

# **Dokumentation**

## **Praktikum Software Entwicklung**

Dozent: Prof. Dr. Richard Göbel

Beteiligte Studenten:

Christoph Piechula

Christoph Cwelich

Christopher Pahl

Eduard Schneider

Florian Bauer

Sabrina Biersack

7. Mai 2012

# Inhaltsverzeichnis

<b>I. Spezifikation</b>	<b>3</b>
1. Übersicht	4
2. Datenmodell der Metadaten	6
3. Anforderungen	7
3.1. Crawlermodul . . . . .	7
3.1.1. Steuerung . . . . .	7
3.1.2. Ausführung des Crawlvorgangs . . . . .	8
3.2. Filtermodule . . . . .	9
3.3. Dateiarchiv . . . . .	10
3.3.1. Versionierung . . . . .	11
3.4. Programmierschnittstelle - Java-Client . . . . .	11
3.5. Server . . . . .	13
3.6. XML-Dateien . . . . .	14
3.7. Datenbank . . . . .	14
<b>4. Entwicklungsumgebung</b>	<b>16</b>
4.1. Programmiersprachen . . . . .	16
4.1.1. Python . . . . .	16
4.1.2. Java . . . . .	16
4.2. Dokumentation . . . . .	16
4.3. Teamsynchronisation . . . . .	16
4.4. Sprache . . . . .	17

**Teil I.**

**Spezifikation**

# 1. Übersicht

Über einen konfigurierbaren Crawler können HTML-Inhalte von Webseiten bis zu einer bestimmten Tiefe aus dem Netz in ein Archiv geladen werden. Da dies parallelisiert erfolgen soll, müssen die Daten nach dem Herunterladen von temporären Verzeichnissen in das gemeinsame Archivverzeichnis synchronisiert werden. Der aus der URL extrahierte Pfad der HTML-Dateien wird dabei auf das Archiv abgebildet, wobei jede HTML-Datei in einen eigenen Archivordner verschoben wird.

Beim Crawlvorgang werden zusätzlich Metadaten der HTML-Seiten erstellt. Diese werden in einer Datenbank und als XML-Datei im jeweiligen HTML-Ordner gespeichert. Die Datenbank soll dabei wieder aus den XML-Daten rekonstruierbar sein.

Außerdem sollen Filter eingehängt werden können, die bereits in den TMP-Ordern ungültige Dateien löschen. Über eine Java-Schnittstelle kann anschließend wieder auf die Daten zugegriffen werden. Hierzu müssen sich Clients über eine vorgegebene Schnittstelle beim Textarchiv anmelden. Die Clients werden anschließend bei Änderungen oder neuen Einträgen benachrichtigt und können sich selbstständig über die Schnittstelle einen beliebigen Stand der Daten herunterladen und an Analysetools weitergeben. Dabei sollen auch neue Dateien den Archivordnern hinzugefügt werden können. Ebenso sollen die oben genannten XML-Daten um neue Nodes erweiterbar sein. Zu Vorführzwecken wird ein Testanalysetool erstellt.

