

Projektdokumentation

Entwicklung des SmartArchives

Python-Programm zur Speicherung von HTML-Codes von Webseiten



Abschlussprüfung Jahr 2024

Prüfungsteilnehmer

Name: Luka
Adresse: Berlin
E-Mail:
Ausbildungsberuf: Fachinformatiker für Anwendungsentwicklung
Azubi-Nummer:

Prüfungsausschuss:

Ausbildungsbetrieb

Firma:
Adresse:
E-Mail:

Praktikumsbetrieb

Firma:
Adresse:
E-Mail:

Inhaltverzeichnis

1	Einleitung.....	1
1.1	Projektumfeld.....	1
1.2	Problemanalyse.....	2
1.3	Projektschnittstellen.....	2
2	Projektdefinition.....	2
2.1	IST Analyse.....	2
2.2	SOLL-Konzept.....	3
2.3	Projektziele.....	3
2.4	Projektabgrenzung.....	3
3	Projektplanung.....	4
3.1	Vorgehensmodell	4
3.2	Projektstrukturplan.....	4
3.3	Zeitplan.....	4
3.4	Gant-Diagramm.....	4
3.5	Kostenplan.....	4
3.6	Qualitätssichernde Maßnahmen.....	5
3.7	Entwürfe.....	5
3.7.1	UML-Diagramme	5
3.7.2	Flussdiagramme.....	6
3.7.3	Klassen Diagramm	8
4	Durchführung.....	9
4.1	Implementierung.....	9
4.1.1	Software-Architektur.....	9
4.1.2	Implementierung des Erhaltens von HTML-Code	9
4.1.3	Implementierung der Erstellung von der URL-Liste.....	10
4.1.4	Implementierung der HTML-Dateierstellung.....	12
4.2	Qualitätskontrolle.....	12
4.3	Außerplanmäßige Änderungen.....	13
5	Abschlussphase.....	13
5.1	Projektübergabe.....	13
5.2	Fachlicher Soll-Ist-Vergleich.....	13
5.3	Zeitlicher Soll-Ist-Vergleich.....	13
5.4	Fazit.....	14
5.4.1	Zeitlicher Fazit.....	14
5.4.2	Sachlicher Fazit.....	14
5.4.3	Persönlicher Fazit.....	14
5.5	Geplante Präsentationsmittel.....	14
6	Anhang.....	i
6.1	Abbildungen.....	i
6.2	Tabellen	vii
7	Quellen.....	xi

Abbildungsverzeichnis

Abbildung 1 Projektstrukturplan	i
Abbildung 2 Use-Case Diagramm	6
Abbildung 3 Flussdiagramm des gesamten Programms	ii
Abbildung 4 Flussdiagramm zur Dateierstellung	ii
Abbildung 5 Flussdiagramm zur rekursiven Durchsuchung der Webseite	iii
Abbildung 6 Klassen Diagramm	8
Abbildung 7 Bildschirm Foto 1: HTML-Datei, Beginn von der Hauptseite	iv
Abbildung 8 Bildschirm Foto 2: HTML-Datei, Beginn von der 2. Seite	v
Abbildung 9 Der korrigierte Quellcode der get_html_code-Funktion	vi
Abbildung 10 Der ursprüngliche Quellcode der get_html_code-Funktion	vi

Tabellenverzeichnis

Tabelle 1 Zeitplan grob	4
Tabelle 2 Zeitplan detailliert	vii
Tabelle 3 Gantt-Diagramm	viii
Tabelle 4 Kostenplan	ix
Tabelle 5 Qualitätssicherungsplan	5
Tabelle 6 Qualitätskontrolle	13
Tabelle 7 Zeitlicher Soll-Ist-Vergleich	x

1. Einleitung

1.1 Projektumfeld

Das Projekt wurde am **XXXXXXXXXXXXXX** im **XX**
XXXXXXXXXXXXXXXXXXXXXXXXXXXX umgesetzt. **XXXXXXXXXXXXXXXXXXXX** ist ein Zusammenschluss aller
 Universitäts- und öffentlichen Bibliotheken sowie zahlreicher Spezialbibliotheken in Berlin und
 Brandenburg. **XXXXXXXXXXXX** stellt in Zusammenarbeit mit GLAM-Einrichtungen (Galerien,
 Bibliotheken, Archiven und Museen) einen offenen und bequemen Zugang zu Daten, Codes, Methoden
 und Ergebnissen wissenschaftlicher Forschung sowie zur Digitalisierung und Erhaltung unseres
 kulturellen Erbes sicher.

In der Welt der Informationstechnologie besteht eines der dringendsten Probleme darin, die Zugänglichkeit und Relevanz von Inhalten aufrechtzuerhalten, die auf alten Websites veröffentlicht wurden, die Materialien für Kongresse, Konferenzen und Seminare gewidmet waren. Obwohl diese Websites nur selten besucht werden, ist eine laufende Zahlung erforderlich, um sie online zu halten. Darüber hinaus können sich schnell ändernde Technologien dazu führen, dass die Inhalte im Laufe der Zeit nicht mehr verfügbar sind, obwohl die Seiten statisch sind. In diesem Zusammenhang stellt sich die Aufgabe, Methoden zur Speicherung von Webseiten in Form von PDF- oder HTML-Dateien auf lokalen Medien mithilfe spezieller Softwaretools zu entwickeln. Dieser Speicheransatz minimiert die Kosten für die Pflege von Inhalten. Die Formate sind weit verbreitet und werden von verschiedenen Anwendungen unterstützt, was einen relativ stabilen Zugriff auf die Inhalte ermöglicht. Dieser Ansatz hat Vorteile sowohl für Forscher, die die Originaldaten für die Analyse aufbewahren, als auch für die Gesellschaft als Ganzes, indem er den Zugang zu historischen Informationen und Wissen langfristig sicherstellt.

Um ein Programm zum Speichern von HTML-Code statischer Seiten in Python zu erstellen, wird vorgeschlagen, nur die in seiner Basisverteilung verfügbaren Standardbibliotheken zu verwenden. Als Open-Source-Sprache fallen für Python bei der Entwicklung und Nutzung von Software keine Lizenzgebühren an. Die Standardbibliotheken von Python umfassen viele Module für Netzwerk, Dateien und Datenverarbeitung, sodass die Funktionalität zum Speichern von HTML-Code effektiv implementiert werden kann. Dieser Ansatz gewährleistet nicht nur die Zuverlässigkeit und Sicherheit des Programms, sondern minimiert auch die Abhängigkeit von Komponenten Dritter und die Kosten

für deren Support. Dies macht eine Python-basierte Lösung unter Verwendung von Standardbibliotheken im Hinblick auf Ressourceneinsparungen und Projektnachhaltigkeit attraktiv.

1.2 Problemanalyse

Um die Aufgabe zu spezifizieren, wurden mehrere Hauptschritte identifiziert: Lesen des HTML-Codes der Hauptseite der Website, rekursives Verfolgen von Links zu untergeordneten Seiten und Lesen von Links von diesen sowie anschließendes Speichern aller HTML-Codes lokal auf dem Computer. Moderne Technologien bieten viele vorgefertigte Lösungen für solche Aufgaben, wie beispielsweise die Bibliothek `requests`, die eine praktische Schnittstelle für HTTP-Anfragen bietet, `PyWebCopy` und `BeautifulSoup` in Python, die den Prozess des rekursiven Herunterladens und Speicherns von Webseiten automatisieren. Allerdings ist zu beachten, dass diese Lösungen nicht kostenlos sind und Lizenzgebühren oder Abonnements erfordern. Dies macht die Entwicklung eines eigenen Programms auf Basis von Standard-Python-Bibliotheken im Hinblick auf Kosteneinsparungen und Projektmanagement attraktiver. Gleichzeitig gewährleistet die Verwendung von Standardbibliotheken die Zuverlässigkeit und Stabilität der Softwarelösung und befreit Benutzer von Abhängigkeiten zu Komponenten Dritter und deren Änderungen.

1.3 Projektschnittstellen

Die Anwendung ist für die Verwendung in der PyCharm-Umgebung über eine Konsolenschnittstelle konzipiert, ohne dass eine grafische Benutzeroberfläche (GUI) erstellt werden muss. Dies entspricht den Anforderungen des Projekts und erleichtert einem Benutzer des Zuse Institute Berlin mit Python-Programmierkenntnissen die Interaktion mit dem Programm, ohne dass zusätzliche Ressourcen für die Erstellung und Pflege einer GUI erforderlich sind.

