

University of Applied Sciences

SEARCH-Tag EXCESS SECH-Browser

Programmier Leitfaden Spezifikation, Konstruktion, Testfälle

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

Version 0.3

Inhaltsverzeichnis

I. Einleitung	1
1. Über dieses Dokument	2
1.1. Teams	2
1.1.1. Team:Content	2
1.1.2. Team:UI	3
1.2. Inhalt	4
2. SECH-Browser	5
2.1. Projektziel	5
2.1.1. Anwendungsszenarien	6
2.1.1.1. Fremdenverkehr	6
2.2. Aufgabenstellung Wintersemester 2015/16	6
3. Einführung in die Programmierung eines EEXCESS-Clients	8
3.1. Grundlegendes	8
3.2. Informationsanfrage	8
3.2.1. /recommend-Anfrage im SECH-Browser Projekt	9
3.3. Zeitplanung	10
II. Spezifikation	11
4. SEARCh-Tag	12
4.1. Allgemeines	12
4.2. SEARCh-Head	12
4.3. SEARCh-Link	13
4.4. SEARCh-Section	13
4.5. SEARCh-Tag	14
5. Bestimmung der Lizenzen der erweiternden Daten	15
6. Recherche der Partner und Provider	16
7. Use Cases	18
7.1. Ranking	18
7.2. Browser	19
7.3. WebKit	21

8. Ranking der Vorschläge	23
III. Konstruktion	26
9. Funktionalitäten	27
9.1. Übersicht	27
9.2. Addressbar	28
9.2.1. Funktionen	28
9.2.2. Implementierung	28
9.3. Lesezeichen	30
9.3.1. Lesezeichen hinzufügen	30
9.3.2. Lesezeichen bearbeiten	30
9.3.3. Lesezeichen anzeigen	31
9.3.4. Lesezeichen löschen	31
10. Einstellungsmöglichkeiten	33
11. WkWebKit JavaScript	34
12. Websiteanalyse und Sechobjekterstellung	36
12.1. Vorbereitung	36
12.2. Analyse des SEARCh-Bodys	36
12.3. Erstellung des Sechobjects	38
13. Server Anfragen	39
13.1. Beschreibung	39
13.2. Aufbau des Prototypen	39
13.3. Netzwerkverbindungen	39
13.4. Verarbeiten/Aufbauen der Anfragen	40
13.4.1. Anfrage	40
13.4.2. DetailAnfrage	41
13.5. JSON Objekt	42
13.6. Prototyp UI	42
14. Klassen Implementierung	44
15. Ranking	45
15.1. Sequenzdiagramm	45
15.2. Klassendiagramm	46
15.3. Abruf der Daten vom Server Sequenzdiagramm	46
15.4. Abruf der Daten vom Server Klassendiagramm	47
16. Entwicklung des Layouts des Browsers	49
16.1. Übersicht	49

16.2. Animation der TableView	51
16.3. Constraints setzen	53
16.4. PopOver Funktion	55
IV. Test	57
17. Testen der Applikation	58
17.1. Einleitung	58
17.2. Testfälle	58
17.3. Auswertung der Testfälle	65

Teil I.

Einleitung

1. Über dieses Dokument

Dieses Dokument dient als Container für die im Laufe der Arbeiten an dem SECH-Projekt entstehenden Unterlagen. Die Quellen für dieses Dokument liegen auf GitHub unter der URL <https://github.com/SECH-Tag-EEXCESS-Browser>.

1.1. Teams

Der Studiengang Mobile Computing der Hochschule Hof hat sich dazu in zwei Teams aufgeteilt:

1.1.1. Team:Content

Teamleiter: Gottfried von Recum

Teammitglieder: Alexander Pöhlmann, Tim Pohrer, Lothar Mödl, Burak Erol

Das Team:Content kümmert sich um die Erstellung und Verwaltung des darzustellenden Inhaltes. Es generiert aus den SEAR^CH-Tags, die das Team:UI aus der besuchten Webseite ausliest und ihm zur Verfügung stellt die Suchanfrage an den EEXCESS Server und interpretiert seine Antwort. Es übernimmt alle Aufgaben zwischen den beiden Aufgabengebieten des Teams:UI.

Im Einzelnen sind das:

- System-/Softwarearchitektur planen - (Recum, Mairhörmann)
- Entwicklung der SEAR^CH-Tags - (Recum, Mairhörmann)
- Vorbereitung des Layouts - (Recum, Mairhörmann)
- Entwicklung Suchanfragenarchitektur - (Recum)
- Umsetzung Systemarchitektur - (Pöhlmann)
- Umsetzung Suchanfragenarchitektur - (Pöhlmann)
- Recherche Preferences - (Pohrer)
- Umsetzung Preferences - (Pohrer)
- Recherche Provider - (Erol)

- Recherche Querry- und Responseformat - (Mödl)
- Ranking entwickeln - (Recum, Mödl)
- Rankingalgorithmus implemetieren - (Mödl, Pohrer)
- Ranking visualisieren - (Erol)
- Zusammenführung Programmteile - (Mödl)
- Eingliederung Ranking - (Mödl)
- Bugfixing - (Mödl, Pöhlmann)
- Dokumentationsteile zusammenführen - (Recum)

1.1.2. Team:UI

Teamleiter: Brian Mairhörmann

Teammitglieder: Andreas Netsch, Andreas Ziemer, Patrick Büttner, Philipp Winterholer

Das Team:UI ist für die benutzergerechte Bedienung und Darstellung der SEARCh-Tags zuständig. Das Team analysiert die Websites auf SEARCh-Tags, bringt diese auf Programmebene und übergibt sie zur Abfrage an das Team:Content. Anschließend werden die Antworten des Antwortens des EEXCESS-Servers auf der Benutzeroberfläche dargestellt.

Es folgt eine Aufzählung der Aufgaben, welche im Team:UI erledigt wurden, und deren bearbeitende Person/en:

- System-/Softwarearchitektur - (Mairhörmann, Recum)
- Entwicklung der SEARCh-Tags - (Mairhörmann, Recum)
- Vorbereitung des Layouts - (Mairhörmann, Recum)
- Implementierung der Adressleiste - (Ziemer)
- Layout und Implementierung des Einstellungsmenüs - (Ziemer)
- Layout und Implementierung der Lesezeichenfunktionen - (Büttner)
- Entwicklung des Layouts des Browser - (Erol, Netsch, Winterholler)
- Einrichten der spezifischen, zum testen der Regex und Websiteanalyse dienenden, und allgemeinen, zum entwickeln der Software dienenden, Testwebsites - (Ziemer)
- Erstellung von spezifischen Testwebsites - (Büttner)
- Implementierung von RegEx-Methoden für HTML-Analyse - (Mairhörmann)
- Implementierung des Auslesens der SearchTags aus HTML-Code - (Mairhörmann)

- Implementierung der TableView zur Anzeige von SEARCh-Tags und PopOver-Funktion - (Netsch, Winterholler)
- Umstellung des Browsers von UIWebView auf WKWebKit - (Netsch, Ziemer)
- Animation der TableView - (Erol)
- Constraints setzen - (Erol, Winterholler)
- Anzeige der Anzahl der SEARCh-Tags - (Winterholler)
- Visualisierung der SEARCh-Tags im Browser - (Mairhörmann, Netsch)

Das Dokument wird von den bearbeitenden Gruppen parallel zu den Entwicklungsarbeiten weiterentwickelt.

1.2. Inhalt

Aktuell beschreibt das Dokument den Entwicklungsstand zum Wintersemester 2015 und besteht aus folgenden Teilen:

1. Beschreibung des Projektziels.
2. Beschreibung der Aufgabenstellung für das Wintersemester 2015.
3. Einführung in die Programmierung eines EEXCESS Clients.
4. Spezifikation der beiden Teilsysteme.
5. Konstruktion der beiden Teilsysteme.
6. Testfälle und ihre Auswertung

2. SECH–Browser

2.1. Projektziel

Das Web besteht aus einer Vielzahl von Seiten mit unterschiedlichsten Informationen. Betrachtet man diese Seiten genauer so stellt man fest, dass ihr informationstragender Inhalt praktisch immer statisch ist. Er wird zum Zeitpunkt der Erstellung der Web–Seite von einem Autor festgelegt und bleibt bis zum nächsten Update in genau diesem Zustand. Auch wenn die Seiten dynamisch generiert werden, der dafür verwendete Inhalt ist vorher von einem Autor erstellt worden. Dass diese Informationen statisch sind bedeutet auch, dass die Seiten für alle Betrachter weltweit identisch sind.

Ziel des SECH–Browsers ist es, diese statische Präsentation der Informationen aufzuheben und die von einem Autor vorgegebenen Inhalte automatisch um benutzerspezifische Informationen, den *SECH–Annotations*¹, zu ergänzen. Er folgt damit der Idee des „taking the content to the user“² des EEXCESS–Projektes.

Die Auswahl der zusätzlichen Informationen wird dabei von dem SECH–Browser, also dem Client, und nicht dem Server vorgenommen. Personenbezogene Daten verlassen also nur dann den eigenen Rechner, wenn es für die Suche nach den Daten der SECH–Annotations notwendig ist. Somit ist ein Maximum an Datenschutz gegeben.

In der ersten Version basiert die Erzeugung der SECH–Annotations durch den SECH–Browser auf Informationen, die der Seitenautor in einem speziellen Format im Text hinterlegt hat. Diese Informationen, die SEAR^CH–Tags³, verwendet der SECH–Browser dann dazu, benutzerspezifische Anfragen an eine Wissensquelle, beispielsweise dem Privacy–Proxy des EEXCESS–Projektes (<http://eexcess.eu>), zu stellen und mit den Ergebnissen die ursprüngliche Darstellung der WWW–Seite zu ergänzen.

In den weiteren Entwicklungsschritten soll der SECH–Browser dann die für die Erzeugung der SECH–Annotations notwendigen Informationen selbständig aus den Textinhalten ableiten. Dafür können dann beispielsweise Techniken aus den Bereichen des Text–beziehungsweise Opinion–Minings verwendet werden.

¹SECH steht dabei für **S**elf **E**mbeding **C**haracteristic **H**yperlinks.

²<http://eexcess.eu>

³SEAR^CH steht dabei für **S**elf **E**mbeding **A**nnotation and **R**ecommendation based **C**haracteristic **H**yperlinks.

2.1.1. Anwendungsszenarien

Im folgenden soll an Hand von einigen Beispielen dargestellt werden, wie die zusätzlichen Funktionalitäten des SECH-Browsers den Anwender unterstützen können.

2.1.1.1. Fremdenverkehr

Web-Seiten von Fremdenverkehrsverbänden werden von einer Vielzahl von verschiedenen Benutzergruppen besucht:

- Besucher, die im näheren Umkreis leben.
- Besucher von weiter her.
- Besuchern mit unterschiedlichen Muttersprachen.
- Besucher verschiedener Altergruppen.
- ...

Die verschiedenen Benutzergruppen haben in der Regel auch verschiedene Anforderungen an die zur Verfügung gestellten Informationen:

- Ein Besucher, der nicht im näheren Umkreis lebt, benötigt in der Regel eher Informationen über die Hotels des Gebietes als ein Besucher aus dem Umkreis.
- Während Jugendliche sich eher für die Angebote aus dem Bereich der Fun-Sport-Arten interessieren, könnten Senioren eher an Informationen über die kulturellen Angebote der Region interessiert sein.
- Fremdsprachige Benutzer sollten vorrangig Informationen in ihrer Landessprache präsentiert werden.

Liegt ein gutes Design der Web-Seite vor, können die Besucher weitere Teile der für sie relevanten Daten durch entsprechende Verlinkungen auf andere Web-Seiten finden. Aber auch diese zusätzlichen Informationen sind statisch, für alle Besucher der Seite identisch und somit nicht an die Bedürfnisse der einzelnen Besucher angepasst.

Auch in diesem Anwendungsfall wäre es viel sinnvoller, wenn der SECH-Browser an Hand der ihm bekannten benutzerspezifischen Informationen selbstständig erkennt, welche zusätzlichen Informationen für den Besucher interessant sein könnten und diese dann entsprechend aufbereitet darstellen.

2.2. Aufgabenstellung Wintersemester 2015/16

Die im Wintersemester 2015/16 zu entwickelnde Version des SECH-Browsers umfasst folgende Schritte:

1. Erstellung eines iOS–Programms zur Darstellung von Web–Seiten.
2. Erstellung eines iOS/OS X Programms zur Generierung von Anfragen an den Privacy–Proxy des EEXCESS–Projekts und der Darstellung der jeweiligen Antworten.
3. Definition der Syntax und Semantik des SEAR^CH–Tags zur Beschreibung der vom Autor vorgegebenen Informationen zur Erstellung einer SECH–Annotation.
4. Erzeugung der um benutzerorientierte Informationen angereicherten Anfragen für den Privacy–Proxy.
5. Integration der einzelnen Teilsysteme in einen lauffähigen Prototypen.

Für die Software–Bausteine sind entsprechende Spezifikations–, Konstruktions– und Testdokumente zu erstellen. Diese Dokumente orientieren sich an den Inhalten der Vorlesung „Software Engineering II“.

Erweiterungen, wie beispielsweise

1. Rating des Benutzers über die Güte der SECH–Annotations einholen;
2. Verwendung des Ratings um darauf folgende, ähnliche SECH–Annotations zu verbessern;

können in das Projekt einfließen.

3. Einführung in die Programmierung eines EEXCESS–Clients

3.1. Grundlegendes

Die Kommunikation eines Clients mit dem EEXCESS–Server basiert auf dem Austausch von JSON–Objekten. Im Rahmen des SECH–Browser–Projektes werden nur die Dienste des Privacy–Proxy–Service¹ in Anspruch genommen. Die dafür benötigten JSON–Objekte sind in der Online–Dokumentation des EEXCESS–Projektes auf GitHub beschrieben.

3.2. Informationsanfrage

Informationsanfragen an den PP–Server geschehen in der Regel in zwei Schritten.

