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 0.4.1

\SECH, \SEARCH, ...

Codeelemente bitte durch \lstinlinel…l einbinden

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. Kapitel

\SECH

Im Folgenden werden die Use-Case-Diagramme des Sech-Browsers übersichtlich vorgestellt. Diese veranschaulicht alle verschiedenen Funktionalitäten, die der Benutzer tätigen kann. Diese veranschaulichen?

4.1. Menüführung des Browsers Teil 1

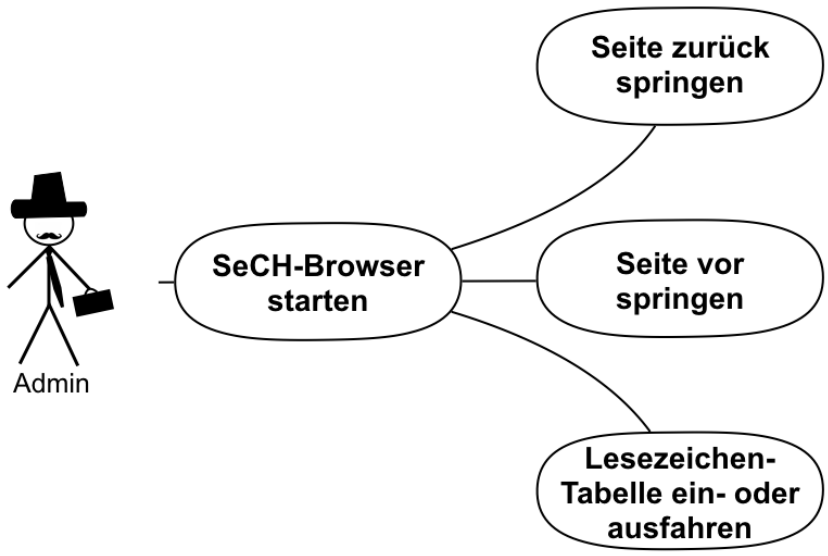

Abbildung 4.1.: Use–Case–Diagramm — Menüführung Teil 1

Dieses Use-Case Diagramm zeigt die wesentlichen Funktionen des Browsers an. Der Nutzer kann auf einer Homepage eine Seite zurück- und vorspringen. Weiterhin hat er die Möglichkeit die Lesezeichen-Tabelle auszufahren und im Anschluss diese wieder einzufahren.

4.2. Lesezeichenverwaltung des Browsers

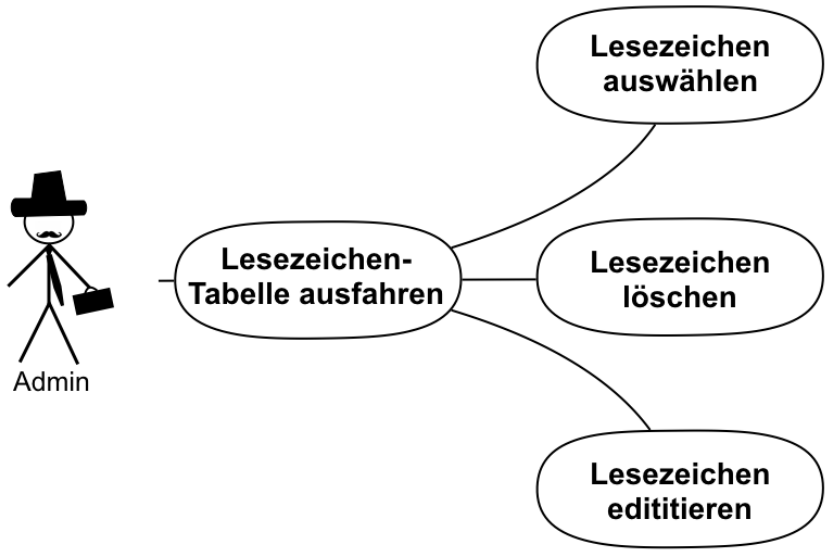

Abbildung 4.2.: Use–Case–Diagramm — Lesezeichenverwaltung

Ist die Lesezeichen-Tabelle ausgefahren, so kann der Benutzer ein persönlich angelegtes Lesezeichen anklicken und es wird die von ihm gewünschte Seite geladen. Das Löschen und das Editieren, ausgewählter Lesezeichen, ist ebenfalls in der Tabelle verwirklichbar.
Kommas???