2. Projektdefinition

2.1 IST Analyse

Vor Beginn des Projekts wurde die PyCharm-Entwicklungsumgebung vorbereitet und konfiguriert, die ein komfortables Programmieren und Debuggen der Anwendung in Python ermöglicht. Wir haben auch den Zugriff auf den Python Package Index (PyPI) konfiguriert, ein zentrales Repository für die Installation und Verwaltung von Python-Bibliotheken und -Modulen von Drittanbietern. Für die Versionskontrolle und Zusammenarbeit am Projekt wurde GitLab verwendet, das Tools für effektives Änderungsmanagement bereitstellt.

2.2 SOLL-Konzept

Das Programm interagiert mit dem Benutzer, indem es die URL, den Pfad und den Dateinamen der Webseite anfordert, um den Inhalt der Website zu speichern. Anschließend crawlt das Programm mithilfe eines rekursiven Algorithmus alle Links auf der Seite und den Unterseiten und liest den HTML-Code. Wenn der HTML-Code erfolgreich gelesen wurde, wird der Code in einer vom Benutzer angegebenen Datei gespeichert. Wenn beim Öffnen von Links oder beim Lesen von Code Fehler auftreten, generiert das Programm eine Fehlermeldung und sorgt so für eine informative Interaktion mit dem Benutzer.

2.3 Projektziele

Das Ziel des SmartArchive-Projekts ist die Entwicklung eines Python-Programms unter Verwendung von Standardbibliotheken, das den HTML-Inhalt einer statischen Website sowie aller untergeordneten Seiten erfasst und in einer einzigen HTML-Datei speichert. Benutzer sollen damit die Möglichkeit haben, statische Webinhalte effektiv zu archivieren. Um das Ziel des Programms zu erreichen, ist es notwendig, folgende Funktionen zu implementieren:

- Eingeben der URL und des Pfades zum Speichern der Dateien.
- Abrufen des HTML-Codes von Webseiten
- Rekursive Seitenanalyse zur Erstellung einer Liste von URLs sowohl für die Hauptwebseite als auch für Unterseiten
- Anschließendes Überprüfen vorhandener URLs und Entfernen doppelter Einträge.
- Protokollierung von Fehlern und Warnungen, um einen zuverlässigen Betrieb des Programms sicherzustellen.

Zur Entwicklung des Programms wird die Entwicklungsumgebung PyCharm Edu sowie Python-Pakete zur Implementierung der Funktionalität des Programms verwendet. Es ist geplant, GitLab zur Verwaltung und Kontrolle von Codeversionen zu nutzen, um ein effektives kollaboratives Projektmanagement, Code-Branching und kollaborative Entwicklung im Team und Änderungsmanagement zu gewährleisten.

2.4 Projektabgrenzung

Bei diesem Projekt geht es nicht um die Entwicklung einer Benutzeroberfläche, da sich das Programm an Benutzer mit Kenntnissen in Python und PyCharm richtet. Die Schnittstellenentwicklung kann als

mögliche nächste Phase des Projekts in Betracht gezogen werden, wenn es notwendig ist, die Zielgruppe der Benutzer zu erweitern und eine bequemere Interaktion mit dem Programm zu ermöglichen.

3. Projektplanung

3.1 Vorgehensmodell

Das Projekt wurde nach dem klassischen Wasserfallmodell mit vier Phasen durchgeführt. Das Wasserfallmodell ist ein bekanntes Vorgehensmodell der klassischen Softwareentwicklung (Fittkau und Ruf [2008], S. 31), bei dem die Projektphasen - Definition, Planung, Umsetzung und Abschluss - sequenziell durchlaufen und bearbeitet werden. Dieser methodische Ansatz erwies sich als erfolgreich für mein kleines Projekt.

3.2 Projektstrukturplan

Zur besseren Planung wurde ein Projekt-Struktur-Plan erstellt, siehe [Abbildung 1](#) im Anhang.

3.3 Zeitplan

Basierend auf dem Wasserfallmodell wurde der folgende Zeitplan erstellt:

Tabelle 1 Zeitplan grob

Hauptaufgaben	Dauer
Definition	6h
Planung	9h
Realisierung	51h
Abschluss	14h

Der detaillierte Zeitplan findet sich im Anhang, [Tabelle 2](#).

3.4 Gantt-Diagramm

Zur besseren Visualisierung wurde der Zeitplan als Gantt-Diagramm dargestellt, siehe Anhang [Tabelle 3](#).

3.5 Kostenplan

Für die Umsetzung des Projekts wurden verschiedene Ressourcen verwendet, die in Hardware, Software und personelle Ressourcen unterteilt werden können. Die erforderliche Hardware bestand aus einem Dell-Computer und einem Büroarbeitsplatz. Der Computer war mit einem Monitor, einer Maus und Tastatur sowie einem Headset für die Kommunikation mit Kollegen ausgestattet. Zur

Entwicklung wurde die Entwicklungsumgebung PyCharm Edu Version 2022.2.2 verwendet. Die Versionsverwaltung wurde mit GitLab realisiert. Für die Dokumentation wurden verschiedene Softwaretools verwendet: Die Benutzerdokumentation wurde mit Microsoft Office Word erstellt, Diagramme wurden mit dem yEd Graph Editor erstellt, und Grafiken wurden mit Paint Brush erstellt. Die Kosten für die Umsetzung des Projekts umfassten Personalkosten, einschließlich des Autors und des Projektleiters. Da ausschließlich lizenzierte Software oder Software mit gültigen Lizenzen verwendet wurde und keine Dienstreisen und damit verbundenen Transportkosten anfielen, entstanden keine zusätzlichen Kosten (Anhang, [Tabelle 4](#)). Da den Praktikanten keine konkreten Zahlen mitgeteilt wurden, enthält der Kostenplan Schätzungen aus folgenden Quellen:

<https://www.meinpraktikum.de/ratgeber/mindestlohn/praktikum/faq>

<https://www.berlin.de/sen/arbeit/beschaefigung/mindestlohngesetze/>

3.6 Qualitätssichernde Maßnahmen

Tabelle 5 Qualitätssicherungsplan

Konstruktive Maßnahmen	Analytische Maßnahmen
Detaillierte Anforderungsliste, siehe Anhang.	Entwicklertests
Detaillierte Planung	White-Box-Tests durch Kollegen
Einhaltung der Coding-Standards von Python	Code Review durch Projektbetreuer
Einhaltung der Firmenrichtlinien bezüglich Entwicklerarbeit	Interne Qualitätskontrollen, Überprüfung durch Vorgesetzten
Formatierung des Codes: Einsatz der Autoformat-Funktion der IDE	Black-Box-Tests durch Tester
Pflege sinnvoller Kommentare im Code	Kommentare zur Erklärung von Schlüsselkonzepten und komplexen Funktionen im Code.