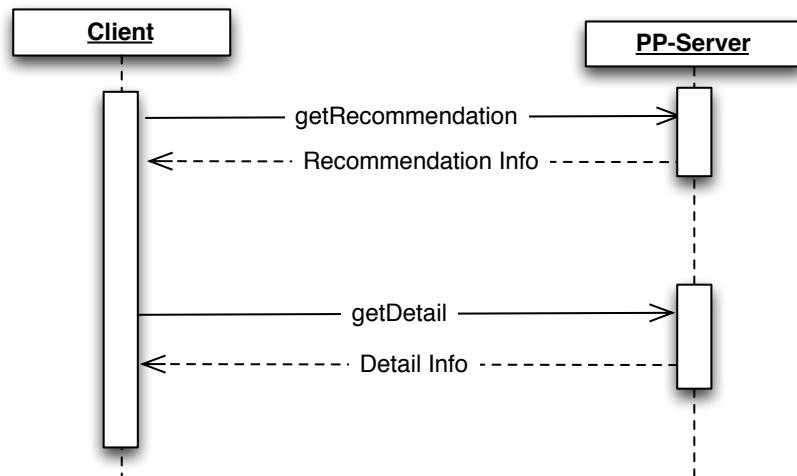


Abbildung 3.1.: Beispiel für Informationsanfrage an den EEXCESS PP–Service

Im ersten Schritt wird eine durch Suchparameter beschriebene /recommend–Anfrage an der Server gestellt. Die Antwort darauf besteht aus einem JSON–Objekt. Es enthält, neben den eigentlichen Ergebnissen der Suchanfrage, auch eine eindeutige ID zur

¹Im Folgenden mit PP–Service abgekürzt.

späteren Referenzierung der Anfrage. Die zurückgelieferten Ergebnisse bestehen ihrerseits wieder aus verschiedenen Elementen, unter anderem einem Titel der das Element näher beschreibt, einer URI die angibt wo das Objekt gespeichert ist und einer eindeutigen ID zur Referenzierung des Objekts. Diese Informationen würden bereits ausreichen, um die einzelnen Ergebnisse der Suchanfrage aus dem Netz zu laden. Da der Benutzer aber selber entscheiden soll, ob die angezeigten Zusatzinformationen für ihn interessant sein könnten, muss er vor der Darstellung eine Auswahl treffen können. Dies ist anhand des Titels in der Regel nur schwer möglich!

In einem zweiten, optionalen Schritt kann daher für jedes Ergebnis einer /recommend–Anfrage eine detaillierte Kurzbeschreibung vom PP–Server abgerufen werden. Der dafür vorgesehene /getDetails–Befehl übergibt die ID der Suchanfrage und die documentBadge–Informationen des ursprünglichen Ergebnisses an den PP–Server und bekommt, falls bei der Datenquelle hinterlegt, eine Kurzbeschreibung des Ergebnisses. An Hand dieser Kurzbeschreibung kann der Anwender dann entscheiden, ob das eigentliche Informations–Objekt über die zugeordnete URI geladen werden soll oder nicht.

3.2.1. /recommend–Anfrage im SECH–Browser Projekt

Ein minimales JSON–Objekt für eine /recommend Anfrage an den PP–Server besteht aus 3 Teilen:

1. Den `origin` Informationen, die den Client näher beschreiben.
2. Dem `loggingLevel`, der angibt ob der PP–Server die Anfragen aufzeichnen darf oder nicht.
3. Den `contextKeywords`, die die eigentlichen Suchbegriffe beinhalten.

Über weitere Parameter können die /recommend–Anfrage noch genauer definiert werden. Im SECH–Browser werden folgende Parameter verwendet:

1. `numResults`, um die Anzahl der vom PP–Server zurückgegebenen Resultate begrenzen zu können.
2. Das `isMainTopic` Attribut der `contextKeywords` Einträge, um bei mehreren Suchparametern eine Priorisierung durchführen zu können.
3. `interests`, um die Anfragen genauer an die Vorlieben des Benutzers anpassen zu können.
4. `languages`, um bevorzugt SECH–Annotations in der Muttersprache des Benutzers zu verwenden.
5. `timeRange`, um die SECH–Annotations zeitlich eingrenzen zu können.

3.3. Zeitplanung

Um alle Aufgaben bestmöglich erledigen zu können wird zunächst eine Zeitplanung erstellt. Diese gilt als Richtschnur und wird wiederkehrend überprüft.

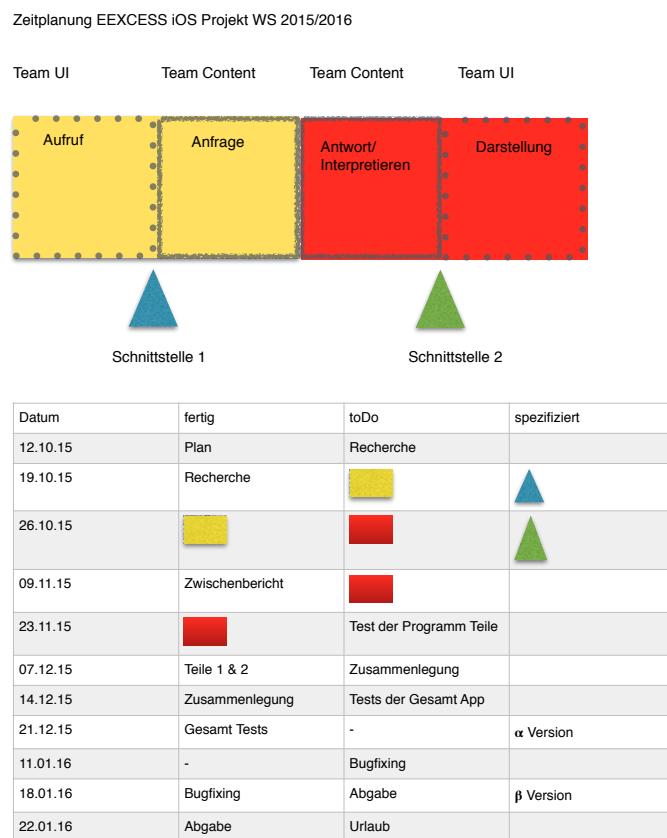


Abbildung 3.2.: Zeitplanung für das SECH-Browser Projekt

Teil II.

Spezifikation

4. SEARCh-Tag

4.1. Allgemeines

Ein SEARCh-Tag gibt dem Ersteller einer HTML5 Seite die Möglichkeit seinen Content dynamisch anzureichern. Dazu kann er einige Attribute definieren. Ein SEARCh-Tag besteht aus bis zu drei Teilen, die an verschiedenen Stellen in den HTML5 Quellcode der Seite eingefügt werden können. Die generelle Form ist

```
<search attribute="">...</search>
```

Die drei Teile sind angelehnt an HTML5:

```
<search-head>
<search-section>
<search-link>
```

Jeder Teil eines SEARCh-Tags hat stets das Attribut `topic="text"` wobei `text` ein frei wählbarer Begriff ist, der zur Contentgenerierung verwendet werden soll.

Die vier optionalen Filter Attribute `type=""`, `mediaType=""`, `provider=""` und `licence=""` dienen der Optimierung der Suchanfrage und anschließenden Antwortauswahl.

4.2. SEARCh-Head

[optional] Der `<search-head>` Teil wird **optional** in den Header der Seite eingefügt um allgemeine Attribute für die Seite als Ganzes definieren zu können.

```
<head>
    <search-head topic="text">
</head>
```

oder

```
<head>
    <search-head topic="Programmierer" type="misc" mediaType="unknown"
                 provider="ZBW" licence="unknown">
</head>
```

4.3. SEARCh-Link

[obligatorisch] Der **obligatorische** `<search-link>` Teil eines SEARCh-Tags umschließt einen Bereich, der mit Inhalt angereichert werden soll. Dieser kann hervorgehoben dargestellt werden.

```
<body>
    <search-link topic="text">...</search-link>
</body>
```

oder

```
<body>
    <search-link topic="Ada" type="person" mediaType="image"
        provider="ZBW" licence="unknown">Ada Lovelace</search-link>
</body>
```

4.4. SEARCh-Section

[optional] Der `<search-section>` Teil wir **optional** im Body des HTML5 Quellcodes platziert und umschließt einen oder mehrere `<search-link>` Teile. Damit kann dem SEARCh-Link Teil ein erweiterter Themenbezug beiseite gestellt werden.

```
<body>
    <search-section topic="text">
        ...
    </search-section>
</body>
```

oder

```
<html>
<head>
    <search-meta topic="Programmierer" type="misc" mediaType="unknown"
        provider="ZBW" licence="unknown">
</head>
<body>
    ...
    <search-section topic="Grossbritannien" type="location"
        mediaType="unknown" provider="ZBW" licence="unknown">
    ...
    <search-link topic="Ada" type="person" mediaType="image"
        provider="ZBW" licence="unknown">Ada Lovelace</search-link>
    ...

```

```
</search-section>  
</body>  
</html>
```

4.5. SEARCh-Tag

Der SEARCh-Tag schließlich besteht aus allen drei Teilen (soweit vorhanden), dabei wird der SEARCh-Link Teil am stärksten gewichtet.

Dabei akkumulieren die `topics` und `types`. Die `mediaTypes`, `provider` und `licences` hingegen ersetzen sich mit zunehmender Priorität in Richtung SEARCh-Link.

5. Bestimmung der Lizenzen der erweiternden Daten

Für einige Benutzer des SECH-Browsers ist es für die weitere Verwendung der erweiterten Daten notwendig zu wissen, welchen Lizenzen diese unterliegen. Hierfür wird die Methode „getLicenceType(licenseURL: String) -> LicenceType“ verwendet. Diese Methode erhält als Eingabeparameter die URL der Lizenz des jeweiligen Objekts als String, welche vom EEXCESS-Server zurückgeliefert wird, und liefert einen Enum-Type zurück. Als Enum-Type sind folgende verfügbar:

- CreativeCommon
- Restricted
- Europeana
- Other

Diese Lizenzen geben an, ob die Daten ohne Weiteres für andere Zwecke verwendet werden dürfen oder irgendwelchen Lizenzbestimmungen unterliegen.

6. Recherche der Partner und Provider

Die Recherche zeigt auf zwei verschiedenen Excel-Dokumenten eine Liste aller Partner und Provider die für die Suchabfragen des Browsers kontaktiert werden. Es wird aufgelistet was die Webseite Inhaltlich spiegelt. Die Dokumente sollen eine Übersicht dieser Seiten für die Suchabfragen zeigen. Die Webseiten unterscheiden sich im Kontext, denn es befinden sich sowohl wissenschaftliche, technische, kulturelle als auch Personen spezifische Inhalte auf diesen verschiedenen Providern.

Partner	Links	Inhalt der Seite
Wikipedia-Local	http://www.wikipedia.at	Suchergebnisse verschiedener Art. Von Personen bis hin zu geschichtlichen Ereignissen.
Deutsche Digitale Bibliothek	https://www.deutsche-digitale-bibliothek.de	Suchergebnisse liefern kulturelle und wissenschaftliche Ergebnisse Deutschlands, wie z.B. über Bücher, Archivalien, Bilder, Skulpturen und Musikstücke.
KIMPortal	https://www.kgportal.bl.ch/startseite	Es finden technisch orientierte Suchabfragen statt.
Mendeley	https://www.mendeley.com	Literaturverwaltungsprogramm, welche das Organisieren, Austauschen und Zitieren von wissenschaftlichen Artikeln und PDF-Dokumenten zeigt. Im weiteren kann man hier auf Statistiken zugreifen.
Europeana	http://www.europeana.eu/portal/	Kulturelle Ergebnisse werden aufgezeigt. Suchabfragen werden mit Fotos dargestellt und viele Bilder werden beschrieben
Wissensmedia	http://www.wissens-server.com	Hier werden verschiedene wissenschaftliche Ergebnisse zurück geliefert.
ZBW	http://www.econbiz.de	Ekonomische Literatur werden dargestellt. Suchabfrage liefert Ergebnisse zu ekonomischen Ergebnissen zurück.
Opensearch	http://www.opensearch.org/Home	Opensearch ist eine Kollektion für einfache Formate für das Teilen der Suchergebnisse.

Tabelle 6.1.: Liste der Partner

Provider	Links	Inhalt der Seite
Europeana	http://www.europeana.eu/portal/	Kulturelle Ergebnisse - Suchabfragen werden fotografisch dargestellt und mehrere Bilder Verfügung eine genaue Beschreibung.
KIMPortal	https://www.openinteractive.ch	Technisch orientierte Suchabfragen werden zurückgeliefert.
ZBW	http://www.econbiz.de	Ekonomische Literatur - Suchabfrage liefert Ergebnisse zu ekonomischen Ergebnissen.
Mendeley	https://www.mendeley.com	Literaturverwaltungsprogramm zum organisieren, austauschen und zitieren von wissenschaftlichen Artikeln und PDF-Dokumenten. Im Weiteren können auf Statistiken zurück gegriffen werden.
Wissenmedia	http://www.wissens-server.com	Wissenschaftliche Ergebnisse werden dargestellt.

Tabelle 6.2.: Liste der Provider

7. Use Cases

7.1. Ranking

Nachdem das System alle Daten vom Server abgefragt hat, werden die Responses gerankt d.h. Suchergebnisse von denen das System denkt, dass diese besser zum Nutzer passen, werden höher gerankt als Suchergebnisse, die wahrscheinlich schlechter zum Nutzer passen. Das Suchergebnis mit dem höchsten Ranking wird dem Nutzer als erstes präsentiert.

Folgendes Use-Case-Diagramm veranschaulicht nochmal das Schema des Rankings:

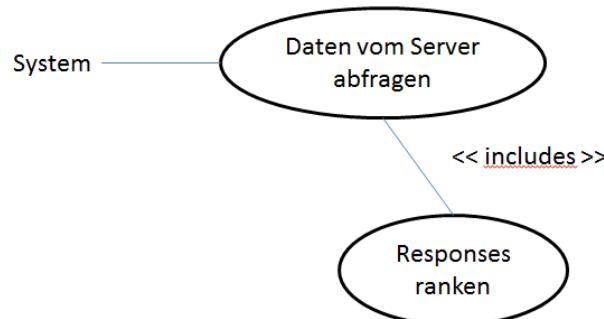


Abbildung 7.1.: Use Case des Rankings

7.2. Browser

Denkbare Use Cases des Browsers sind:

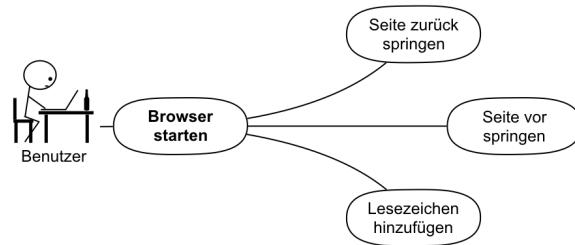


Abbildung 7.2.: Menüführung des Browsers (Navigation)

Dieses Use-Case Diagramm zeigt die wesentlichen Funktionen des Browsers auf einer Homepage eine Seite zurück zu gehen oder eine Seite vor zu wechseln. Im weiteren hat er die Möglichkeit ein eigenes Lesezeichen anzulegen.

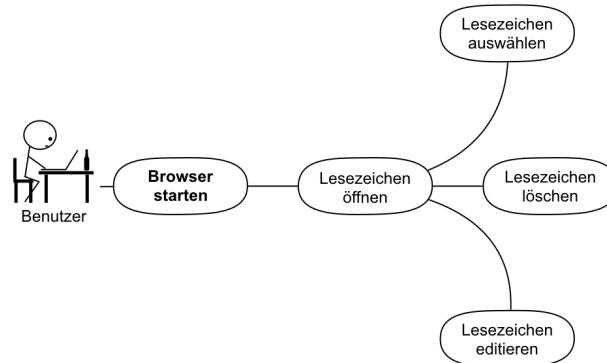


Abbildung 7.3.: Lesezeichenverwaltung des Browsers

In diesem Use-Case Diagramm wird die Lesezeichenverwaltung veranschaulicht. Der Nutzer kann ein angelegtes Lesezeichen auswählen, diese löschen oder editieren.