4.3. Menüführung des Browsers Teil 2

In diesem Use-Case Diagramm werden weitere Navigationselemente des Browsers vorgestellt. Der Nutzer kann eigene Lesezeichen hinzufügen, die von ihm definierte Startseite laden, eine URL in die Address-Bar eingeben und diese dann laden, die aktuelle Seite neu laden, die Search-Tabelle aus- und einzufahren und das Fenster für Optionen öffnen.
\SEARCH einfahren?

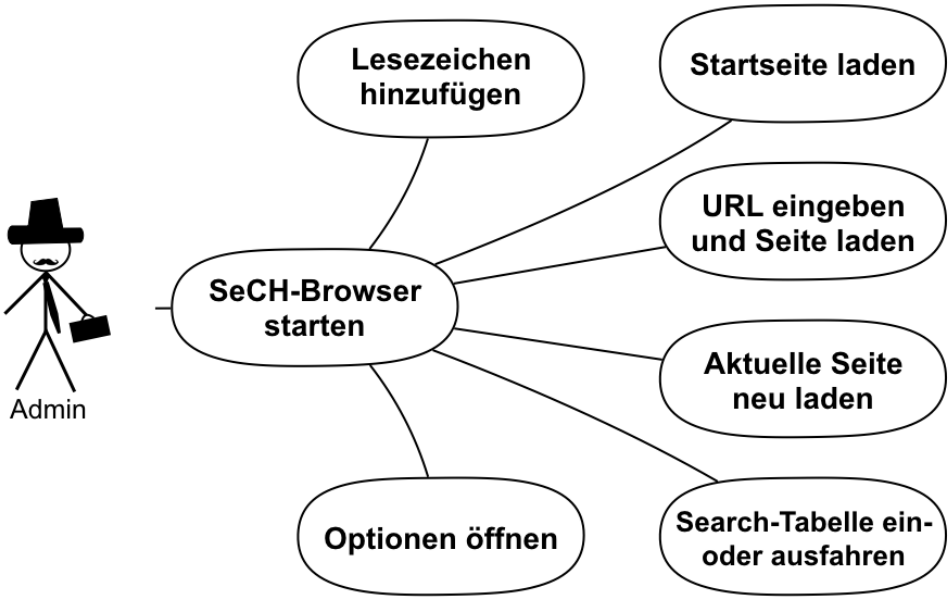

Abbildung 4.3.: Use–Case–Diagramm — Menüführung Teil 2

4.4. Search–Tagverwaltung des Browsers

Der Nutzer kann durch zwei verschiedene Wege Suchergebnisse für das gewünschte Search-Tag anzeigen lassen. Zunächst muss der Nutzer durch Eingabe und Bestätigung einer URL die Seite aufrufen. Erste Variante ist das gewünschte Search-Tag, welches hervorgehoben wird, anzuklicken. Zweite Variante ist durch Ausfahren der Search-Tabelle das gewünschte Search-Tag auszuwählen. Beide Wege führen zum selben Ergebnis, und zwar die Anzeige des ersten Suchergebnisses des jeweiligen Search-Tags. Zusätzlich kann der Nutzer alle weitere Suchergebnisse zu einem Search-Tag anzeigen lassen.

Die erste …

Die zweite …

4.5. Browser-Einstellungen

Befindet sich der Nutzer in den Browser-Einstellungen, so hat er die Möglichkeit bestimmte Suchmaschinen, welche für die Suchergebnisse der Search-Tags verwendet werden, zu aktivieren oder diese zu deaktivieren. Im Weiteren kann der Benutzer sein persönliches