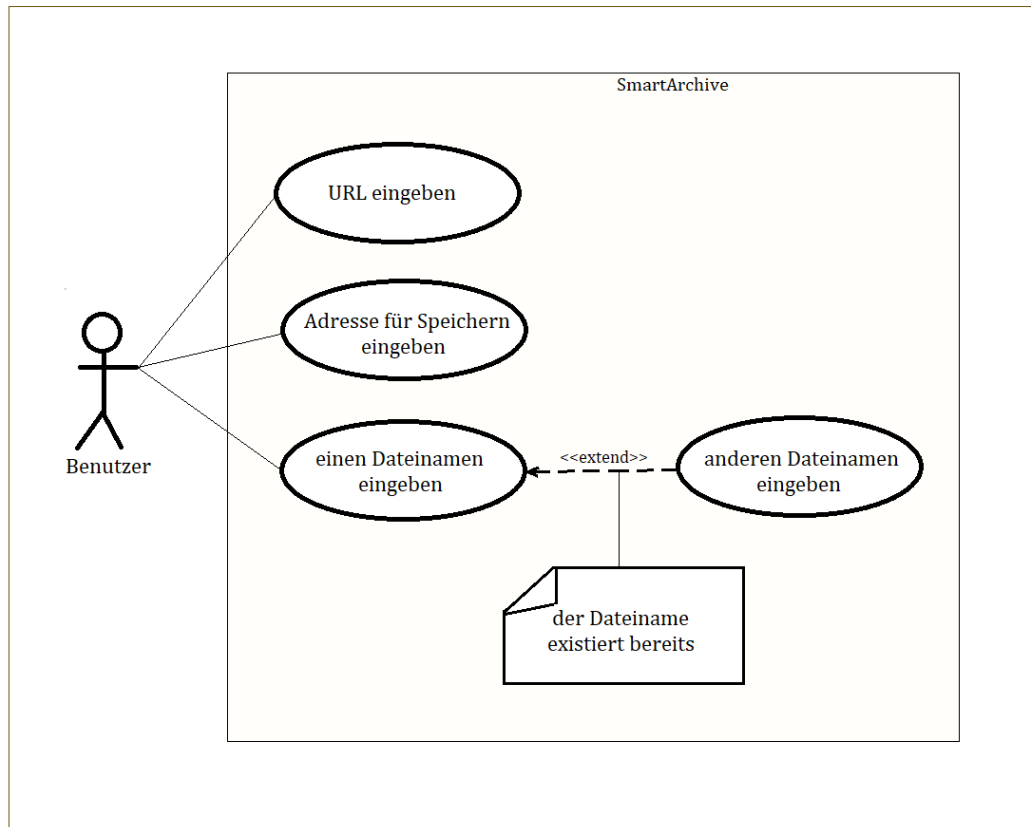
3.7 Entwürfe

3.7.1 UML-Diagramm

Anwendungsfall 1: Ein Benutzer interagiert mit einem System, das Informationen über eine Webseite zur Archivierung anfordert. Der Benutzer gibt die URL der Seite ein, die archiviert werden soll.

Anwendungsfall 2: Das System fragt den Benutzer nach der Adresse, unter der die HTML-Codes von Webseiten gespeichert werden sollen. Der Benutzer gibt die erforderliche Adresse zum Speichern von Dateien ein.

Abbildung 2: Use-Case Diagramm



Anwendungsfall 3: Wenn das System nach einem Dateinamen zum Speichern fragt, gibt der Benutzer den entsprechenden Namen ein. Wenn bereits eine Datei mit demselben Namen existiert, warnt das Programm den Benutzer und fordert ihn auf, einen anderen Namen für die Datei einzugeben.

Diese Anwendungsfälle decken die grundlegenden Schritte der Benutzerinteraktion mit dem SmartArchive-System ab und machen den Prozess der Archivierung von Webseiten bequem und verständlich.

3.7.2 Flussdiagramme

Das Flussdiagramm des Programms „SmartArchive“ in der ersten Entwicklungsphase (Anhang, [Abbildung 3](#)) umfasst die wichtigsten Schritte und Vorgänge, die für die Archivierung von Webseiten erforderlich sind. Der Prozess beginnt mit der Eingabephase, in der der Benutzer die erforderlichen Informationen angibt, beispielsweise die URL der zu archivierende Seite und den Pfad zum Speichern der Seitencoddatei.

Als nächstes implementiert das Programm Blöcke zum Erstellen einer Datei, in der der HTML-Code der Seite gespeichert wird, sowie einen Block zur Überprüfung der Rekursionstiefe. Wenn die vom Benutzer angegebene Rekursionstiefe nicht erreicht wird, fährt das Programm mit dem Lesen der HTML-Seite und dem Hinzufügen der URL zur Liste zur weiteren Analyse fort.

Wenn die Rekursionstiefe erreicht ist, öffnet das Programm die erstellte Datei und schreibt nacheinander die HTML-Codes aller angezeigten Seiten auf. Dadurch wird sichergestellt, dass die Inhalte jeder Seite innerhalb der angegebenen Rekursionstiefe separat gespeichert werden.

Das Programm endet, nachdem alle Phasen der Archivierung und Aufzeichnung von Dateien abgeschlossen sind, was durch den Block „Programmende“ angezeigt wird. Dieses grobe Programmdiagramm ermöglicht Ihnen die visuelle Darstellung der Hauptphasen des SmartArchive-Systems und bietet eine grundlegende Roadmap für die weitere Detail- und Funktionsentwicklung. Jeder der Blöcke in diesem Diagramm stellt ein unabhängiges Modul dar, das aus mehreren aufeinanderfolgenden Aktionen besteht.

Das Diagramm in [Abbildung 4](#), Anhang, beschreibt eine Funktion, die eine Datei zum Schreiben erstellt, indem sie den Benutzer zur Eingabe des Pfads und Dateinamens auffordert. Die Funktion prüft, ob bereits eine Datei mit demselben Namen existiert. Wenn die Datei bereits vorhanden ist, fordert die Funktion den Benutzer zur Eingabe eines neuen Namens auf. Nach erfolgreichem Abrufen des Namens erstellt die Funktion die Datei und informiert den Benutzer über diese Aktion. Dies stellt sicher, dass Dateien korrekt erstellt werden, ohne dass vorhandene Dateien überschrieben werden.

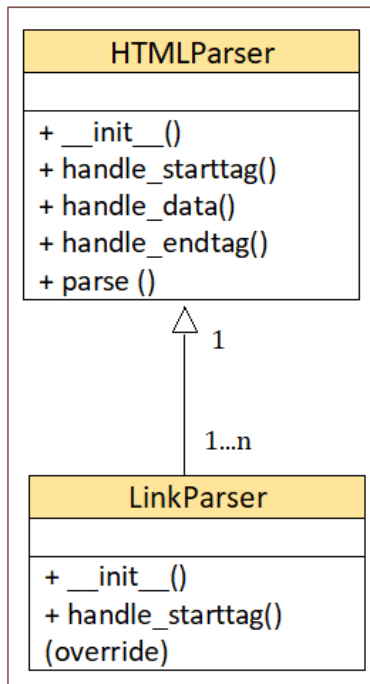
Das nächste Flussdiagramm ([Abbildung 5](#), Anhang) beschreibt eine Funktion, die Seiten rekursiv scannt und eine Liste von URLs erstellt. Es nimmt die URL einer Website und deren HTML-Text als Parameter an. Der erste Schritt besteht darin, eine leere Liste zum Aufzeichnen von URLs zu erstellen und die Homepage-URL zu dieser Liste hinzuzufügen. Das Programm prüft die aktuelle Rekursionstiefe. Ist die Tiefe noch nicht erreicht, analysiert das Programm den HTML-Code der Hauptseite und extrahiert daraus Links. Für jede gefundene Referenz wird geprüft, ob sie absolut ist. Wenn die Referenz nicht absolut ist, wird sie übersprungen. Handelt es sich um einen absoluten Link, prüft das Programm, ob der HTML-Code der Seite über diesen Link bezogen werden kann. Wenn möglich, wird der Link zur Liste hinzugefügt und die Funktion ruft sich selbst rekursiv auf, um diese Seite zu analysieren. Wenn nicht, wird der Link übersprungen.

Somit durchläuft das Programm rekursiv alle verfügbaren Links, bis es eine bestimmte Rekursionstiefe erreicht. Nach Abschluss der Untersuchung des HTML-Codes der Hauptseite gibt die Funktion eine Liste aller innerhalb der angegebenen Rekursionstiefe gefundenen URLs zurück.

3.7.3 Klassen Diagramm

In der Funktion `creating_list_urls()` zum Parsen einer Webseite wird die integrierte `HTMLParser`-Klasse aus dem Modul `html.parser` der Bibliothek `urllib.parse` verwendet.

Abbildung 6 Klassen Diagramm



Wir haben unsere eigene `LinkParser`-Klasse erstellt, die von `HTMLParser` erbt. Dies ist eine integrierte Klasse aus dem Modul `html.parser`, die ein grundlegendes Framework zum Parsen von in HTML und XHTML formatierten Textdateien bereitstellt. Es definiert Methoden zur Verarbeitung verschiedener HTML-Elemente und -Daten. Einer `HTMLParser`-Instanz werden HTML-Daten zugeführt und sie ruft Handler-Methoden auf, wenn Start-Tags, End-Tags, Text, Kommentare und andere Markup-Elemente gefunden werden.

Die Klasse `LinkParser` vererbt von `HTMLParser` und überschreibt die Methode `handle_starttag`, um sich während des Analysevorgangs speziell auf das Extrahieren von URLs aus Ankertags (`<a>`) zu konzentrieren.

Methoden und Attribute:

`__init__()`: Sowohl `HTMLParser` als auch `LinkParser` verfügen über eine `__init__()`-Methode, die die Instanz initialisiert und alle internen Datenstrukturen einrichtet.