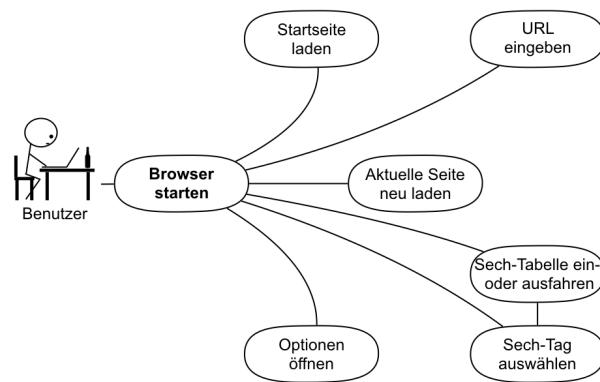


Abbildung 7.4.: URL Navigation und SECH-Tabelle des Browsers

Dieses Use-Case Diagramm zeigt weitere Navigationselemente im Browser. Er kann die von ihm definierte Startseite laden, eine URL in die Suchleiste eingeben und bestätigen oder die aktuelle Seite neu laden und die Optionen öffnen. Des Weiteren kann er die SECH-Tabelle ein- und ausfahren in der sich die SECH-Tags befinden und er eines dieser auswählen kann.

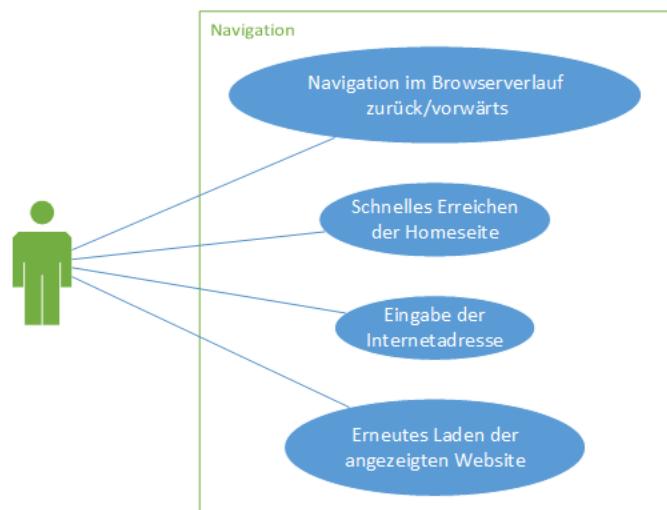


Abbildung 7.5.: Use Case der Browser Navigation

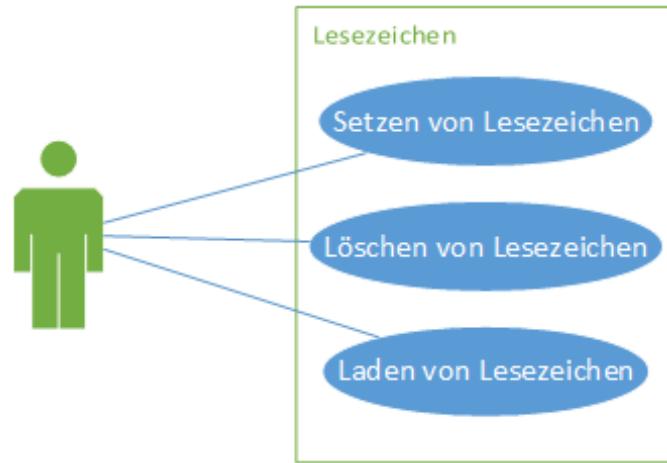


Abbildung 7.6.: Use Case der Browser Lesezeichen



Abbildung 7.7.: Use Case der Browser Einstellungen

7.3. WebKit

Ablauf der Interaktion zwischen WebKit und SEARC^H-Tag:

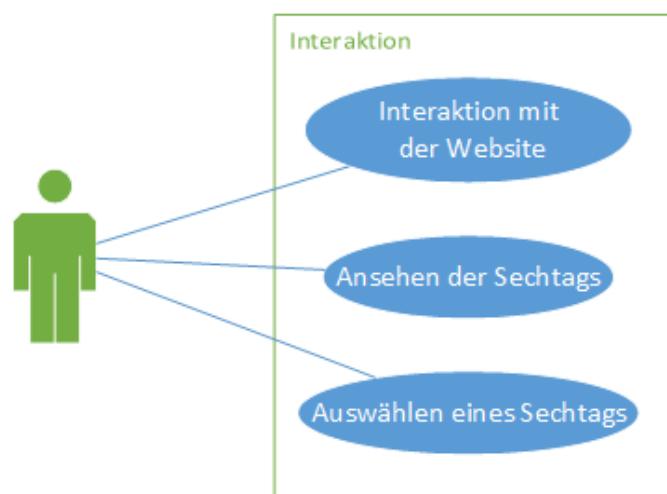


Abbildung 7.8.: Interaktion zwischen Webkit und SEARCh-Tag

8. Ranking der Vorschläge

Die Suchergebnisse für einen Begriff oder für Wortgruppen müssen nach den Interessen des Nutzers sortiert werden. Die Auswahl der Ergebnisse basiert auf verschiedenen Regeln. Diese Regeln sind Funktionen, die überprüfen, ob ein Kriterium erfüllt ist, oder nicht. Bei Vollendung einer Regel wird der Wert 1 oder 0 zurückgeliefert. 1 bedeutet, die Regel trifft zu und ist erfüllt, 0 bedeutet, die Regel wurde nicht erfüllt und trifft nicht zu. Für jeden Datensatz können alle Regeln durchlaufen werden, müssen aber nicht. Außerdem besitzt jede Regel eine Gewichtung, die aussagt, wie sehr diese Regel die Interessen des Nutzers wiederspiegelt. Nachdem alle Regeln abgeprüft wurden, wird der jeweilige Wert mit dem prozentualen Wert der Gewichtung multipliziert. Dies hat zur Folge, dass die negativ bewerteten Regeln eliminiert werden. Nach dieser Rechenoperation folgt eine Zweite. Diese legt fest, dass alle Teilergebnisse aufsummiert werden und im Anschluss durch die gesamte Anzahl aller abgeprüften Regeln dividiert wird. Der daraus resultierende Wert beschreibt als Prozentangabe, in wie fern dieser Datensatz den Nutzer interessieren könnte. Der oben genannte Prozess muss für jeden Datensatz wiederholt werden. Am Ende werden alle Ergebnisse sortiert, sodass das Ergebnis mit dem höchsten Prozentwert ganz oben landet und am Ende dem Nutzer präsentiert werden kann.

Dieses Diagramm stellt den allgemeinen Ablauf des Rankings dar. Im Folgenden wird das Recommendation Ranking im Detail erklärt.

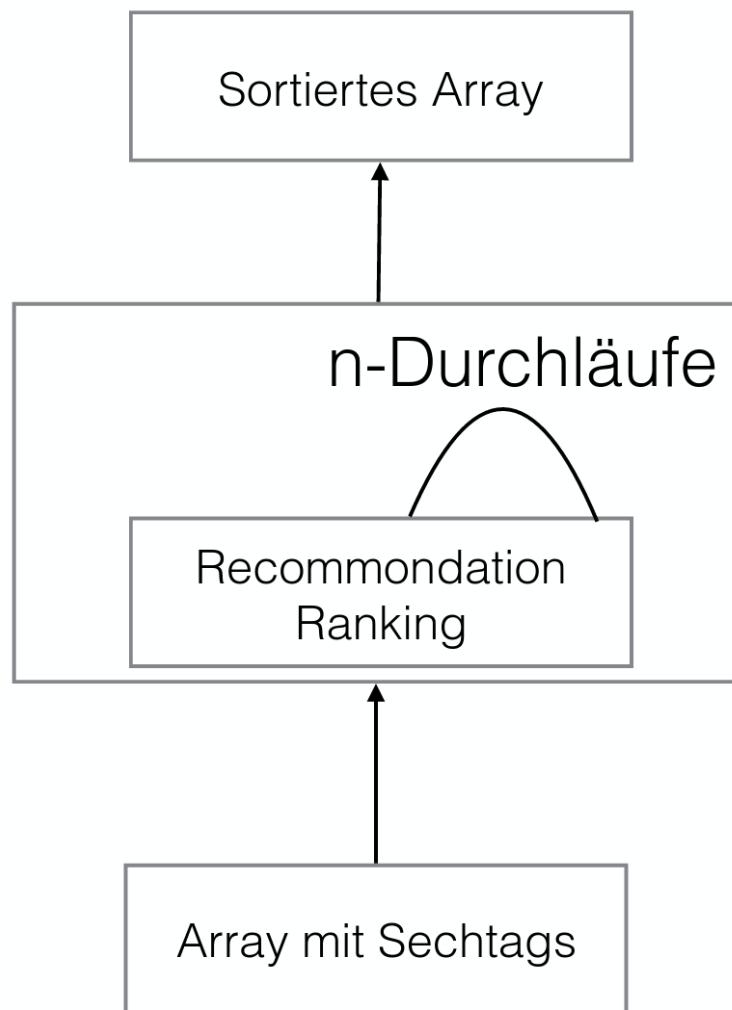


Abbildung 8.1.: Ablauf des Rankings

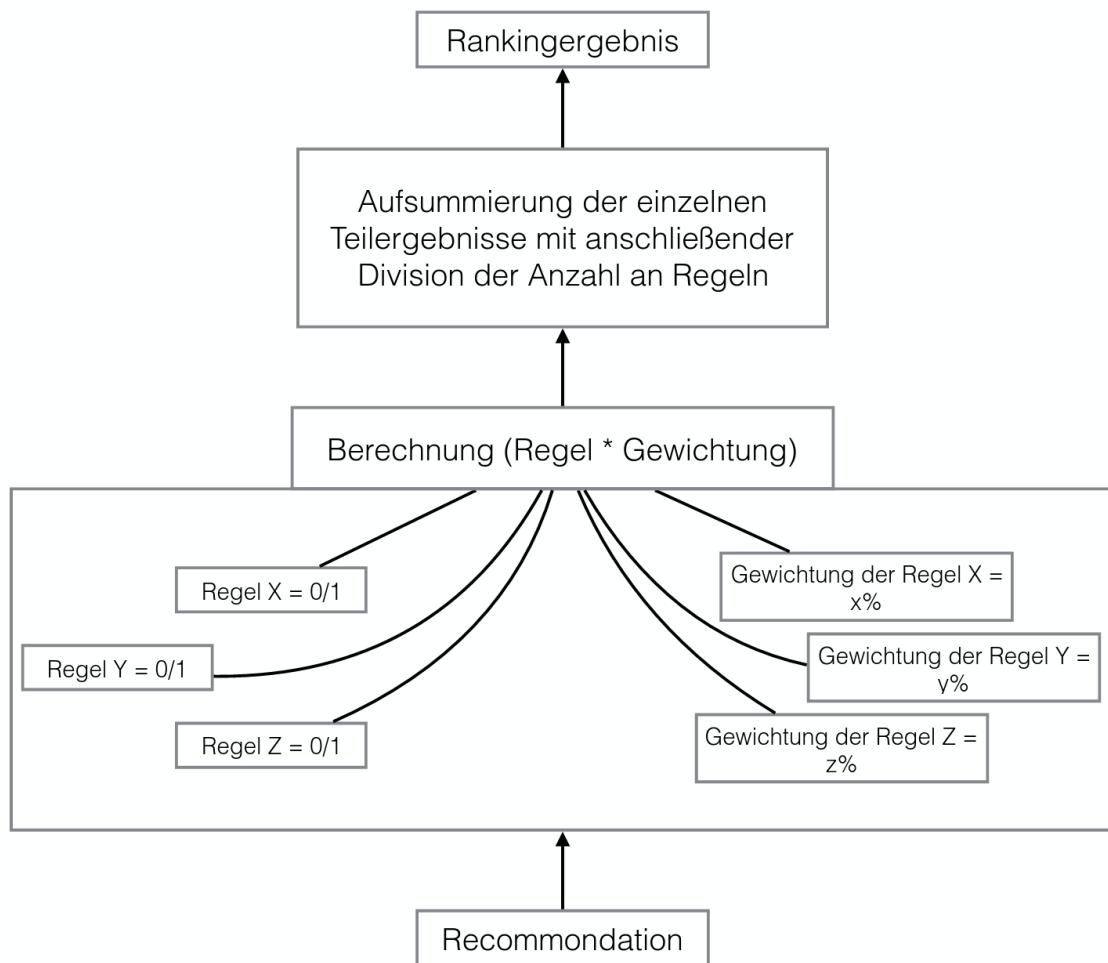


Abbildung 8.2.: Detailablauf Ranking

Teil III.

Konstruktion

9. Funktionalitäten

9.1. Übersicht

- WebView
- Back/Forward Button
- Lesezeichen
- Lesezeichen hinzufügen
- Home-Button
- AddressBar
- Reload
- Menü

Mit Hilfe der WebView ist es möglich, die gewünschte Webseite anzuzeigen. Befindet man sich auf einer Webseite mit SEARCH-Tags, so erhält man zu diesen zusätzliche Informationen.

Durch den Back- beziehungsweise Forward-Button springt man eine Seite zurück beziehungsweise vor.

In der AddressBar gibt man die URL der gewünschten Webseite an. Nach einem Klick in die WebView wird die URL geprüft und die Seite geladen.

Bei Eingabe eines Suchbegriffs zum Beispiel Bamberg wird man auf www.google.de weitergeleitet.

Nach einem Klick auf das Lesezeichensymbol werden die Lesezeichen in einer Tabelle angezeigt und können direkt darüber geladen werden.

Über Lesezeichen hinzufügen kann man der aktuellen Website einen Namen geben und sie unter diesem speichern.

Durch Klick auf den Home-Button wird man auf die aktuell festgelegte Startseite weitergeleitet.

Durch den Reload-Button ist es möglich, die aktuelle Webseite neu zu laden.

9.2. Addressbar

9.2.1. Funktionen

Die Addressbar prüft die Eingaben des Nutzers und schickt diese weiter an die WebView um die Seite anzeigen zu lassen. Bei Eingabe einer URL wird diese geprüft und an die WebView weitergeschickt um die Webseite anzuzeigen. Dabei wird die URL nur auf Vollständigkeit geprüft. Falls es dabei zu Fehlern kommt (fehlen von "http://www", "http://" oder "www") wird die URL ergänzt und erst dann weitergeliefert. Wenn es sich bei der Eingabe um keine URL handelt, wird die Eingabe an Google weitergeleitet und die Ergebnisse angezeigt.