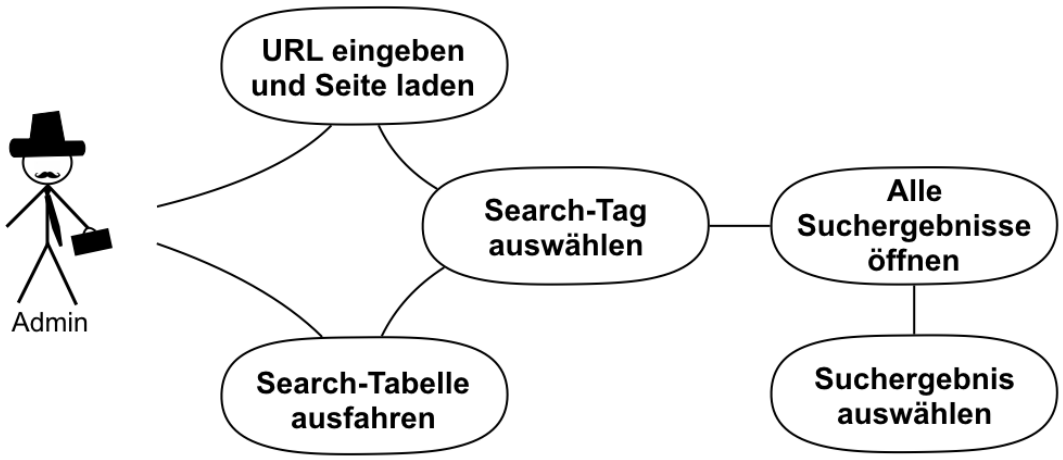

Abbildung 4.4.: Use–Case–Diagramm — Search–Tagverwaltung

Satzbau …

Nutzerprofil verwalten und seine Startseite für die Applikation festlegen, welche direkt nach dem Öffnen der Anwendung als erste Seite geladen wird.

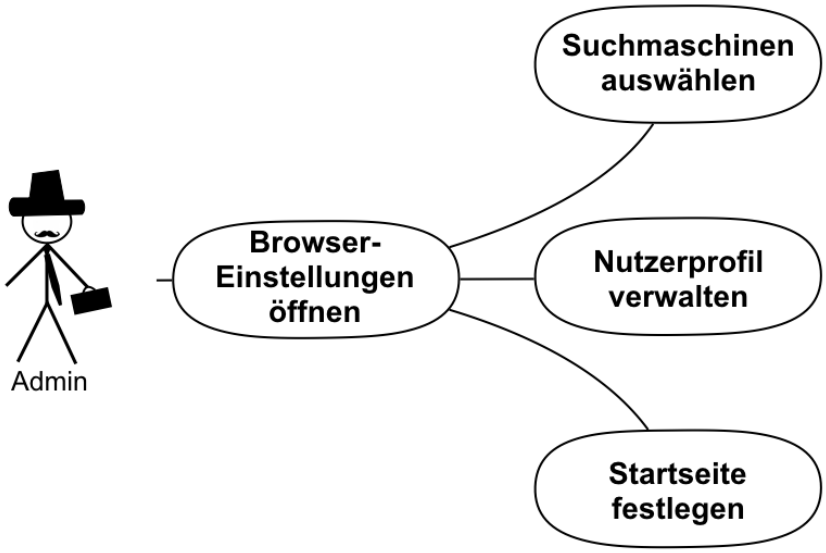

Abbildung 4.5.: Use–Case–Diagramm — Browser–Einstellungen

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 UIs gegeben, der Verlauf zur Anzeige eines Search-Links erklärt, sowie die Packagebedeutungen innerhalb des UIs erläutert. \SEARCH