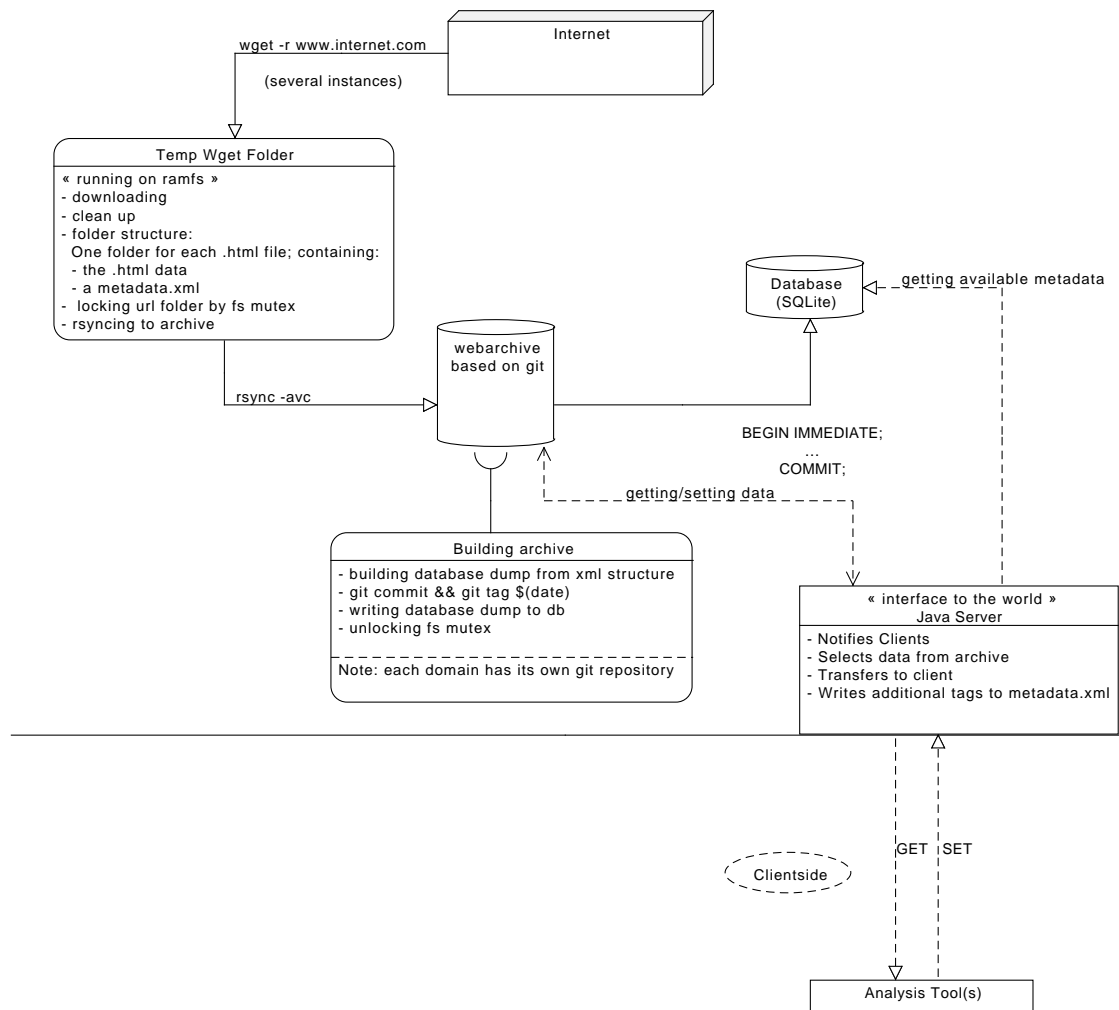


Abbildung 1.1.: Diagramm: Grundlegendes Design

## 2. Datenmodell der Metadaten

An dieser Stelle wird bereits ein vereinfachtes Datenmodell festgelegt, welches später auf Datenbank und XML-Daten umgesetzt wird und zentrale Metadaten über HTML-Seiten erfassen soll.

Für jede HTML-Datei wird dabei ein solcher Satz von Metadaten angelegt. Die Metadaten sollen vor allen Dingen Such- und Sortieraufgaben erleichtern.

Name	Datentyp	Beschreibung
url	String	Original URL der HTML-Datei
title	String	Soweit vorhanden wird der Titel der Seite gespeichert
path	String	Dateipfad zum Archivordner der Datei im Webarchiv
crawlTime	Timestamp oder Integer (UTC in ms)	Beginn des Crawls
commitTag	String Form:	Der Committag dient zum Wiederauffinden in der Versionsverwaltung. Der Tag setzt sich aus Domainnamme und dem Zeitpunkt des Crawlvorgangs zusammen. Form: commitTag := ;domainName;crawlTime; crawlTime := ;YYYY-MM-DD HH:MM:SS;

Tabelle 2.1.: Metadaten

## 3. Anforderungen

Im folgenden sind die Anforderungen an die Software spezifiziert. Soweit möglich wurden die Anforderungen schon in einzelne Komponenten und Module gegliedert. Eine grobe Übersicht gibt auch Diagramm im Abschnitt 1.

Alle Anforderungen werden mit einer Kennnummer in der Form

*/ < Gruppenprefix > . < Anforderungsnummer > [. < Unternummer > ]/*

gekennzeichnet, damit diese später wieder referenziert werden können.

### 3.1. Crawlermodul

Dieses Modul soll als eigenständiger Prozess laufen und in regelmäßigen Abständen Crawlvorgänge starten und die Daten in das Archiv schreiben.