9.2.2. Implementierung

Die in die Addressbar oder über den HomeButton eingegebene URL wird über die Funktion loadURL() an checkURL weitergeleitet. Hier wird der String auf Fehler überprüft und der geänderte String zurückgegeben. Danach wird er mit loadRequest() an die WebView weitergegeben.

addressBar/homBTN():
Eingabe der URL als String.

loadURL():
Aufrufe der Verarbeitungsschritte für die URL.

checkURL():
Aufgeteilt in einzelne Funktionen, welche mit Hilfe von Regulären Ausdrücken die URL nach fehlenden Elementen prüfen. Falls es Element fehlt wird dieses hinzugefügt. Wenn es sich um keine URL handelt, wird eine URL für die Googlesuche erstellt.

validateHTTPWWW():
Prüfen nach fehlendem "http://www".

validateHTTP():
Prüfen nach fehlendem "http://".

validateWWW():
Prüfen nach fehlendem "www".

loadRequest:
Weitergabe der geprüften und verbesserten URL an die WebView.

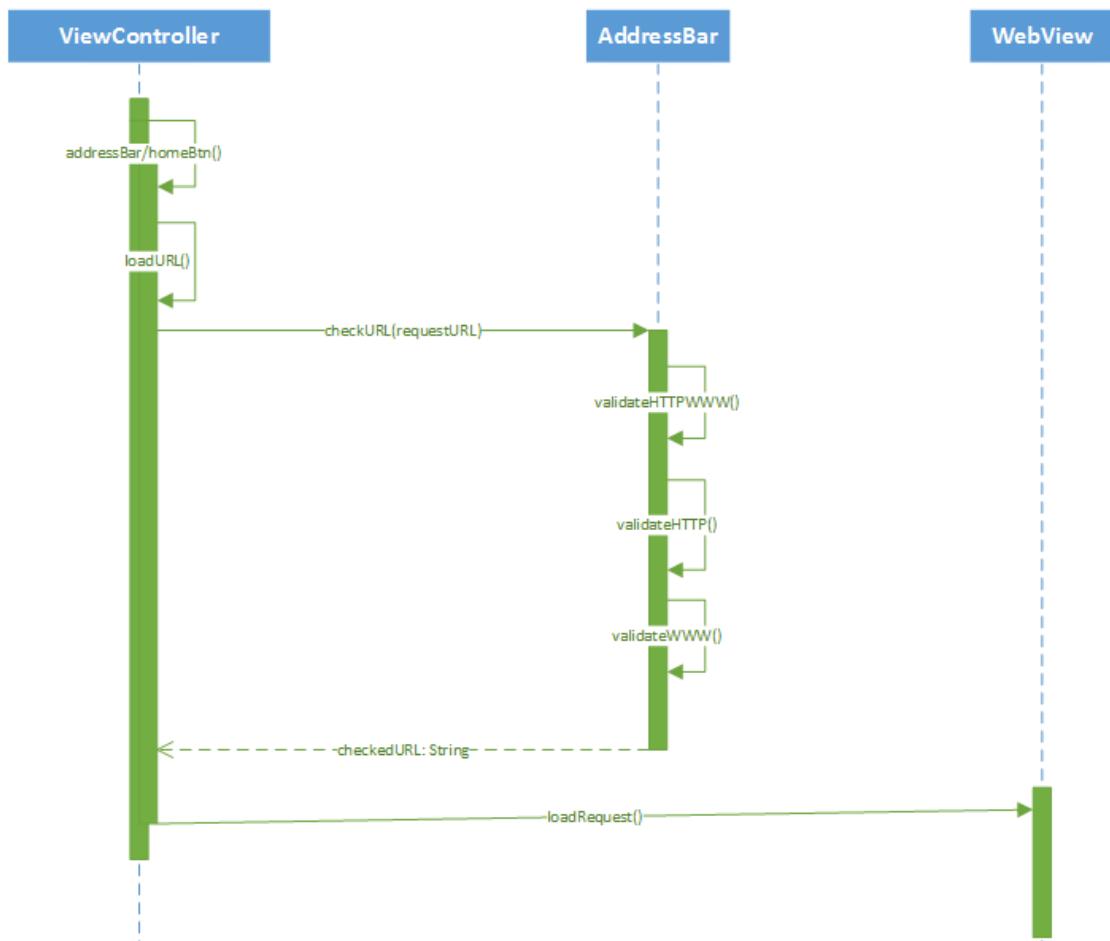


Abbildung 9.1.: Sequenzdiagramm Adressbar

9.3. Lesezeichen

9.3.1. Lesezeichen hinzufügen

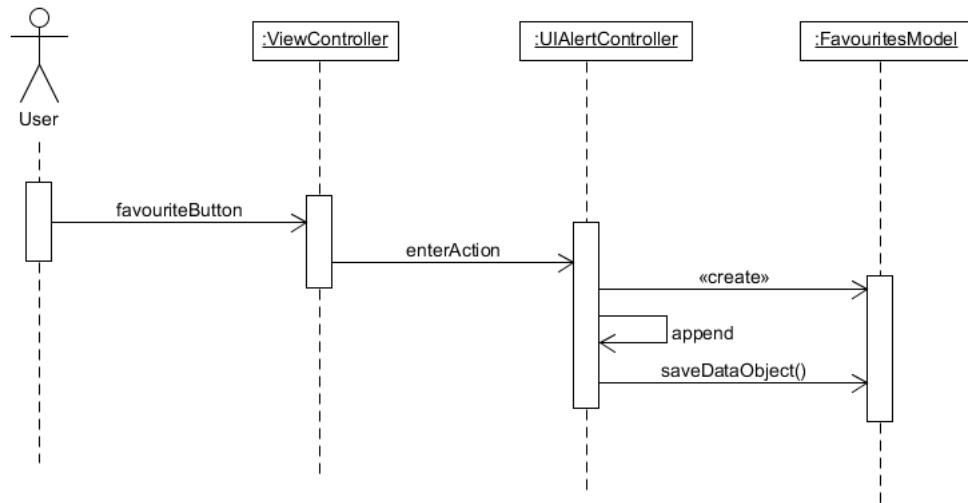


Abbildung 9.2.: Lesezeichen hinzufügen

Nachdem der User auf den Plusbutton zum Hinzufügen eines neuen Lesezeichens klickt, wird im ViewController die IBAction des Buttons aufgerufen. Hier wird die enterAction des UIAlertControllers aufgerufen. Nun kann der Nutzer den gewünschten Titel des Lesezeichens eingeben. Die URL wird automatisch übernommen. Nachdem man auf Enter zum Bestätigen klickt, wird ein neues Objekt des FavouriteModels erzeugt. Anschließend wird das erstellte Lesezeichen in das Model geschrieben und persistent gespeichert.

9.3.2. Lesezeichen bearbeiten

Nachdem der User auf das Lesezeichensymbol zum Anzeigen der gespeicherten Lesezeichen klickt, wird im ViewController die prepareForSegue-Methode aufgerufen. Nun wird der FavouriteTableViewCellController geladen. Hier wird die Methode editActionsForRowAtIndexPath indexPath: NSIndexPath aufgerufen. Nachdem man den entsprechenden Eintrag in der TableView auswählt, öffnet sich ein AlertFenster. Hier ist es möglich, den Titel zu ändern. Nach dem Bestätigen wird dies persistent gespeichert und die TableView aktualisiert.

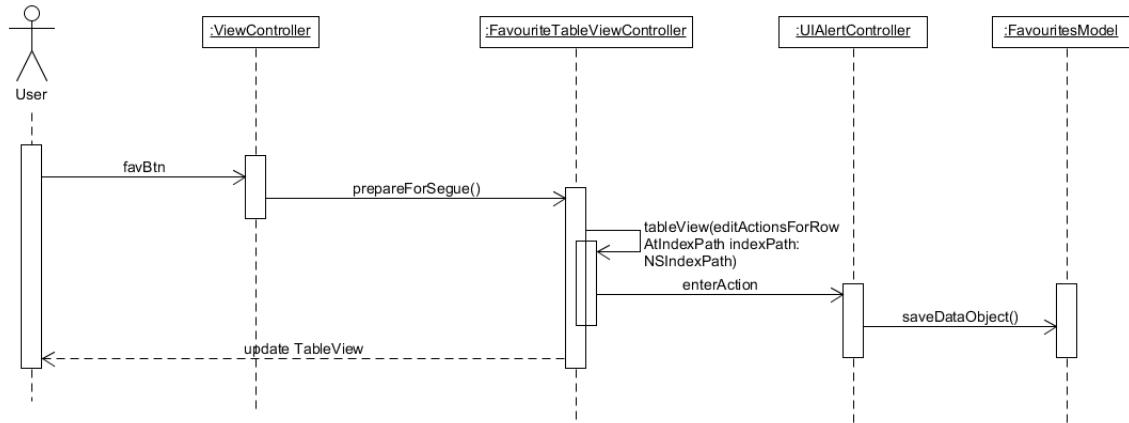


Abbildung 9.3.: Lesezeichen bearbeiten

9.3.3. Lesezeichen anzeigen

Nachdem der User auf das Lesezeichensymbol zum Anzeigen der gespeicherten Lesezeichen klickt, wird im ViewController die `prepareForSegue`-Methode aufgerufen. Nun wird der FavouriteTableViewController geladen. Hier wird die Methode `tableView(didSelectRowAtIndexPath: IndexPath)` aufgerufen. Hier wird überprüft, ob der Nutzer auf eine Zelle der TableView geklickt hat. Sobald man auf eine Zelle klickt, wird die entsprechende URL im FavouriteModel gespeichert und die Delegate-Methode `receiveInfo()` im ViewController aufgerufen. Dort wird die Methode `loadURL` mit der entsprechenden URL aufgerufen. Anschließend gelangt der Nutzer zum ViewController und die URL wird in der WebView angezeigt.

9.3.4. Lesezeichen löschen

Nachdem der User auf das Lesezeichensymbol zum Anzeigen der gespeicherten Lesezeichen klickt, wird im ViewController die `prepareForSegue`-Methode aufgerufen. Nun wird der FavouriteTableViewController geladen. Hier wird die Methode `editActionsForRowAtIndexPath: IndexPath` aufgerufen. Hier wird der ausgewählte Eintrag gelöscht. Dies wird persistent gespeichert und die TableView aktualisiert.

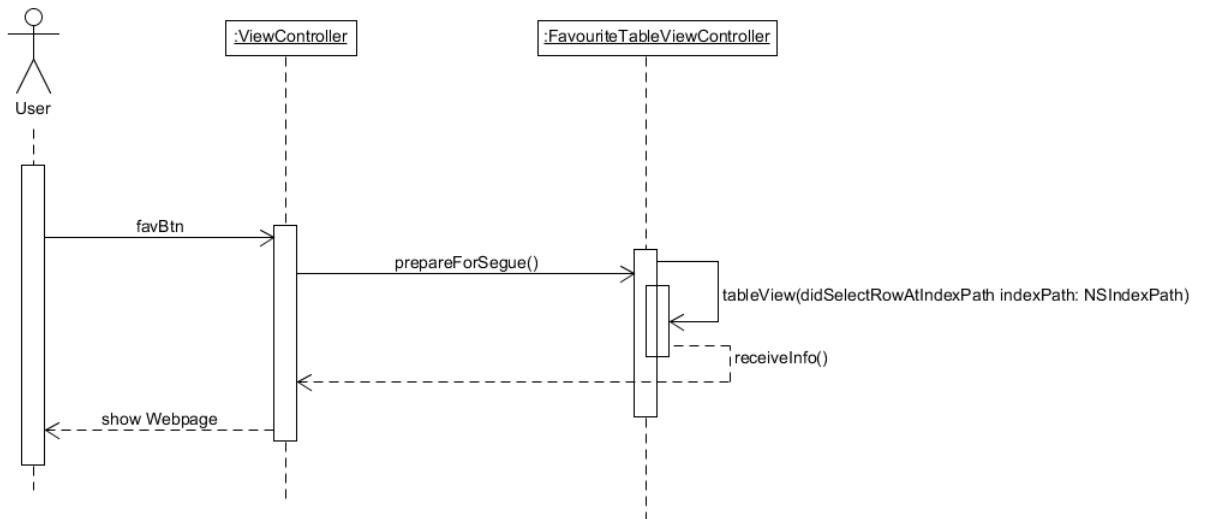


Abbildung 9.4.: Lesezeichen anzeigen

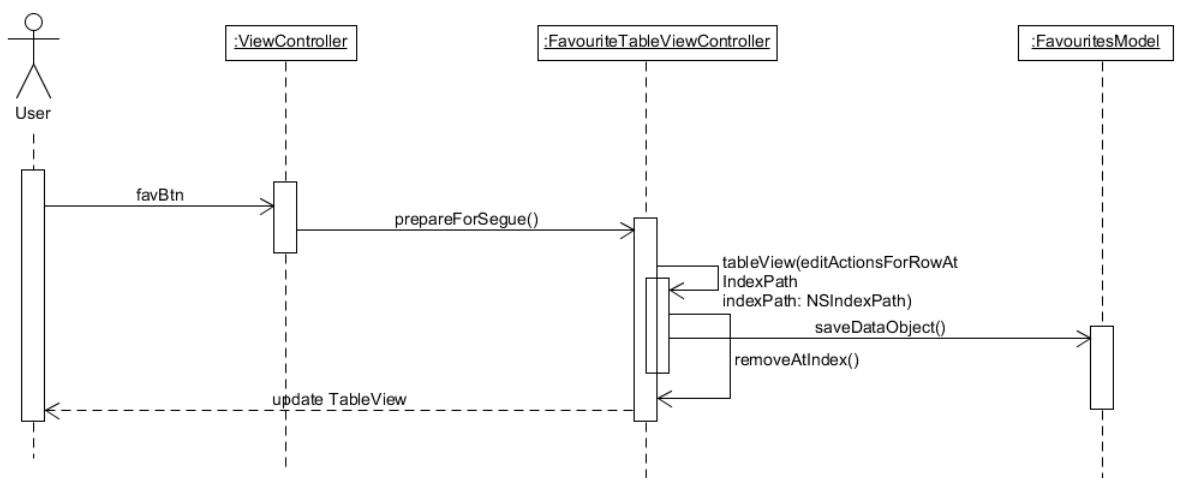


Abbildung 9.5.: Lesezeichen löschen

10. Einstellungsmöglichkeiten

Der Nutzer hat die Möglichkeit bestimmte Einstellungen vorzunehmen, welche die Ergebnisse der Suchanfragen beeinflussen.

Folgende Einstellungen können getätigt werden:

- Geschlecht des Nutzers (männlich, weiblich)
- vom Nutzer bevorzugte Sprache (deutsch, englisch)
- Stadt
- Land
- Geburtsdatum des Nutzers

Nachdem der Nutzer alle Einstellungen getätigt hat, werden die entsprechenden Attribute in ein Dictionary der Form [String:String] hinzugefügt und die Methode ”createJSONForRequest (keyWordsWithKeys:[String:AnyObject], detail:Bool, pref:[String:String]) ->JSONObject?” der Klasse ”MainController” wird aufgerufen. Diese Methode berechnet zu erst mit Hilfe der Methode ”calculateAgeRange(ageInYears:Int) ->Int” die sog. ”Age-Range”. Die ”Age-Range” ist eine als String gespeicherte Zahl und gibt an ob es sich beim Nutzer um ein Kind, jungen Erwachsenen oder Erwachsenen handelt. Welche ”Age-Range” jedem Alter zugeordnet ist, kann folgender Tabelle entnommen werden:

Alter	Bezeichnung	Age-Range
0-17	Kind	0
18-25	junger Erwachsener	1
ab 26	Erwachsener	2

Tabelle 10.1.: mögliche Altersbereiche

Danach wird das entsprechende JSON-Objekt mit den dazugehörigen Attributen erzeugt.