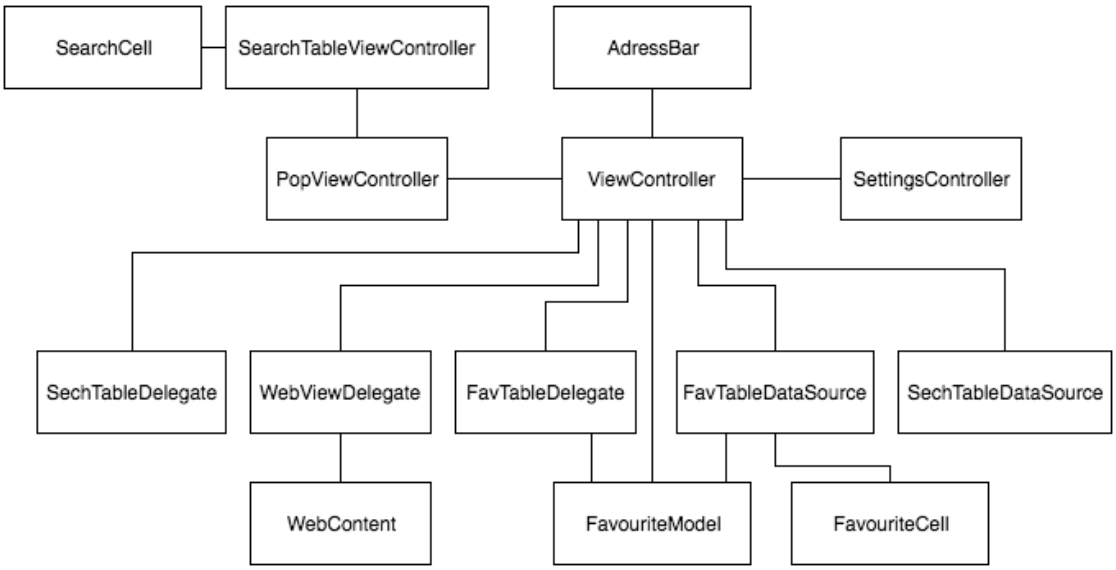

Abbildung 6.1.: Klassenabhängigkeiten im UI

Typo

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

6.1. ViewController

Der ViewController dient zur Steuerung der Funktionen welche direkt mit dem Interface zusammen gehören. Dazu gehören IBActions welche von Buttons oder anderen Interkationsflächen angestoßen werden können, sowie Funktionen welche direkt mit diesen zusammenhängen, Beziehungsweise aus diesen resultieren. Der ViewController ist die zweite Schicht nach dem UI. Im Folgenden werden die einzelnen Viewcontroller mit ihren jeweiligen Funktionen beschrieben. Der Viewcontroller ist der Hauptviewcontroller für ViewController?
die normale Browseransicht mit der Adresszeile und dem WkWebView. Er verwaltet die
Typo Kopzeile mit Adresszeile sowie den Buttons für Vorwärts, Zurück, Lesezeichen, Lesezeichen hinzufügen, Reload, Sech-Tabelle sowie die Settings. Die datei Main.js welche aus

Typo

??

dem Viewcontroller heraus aufgerufen wird, markiert alle gefundenen Search-Tags. Außerdem wird im Falle eines Klicks die Position sowie die id des Search-Tags übermittelt. Der PopViewController ist zuständig für die Verwaltung eines neuen PopViews, beim klicken auf einen Searchtag. Er verwaltet die Anzeige des ausgewählten Searchlinks in einem neuen WkWebView sowie den Button für die Auswahl der verschiedenen Suchergebnisse zu einem Searchtag. Der SearchTableViewController verwaltet die einzellnen Suchergebnisse zu einem Searchtag. Alle Ergebnisse werden sortiert in einer Tabelle mit Bild, Title und Link angezeigt. Der Settingscontroller is für die verwaltung der Browsereinstellungen zuständig.

Schreibweise ...

6.1.1. ViewController

6.1.2. SearchTableViewController

6.1.3. PopViewController

6.1.4. SettingsController

6.1.5. main.js

6.2. Delegate

Schreibweise

Die Delegates im SechBrowser dienen zur Bereitstellung einer einhaltlichen 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 SearchExtraction angestoßen. Die readHead.js enthält den Javascript Code zur Auslesen des Headbereiches einer HTML-Seit. 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 enhält die Klasse die Funktionen, um Lesezeichen bearbeiten und löschen zu können. Der SechTableDelegate enthält die Routine zum Darstellen von Sechlinks aus der SechTabelle.

Absätze?