`handle_starttag()`: Diese Methode vom `HTMLParser` wird immer dann aufgerufen, wenn ein öffnendes HTML-Tag gefunden wird. Als Argumente erhält es den Tag-Namen und eine Liste von Attributen.

In der Klasse `LinkParser` diese überschriebene Methode prüft speziell, ob das Tag ein Ankertag (`<a>`) ist. Wenn ja, extrahiert es die URL aus dem `href`-Attribut und hängt sie an eine interne Linkliste an.

`handle_data()` und `handle_endtag()`: Diese Methoden sind auch in `HTMLParser` definiert und werden für andere HTML-Elemente und Daten aufgerufen.

`parse()`: Diese Methode ist in `HTMLParser` definiert und wird verwendet, um den Parsing-Prozess für eine bestimmte HTML-Zeichenfolge zu initiieren. Die `LinkParser`-Klasse verfügt über keine `parse()`-

Methode, ruft jedoch während ihrer eigenen `__init__()`-Methode indirekt die `parse()`-Methode der Klasse `HTMLParser` auf.

4. Durchführung

4.1 Implementierung

4.1.1 Software-Architektur

Verwendete Bibliotheken:

Das Programm verwendet ausschließlich Standard-Python-Bibliotheken wie `urllib.request`, `urllib.parse`, `urllib.error`, um URLs und HTML-Code von Webseiten zu verarbeiten und Links aus HTML zu extrahieren.

Intuitive Funktionsnamen:

Alle Funktionen haben klare und aussagekräftige Namen, wodurch der Code für andere Entwickler besser lesbar und verständlich ist. Beispielsweise geben die Funktionen `get_user_input()`, `get_html_code` ihre Funktionalität deutlich an.

Funktion `creating_list_urls()`: die Funktion verwendet die Standard-Python-Bibliothek `urllib.parse`. Die Funktion verwendet HTML-Text als Eingabe. Die Funktion findet rekursiv alle Links zu Unterseiten in diesem Text mit einer Standardtiefe von 2, prüft sie auf Eindeutigkeit und fügt sie der Liste hinzu. Die Funktion gibt eine Liste eindeutiger Arbeitslinks zurück.

Programmstruktur:

Das Programm ist in separate Funktionen unterteilt, die bestimmte Aufgaben ausführen. Die gesamte Logik zur Verarbeitung von HTML-Code, zum Extrahieren von Links und zum Speichern von Daten ist in diesen Funktionen untergebracht, was die Wartung und Erweiterung des Programms erleichtert.

Funktionskommentare:

Jede Funktion wird von Kommentaren begleitet, die ihren Zweck, ihre Parameter und ihre Betriebslogik erläutern.

4.1.2 Implementierung des Erhaltens von HTML-Code

Die Funktion `get_html_code()` zum Abrufen des HTML-Codes einer Webseite unter einer angegebenen URL wird mithilfe der URL-Bibliothek implementiert. Es nimmt einen Link als Argument und gibt den HTML-Code der Seite zurück. Wenn die Seitenanforderung erfolgreich ist, gibt die Funktion den entsprechenden HTML-Code zurück. Wenn jedoch ein Fehler vom Typ `urllib.error.URLError` auftritt, fängt die Funktion den Fehler ab und behandelt ihn, gibt eine Fehlermeldung aus und gibt `None` zurück.

Die Funktion `get_html_code()` wird innerhalb einer anderen Funktion `save_html_codes()` angerufen, die eine Liste URLs und die Adresse der Datei zum Schreiben von Code annimmt und ihre Codes nacheinander in dieser Datei speichert.

4.1.3 Implementierung der Erstellung von der URL-Liste

Die Funktion `creating_list_urls()` für rekursiven Scan der eingegebenen Seite verwendet die `HTMLParser`-Klasse aus der Standardbibliothek `urllib.parse`, um HTML-Code zu analysieren und Links aus dem `href`-Attribut von `<a>`-Ankertags zu extrahieren. Hier wird eine Instanz der `LinkParser`-Klasse erstellt, die eine Unterklasse von `HTMLParser` ist. Diese Klasse wurde definiert, um bestimmte HTML-Tags zu verarbeiten. `parser.feed(html_text)` wird aufgerufen, um den übergebenen HTML-Text zu verarbeiten und die `handle_starttag`-Methode der `LinkParser`-Klasse für jedes Starttag im HTML-Text aufzurufen:

```
# Parsing of HTML-code with urllib.parse
from html.parser import HTMLParser

class LinkParser(HTMLParser):
    def __init__(self):
        super().__init__()
        self.links = []

    def handle_starttag(self, tag, attrs):
        if tag == 'a':
            for attr, value in attrs:
                if attr == 'href':
                    self.links.append(value)

# Create a parser instance and parse the HTML code
parser = LinkParser()
parser.feed(html_text)
```

`parser.links` ist eine Liste, die von der `LinkParser`-Klasse erstellt wurde. In der `handle_starttag`-Methode der `LinkParser`-Klasse werden Links (`<a>`-Tags) aus dem HTML-Code extrahiert und in diese Liste `parser.links` eingefügt. Die Schleife `for link in parser.links:` durchläuft alle Links, die aus dem HTML-Text extrahiert wurden. In der Funktion `creating_list_urls()` wird auch überprüft, ob der Link eine Schemakomponente hat oder nicht. Wenn ein Link relativ ist und kein Schema wie "http" oder "https" hat, wird er übersprungen. Das bedeutet, dass nur Links mit einem expliziten Schema verarbeitet werden, was typischerweise für externe Links oder vollständige URLs der Fall ist:

```
# Sort through the links, open them, find new links
for link in parser.links:
    try:
        if urllib.parse.urlparse(link).scheme == '':
            continue # Skip relative link
```

Der folgende Codeausschnitt zeigt die tatsächliche Implementierung des rekursiven Seitenscans:

```
# Get HTML code of the page from the link
    this_html_code = get_html_code(link)
# If the HTML code is received successfully, add links to the
# general list
    if this_html_code:
        all_links.append(link)
        # Recursively calls creating_list_urls
        # to process internal links
        inner_links = creating_list_urls(this_html_code, link,
visited_urls, depth + 1, max_depth)
        all_links.extend(inner_links)
    except Exception as e:
        print("Error processing URL:", link, e)
```

Die Liste `parser.links` enthält alle extrahierten Links aus dem HTML-Code der aktuellen Seite. Die Schleife durchläuft jeden dieser Links, um sie weiter zu verarbeiten. Für jeden Link wird der HTML-Code der verlinkten Seite abgerufen, indem die `get_html_code`-Funktion verwendet wird. Nachdem der HTML-Code der verlinkten Seite erhalten wurde, wird die Funktion `creating_list_urls()` rekursiv aufgerufen, um die internen Links dieser Seite zu extrahieren. Dies geschieht, indem die `this_html_code` der verlinkten Seite und die Tiefe der Rekursion aktualisiert werden. Wenn der HTML-Code erfolgreich erhalten wurde, werden die Links zu dieser Seite der allgemeinen Liste `all_links` hinzugefügt. Auf diese Weise werden alle relevanten Links von allen Ebenen der Webseite gesammelt und verarbeitet, während die Rekursion tiefer in die Seitenstruktur eindringt.

Beim Schreiben der Funktion `get_html_code` zum sequentiellen Speichern von HTML-Codes mehrerer Seiten in eine Datei haben wir bestimmte Optimierungen vorgenommen, um Wiederholungen zu vermeiden und die Korrektheit der gespeicherten Inhalte sicherzustellen. Um dies zu erreichen, wurden folgende Änderungen durchgeführt:

- Anstelle der Variablen `html_code` wird in der Zeile `f.write(this_html_code + "\n\n")` nun die Variable `this_html_code` verwendet.

- Das schließende HTML-Tag `</body>\n</html>` wurde außerhalb des Zyklus verschoben, damit es nach dem Speichern aller Codes und nicht nach jeder Seite geschlossen wird. Jetzt speichert die Funktion die HTML-Codes aller Seiten korrekt in einer Datei ([Abbildungen 7](#) und [8](#), Anhang).

Der korrigierte und der ursprüngliche Quellcode der Funktion sind in [Abbildung 9](#) und [10](#) dargestellt.

