Rankings

Die Funktionalität des Rankings wird mit Hilfe von aktuell drei verschiedenen Regeln realisiert:

- Language
- Mendeley
- MediaType

Jede dieser Regeln besitzt einen Faktor für die Gewichtung. Dieser legt fest, wie wichtig eine Regel ist. Für jeden Datensatz werden die oben genannten Regeln mit den entsprechenden Eingangsparametern erzeugt. Im weiteren Fortschritt des Projekts soll das Ranking dahingehend erweitert werden, dass nicht für jeden Datensatz alle Regeln erzeugt werden, sondern nur diejenigen, die für den jeweiligen Datensatz notwendig sind. So wird zum Beispiel für einen Datensatz, der nicht von der Suchmaschine Mendeley kommt, auch nicht die Regel „Mendeley“ erzeugt.

Die erstellten Regeln pro Datensatz werden in einem Hilfsarray zwischengespeichert (`SearchRules`). Der Hauptteil des Rankings beginnt erst mit der Berechnung. Trifft eine Regel zu, also wenn der erwartete Wert der jeweiligen Regel dem tatsächlichen Wert des Suchergebnisses entspricht, so wird eine 1 (wahr) von der Regel zurückgeliefert. Trifft eine Regel nicht zu, so wird eine 0 (nicht wahr) zurückgeliefert. Die Rückgabewerte werden mit Hilfe des Enums `RuleMatch` realisiert. Dieser Wert wird mit der Gewichtung der jeweiligen Regel multipliziert. Im Anschluss werden alle Ergebnisse aller Regeln, die zu einem Datensatz gehören, aufsummiert und durch die Anzahl der verwendeten Regeln dividiert. Dieser errechnete Wert gibt im Verhältnis wieder, in wie weit das Suchergebnis für den Nutzer interessant sein könnte. Nach dem Endergebnis werden alle Suchergebnisse zugehörig zu einem SEARCH-Link absteigend nach diesem Wert sortiert und im Anschluss wieder dem Aufrufer zurückgegeben, der sie dann dem Nutzer präsentiert.

7.8. SettingsModel

Im `SettingsModel` werden die verschiedenen Einstellungen für den Browser gespeichert. Der Nutzer kann diese über das Einstellungsmenü des Geräts ändern.

7.8.1. Einstellungen

Die Einstellungen sind in drei verschiedene Kategorien eingeteilt.

- Suchmaschinen
- Nutzerprofil
- Browsereinstellungen

7.8.1.1. Suchmaschinen

Hier kann der Nutzer die verschiedenen Suchmaschinen einstellen, die für die SEARCH-Tags verwendet werden sollen. Dabei kann entweder jede Suchmaschine einzeln gewählt werden, oder den Empfehlungen des Autors gefolgt werden.

Zur Auswahl stehen:

- DuckDuckGo
- Eexcess
- Faroo

Sobald eine Suchmaschine gewählt wurde, werden die Empfehlungen des Autors ignoriert.

7.8.1.2. Nutzerprofil

Hier können die Nutzerdaten angegeben werden. Davon wird bisher nur die Sprache verwendet.

Einstellungsmöglichkeiten:

- Name
- Alter
- Stadt
- Land
- Sprache

7.8.1.3. Browsereinstellungen

Hier kann die Startseite des Browser festgelegt werden. Die hier gesetzte Seite wird noch nicht im Browser als Startseite übernommen.

Abbildungsverzeichnis