11. WkWebKit JavaScript

WkWebKit ersetzt den ursprünglichen UIWebView und wurde in unserer App nachträglich hinzugefügt, da war anfangs einen UIWebView verwendet haben. Dieser bietet aber weniger Funktionen und wird somit von Apple nicht mehr empfohlen. Die Einbindung des WkWebView ist zum derzeitigen Stand jedoch noch nicht über den Interface Builder möglich und muss somit programatisch erstellt werden. Erzeugen der verwendeten Variable myWebView als WKWebView:

```
var myWebView: WKWebView?
```

Zuweisung des WebViewDelegates ind er Funktion viewDidLoad():

```
myWebViewDelegate = WebViewDelegate()  
myWebViewDelegate.viewCtrl = self
```

Um JavaScript in die WkWebView einzubinden muss eine WkWebViewConfiguration erstellt werden:

```
let config = WKWebViewConfiguration()
```

Als nächstes muss der Pfad der JavaScript Datei angegeben werden mit dieser wird der ScriptContent extrahiert. Des weiteren muss der Zeitpunkt des Einfügens in das Dokument angegeben werden. Dies wird der WkWebViewConfiguration hinzugefügt genauso wie ein ScriptMessageHandler der auf klick Ereignisse reagiert:

```
let scriptURL = NSBundle mainBundle().pathForResource("main", ofType:  
"js")  
  
let scriptContent = try! String(contentsOfFile: scriptURL!, encoding:  
NSUTF8StringEncoding)  
  
let script = WKUserScript(source: scriptContent, injectionTime:  
.AtDocumentStart, forMainFrameOnly: true)  
  
config.userContentController.addUserScript(script)  
  
config.userContentController.addScriptMessageHandler(self, name:  
"onclick")
```

Die WkWebViewConfiguration hat nun alle erforderlichen Komponenten und kann zusammen mit der Angabe des Frames welcher hier die Grenzen des ContainerViews sind als SubView des ContainerViews gesetzt werden:

```
self.myWebView = WKWebView(frame: containerView.bounds,
configuration: config)
self.containerView.addSubview(myWebView!)
```

Somit werden auch die Constraints nur für den ContainerView gesetzt. Die Funktion userContentController(...) empfängt Nachrichten von JavaScript fügt diese der var headline hinzu und vergleicht die die SEARCH-Tags mit der headline und öffnet den PopView mit dem angeklickten SEARCH-Tag:

```
func userContentController(userContentController: WKUserContentController,
    didReceiveScriptMessage message: WKScriptMessage) {
    print("JavaScript is sending a message \(message.body)")
    self.headLine = message.body as! String

    let sechTags = tableViewDataSource.sechTags

    for(var i = 0; i < sechTags.count; i++){
        if(sechTags[i] == self.headLine){
            self.indexPathForSelectedSearchTag = i
        }
    }
    performSegueWithIdentifier("showPopView", sender: self)
}
```

Die Implementiere JavaScript-Datei main.js sucht html Tags der Art SEARCH-Link und markiert diese Rot, und sendet eine Nachricht an das wkWebkit wenn ein SEARCH-Tag angeklickt wird:

```
window.onload = function() {
    var elements = document.getElementsByTagName("search-link");
    for (var i=0; i<elements.length;i++){
        elements[i].style.color = "Red";
        elements[i].addEventListener("click", function(){
            var currentTopic = this.getAttribute("topic");
            webkit.messageHandlers.onclick.postMessage(currentTopic);});
    }
}
```

12. Websiteanalyse und Sechobjekterstellung

12.1. Vorbereitung

Nachdem die Methode `getSechObjects()` im `SECH-Manager`, durch den `WebViewDelegate`, aufgerufen wurde, wird zuerst mit der Analyse der HTML-Heads begonnen. Dies geschieht mit der Methode `getSechHead()`, welche ein Objekt vom Typ Tag zurückliefert. Außerdem wird in dieser Methode der Standardfilter für die EEXCESS-Anfrage erstellt und zwischen gespeichert. Falls im Body weiter Filter verwendet werden wird dieser Standardfilter überschrieben.

12.2. Analyse des `SEARCH-Body`s

Funktion `makeSech()`: In dieser Methode wird der HTML-Body in `SEARCH-Links` und `SEARCH-Sections` aufgeteilt, entsprechend analysiert und es werden `SECH-Objekte` erzeugt. Als erstes übergibt der `SECH-Manager` dem `HTML-Manager` den HTML-Body um ein Array von String zurück zu bekommen. Dieses Array enthält vier unterscheidbare String-Elemente:

- HTML-Opening-Tag einer Section mit Attributen der Section
- HTML-Closing-Tag einer Section
- HTML-Opening-Tag eines Links mit Attributen des Links
- HTML-Closing-Tag eines Links

Die Methode iteriert nun über das Array und entscheidet je nach Art des String-Elements was zu tun ist:

- HTML-Opening-Tag SECTION: Der `HTML-Manager` findet heraus welche Attribute die Section hat, speichert die Section zwischen und setzt gegebenenfalls einen neuen Filter. Außerdem wird die Variable `sectionIsAvailable` auf true gesetzt um in den folgenden Iterationen entscheiden zu können ob ein Link in einer Section steht oder nicht.
- HTML-Closing-Tag SECTION: Der Filter wird wieder zurückgesetzt und es ist keine Section mehr verfügbar.
- HTML-Opening-Tag LINK: Falls eine dieser Aufrufe zu tragen kommt und `sectionIsAvailable` gleich true ist, findet der `HTML-Manager` die Attribute des Links

und es wird ein SECH-Objekt erzeugt. Bei der Erzeugung werden der zwischen gespeicherte Head und die Section, sowie ein gegebenenfalls aktualisierter Filter übergeben. Dieses erzeugte Objekt wird dann an ein Array angehängt welches bei Abschluss der Websiteanalyse zurückgegeben wird.

- HTML-Closing-Tag LINK: Dieses Element kann ignoriert werden, da innerhalb eines Links keine weiteren HTML-Elemente von belang stehen.

12.3. Erstellung des Sechobjects

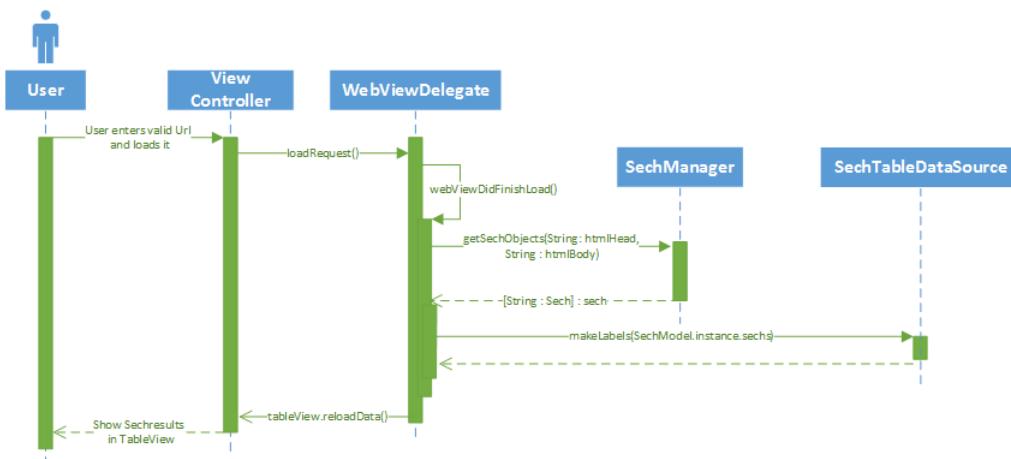


Abbildung 12.1.: Ablauf der Erstellung eines SECH-Objektes

Sobald der User eine gültige URL eingibt und lädt, wird die Laderoutine `loadRequest()` angestoßen. Ist die Seite fertig geladen wird in der Delegatemethode `webViewDidFinishLoad()` des WebviewDelegates das Auslesen bzw. Laden der SEARCh-Tags begonnen.

Dem SEARCh-Manager wird der HTML-Head und Body der eben geladenen Website übergeben und es werden mit Hilfe des HTMLManagers SEARCh-Objekte erstellt. Diese SEARCh-Objekte werden in einem Dictionary zurückgegeben.

Sobald ein SECH-Object-Dictionary zurückgegeben wurde, werden die Ergebnisse gerankt und an den SechTableDataSource übergeben. Anschließend wird mit `reloadData()` die Tabelle der SECH-Tags im ViewController gefüllt.

13. Server Anfragen

13.1. Beschreibung

Der Content-Prototyp ermöglicht es, Anfragen an den Server des EEXCESS-Projektes zu schicken. Der Prototyp unterstützt das PP-QF2 und somit auch das PP-RF2.

13.2. Aufbau des Prototypen

Der zunächst erstellte Prototyp ist grob wie folgt aufgebaut:

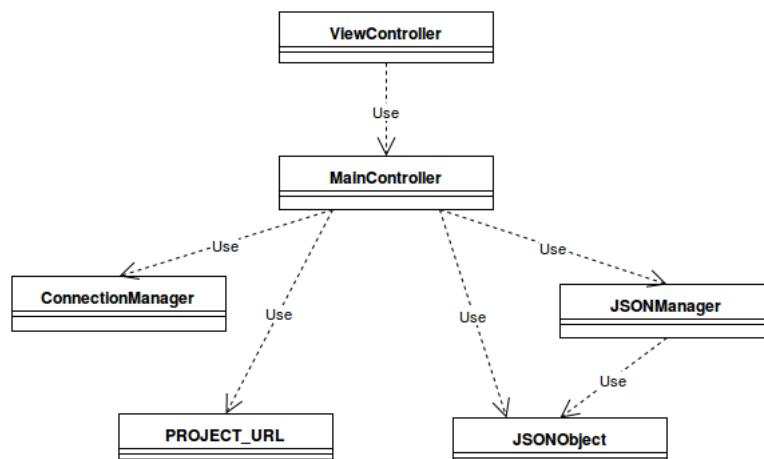


Abbildung 13.1.: Aufbau des Prototypen

13.3. Netzwerkverbindungen

Die Klasse ConnectionManager ist für das Erstellen einer Verbindung zum Server zuständig. Sie besitzt eine öffentliche Methode „makeHTTP_Request“. Zusätzlich gibt es eine Klasse PROJECT_URL für die URLs, welche im Projekt benötigt werden.

13.4. Verarbeiten/Aufbauen der Anfragen

13.4.1. Anfrage

1. Auslesen der Suchbegriffe
 - Beachten der Trennung der Suchbegriffgruppen
 - Einfügen in ein Dictionary, als Schlüssel wird der Suchbegriff genommen
 - Die Schlüssel werden in die ComboBox für das Bearbeiten der Werte eingefügt
2. Die Suchbegriffe der Parameter werden bearbeitet (isMainTopic und Type) (optional)
3. Eingabe der Preferenzen (optional)
4. Abschicken der Anfrage:
 - a) Suchwörter, Preferenzen und die generierte QueryID werden an den MainController weitergegeben
 - b) Übergabe der Werte an den JSONManager
 - c) Die Schlüssel werden in die ComboBox für das Bearbeiten der Werte eingefügt
 - d) Werte werden in das JSONObject eingefügt und zurückgegeben
 - e) JSONObject wird über den MainController an den ConnectionManager weitergegeben
 - f) Der ConnectionManager verschickt die Daten an den Server
5. Die Antwort vom Server wird im ViewController:53 entgegen genommen und dargestellt
6. Speichert JSON in das Dictionary „mapOfJSONs“ in der Klasse „MainController“
7. Fügt die DocumentBadges in die die ComboBox für die DetailAnfrage

13.4.2. DetailAnfrage

1. Auswahl des DocumentBag oder der „take all“ Option
2. Betätigen des „Search Details“ -Buttons
 - a) Übergibt den DocumentBag/DocumentBags an den JSONManager über den MainController
 - b) JSONObject wird erzeugt und zurückgegeben
 - c) DetailAnfrage wird an den Server übermittelt
3. Antwort wird vom ViewController entgegengenommen und im TextFeld dargestellt

13.5. JSON Objekt

Der JSON wird in der Klasse `JSONObject` gehalten. Ein Dictionary in der Klasse speichert alle Schlüssel/Wert-Paare des `JSONObject`s. Es gibt drei Varianten für das Erstellen eines Objekts der Klasse `JSONObject`:

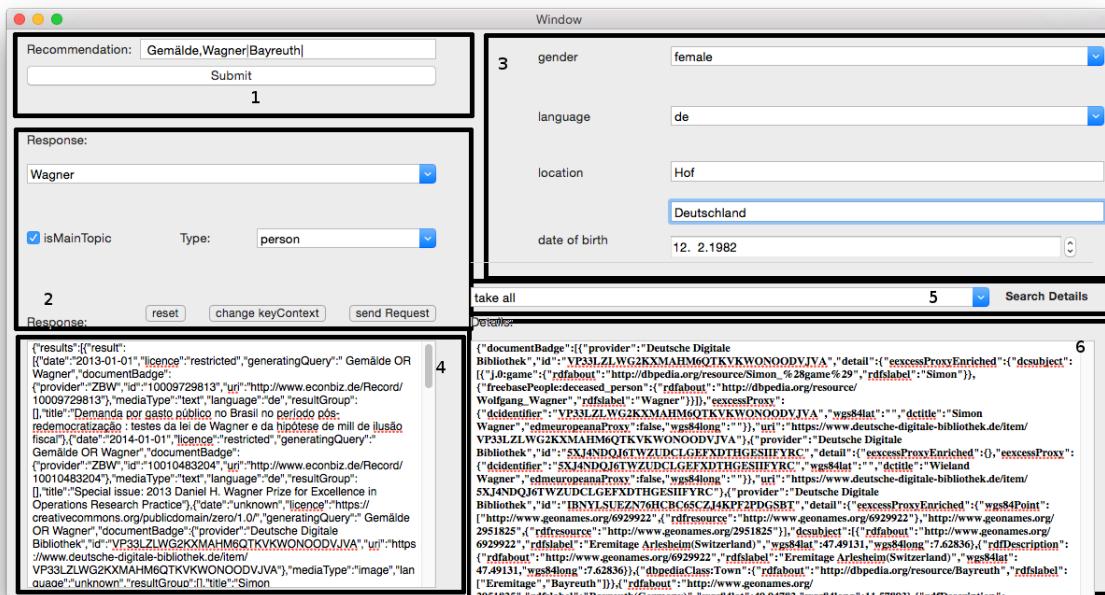
- ohne Übergabewert
- mit einem Dictionary
- mit einem `NSData`-Objekt, welches ein Dictionary enthält

Die Klasse `JSONObject` besitzt verschiedene Methoden:

- Auslesen von Werten (spezialisiert auf bestimmte Datentypen)
- Einer `found()`-Methode zum Überprüfen, ob der Schlüssel vorhanden ist
- Zum Erweitern des JSONs mit Werten
- Convert-Methoden für das Umwandeln von `JSONObject` in String und `NSData`

Die Klasse `JSONObject` kann ohne großen Aufwand um weitere Methoden ergänzt werden, wie zum Beispiel um eine Methode, `getImage(key:String)` zu erweitern.