#### 3.1.1. Steuerung

**/Cr-1/ Config-file** Die Steuerung des bzw. der Crawler erfolgt über eine Config-Datei. Ist die Config-Datei nicht vorhanden oder kann sie nicht gelesen werden, so werden hart codierte Werte verwendet. Es können folgende Parameter eingestellt werden:

**/Cr-1.1/ Suchtiefe** Suchtiefe bis zu der Links gefolgt werden soll.

**/Cr-1.2/ Crawlintervall** Zeitintervalle zwischen den Crawlvorgängen.

**/Cr-1.3/ Maximale Instanzen** maximale Anzahl der gleichzeitig gestarteten Crawlerinstanzen.

**/Cr-1.4/ Filtereinstellungen** Liste von Verwendeten Ausschlussfiltern und deren Modulpfad.

**/Cr-2/ Kommandozeileninterface** Optional können beim Start mittels Kommandozeile zusätzliche Parameter übergeben werden.

**/Cr-2.1/ Überschreiben** Werte aus dem Config-file werden damit überschrieben.

**/Cr-2.2/ URL-Liste** Es kann eine Liste von URLs übergeben werden, die als Startpunkte für die Crawler verwendet werden sollen. Die Übergabe von mindestens einem Element ist aber immer notwendig.

**/Cr-2.3/ DB-Recovery** Es kann ein Datenbank-Recovery erzwungen werden. Siehe auch /Db-6/

### 3.1.2. Ausführung des Crawlvorgangs

**/Cr-3/ Start in Intervallen** Der Crawlvorgang wird immer wieder nach einem gemäß ?? festgelegtem Interval neu in Gang gesetzt.

**/Cr-4/ Zeitpunkt speichern** Der Startzeitpunkt wird sofort nach dem Beginn des Crawlvorgangs zwischengespeichert. Siehe auch Abschnitt 2.

**/Cr-5/ Parallelität** Die Ausführung der Crawlvorgänge soll parallel durchgeführt werden.

**/Cr-6/ URL-Queue** Alle aus /Cr-2.2/ werden in eine Warteschlange geschrieben.

**/Cr-7/ Instanziierung** Für die Crawlerinstanzen wird ein externes Tool verwendet: WGET

**/Cr-7.1/ Temporäre Ordner anlegen** Für jede Crawlerinstanz wird ein eigenes temporäres Verzeichnis zum Speichern der Downloads angelegt. Im folgenden mit TMP abgekürzt.

**/Cr-7.2/ Start** Es werden solange Crawlerinstanzen mit URLs als Startpunkt aus der URL-Queue erzeugt, bis diese leer ist.

**/Cr-7.3/ maximale Instanzen** Wird die Obergrenze an maximal erzeugbaren Instanzen erreicht (siehe /Cr-1.3/), wird immer solange mit dem Erzeugen neuer Instanzen gewartet, bis die nächste Crawlerinstanz fertig ist.

**/Cr-7.4/ Domain-Überschneidungen** Um Überschneidungen zu vermeiden, werden Domainnamen anderer Instanzen ausgeschlossen.

**/Cr-8/ Crawlen** Jede gestartete Instanz beginnt nun den Crawlvorgang.

**/Cr-8.1/ Herunterladen** Jede Instanz kopiert die heruntergeladene HTML-Dateien der Seite in ein temporäres Verzeichnis je Crawlerinstanz.

**/Cr-8.2/ Ordnerstruktur** Dabei wird die online vorhandene URL-Pfadstruktur der HTML-Dateien auf das Dateisystem abgebildet. Je Domain wird dadurch ein Hauptverzeichnis im TMP-Ordner erzeugt.

**/Cr-9/ Filterung** Für alle HTML-Dateien wird eine Liste von Filtern durchlaufen. Jeder Filter prüft, ob die HTML-Datei behalten oder verworfen werden soll. Verworfenen Dateien werden sogleich gelöscht.

**/Cr-10/ Bereinigung** In diesem Teil werden die heruntergeladenen Ordner bereinigt, also leere Ordner gelöscht.



**/Cr-11/ Normalisierung /Cr-11.1/ Datei umbenennen** Das HTML-file in "data.html" umbenannt.

**/Cr-11.2/ Archivordner erzeugen** Es wird ein Archivordner (im Folgenden kurz ARC genannt) mit dem Namen des HTML-files (inklusive Dateiendung) erzeugt.