4.1.4 Implementierung der HTML-Dateierstellung

Insgesamt bietet die Funktion `create_file_if_not_exists()` eine effiziente Möglichkeit, eine Datei zu erstellen und sicherzustellen, dass der Dateiname eindeutig ist, um Überschreibungsfehler zu vermeiden.

Zuerst wird die Python-Bibliothek `os` importiert, die Funktionen zur Interaktion mit dem Betriebssystem bereitstellt, wie z.B. das Überprüfen von Dateixistenz, Erstellen von Dateien usw. Durch die Verwendung von `os.path.join` wird der vollständige Pfad zur Datei aus dem angegebenen Pfad und Dateinamen erstellt. Es wird überprüft, ob die Datei bereits existiert. Falls ja, wird der Benutzer aufgefordert, einen anderen Dateinamen einzugeben, bis ein eindeutiger Dateiname gefunden wird. Erstellung der Datei: Sobald ein eindeutiger Dateiname gefunden wurde, wird die Datei erstellt und der vollständige Pfad zur erstellten Datei zurückgegeben. Wenn die Datei erfolgreich erstellt wurde, wird eine Erfolgsmeldung angezeigt, andernfalls eine Fehlermeldung. Die Funktion gibt den vollständigen Pfad zur erstellten Datei zurück, wenn sie erfolgreich erstellt wurde, andernfalls `None`, um den Fehler weiter zu behandeln:

```
Input your URL: https://www.projekt-cib.de/wordpress/
Enter the path to save the file: C:/Users/tn/OneDrive/Desktop
Enter the file name: 01.html
File '01.html' already exists. Enter a different file name: 77.html
File 77.html was successfully created at C:/Users/tn/OneDrive/Desktop
```

4.2 Qualitätskontrolle

Der Kollege Stefan Lohrum hat wie geplant getestet, die Fehler, die er gefunden hat, wurden korrigiert und behoben.

Tabelle 6 Qualitätskontrolle

Konstruktive Maßnahmen		Analytische Maßnahmen	
Detaillierte Anforderungsliste, siehe Anhang.	ok	Entwicklertests	ok
Detaillierte Planung	ok	White-Box-Tests durch Kollegen	ok
Einhaltung der Coding-Standards von Python	ok	Code Review durch Projektbetreuer	ok
Einhaltung der Firmenrichtlinien bezüglich Entwicklerarbeit	ok	Interne Qualitätskontrollen, Überprüfung durch Vorgesetzten	ok
Formatierung des Codes: Einsatz der Autoformat-Funktion von PyCharm	ok	Black-Box-Tests durch Tester	ok
Pflege sinnvoller Kommentare im Code	ok	Kommentare zur Erklärung von Schlüsselkonzepten und komplexen Funktionen im Code.	ok

4.3 Außerplanmäßige Änderungen

Das Projekt verlief genau nach Plan, es gab keine außerplanmäßigen Änderungen.

5 Abschlussphase

5.1 Projektübergabe

Der Code wurde auf GitLab, wie angefordert wurde, hochgeladen.

Die Dokumentation wurde als PDF in dieselbe Hochladestation abgegeben.

5.2 Fachlicher Soll-Ist-Vergleich

Alle geplanten Funktionen konnten erfolgreich entwickelt werden.

5.3 Zeitlicher Soll-Ist-Vergleich

Die [Tabelle 7](#) für Zeitlicher Soll-Ist-Vergleich ist im Anhang beigelegt.

Das Projekt SmartArchive wurde im vorgegebenen Zeitrahmen erfolgreich umgesetzt und die in der Analysephase gestellten Anforderungen wurden vollständig erfüllt. Während der Arbeit am Projekt wurden jedoch einige Abweichungen vom ursprünglichen Plan festgestellt. In der Phase der Integration einzelner Blöcke in ein einziges Programm wurde mehr Zeit aufgewendet als ursprünglich geplant. Ausschlaggebend hierfür war die Notwendigkeit, die Zusammenarbeit aller Komponenten gründlich zu testen und sicherzustellen, dass sie fehlerfrei interagieren. Einige Aufgaben, wie zum Beispiel die Entwicklung einzelner Features oder Module, gestalteten sich einfacher und nahmen weniger Zeit in Anspruch als geplant.

5.4 Fazit

5.4.1. Zeitlicher Fazit

Trotz einiger Änderungen beim Zeitaufwand für einzelne Aufgaben wurde der Gesamtzeitrahmen des Projekts SmartArchive eingehalten.

5.4.2 Sachlicher Fazit

Durch sorgfältige Strukturierung und Management-Feedback lässt das Projekt die Tür für zukünftige Erweiterungen und Integration neuer Funktionen offen.

5.4.3 Persönlicher Fazit

Im Laufe des Projekts habe ich meine Fähigkeiten in der Python-Programmierung deutlich verbessert. Die Erfahrung mit der Codeüberprüfung und das Feedback des Vorgesetzten haben mir gezeigt, wie wichtig es ist, bei der Arbeit an größeren Programmen viele Faktoren zu berücksichtigen und wie wichtig eine sorgfältige Vorplanung der Schritte ist. Das Wasserfall-Projektentwicklungsmodell erwies sich für dieses kleine Projekt als effektiv.