13.6. Prototyp UI



1. Eingabe der Suchbegriffe. Mit „,“ werden die Begriffe getrennt und mit „—“ werden die Suchbegriffgruppen getrennt
2. Hier können den Suchbegriffen noch Eigenschaften hinzugefügt werden. Nach jeder Änderung muss der „change keyContext“ -Button gedrückt werden. Mit dem Button „send Request“ wird die Anfrage verschickt. Der Button „reset“ besitzt keine Funktion.
3. In diesem Bereich können zusätzliche Informationen für die Anfrage angegeben werden.
4. Anzeige der Antwort vom Server
5. Auswahlmöglichkeiten für die Detailanfrage und der „Search Details“ -Button
6. Anzeige der DetailAntwort vom Server

14. Klassen Implementierung

Es entsteht ein Objekt der Klasse `SEARCHModel`:

```
class SEARCHModel {  
    static let LINK_TAG = "LINK"  
    static let SECTION_TAG = "SECTION"  
    static let HEAD_TAG = "HEAD"  
  
    var id = String()  
    var tags = [String : Tag]()  
    // String is id (link, section, head) and Tag is Tag-Object  
    var filters = Filter()  
  
}  
  
class Tag {  
    var topic = String()  
    var type = String()  
    var isMainTopic = false  
}  
  
class Filter {  
    var mediaType = String()  
    var provider = String()  
    var licence = String()  
}
```

15. Ranking

15.1. Sequenzdiagramm

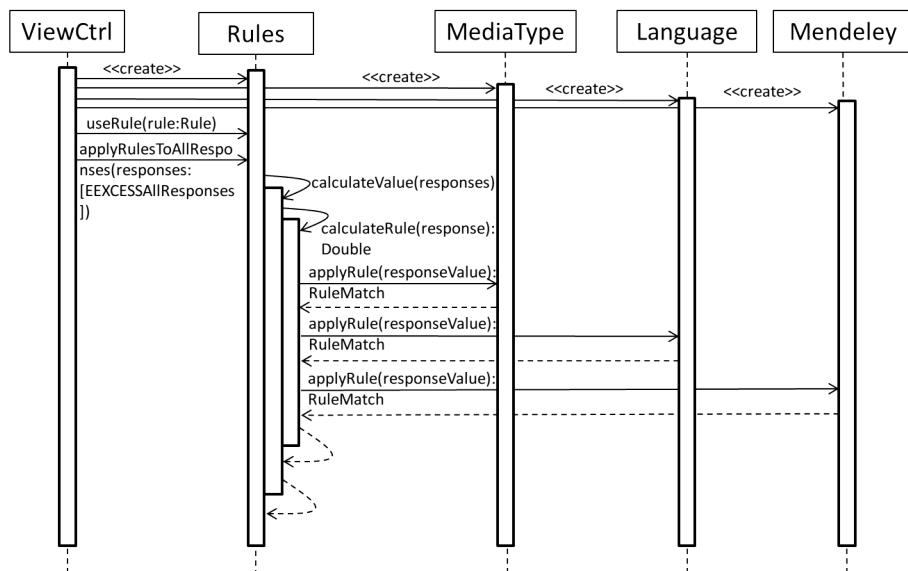


Abbildung 15.1.: Sequenzdiagramm des Rankings

Dieses Sequenzdiagramm zeigt einen exemplarischen Durchlauf des Rankings der Vorschläge für den Benutzer. Nachdem die Daten aus dem Internet abgerufen wurden und an den ViewController (Team UI) übergeben wurden folgt die Sortierung der Ergebnisse nach den Interessen des Nutzers. Es existieren eine Reihe an verschiedenen Regeln, die angewendet werden können. Die Anzahl an Regeln kann variieren. Welche Regeln angewendet werden, wird vom ViewController festgelegt mit Hilfe der Methode „`useRule(rule:Rule)`“. Nachdem die Regeln, die angewendet werden sollen, festgelegt wurden, müssen diese angewendet werden. Dies geschieht mit der Methode „`applyRulesToAllResponses(responses:[EXCESSAllResponses])`“. Als Argument müssen alle zuvor erhaltenen Ergebnisse mit übergeben werden. Die Klasse Rules übernimmt im Anschluss die Berechnungen, die für das Ranking der Ergebnisse notwendig sind. Die Methode „`calculateValue(responses)`“ berechnet in Verbindung mit der Methode „`calculateRule(response): Double`“ einen Durchschnittswert für jede einzelne Antwort für jeden einzelnen SEARCH-Tag, nachdem jede Regel angewendet wurde. Eine Regel wird durch den Aufruf der Methode „`applyRule(responseValue): RuleMatch`“ angewandt, welche jede Regel besitzt und liefert den Wert 0 oder 1 zurück. Dieser bedeutet, ob die Regel

auf die Interessen des Nutzers zutrifft oder nicht. Zum Abschluss des Rankings werden die einzelnen Ergebnisse entsprechend nach dem zuvor errechneten Durchschnittswert sortiert. Im Anschluss erhält das Team UI die sortierten Werte zurück und kann sie dem Nutzer darstellen.

15.2. Klassendiagramm

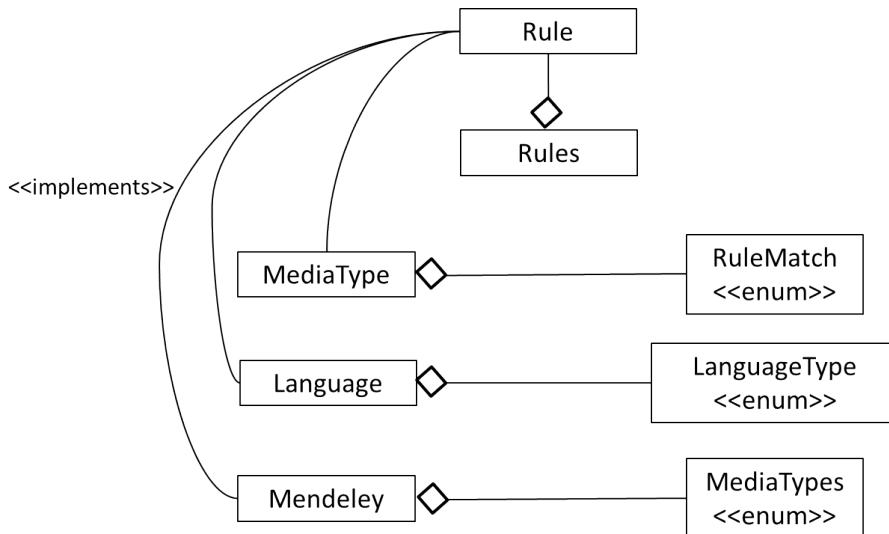


Abbildung 15.2.: Klassendiagramm des Rankings

15.3. Abruf der Daten vom Server Sequenzdiagramm

Dieses Sequenzdiagramm zeigt einen exemplarischen Ablauf zum Abrufen der Daten vom Server. Zuerst wird eine Instanz des TaskControllers erzeugt. Dieser stellt die Schnittstelle zum View zur Verfügung. Er empfängt und sendet Daten von bzw. an den View. Die Methode „getRecommendations([SEARCHModel], setRecommendations:(String, [Response]))“ bildet die Schnittstelle, bei der die einzelnen, aus der HTML Seite ausgelesenen, SEARCH-Tags in Form von SEARCHModels übergeben werden, sowie die Methode „setRecommendations(String, [Responses])“, die für die Rückgabe der heruntergeladen Ergebnisse zuständig ist. Im Anschluss wird eine Instanz des JSONConnectionControllers sowie des EEXCESSRecommendationJSONControllers erzeugt. Der JSONConnectionCtrl ist für den Aufbau der Verbindung zum Server zuständig. Beim Erzeugen es EEXCESSRecommendationJSONControllers werden die SEARCHModels in einen JSON String konvertiert. Mit Hilfe der Methode „post(AnyObject, String, postCompleted(Bool, NSData))“ wird zuvor erzeugte JSON String an den Server gesendet. Nach

Erhalt der Daten wird die Methode `postCompleted(Bool, NSData)` mit diesen heruntergeladenen Daten aufgerufen. Innerhalb von `postCompleted(...)` wird wiederum eine Instanz des `EEXCESSRecommendationControllers` erzeugt, welcher die JSON Daten parst und in die interne Struktur der App überführt. Hierfür wird für jede einzelne Antwort eines `SEARCH`-Tags eine Instanz von `EEXCESSSingleResponse` erzeugt. Alle Antworten, die zu einem `SEARCH`-Tag gehören, werden in einer neuen Instanz von `EEXCESSAllResponses` gespeichert. Das komplette Array von allen `EEXCESSAllResponses` wird an den `TaskCtrl` zurückgegeben. Dieses Array wird mit der zu Beginn erhaltenen Methode „`setRecommendations(String, [EEXCESSAllResponses])`“ an den View zurückgegeben, welcher die Ergebnisse dann darstellt.

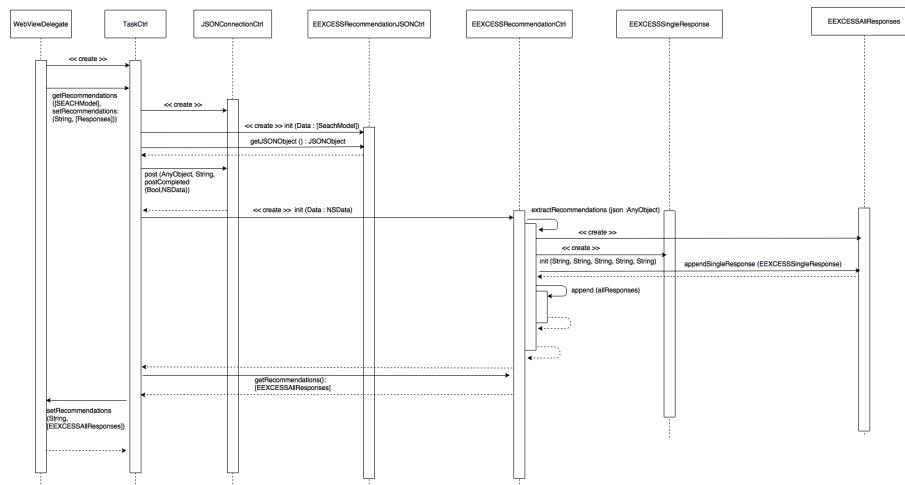


Abbildung 15.3.: Sequenzdiagramm einer Serveranfrage

15.4. Abruf der Daten vom Server Klassendiagramm

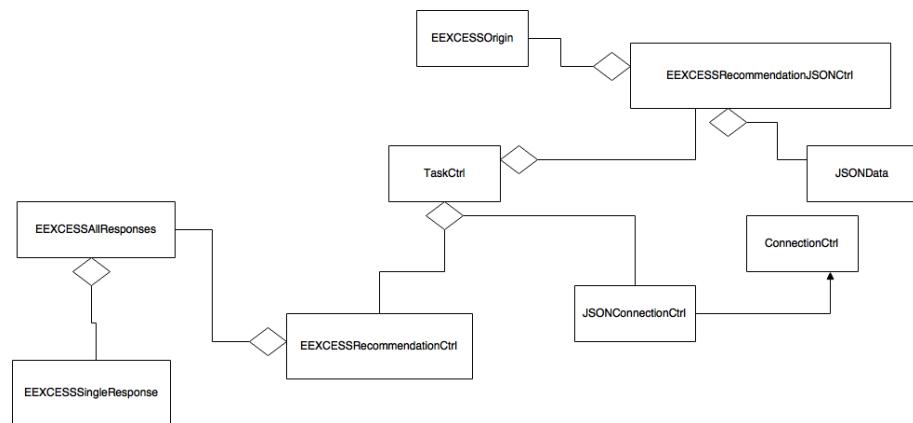


Abbildung 15.4.: Klassendiagramm einer Serveranfrage

16. Entwicklung des Layouts des Browsers

16.1. Übersicht

Das Layout des Browsers besteht aus folgenden Elementen:

- Navigation Controller
- Navigation Bar
- Back/Forward Button
- Lesezeichen Button
- Lesezeichen hinzufügen Button
- Home Button
- AddressBar
- Reload Buttons
- SECH Button
- Optionen Button
- Container
- WKWebView

Die oben aufgelisteten Elemente stellen das komplette Layout des Browser dar. Der Navigation Controller dient in diesem dazu, dem Browser einen automatisch generierten Navigation Bar hinzuzufügen, welche im oberen Bildschirm fest verankert ist und eine nicht veränderbare Höhe und Breite liefert. Diese Navigation Bar besitzt einen Back Button, um auf einer Seite zurück zu springen und einen Forward Button, um eine Seite vor zu blättern. Bedient man den Lesezeichen Button so öffnet sich eine TableView mit aller gespeicherten Seiten chronologisch absteigend in der er eine Seite auswählen, löschen oder editieren kann. Will der Nutzer die Liste um ein weiteres Element ergänzen, so muss er den Plus Button bedienen und es öffnet sich ein Pop Up Fenster, der den Link der aktuell besuchten Seite automatisch übernimmt und der Nutzer diesem lediglich nur noch einen Titel vergeben und abspeichern muss. Mit dem Home Button gelangt der Nutzer zu seiner definierten Startseite. Die AddressBar dient für die Eingabe eines Weblinks der durch bestätigen zu der eingegebenen Seite gelangt. Um die aktuelle Seite neu zu laden, muss der Reload Button bedient werden. Will der Benutzer alle SEARCh-Tags

auf der aktuellen Seite anzeigen, so bedient er den SECH-Button. Ist der SECH-Button angeklickt worden, so klappt sich von der rechten Bildschirmseite eine TableView mit allen SEARCH-Tags auf und bietet dem Nutzer eine Übersicht dieser. Wird eines der SEARCH-Tags in dieser TableView angeklickt, so öffnet sich ebenfalls ein PopUp Fenster zu diesem Tag, der weitere Informationen zurückliefert. Um die SEARCH-Tag Tabelle wieder einzuklappen wird ein weiteres Klicken auf den SECH-Button benötigt und diese führt sich ein. Durch einen Klick auf den Optionen Button öffnet sich auch hier ein Pop Up Fenster, der benutzerspezifische Informationen mitliefert und übersichtlich darstellt. Um den WKWebView eine dynamische Höhe und Breite programmatisch zu übergeben, wurde ein Container, eine Schicht unter diesem, mit festen Constraints übergeben. Diese Constraints passen sich zu den benachbarten Elementen, wie der ausgefahrenen SECH-Tabelle oder der Navigation Bar, an und das WKWebView übernimmt die Constraints dieses darunter liegenden Containers. Dreht man das Endgerät beispielsweise von Hochformat in den Querformat, ändern sich Höhe und Breite des Containers und diese werden vom WKWebView übernommen.