**/Cr-11.3/ Datei verschieben** Die data-Datei wird in den soeben erzeugte ARC verschoben.

**/Cr-12/ Extraktion der Metadaten** In diesem Vorgang werden die im Datenmodell (Abschnitt 2) definierten Metadaten aus dem HTML extrahiert und zwischengespeichert.

**/Cr-12.1/ url** Die Original-URL wird aus dem aktuellen Ordnerpfad abgeleitet.

**/Cr-12.2/ title** muss der Titel aus dem Inhalt geparkt werden.

**/Cr-12.3/ path** Es wird der aktuelle absolute Ordnerpfad auf das ARC gespeichert.

**/Cr-12.4/ crawlTime** Es wird der Beginn des Crawlvorgangs (/Cr-4/) als Timestamp bzw. als Ganzzahl in ms gespeichert.

**/Cr-12.5/ commitTag** Der commitTag wird aus Domainordnername und crawlTime (im Textformat) zusammen.

**/Cr-13/ Erzeugung von XML-Dateien** Die zwischengespeicherten Metadaten werden in XML-Metadateien (siehe auch Abschnitt 3.6) geschrieben und jeweils im zugehörigen ARC gespeichert.

**/Cr-14/ Synchronisation** Die vorbereiteten Daten in den TMP-Ordern werden nun in das vorhandene Archiv synchronisiert (mit RSYNC). (Siehe auch Abschnitt 3.3)

**/Cr-14.1/ Update** Veraltete ARCs werden komplett durch die neuen ersetzt (und dadurch von der Versionierung in einen älteren Versionstand verschoben).

**/Cr-14.2/ Nicht vorhandene Ordner** ARCs, welche im TMP-Ordner aber nicht im Webarchiv vorhanden sind, werden komplett aus dem Archiv gelöscht (und landen ebenfalls in der Versionierung)

**/Cr-15/ Datenbankaktualisierung** Die Datenbank wird nun mithilfe der zwischengespeicherten Metadaten aktualisiert.

## 3.2. Filtermodule

**/Fi-1/ Schnittstelle** Alle Filtermodule sollen eine vorgegebene Funktionsschnittstelle erfüllen, welcher ein Dateipfad übergeben werden kann und einen Wahrheitswert zurückgibt.

**/Fi-2/ Konfiguration** Ein Filtermodul kann über eigenes Config-file verfügen, welches im Verzeichnis der anderen Config-dateien gespeichert werden soll.

**/Fi-3/ HTML Überprüfung** Ein Filter soll ein gegebenes File überprüfen und einen Wahrheitswert zurückgeben, ob dieses behalten oder verworfen werden soll. Die Prüfmethode ist dabei vom einzelnen Zweck des Filters abhängig.

**/Fi-4/ Implementierung Testfilter: Werbefilter**

### 3.3. Dateiarchiv

Das Dateiarchiv stellt den zentralen Speicherort aller Quell-, Meta-XML und sonstiger hinzugefügter Daten dar. Die Dateistruktur wird bereits vom Crawlvorgang vorgegeben, wird hier aber nochmal kurz erläutert:

- Auf der obersten Ebene stehen die Domain-Ordner.
- Darunter wird die von der URL abgebildeten Struktur innerhalb der Domain nachgebildet.
- Die einzelnen HTML-Dateien werden durch ARC-Ordner ersetzt, bzw. in diese verschoben.
- Jedes ARC enthält die Quell-HTML-Datei, die aber in data.html unbenannt wurde. sowie die dazugehörige XML-Datei (data.xml). Ein ARC kann aber auch weitere Dateien enthalten, welche nach dem Crawlen hinzugefügt werden können.

**/Ar-1/ Berechtigungen** Generell dürfen keine Änderungen an bestehenden Dateien durchgeführt werden (Ausnahme: siehe /Xm-3/).

**/Ar-1.1/ Externe Benutzer** Von externen Nutzern (Java-Clients) dürfen nur Dateien in *aktuelle* ARChinzugefügt werden.

**/Ar-1.2/ Crawler** Crawler dürfen neue ARC-Ordner hinzufügen oder alte löschen bzw. ersetzen, wobei immer das gesamte ARC ersetzt wird.

**/Ar-2/ Synchronisation** Um gleichzeitige Zugriffe auf Dateien zu verhindern, muss Synchronisiert werden.

**/Ar-2.1/ Sicherungsmechanismus** Zum Schutz der kritischen Stellen wird mit Locks bzw. Mutexen gearbeitet.

**/Ar-2.2/ Umfang des Schutzes** Es wird immer der gesamte Domain-Ordner gesperrt.

**/Ar-2.3/ Lesezugriffe** Es muss vor jedem Lesezugriff ein Lock gesetzt und sofort wieder entfernt werden, nachdem die Daten ausgelesen wurden.