6.2.1. AppDelegate

6.2.2. WebViewDelegate

6.2.3. readHead.js

6.2.4. FavTableDelegate

6.2.5. SechTableDelegate

6.3. DataSource

Die DataSources im SechBrowser dienen zur Bereistellung der Daten in den Tabellen. Von den einzelnen Klassen werden jeweils Funktionen zum befüllen der einzelnen Zellen aufgerufen. In der Klasse FavTableDataSource werden alle gespeicherten Lesezeichen geladen und danach in die einzelnen Zellen der Tabelle geladen. Im SechTableDataSource werden alle SechTags die auf der Seite gefunden wurden in den Zellen der SechTabelle dargestellt. Zusätzliche befinden sich noch Funktionen zum Erweitern der sechTags in der Klasse.

6.3.1. FavTableDataSource

6.3.2. SechTableDataSource

6.4. Components

In den Components werden die spezifischen Zellen für die Lesezeichen-Tabelle und Search-Tag-Tabelle modelliert, die in den jeweiligen Delegates verwendet werden. Weiterhin ist in Components die Validierung für die AddressBar definiert, welche die Eingaben in der Suchleiste überprüft und verwaltet.

6.5. Persistence

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

6.5.1. FavoritesModel

6.6. WebContent

Der WebContent stellt die Schnittstelle zwischen dem UI 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 geteilt.
Aus dem WebContet wird in der SearchExtraction ein SearchModel erzeugt.

7. Programmlogik

7.1. TaskCtrl

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 SearchModel-Objekten zusammengebaut und in einem SearchModels-Objekt gesammelt.
Für den Ablauf der Analyse und Erzeugung ist der SearchManager zuständig. Die Methoden für die Analyse des HTML-Codes stellt die RegexForSEARCH zur Verfügung.

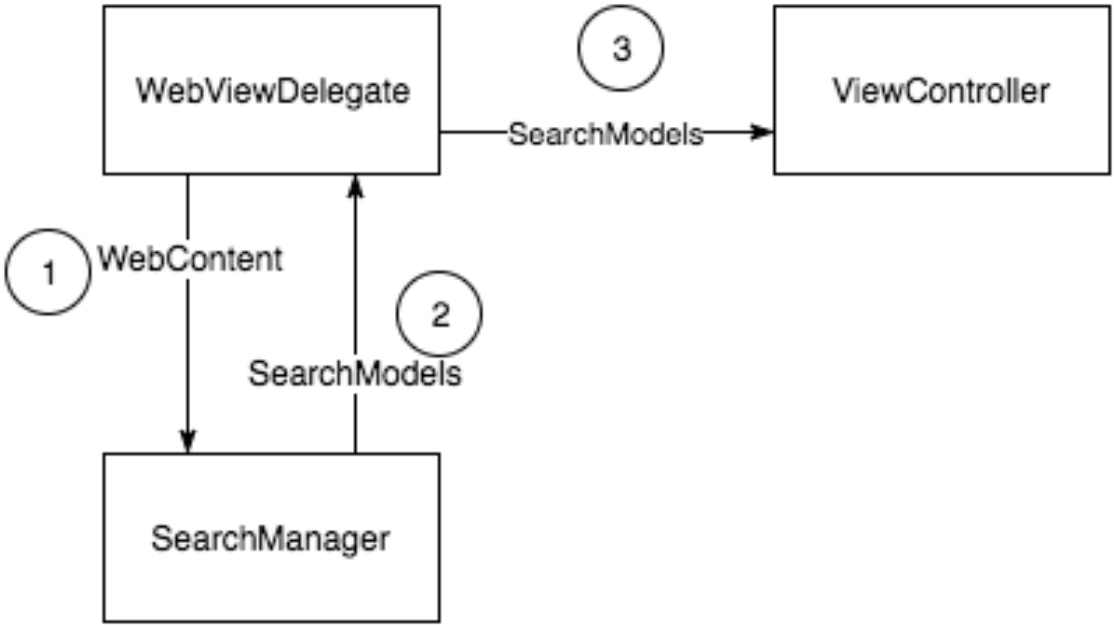

Abbildung 7.1.: Umwandeln eines WebContent-Objekts in ein SearchModels-Objekt

Die obige Darstellung beschreibt den Ablauf und die Übergabestellen der Schnittstellen WebContent und SearchModels. Der WebContent, erzeugt vom WebViewDelegate, wird in Schritt 1 dem SearchManager übergeben. Der SearchManager erzeugt aus dem WebContent die SearchModels und gibt diese in Schritt 2 an den WebViewDelegate zurück. Der WebViewDelegate übergibt dann das SearchModels-Objekt an den ViewController, wo es zur Verfügung für die anderen Packages steht.