5.5 Geplante Präsentationsmittel

Laptop, Beamer

6 ANHANG

6.1 Abbildungen

Abbildung 1 Projektstrukturplan des Programms SmartArchive

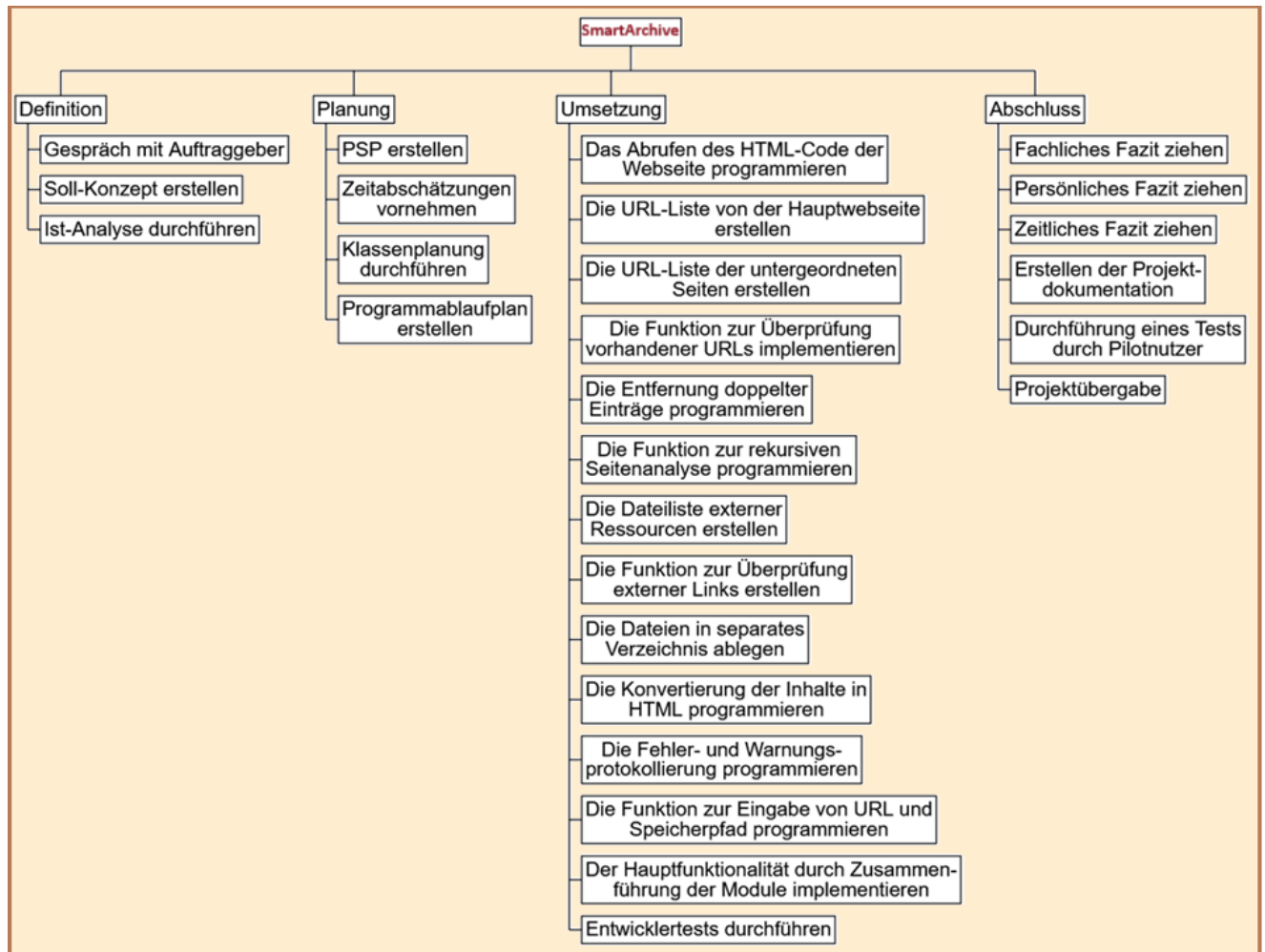


Abbildung 3 Flussdiagramm des gesamten Programms

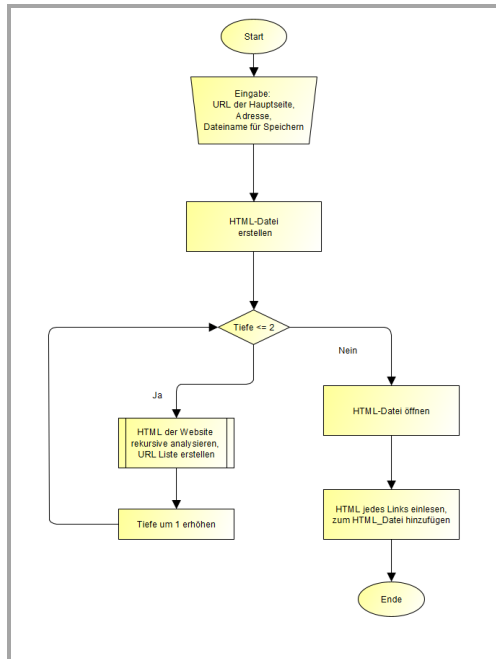


Abbildung 4 Flussdiagramm zur Dateierstellung

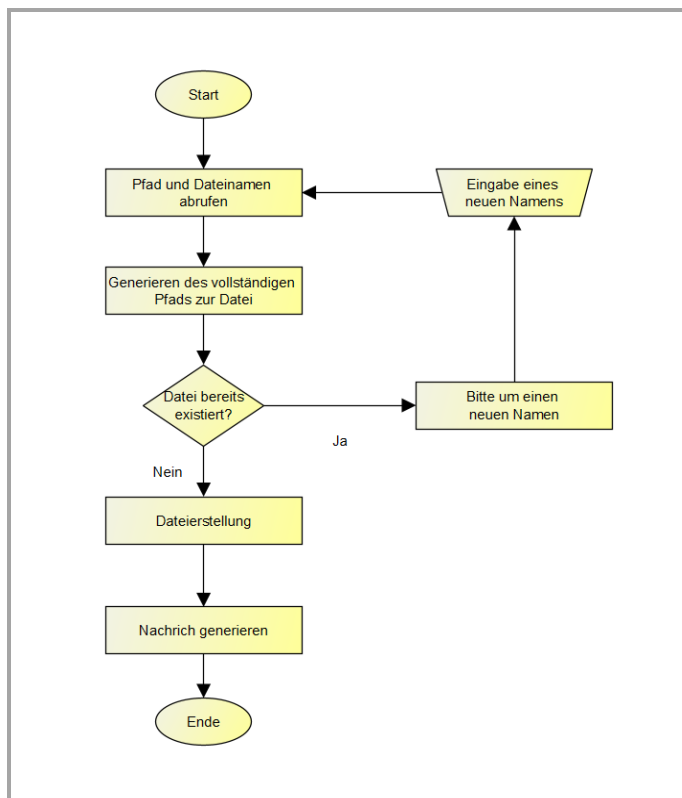


Abbildung 5 Flussdiagramm zur rekursiven Durchsuchung der Webseite

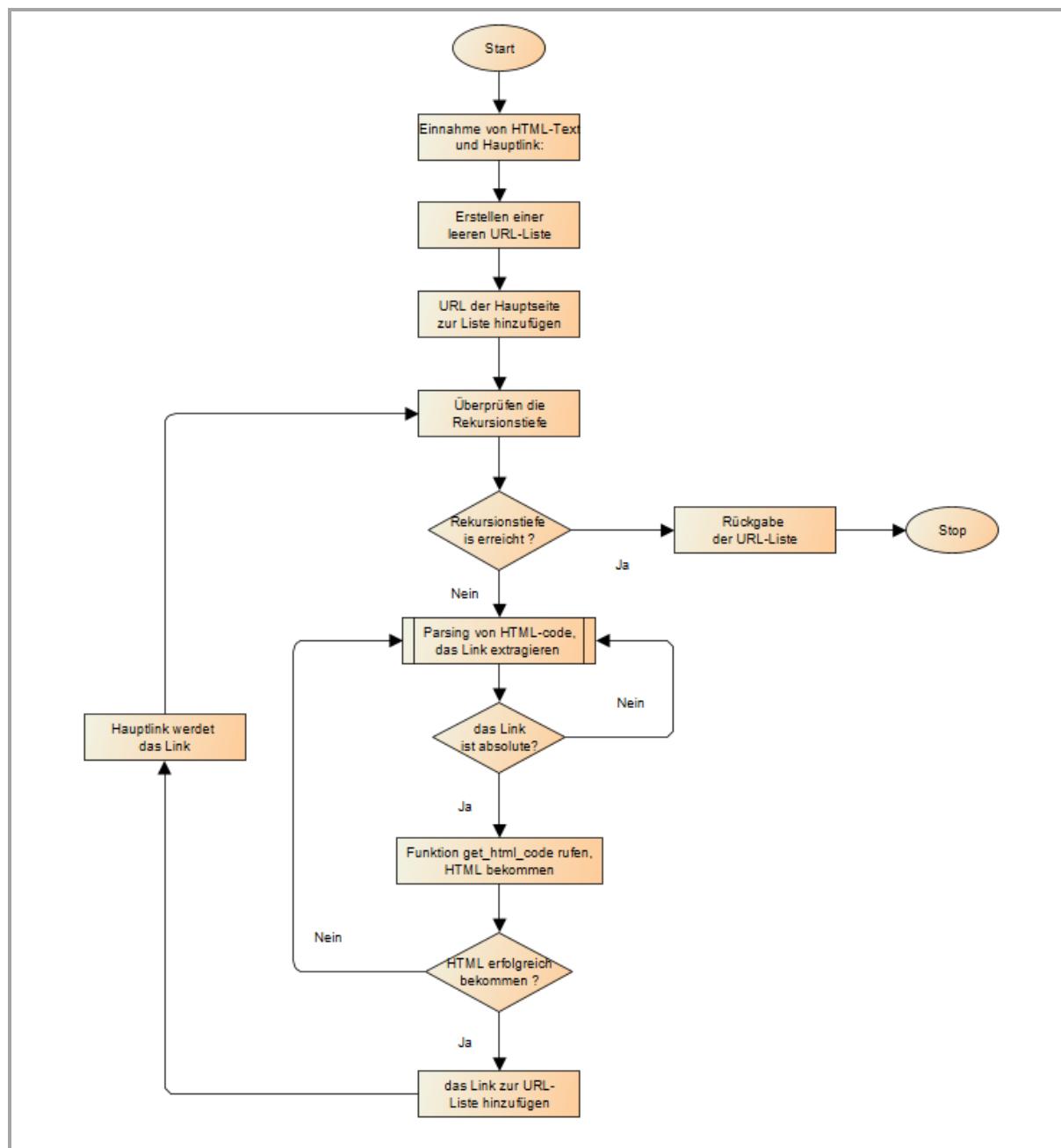
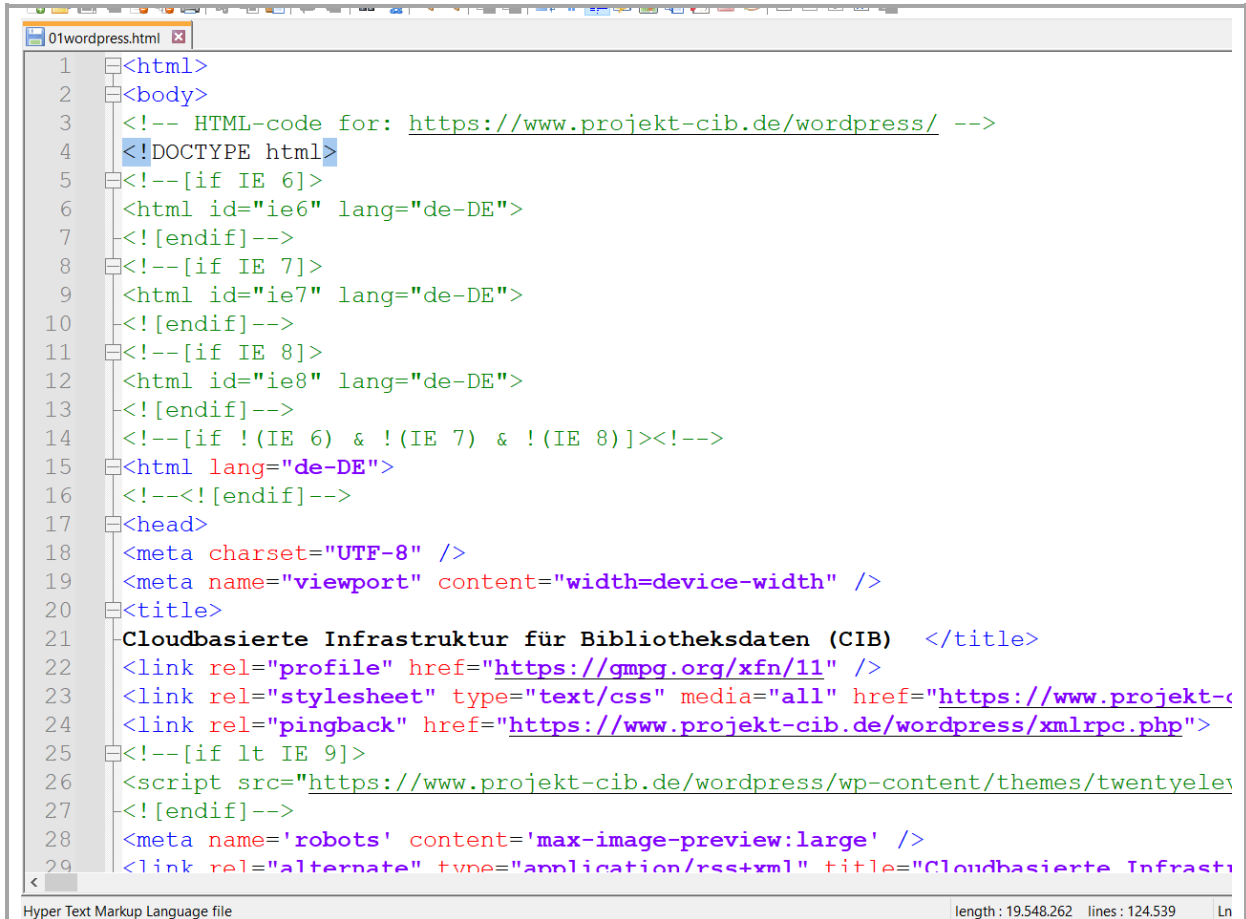