**/Ar-2.4/ Schreibzugriffe** Es muss vor jedem Schreibvorgang ein Lock gesetzt und nach dem Beenden des Vorgangs wieder entfernt werden.

**/Ar-3/ Dateisystem** Beim darunterliegenden Dateisystem wird von einem vorhandenen Unix-Filesystem ausgegangen.

**/Ar-4/ Komprimierung** Eine explizite Dateikomprimierung wird nicht vorgesehen, ist aber zum Teil schon durch die Versionierung gegeben, da alte Revisionen gepackt abgelegt werden.

### 3.3.1. Versionierung

**/Ar-5/ Versionsverwaltungssystem** Es wird das Versionsverwaltungssystem GIT verwendet.

**/Ar-6/ Domainversionierung** Jeder Domain-Ordner wird über eine dezentrale Versionsverwaltung (git) versioniert. Damit ist auch das Wiederherstellen älterer Versionen möglich.

**/Ar-7/ Hinzufügen** Beim Hinzufügen von neuen Dateien müssen diese der Versionsverwaltung bekanntgemacht werden (git add).

**/Ar-8/ Commit** Änderungen müssen stets mit einem Commit bestätigt werden. Dabei wird zur Identifikation immer der in Abschnitt 2 definierte commitTag verwendet. Dabei ist zu beachten:

**Während des Crawlvorgangs** Das Erzeugen neuer commitTags während des Crawlvorgangs ist im Abschnitt 3.1.2 beschrieben

**/Ar-8.1/ Nach dem Crawlen** Beim nachträglichen Hinzufügen von Dateien wird der alte commitTag wiederverwendet.

**/Ar-9/ Ändern und Löschen** Beim Überschreiben und Löschen von Dateien müssen keine besonderen Vorkehrungen getroffen werden. Diese werden von der Versionierung in eine tiefere Versionsebene verschoben.

**/Ar-10/ Schreiben in veraltete Verzeichnisse** Sollte in Ausnahmefällen in veraltete Verzeichnisse geschrieben werden, zum Beispiel weil die Daten von den Crawlern während einer Analyse geändert wurden, dann werden die Daten verworfen und der Schreiber muss benachrichtigt werden. (z.B. mittels einer Exception)

## 3.4. Programmierschnittstelle - Java-Client

Diese Schnittstelle soll die Anbindung der Analysemethoden ermöglichen und macht gleichzeitig einen Zugriff über das Netzwerk möglich.

**/CI-1/ Konfiguration** Es sind die IP und der Port des Servers in einer Config-Datei zu hinterlegen.

**/CI-2/ Client-API** Für Benutzer des Clients wird eine Programmierschnittstelle in Java zur Verfügung gestellt. Die API umfasst dabei folgende Schnittstellen:

**/CI-2.1/ MetaData** Grundlegende Methoden einer Metadatenklasse. Neben den Metadateninformationen soll diese Klasse auch als Schlüsselement zum Zugriff auf die Archivordner und XML-Dateien dienen.