7.2.1. SearchManager

7.2.2. RegexForSEARCH

7.3. SEARCHModels

Das SearchModels stellt die Schnittstelle zwischen der SearchExtraction und dem TaskController dar. Von der SearchExtraction erzeugt beinhaltet ein SearchModels-Objekt eine Sammlung von SearchModel-Objekten erzeugt aus einer Website. In einem SearchModel-Objekt werden Search-Tag-Links mit der zugehörigen Search-Tag-Section/-Head und 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 wir der Index gesetzt.

7.4. QueryCreation

7.4.1. QueryCreationCtrl

7.5. SearchQuerys

7.6. QueryResolution

7.6.1. JSONData

7.6.2. QueryBuild

7.6.2.1. AbstractBuilder

7.6.2.2. AbstractJSONBuilder

7.6.2.3. AbstractURLBuilder

7.6.2.4. EexcessJSONBuilder

7.6.2.5. FarooURLBuilder

7.6.2.6. DuckDuckGoURLBuilder

7.6.2.7. EEXCESSOrigin

7.6.3. QuerySend

7.6.3.1. AbstractConnectionCtrl

7.6.3.2. JsonConnectionCtrl

7.6.3.3. URLConnectionCtrl

7.6.4. ResponseParse

7.6.4.1. FarooResponseParser

7.6.4.2. DuckDuckGoResponseParser

7.6.4.3. EexcessResponseParser

7.6.4.4. AbstractResponseParser

7.7. SearchResults

7.8. Ranking

7.8.1. Rules

7.8.2. SearchRules

7.8.3. RankingPersistence

7.8.3.1. PersistenceController

7.8.3.2. RankingDataObject

7.8.3.3. RankingDataObjectPersistency

Das Ranking dient der Sortierung der Suchergebnisse nach den Interessen des Nutzers. Das Sortieren der Zusatzinformationen wird durchgeführt, bevor die Ergebnisse dem Nutzer präsentiert werden. D.h. es wird in der Klasse „SearchResults“ initialisiert und gestartet, nachdem diese aus dem Internet heruntergeladen wurden.

Die Funktionalität des Rankings wird mit Hilfe von verschiedenen Regeln realisiert. Aktuell existieren drei Regeln: „Language“, „Mendeley“ und „MediaType“. Jede Regel besitzt einen Faktor für die Gewichtung. Dieser legt fest, wie wichtig eine Regel ist. Für jeden Datensatz werden die o.g. 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 z.B. 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, d.h. der erwartete Wert der jeweiligen Regel entspricht dem tatsächlichen Wert des Suchergebnisses, so wird eine 1 von der Regel zurückgeliefert. Trifft eine Regel nicht zu, so wird eine 0 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 prozentual wieder, in wie weit das Suchergebnis für den Nutzer interessant sein könnte. Nach dem Endergebnis werden alle Suchergebnisse zugehörig zu einem Search-Link der Größe nach sortiert und im Anschluss wieder dem Aufrufer zurückgegeben, der sie dann dem Nutzer präsentiert.

Die Klassen „PersistenceController“, „RankingDataObject“ und „RankingDataObjectPersistency“ werden bei dem aktuellen Stand des Projektes noch nicht verwendet. Im weiteren Verlauf werden die Klassen zum Speichern von nutzerspezifischen Informationen verwendet, um so die Suchergebnisse noch besser auf die Interessen des Nutzers zuschneiden zu können.

7.9. SettingsModel

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

Einstellungen Die Einstellungen sind in drei verschiedene Kategorien eingeteilt.

- Suchmaschinen
- Nutzerprofil
- Browsereinstellungen

Suchmaschinen Hier kann der Nutzer die verschiedenen Suchmaschinen einstellen, die für die SearchTags 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.