Abbildung 7 Bildschirm Foto 1: HTML-Datei, Beginn von der Hauptseite



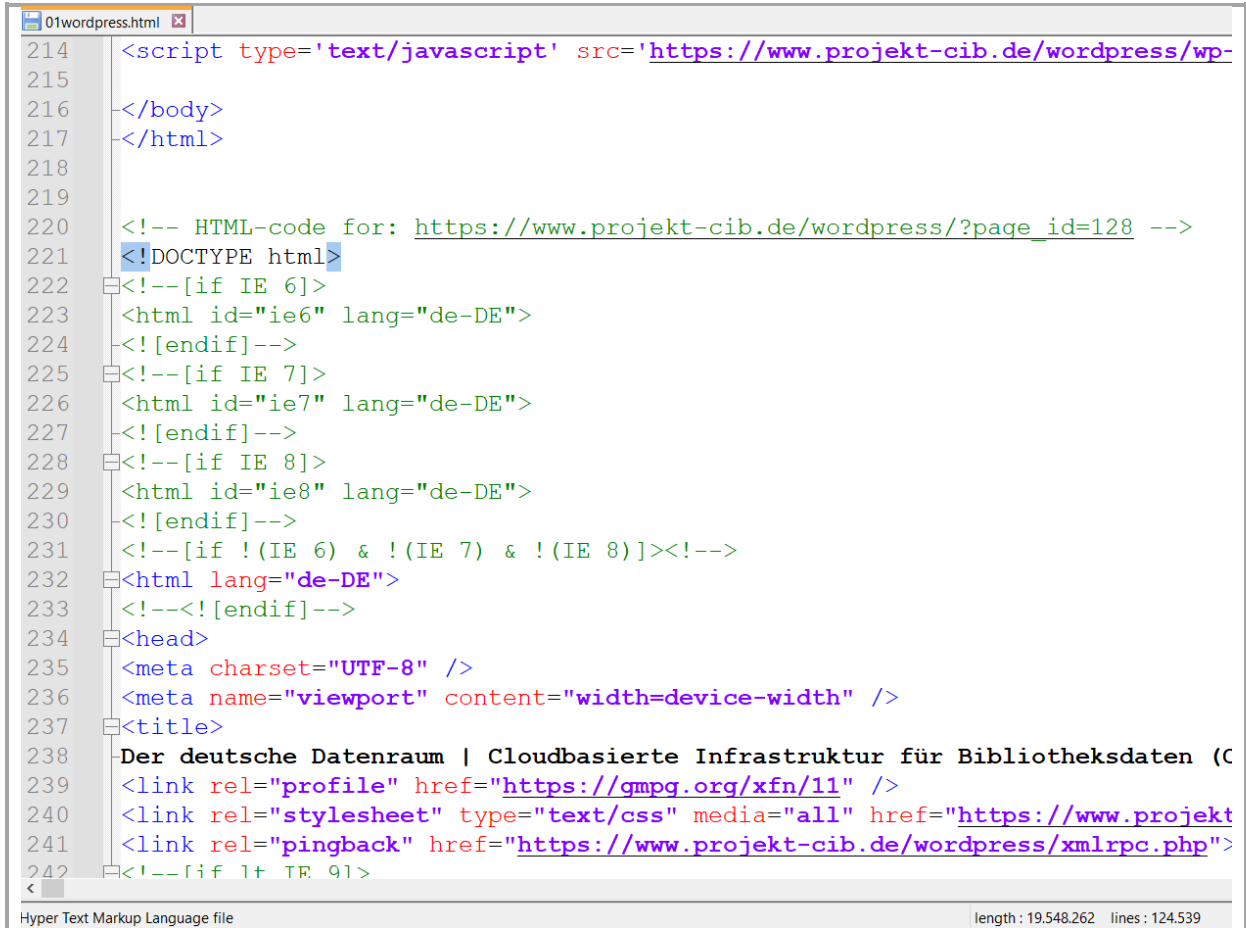
```

1 <html>
2 <body>
3 <!-- HTML-code for: https://www.projekt-cib.de/wordpress/ -->
4 <!DOCTYPE html>
5 <!--[if IE 6]>
6 <html id="ie6" lang="de-DE">
7 <![endif]-->
8 <!--[if IE 7]>
9 <html id="ie7" lang="de-DE">
10 <![endif]-->
11 <!--[if IE 8]>
12 <html id="ie8" lang="de-DE">
13 <![endif]-->
14 <!--[if !(IE 6) & !(IE 7) & !(IE 8)]><!-->
15 <html lang="de-DE">
16 <!--<![endif]-->
17 <head>
18 <meta charset="UTF-8" />
19 <meta name="viewport" content="width=device-width" />
20 <title>
21 Cloudbasierte Infrastruktur für Bibliotheksdaten (CIB) </title>
22 <link rel="profile" href="https://gmpg.org/xfn/11" />
23 <link rel="stylesheet" type="text/css" media="all" href="https://www.projekt-cib.de/wordpress/wp-content/themes/twentyeleven/css/all.css" />
24 <link rel="pingback" href="https://www.projekt-cib.de/wordpress/xmlrpc.php">
25 <!--[if lt IE 9]>
26 <script src="https://www.projekt-cib.de/wordpress/wp-content/themes/twentyeleven/js/html5.js"></script>
27 <![endif]-->
28 <meta name="robots" content="max-image-preview:large" />
29 <link rel="alternate" type="application/rss+xml" title="Cloudbasierte Infrastruktur für Bibliotheksdaten (CIB)" href="https://www.projekt-cib.de/wordpress/feed/" />