- /CI-2.2/ WebarchiveClient** Zentrale Schnittstelle zum Zugriff auf das Webarchiv, Details siehe unten.
- /CI-2.3/ Observer** Implementierungen dieses Observers können sich beim Client anmelden, um über Änderungen informiert zu werden. Die Schnittstelle enthält eine Methode um Update-Informationen zu erhalten.
- /CI-3/ Registrierung am Server** Alle aktiven Java-Clients werden beim Server gespeichert, um diese über Änderungen informieren zu können. Beim Abmelden oder Beenden muss ein Client aus dieser Registrierung gelöscht werden.
- /CI-4/ Observerregistrierung** Benutzer des Clients können sich mittels o.g. Schnittstelle beim Client als Observer an- und abmelden.
- /CI-5/ Benachrichtigungen** Vom Server erhaltene Änderungen (in Form von commitTags) werden an die Observer weitergereicht.
- /CI-6/ Datenbankabfragen** Es sollen auch vorbereitete SQL-Statements an den Server geschickt werden können. Die SQL-Abfrage wird soweit vorbereitet, dass nur noch ein SQL-Bedingungsausdruck für die WHERE-Klausel angegeben werden muss. Optional soll auch eine ORDER-BY-Klausel im selben Stil angegeben werden können. Als return-Wert wird eine Liste von Metadatenobjekten zurückgegeben.
- /CI-7/ Datei Listing** Mittels der Metadatenobjekte kann man sich über eine gesonderte Anfrage eine Auflistung über den Inhalt eines ARC-Ordners zurückgeben lassen.
- /CI-8/ Datei Lesen** Mittels Metadatenobjekt und Dateipfadangabe wird ein Dateistream zum Lesen zurückgegeben.
- /CI-9/ Datei Schreiben** Mittels Metadatenobjekt und Dateipfadangabe wird ein Dateistream zum Schreiben zurückgegeben. Wie in /Ar-1/ beschrieben, dürfen dabei keine Dateien überschrieben werden.
- /CI-10/ Auslesen von zusätzlichen Tags** Durch Übergabe eines Metadata-objekts und eines Tagnamens an eine get-Methode wird ein passender XML-Node herausgesucht.
- /CI-11/ Erweiterung der XML-Daten** Mittels einer set-Methode, die Namen und Inhalt des Tags als Parameter erhält, können neue Tags zur XML-Datei hinzugefügt werden.
- /CI-12/ Test Analysetool** Zur Demonstrations- und Testzwecken der Java-Clientschnittstelle wird ein Analysetool erstellt, welches die Wörter im HTML zählt und das Ergebnis im Archiv als Datei sowie im XML als zusätzliches Tag speichert.

## 3.5. Server

**/Sv-1/ Konfiguration** Es ist der Port des Servers in einer Config-Datei zu hinterlegen. Desweiteren werden darin auch alle Pfade zu externe Ressourcen hinterlegt.

**/Sv-2/ Client-Server Kommunikation** Es muss ein Nachrichtensystem zwischen dem Client und dem Server implementiert werden. Über bestimmte Kennungen (z.B. über enums) ist der Inhalt einer Nachricht zu kennzeichnen. Dabei müssen folgende Informationen ausgetauscht werden können:

**/Sv-2.1/ Austausch von Stream-Daten** Zwischen C. und S. müssen Daten in Form von Streams verschickt werden können. Diese müssen in geeigneter Form bei der Übertragung gepuffert werden.

**/Sv-2.2/ Exceptions** Vom Clientanwender verursachte Exceptions werden an diesen weitergeleitet und im Client erneut geworfen.

**/Sv-2.3/ Datenbankabfragen** Datenbankabfragen werden in Form von SQL vom Client geschickt und diesem in Form von Metadaten-Objekten beantwortet.

**/Sv-2.4/ Änderungen im Archiv** Diese Nachrichtenform enthält Informationen für Client, welche Änderungen im Archiv betreffen.

**/Sv-3/ Klienten registrieren** Der Server hält eine Liste von angemeldeten Java-Clients und verwaltet die Verbindungen der Klienten.

**/Sv-4/ Klienten entfernen** Clients werden aus der Registrierung entfernt bei Verbindungsverlust oder beim Beenden der Clients.

**/Sv-5/ Update-Notifier** Hierfür speichert er sich den Zeitpunkt der letzten Update-Suche und vergleicht ihn mit dem Datum der Datensätze in der Datenbank. Werden Änderungen gefunden, werden das Datum und der Commit-Tag zwischengepuffert.

**/Sv-5.1/ Konfiguration** Der Benachrichtigungsintervall muss in einem Configfile gespeichert werden.

**/Sv-5.2/ Thread** Der Update-Notifier ist ein eigens laufender Thread des Servers.

**/Sv-5.3/ Suchintervalle** Der Notifier in dem vorgegebenen Intervall (z.B. stündlich) in der Datenbank, ob neue Commit-Tags vorhanden sind.

**/Sv-5.4/ Zeitpunkt der letzten Suche** Es wird immer der Zeitpunkt der letzten Suche gespeichert, um die Datenbank auf aktuelle Daten zu prüfen.