Nutzerprofil Hier können die Nutzerdaten angegeben werden. Davon wird bisher nur die Sprach verwendet. Einstellungsmöglichkeiten:

- Name
- Alter
- Stadt
- Land
- Sprache

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

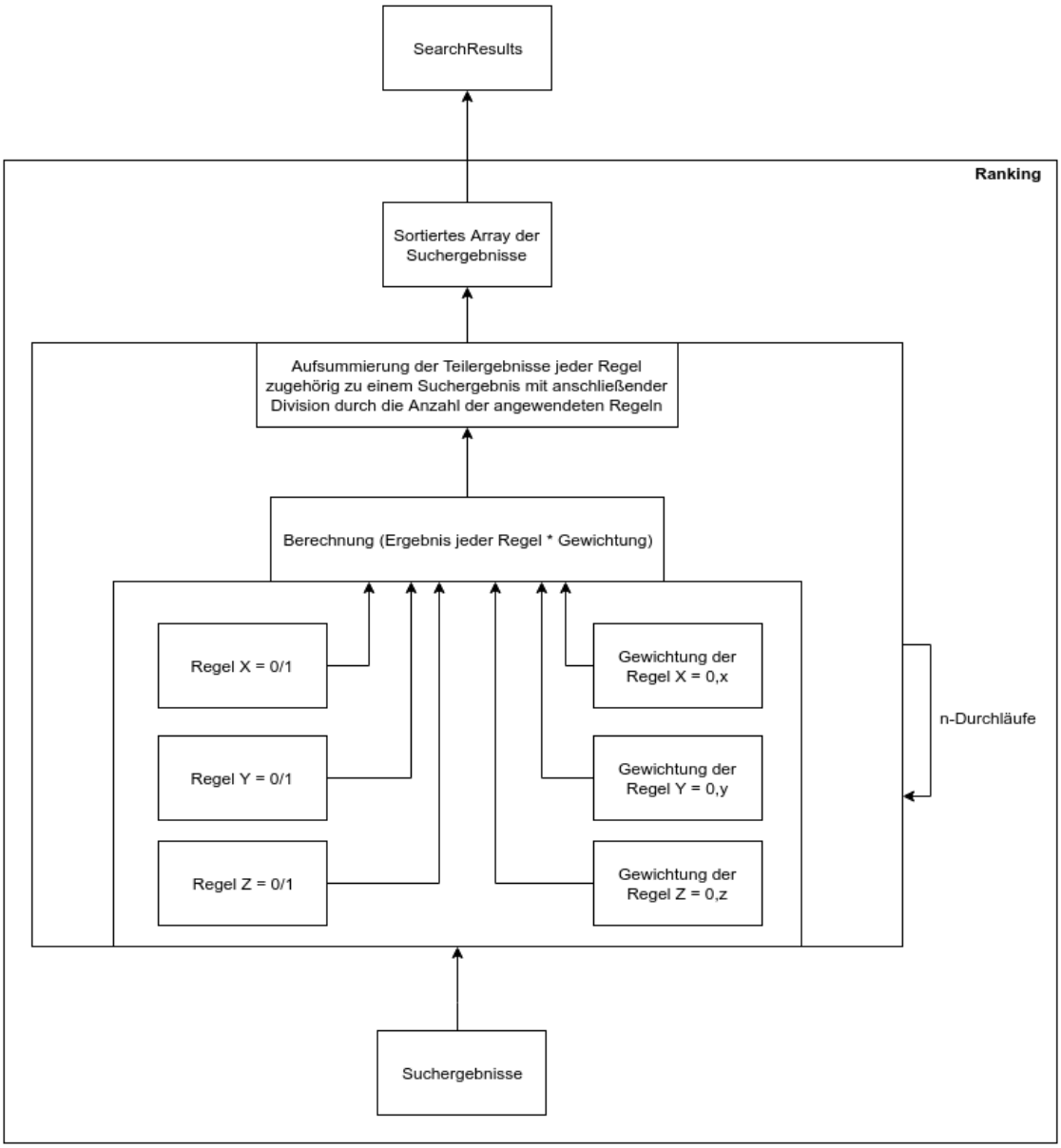

Abbildung 7.2.: Ablauf des Rankings

Position

Abbildungsverzeichnis

Tabellenverzeichnis