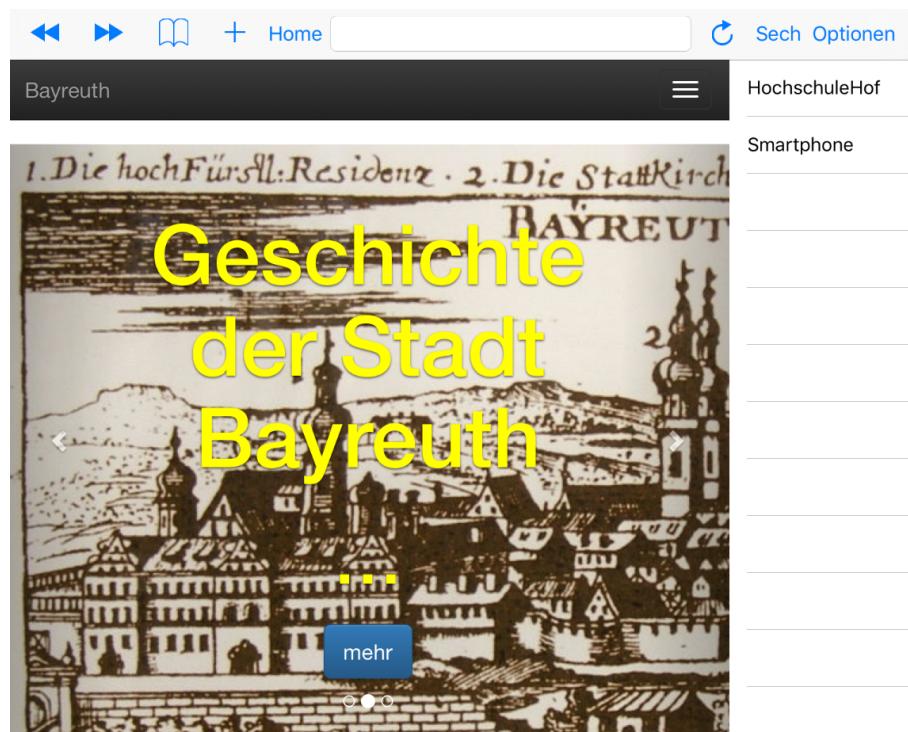


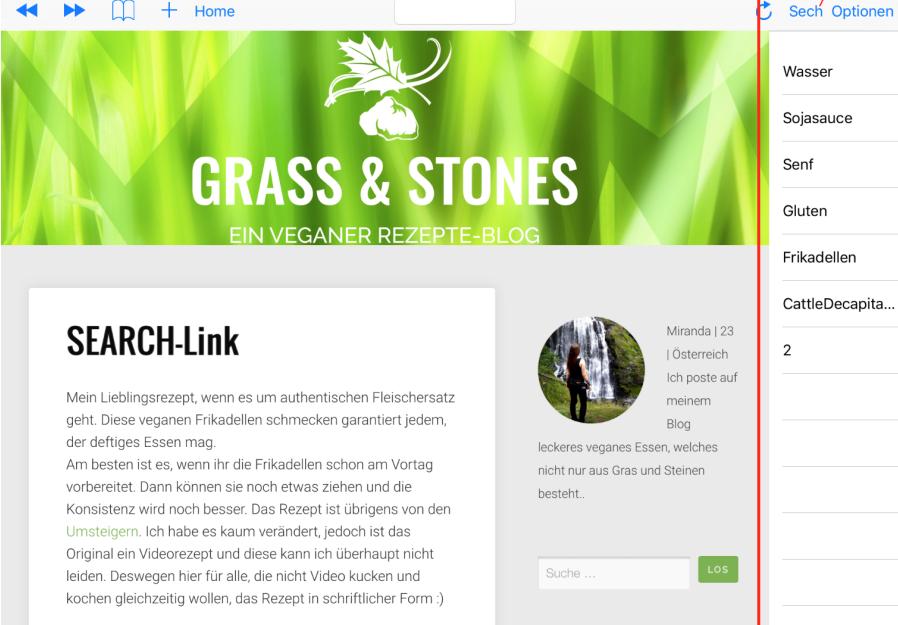
Abbildung 16.1.: Browser im Hochformat

16.2. Animation der TableView

Um eine Übersicht aller gefundenen SEARCH-Tags grafisch darzustellen, wurde ein Label über den SECH-Button ergänzt, welche die Anzahl der SEARCH-Tags in einer roten Schrift angibt. Diese Anzahl soll dem Nutzer zeigen, wie viele SEARCH-Tags sich auf der Webseite befinden. Beim Bedienen des SECH-Buttons, wird die SEARCH-Tabelle animiert herbei gerufen. Diese Tabelle wird rechts im Browser ein- und ausgefahren. Das WKWebView wird dementsprechend vergrößert oder verkleinert. Bedient man diesen SECH-Button, so fährt sich eine Tabelle vom rechten Bildschirm aus und liefert alle Tags in einer TableView zurück. Die zurückgelieferten Tags werden in separaten Zellen auf der TableView abgespeichert, die durch anklicken eine neue Pop-Up Seite, spezifisch zu dem angeklickten Tag, öffnet. Aufgrund der Umstellung der Navigationsleiste wird diese grafische Darstellung der Anzahl an SECH-Tags nicht mehr angezeigt. Die Umstellung der Navigationsleiste war notwendig um die Höhe und Breite des WKWebViews festzulegen, darum ist ein neuer Navigation Controller erzeugt worden, welche die Navigation Bar automatisch mitliefert. In die neue Navigation Bar kann jedoch kein Label hinzugefügt und spezifisch positioniert werden, somit wurde diese Option entfernt.

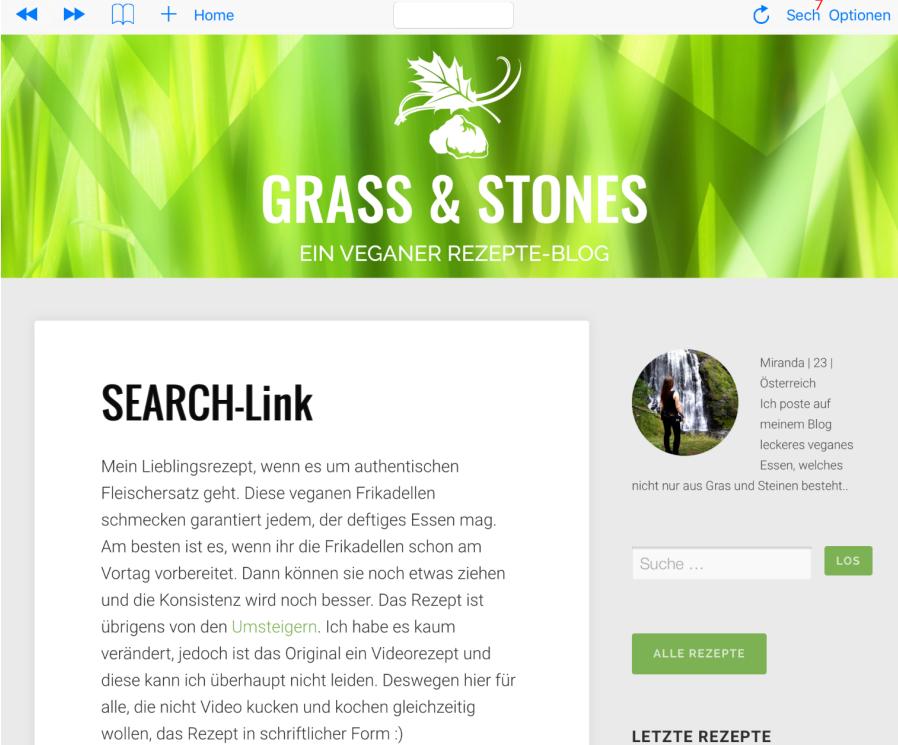


Abbildung 16.2.: Anzahl der SEARCH-Tags



The screenshot shows the GRASS & STONES blog homepage. A sidebar on the right is titled "Sech Optionen" and contains a list of categories: Wasser, Sojasauce, Senf, Gluten, Frikadellen, CattleDecapita..., and 2. Below this is a search input field with the placeholder "Suche ..." and a green "LOS" button. The main content area features a post by user Miranda from Austria, which includes a small profile picture of her standing in front of a waterfall.

Abbildung 16.3.: SEARCH-Tabelle aufgeklappt



This screenshot shows the same blog homepage as above, but the sidebar has been collapsed. The "Sech Optionen" title is still visible at the top of the sidebar, but the category list and search input are no longer present. The main content area remains the same, displaying the post by user Miranda.

Abbildung 16.4.: SEARCH-Tabelle verstekkt

16.3. Constraints setzen

Damit die Bedienelemente und das UI vom Browser auf verschiedenen Endgeräten identisch dargestellt wird, müssen Constraints gesetzt werden. Somit wird eine dynamische und bei bestimmten Bedienelementen eine feste Position gesetzt bzw. angepasst. Komplikationen mit den schon vorhandenen Constraints vom Container im Hintergrund waren vorhanden, somit mussten alle Constraints gelöscht und neu erzeugt werden. Nachdem das WKWebView dennoch keine Höhe und Breite vom darunter liegenden Container angenommen hatte (Vermutung: Im main.storyboard nicht verwendete Constraints, die das Layout ungewollt ändern), musste der Navigation Controller gelöscht und erneut integriert werden. Hierzu wurden folgende Schritte beachtet:

- Referenzen der Bedienelemente im main.storyboard und im ViewController löschen
- Menüleiste samt Buttons im main.storyboard löschen
- Tabelle/Zellen löschen
- Container im Hintergrund löschen
- Navigation Controller ausklinken und entfernen

Navigation Controller neu erstellen:

- Menüleiste samt Buttons einfügen und feste Constraints setzen
- Container einfügen und feste Constraints setzen
- Tabelle/Zellen einfügen und feste Constraints setzen
- Referenzen neu verlinken und alles wieder in den ViewController integrieren

Nach erfolgreichem absolvieren dieser Schritte wurden im ViewController zwei Constraints erzeugt. Diese dienen für die Zuweisung der Höhe und Breite vom Container auf den darüber liegenden WKWebView, welches programmatisch erzeugt wird. Ändern sich die Constraints vom Container, beispielsweise beim rotieren des Endgerätes, so ändert sich auch Höhe und Breite des WKWebViews.

Ein weiteres Problem stellte die Navigation Bar dar, da diese im nachhinein ergänzt wurde, anstatt diese über einen Navigation Controller zu erzeugen. Das Layout wurde nun neu aufgesetzt und eine vom Navigation Controller erzeugte Navigation Bar verwendet. Gleichzeitig wurde das Problem, in der der Container verschoben wurde und das darüber liegende WKWebView verrutscht ist, gelöst, da diese Constraints sich nun von der Navigation Bar aus orientieren.

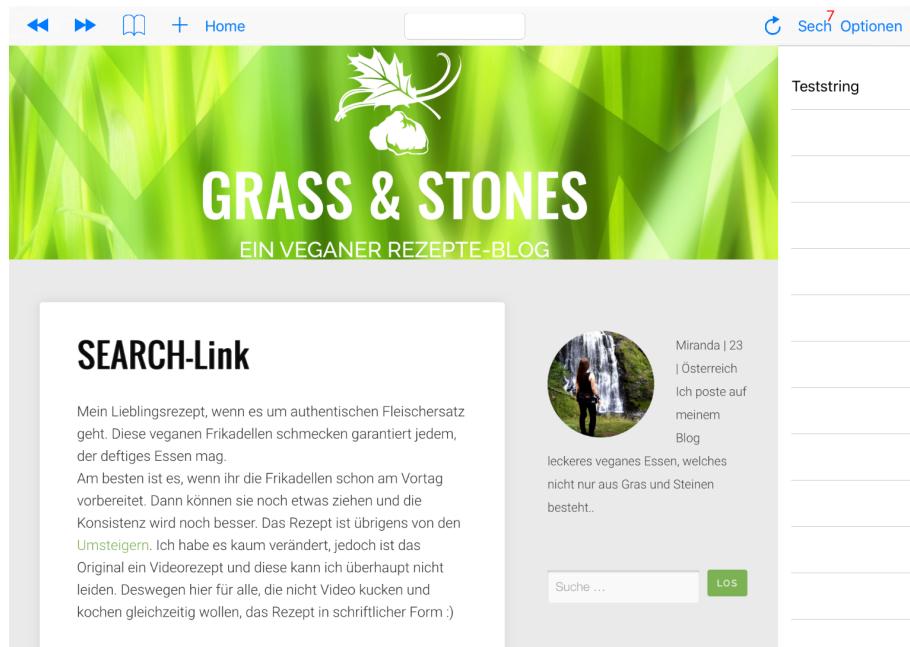


Abbildung 16.5.: Hochformat des Browsers

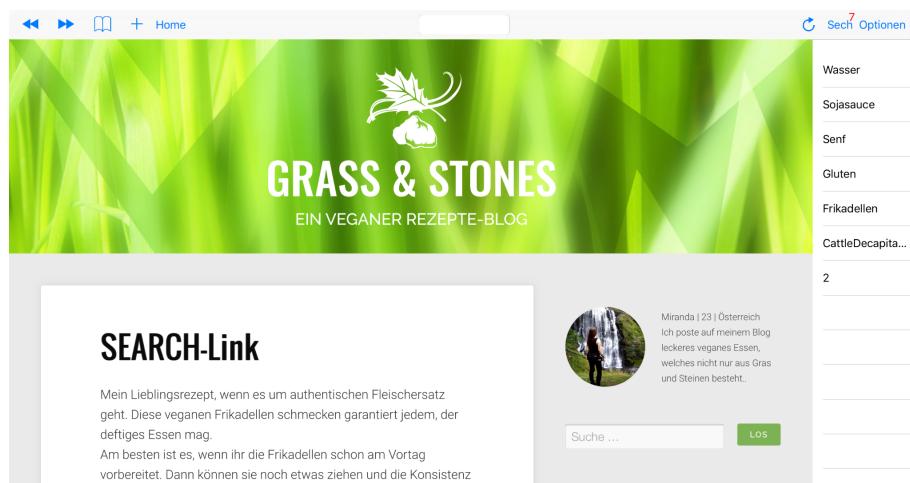


Abbildung 16.6.: Querformat des Browsers

16.4. PopOver Funktion

Damit sich der Nutzer gefundene SEAR^CH-Tags direkt in dem Browserfenster anzeigen lassen kann, wurde die PopOver Funktion eingeführt. Diese stellt den angeklickten SEAR^CH-Tag in einem neuen Fester auf dem Screen dar. Dabei ist es egal ob der Benutzer den SEAR^CH-Tag auf der Website oder in der TableView anklickt. Die folgende Funktion

```
override func prepareForSegue(segue: UIStoryboardSegue,
    sender: AnyObject?) {
    ...
}
```

erzeugt den PopOverView und liefert den angeklickten SEAR^CH-Tag mit. Dieser wird dann als Überschrift wie in Abbildung 1, dargestellt. In dem PopOverView wird oben