**/Sv-5.5/ Klienten informieren** Sollte die Suche Ergebnisse zutage gefördert haben, wird eine Liste mit den neuen commitTags an die registrierten Clients geschickt. Siehe auch /Cl-5/

## 3.6. XML-Dateien

**/Xm-1/ Inhalt** Die XML-Dateien enthalten in ihrem Wurzelknoten eine Meta- und einen Datenknoten.

**/Xm-1.1/ Metaknoten** Der Metaknoten ist für die in 2 beschriebenen Metadaten reserviert.

**/Xm-1.2/ Datenknoten** Der Datenknoten ist anfangs leer und kann von Benutzern um weitere Knoten erweitert werden.

**/Xm-2/ Erstellung eines XML-Schemas** Für die Validierung der XML-Daten muss eine XML-Schema ausgearbeitet werden.

**/Xm-3/ Erweiterbarkeit des Datenelements** Wie oben beschrieben ist beim Design auf Erweiterbarkeit zu achten.

**/Xm-3.1/ Name eines Elements** Der Name eines hinzugefügten Elements ist frei wählbar, darf aber nur einmal auf der Ebene unter dem Datenknoten vorkommen.

**/Xm-3.2/ Inhalt** Struktur und Inhalt der hinzugefügten Elemente ist frei wählbar.

**/Xm-3.3/ Erweiterung des Schemas** Bei Erweiterung des Datenknotens ist das Schema auch entsprechend zu erweitern.

**/Xm-4/ Validierung** Eine Validierung ist dann durchzuführen, nachdem eine XML-Datei erweitert worden ist. Dabei auftretende Fehler werden in eine Log-Datei geschrieben.

**/Xm-5/ Schreibschutz** Es dürfen nur neue Knoten hinzugefügt werden dürfen und

**/Xm-5.1/ Kein Überschreiben** Es dürfen keine vorhandenen Elemente überschrieben oder geändert werden können.

**/Xm-5.2/ Metaknoten gesperrt** Der Metaknoten darf nicht geändert werden.

## 3.7. Datenbank

**/Db-1/ Konfiguration** Die Datenbank wird als CREATE-TABLE-statement in einer SQL-Datei gespeichert.

**/Db-2/ SQL in externen Dateien** Vorbereitete SQL-Statements werden in je einer SQL-Datei in einem Ordner gespeichert. Inlinedefinitionen im Quellcode sind also zu vermeiden.

**/Db-3/ Inhalt** Es werden die Speicherstände aller HTML-Dateien und Versionen im Archiv festgehalten. Die gespeicherten Daten entsprechen den Knoten des Abschnitts 2.

**/Db-4/ Erweiterbarkeit** Eine Möglichkeit zum dynamischen Erweitern der Datenbank ist nicht vorgesehen.

**/Db-5/ Berechtigungen /Db-5.1/ Schreiben** Schreibrechte werden nur dem Crawlmodul erteilt.

**/Db-5.2/ Lesen** Gelesen kann falls notwendig von allen Komponenten werden. Wobei externe Benutzer über die Schnittstellen der Clients SELECT-statements absetzen können.

**/Db-6/ Wiederherstellung** Sollte die Datenbank beschädigt oder geändert werden, dann soll diese wieder aus den im Archiv vorhandenen XML-Metadaten aller Versionen rekonstruiert werden können.

## 4. Entwicklungsumgebung

### 4.1. Programmiersprachen

Es werden die Sprachen Python und Java benutzt.

#### 4.1.1. Python

Python in der Version 2.7 oder 3 wird für die systemnahen Teile verwendet:

- Das gesamte Crawlermodul
- Die Filtermodule
- Der Zugriff auf das Dateisystem (Archiv) und die notwendige Ordnersynchronisation.

Python enthält bereits leistungsfähige Libraries für den Zugriff auf das Dateisystem, SQLite und die Versionsverwaltung git.

#### 4.1.2. Java

Die Client-Server Architektur der oben genannten Programmierschnittstelle werden mit Java 1.7 umgesetzt. Hierzu ist Java besonders geeignet und es ist gewährleistet, eine an der Hochschule Hof allgemein verständliche Schnittstelle zu schaffen.

### 4.2. Dokumentation

Die Dokumentation wird in Latex als ein fortlaufendes Gesamtdokument erstellt, welches je nach Phase um weitere Teile erweitert wird.

### 4.3. Teamsynchronisation

Dokumente und Quellcode werden über ein gemeinsames Repository auf github.com synchronisiert.



## 4.4. Sprache

Die Sprache der Dokumentation ist Deutsch wobei natürlich geläufige Fremdwörter enthalten sind. Quellcode, Kommentare und daraus abgeleitete APIs (Javadoc und Sphinx) sind in Englisch zu verfassen.