```

Hyper Text Markup Language file length : 19.548.262 lines : 124.539 Ln

Abbildung 8 Bildschirm Foto 2: HTML-Datei, Beginn von der 2. Seite



```

214 <script type='text/javascript' src='https://www.projekt-cib.de/wordpress/wp-
215
216 </body>
217 </html>
218
219
220 <!-- HTML-code for: https://www.projekt-cib.de/wordpress/?page_id=128 -->
221 <!DOCTYPE html>
222 <!--[if IE 6]>
223 <html id="ie6" lang="de-DE">
224 <![endif]-->
225 <!--[if IE 7]>
226 <html id="ie7" lang="de-DE">
227 <![endif]-->
228 <!--[if IE 8]>
229 <html id="ie8" lang="de-DE">
230 <![endif]-->
231 <!--[if !(IE 6) & !(IE 7) & !(IE 8)]><!-->
232 <html lang="de-DE">
233 <!--<![endif]-->
234 <head>
235 <meta charset="UTF-8" />
236 <meta name="viewport" content="width=device-width" />
237 <title>
238 Der deutsche Datenraum | Cloudbasierte Infrastruktur für Bibliotheksdaten (C
239 <link rel="profile" href="https://gmpg.org/xfn/11" />
240 <link rel="stylesheet" type="text/css" media="all" href="https://www.projekt
241 <link rel="pingback" href="https://www.projekt-cib.de/wordpress/xmlrpc.php">
242 <!--[if lt IE 9]>

```

Hyper Text Markup Language file length : 19.548.262 lines : 124.539

Abbildung 9 Der korrigierte Quellcode der get_html_code-Funktion

```
def save_html_codes(urls, this_full_path):
    try:
        # Open file to save HTML-codes
        with open(this_full_path, 'w', encoding='utf-8') as f:
            # Write the start of the HTML file
            f.write("<html>\n<body>\n")
            # Get and write HTML for every site
            for url in urls:
                this_html_code = get_html_code(url)
                if this_html_code:
                    f.write(f"<!-- HTML-code for: {url} -->\n")
                    f.write(this_html_code + "\n\n")
            # End of the HTML file
            f.write("</body>\n</html>")

            print(f"HTML codes are saved in the file {this_full_path}")
    except Exception as e:
        print("Error saving HTML codes:", e)
```

Abbildung 10 Der ursprüngliche Quellcode der get_html_code-Funktion

```
def save_html_codes(urls, this_full_path):

    try:

        # Open file to save HTML-codes

        with open(this_full_path, 'w', encoding='utf-8') as f:
            # Write a start of HTML-file
            f.write("<html>\n<body>\n")
            # Get and write HTML for every site
            for url in urls:
                this_html_code = get_html_code(url)
                if this_html_code:
                    f.write(f"<!-- HTML-code for: {url} -->\n")
                    f.write(this_html_code + "\n\n")
            # End of the HTML-file:
            f.write("</body>\n</html>")

            print(f"HTML-code {url} is saved in the file {this_full_path}")
    except Exception as e:
        print("Error saving HTML codes: ", e)
```


6.2 Tabellen

Tabelle 2 Zeitplan detailliert

Prozess	Teilprozess	Stunden
Definitionsphase 6 Stunden	Gespräch mit Auftraggeber	2
	Soll-Konzept erstellen	2
	Ist-Analyse durchführen	2
Planung 9 Stunden	Projektstrukturplan (PSP) erstellen	2
	Zeitabschätzungen vornehmen	2
	Klassenplanung durchführen	2
	Programmablaufplan (PAP) erstellen	3
Umsetzung 51 Stunden	Das Abrufen des HTML-Code der Webseite programmieren	5
	Die URL-Liste von der Hauptwebseite erstellen	5
	Die URL-Liste der untergeordneten Seiten erstellen	4
	Die Funktion zur Überprüfung vorhandener URLs implementieren	5
	Die Entfernung doppelter Einträge programmieren	3
	Die Funktion zur rekursiven Seitenanalyse programmieren	6
	Die Dateiliste externer Ressourcen erstellen	3
	Die Funktion zur Überprüfung externer Links erstellen	3
	Die Dateien in separates Verzeichnis ablegen	3
	Die Konvertierung der Inhalte in PDF oder HTML programmieren	3
	Die Fehler- und Warnungsprotokollierung programmieren	3
	Die Funktion zur Eingabe von URL und Speicherpfad programmieren	3
	Der Hauptfunktionalität durch Zusammenführung der Module implementieren	4
	Entwicklertests durchführen	1
Abschluss 14 Stunden	Fachliches Fazit ziehen	1
	Persönliches Fazit ziehen	1
	Zeitliches Fazit ziehen	1
	Erstellen der Projektdokumentation	9
	Durchführung eines Tests durch Pilotnutzer	1
	Projektübergabe	1

Tabelle 3 Gantt-Diagramm

Gantt-Diagramm		29.04.2024							30.04.2024							02.05.2024							03.05.2024							06.05.2024																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
		x	x							x	x																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															

Gantt-Diagramm		07.05.2024							08.05.2024							10.05.2024							13.05.2024							14.05.2024																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
		x	x	x																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					

Tabelle 4 Kostenplan

Personalkosten			
Personal	Stundenlohn, €	Stundenanzahl	Gesamtlohn, €
Eigener Lohn	13,50	80	1.080,00
Consulting Frau R.	53,50	2	107,00
Consulting Herr L.	53,50	2	107,00
Summe			1.294,00 €
Fixkosten Pauschal			
Kostenart	Kosten/St	Stundenanzahl	Gesamtkosten, €
Bürokosten	2,42	80	193,60
Kosten Arbeitsplatz PC	0,11	80	8,80
Kosten Kommunikation	0,09	2	0,18
Summe			202,58 €
Variable Kosten Pauschal			
Kostenart	Kosten/St, €	Stundenanzahl	Gesamtkosten, €
Materialkosten	0,07	80	5,60
Summe			5,60
Projektgesamtkosten			1.502,18€

Tabelle 7 Zeitlicher Soll-Ist-Vergleich

Prozess	Teilprozess	SOLL, St.	IST, St.
Definitionsphase 6 Stunden	Gespräch mit Auftraggeber	2	2
	Soll-Konzept erstellen	2	2
	Ist-Analyse durchführen	2	2
Planung 9 Stunden	Projektstrukturplan (PSP) erstellen	2	1,5
	Zeitabschätzungen vornehmen	2	2
	Klassenplanung durchführen	2	2
	Programmablaufplan (PAP) erstellen	3	3,5
Umsetzung 51 Stunden	Das Abrufen des HTML-Code der Webseite programmieren	5	2
	Die URL-Liste von der Hauptwebseite erstellen	5	4
	Die URL-Liste der untergeordneten Seiten erstellen	4	4
	Die Funktion zur Überprüfung vorhandener URLs implementieren	5	5
	Die Entfernung doppelter Einträge programmieren	3	4
	Die Funktion zur rekursiven Seitenanalyse programmieren	6	6
	Die Dateiliste externer Ressourcen erstellen	3	3
	Die Funktion zur Überprüfung externer Links erstellen	3	3
	Die Dateien in separates Verzeichnis ablegen	3	5
	Die Konvertierung der Inhalte in PDF oder HTML programmieren	3	3
	Die Fehler- und Warnungsprotokollierung programmieren	3	3
	Die Funktion zur Eingabe von URL und Speicherpfad programmieren	3	2
	Der Hauptfunktionalität durch Zusammenführung der Module implementieren	4	6
	Entwicklertests durchführen	1	1
Abschluss 14 Stunden	Fachliches Fazit ziehen	1	1
	Persönliches Fazit ziehen	1	1
	Zeitliches Fazit ziehen	1	1
	Erstellen der Projektdokumentation	9	9
	Durchführung eines Tests durch Pilotnutzer	1	1
	Projektübergabe	1	1

7 Quellen:

Fittkau und Ruf 2008

Fittkau, Thomas ; Ruf, Walter: Ganzheitliches IT-Projektmanagement. München : Oldenbourg
Wissenschaftsverlag, 2008. – ISBN 978–3–486–58567–4

Magnus Lie Hetland: Beginning Python: From Novice to Professional, Second Edition: APRESS, 2008