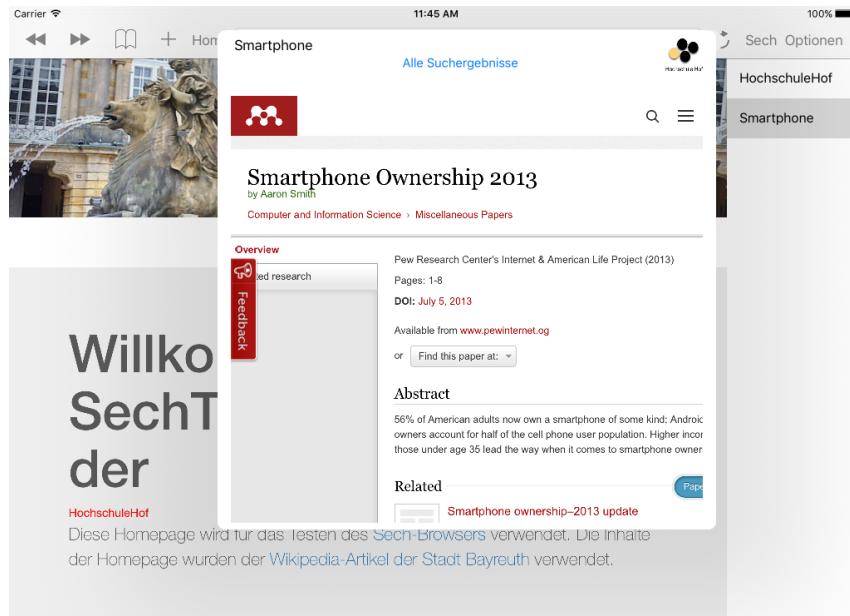


Abbildung 16.7.: Liste für angeklickten SEAR^CH-Tag

links der Titel des angeklickten SEAR^CH-Tags dargestellt. Daneben wird ein Button dargestellt, der eine weitere TableView enthält. Rechts oben wird noch das Logo des aktuellen Browsers angezeigt. Durch einen Klick auf den Button „Alle Suchergebnisse“ wird eine neue TableView geöffnet, die weitere Suchergebnisse zu dem aktuellen Suchbegriff enthält (vgl. Abb. 16.8). Zum Öffnen der Tabelle wurde folgende Funktion auf den Button gelegt

```
@IBAction func openTable(sender: AnyObject) {
    if let popover = popoverContent.popoverPresentationController {
        let viewForSource = sender as! UIView
```

```

popover.sourceView = viewForSource
popover.sourceRect = viewForSource.bounds
popoverContent.preferredContentSize = CGSizeMake(400,500)
}
self.presentViewController(popoverContent, animated: true,
completion: nil)
}

```

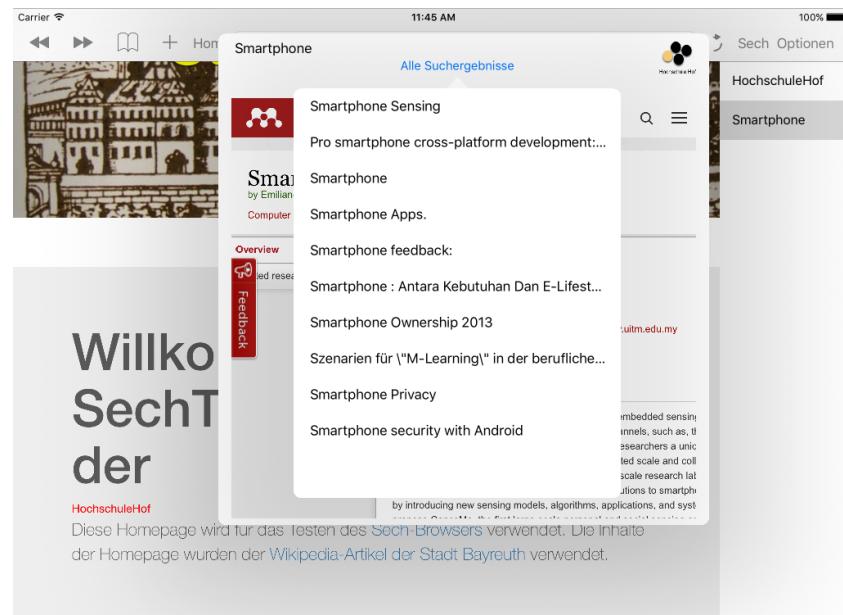


Abbildung 16.8.: Liste aller Suchergebnisse

Durch diese Funktion, hat der Benutzer die Möglichkeit sich unterschiedliche Ergebnisse zu einem **SEARCH**-Tag innerhalb eines Fensters anzeigen zu lassen. Durch einen Tap auf ein anderes Ergebnis in der Tabelle, lädt der darunter liegende Webview neu. Um das Popover Fenster wieder zu schließen, muss der Benutzer nur innerhalb des Views tippen. Dadurch wird wieder der normale WebView angezeigt.

Teil IV.

Test

17. Testen der Applikation

17.1. Einleitung

Zum Testen der Applikation SECH-Browser ist es erforderlich, die in einer exemplarisch erzeugten Webseite (www.sech-browser.de/tests) eingebetteten SEAR^CH-Tags zu überprüfen. Um idealerweise alle Problemfelder bzw. Fehler der App erkennen zu können, ist es erforderlich eine bestmögliche Abdeckung aller möglichen Testfälle zu gewährleisten.

17.2. Testfälle

Um eine bestmögliche Überprüfung der App zu erreichen, wurde eine systematische Implementierung aller denkbaren Testfälle unter <http://www.sech-browser.de/old-index.html> vorgenommen. Wie aus der Definition der SEAR^CH-Tags ersichtlich, besteht das SEAR^CH-Tag aus den drei Teilen:

- SEAR^CH-Head
- SEAR^CH-Section
- SEAR^CH-Link

Weiterhin kann jeder Teil des SEAR^CH-Tags bis zu 5 Attribute enthalten:

- topic
- type
- media-type
- provider
- licence

Um alle möglichen realisierten Kombinationen der Teile der SEAR^CH-Tags mit den zugehörigen Attributen aufrufen zu können, wurde eine übersichtliche Menüstruktur gewählt. Zusätzlich wurden im letzten Menüpunkt 'Fehler' noch syntaktische Fehler in den Tags eingebaut, wie Weglassen von Tag-Klammern sowie Weglassen von Anführungszeichen bei Attributnamen oder falsch geschriebene Attributnamen. Die Menü-Struktur der Testfälle ist nachfolgend dargestellt:

Menü – Testfälle

keine SEARCh-Tags

nur SEARCh-Head

1. mit allen Attributen
2. mit einzelnen Attributen
 - a) nur Topic
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - b) nur Type
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - c) nur mediaType
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - d) nur Provider
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - e) nur Licence
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
3. mit mehreren Attributen
 - a) Topic, Type
 - b) Topic, Type, mediaType
 - c) Topic, Type, mediaType, Provider

nur S**E**AR**C**H-Section

1. mit allen Attributen
2. mit einzelnen Attributen
 - a) nur Topic
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - b) nur Type
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - c) nur mediaType
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - d) nur Provider
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - e) nur Licence
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
3. mit mehreren Attributen
 - a) Topic, Type
 - b) Topic, Type, mediaType
 - c) Topic, Type, mediaType, Provider

nur S**E**AR**C**H-Link

1. mit allen Attributen
2. mit einzelnen Attributen
 - a) nur Topic
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - b) nur Type
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - c) nur mediaType
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - d) nur Provider
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
 - e) nur Licence
 - mit Umlauten
 - mit Leerzeichen
 - mit Sonderzeichen
 - mit Ziffern
3. mit mehreren Attributen
 - a) Topic, Type
 - b) Topic, Type, mediaType
 - c) Topic, Type, mediaType, Provider

SEAR^CH-Head und SEAR^CH-Section

1. Standard
2. mit Umlauten
3. mit Leerzeichen
4. mit Sonderzeichen
5. mit Ziffern

SEAR^CH-Head und SEAR^CH-Link

1. Standard
2. mit Umlauten
3. mit Leerzeichen
4. mit Sonderzeichen
5. mit Ziffern

SEAR^CH-Section und SEAR^CH-Link

1. Standard
2. mit Umlauten
3. mit Leerzeichen
4. mit Sonderzeichen
5. mit Ziffern

SEAR^CH-Head, SEAR^CH-Section, SEAR^CH-Link

1. Standard
2. mit Umlauten
3. mit Leerzeichen
4. mit Sonderzeichen
5. mit Ziffern

Fehler

nur **SEARCh-Head**

1. Anführungszeichen vorne fehlt
2. Anführungszeichen hinten fehlt
3. Tag-Klammer vorne fehlt
4. Tag-Klammer hinten fehlt

nur **SEARCh-Section**

1. Anführungszeichen vorne fehlt
2. Anführungszeichen hinten fehlt
3. Beginn Tag-Klammer vorne fehlt
4. Beginn Tag-Klammer hinten fehlt
5. Ende Tag-Klammer vorne fehlt
6. Ende Tag-Klammer hinten fehlt

nur **SEARCh-Link**

1. Anführungszeichen vorne fehlt
2. Anführungszeichen hinten fehlt
3. Beginn Tag-Klammer vorne fehlt
4. Beginn Tag-Klammer hinten fehlt
5. Ende Tag-Klammer vorne fehlt
6. Ende Tag-Klammer hinten fehlt

SEARCh-Head und SEARCh-Section

1. SEARCh-Head Topic Anführungszeichen vorne fehlt
2. SEARCh-Section Topic Anführungszeichen hinten fehlt
3. SEARCh-Head Tag-Klammer vorne fehlt
4. SEARCh-Section Beginn Tag-Klammer hinten fehlt
5. SEARCh-Section Ende Tag-Klammer hinten fehlt
6. SEARCh-Section nicht geschlossen

SEAR^CH-Head und SEAR^CH-Link

1. SEAR^CH-Head Topic Anführungszeichen vorne fehlt
2. SEAR^CH-Link Topic Anführungszeichen hinten fehlt
3. SEAR^CH-Head Tag-Klammer vorne fehlt
4. SEAR^CH-Link Beginn Tag-Klammer hinten fehlt
5. SEAR^CH-Link Ende Tag-Klammer hinten fehlt
6. SEAR^CH-Link nicht geschlossen

SEAR^CH-Section und SEAR^CH-Link

1. SEAR^CH-Section Topic Anführungszeichen vorne fehlt
2. SEAR^CH-Link Topic Anführungszeichen hinten fehlt
3. SEAR^CH-Section Tag-Klammer vorne fehlt
4. SEAR^CH-Section Ende Tag-Klammer hinten fehlt
5. SEAR^CH-Section nicht geschlossen
6. SEAR^CH-Link Beginn Tag-Klammer hinten fehlt
7. SEAR^CH-Link Ende Tag-Klammer hinten fehlt
8. SEAR^CH-Link nicht geschlossen

SEAR^CH-Head, SEAR^CH-Section, SEAR^CH-Link

1. SEAR^CH-Head falscher Attributname (topics statt topic)
2. SEAR^CH-Section falscher Attributname (provide statt provider)
3. SEAR^CH-Link falscher Attributname (license statt licence)
4. SEAR^CH-Head Tag-Klammer vorne fehlt
5. SEAR^CH-Section Tag-Klammer vorne fehlt
6. SEAR^CH-Section Ende Tag-Klammer hinten fehlt
7. SEAR^CH-Section nicht geschlossen
8. SEAR^CH-Link Beginn Tag-Klammer vorne fehlt
9. SEAR^CH-Link Ende Tag-Klammer hinten fehlt
10. SEAR^CH-Link nicht geschlossen

17.3. Auswertung der Testfälle

Sobald nur ein `SEARCH`-Head vorhanden ist, wird dieser nicht ausgewertet und es werden keine `SEARCH`-Links angezeigt.

Sobald nur eine oder mehrere `SEARCH`-Sections vorhanden sind, werden diese nicht ausgewertet und es werden keine `SEARCH`-Links angezeigt.

Sobald `SEARCH`-Links vorhanden sind, werden diese entsprechend ausgewertet und sie werden korrekt in der TableView der `SEARCH`-Tags angezeigt.

Es treten Fehler auf, sobald im `SEARCH`-Link die Anführungszeichen fehlen, denn es werden trotzdem entsprechende `SEARCH`-Links gefunden und fehlerhaft angezeigt.

Sobald `SEARCH`-Tag-Klammern fehlen, können diese nicht ausgewertet werden und werden nicht angezeigt.

Abbildungsverzeichnis

3.1. Beispiel für Informationsanfrage an den EEXCESS PP–Service	8
3.2. Zeitplanung für das SECH-Browser Projekt	10
7.1. Use Case des Rankings	18
7.2. Menüführung des Browsers (Navigation)	19
7.3. Lesezeichenverwaltung des Browsers	19
7.4. URL Navigation und SECH-Tabelle des Browsers	20
7.5. Use Case der Browser Navigation	20
7.6. Use Case der Browser Lesezeichen	21
7.7. Use Case der Browser Einstellungen	21
7.8. Interaktion zwischen Webkit und SEARCh-Tag	22
8.1. Ablauf des Rankings	24
8.2. Detailablauf Ranking	25
9.1. Sequenzdiagramm Adressbar	29
9.2. Lesezeichen hinzufügen	30
9.3. Lesezeichen bearbeiten	31
9.4. Lesezeichen anzeigen	32
9.5. Lesezeichen löschen	32
12.1. Ablauf der Erstellung eines SECH-Objektes	38
13.1. Aufbau des Prototypen	39
15.1. Sequenzdiagramm des Rankings	45
15.2. Klassendiagramm des Rankings	46
15.3. Sequenzdiagramm einer Serveranfrage	47
15.4. Klassendiagramm einer Serveranfrage	48
16.1. Browser im Hochformat	50
16.2. Anzahl der SEARCh-Tags	51
16.3. SEARCh-Tabelle aufgeklappt	52
16.4. SEARCh-Tabelle versteckt	52
16.5. Hochformat des Browsers	54
16.6. Querformat des Browsers	54
16.7. Liste für angeklickten SEARCh-Tag	55
16.8. Liste aller Suchergebnisse	56

Tabellenverzeichnis

6.1. Liste der Partner	16
6.2. Liste der Provider	17
10.1. mögliche Altersbereiche	33