

Dokumentation & Projekttagebuch

Innovation Lab 3 Jahr 2024

Projekt: WebCrawler

Team: Deyaa Yousef, if21b021

1. Allgemeine Informationen

Projektname: Webcrawler

Supervisor: Harald Wahl

Innovation Lab 3 Sommersemester 2024

Projektteam:

Deyaa Yousef, if21b021@technikum-wien.at

Management-Summary des Projektes

Das Projekt "WebCrawler" zielt darauf ab, eine Anwendung zu entwickeln, die es Nutzern ermöglicht, Websites zu durchsuchen und PDF-Dokumente zu extrahieren. Die Hauptfunktionen umfassen das Crawlen von Webseiten, das Extrahieren und Analysieren von PDF-Dokumenten, die Erstellung von Wortstatistiken und Wortwolken. Ziel ist es, eine effiziente und benutzerfreundliche Lösung zur Datenextraktion und -analyse bereitzustellen.

Rahmenbedingungen und Projektumfeld

Das Projekt wurde unter Berücksichtigung der folgenden Rahmenbedingungen entwickelt:

- Verwendung von Python, JavaScript, HTML, CSS und Bootstrap.
- Einsatz von Flask als Webframework.
- Integration von PDFMiner und PyPDF2 für die PDF-Verarbeitung.
- Nutzung von Flask-Login für die Benutzerauthentifizierung.
- Sicherstellung der Usability und Performance der Anwendung.
- Nutzung von GitHub für die Versionskontrolle und VS Code als Entwicklungsumgebung.
- Verwendung von Trello für das Projektmanagement.

Semester-Roadmap

Sprint 1 (Woche 1):

Ziel: Einrichtung der Entwicklungsumgebung und Implementierung der Basisfunktionalitäten.

- Einrichtung der Entwicklungsumgebung (Python, Flask, VS Code).
- Initialisierung des GitHub-Repositories und Einrichtung von Trello.
- Implementierung der Benutzerregistrierung und des Loginsystems.
- Erstellung der grundlegenden HTML/CSS-Struktur mit Bootstrap.
- Einrichtung der Datenbank (SQLAlchemy) und User-Model.

Aufwand: 35 Stunden

Sprint 2 (Woche 2):

Ziel: Entwicklung der Crawling-Funktionalität und Integration der PDF-Extraktion.

- Implementierung der URL-Eingabe und Crawling-Logik.
- Integration von PDFMiner und PyPDF2 zur PDF-Extraktion.
- Speicherung der gefundenen PDF-Links in der Datenbank.
- Implementierung der Anzeige und Download-Funktion für gefundene PDFs.
- Durchführung erster Tests und Fehlerbehebungen.

Aufwand: 40

Sprint 3 (Woche 3):

Ziel: Erweiterung der Anwendung um Wortstatistiken und Wortwolken.

- Implementierung der Wortstatistiken für die extrahierten PDF-Dokumente.
- Erstellung der Wortwolken mit WordCloud.
- Integration der Suchfunktion zur Suche nach Wörtern in den PDFs.
- Optimierung der Benutzeroberfläche basierend auf den Testergebnissen.
- Durchführung weiterer Tests und Fehlerbehebungen.

Aufwand: 25

Sprint 4 (Woche 4):

Ziel: Finale Optimierungen, Dokumentation und Vorbereitung der Präsentation.

- Durchführung finaler Tests zur Sicherstellung der Funktionalität.
- Behebung aller verbleibenden Fehler und Optimierung der Anwendung.
- Erstellung der Projektdokumentation.
- Vorbereitung der Abschlusspräsentation.
- Abgabe des Projekts.

Aufwand: 50

Collaboration & Tooling

• Version Control: GitHub DYO1407/Web_Crawler (github.com)

• **IDE**: Visual Studio Code

• **Project Management**: Trello

Anmerkungen

- Das Projekt soll sicherstellen, dass die PDF-Dokumente korrekt und effizient verarbeitet werden, um genaue Wortstatistiken zu liefern.
- Es wird darauf geachtet, dass die Anwendung skalierbar ist und bei Bedarf um weitere Funktionen erweitert werden kann.

2. Projekt-Kurzbeschreibung

Ziele des Projekts:

Das Ziel des Projekts "WebCrawler" ist es, eine Webanwendung zu entwickeln, die es Nutzern ermöglicht, Websites zu durchsuchen und PDF-Dokumente zu extrahieren und zu analysieren. Die Anwendung soll folgende Funktionen bieten:

- Benutzerregistrierung und Login-System
- Eingabe von URLs zum Crawlen von Websites
- Einstellungen zur Definition der Crawling-Tiefe
- Anzeige und Download der gefundenen PDF-Dokumente
- Statistiken der häufigsten Wörter in den PDF-Dokumenten
- Erstellung einer Wortwolke aus den gesammelten Daten

Herausforderungen:

Die größten Herausforderungen bestehen darin, eine effiziente und genaue PDF-Extraktion zu gewährleisten und gleichzeitig eine benutzerfreundliche Oberfläche zu bieten. Zudem müssen die Wortstatistiken korrekt berechnet und die Wortwolken präzise erstellt werden.

Mehrwert für die Anwender:

Der größte Mehrwert besteht darin, dass Nutzer auf einfache Weise wertvolle Informationen aus großen Mengen an PDF-Dokumenten extrahieren und analysieren können. Dies spart Zeit und Aufwand und ermöglicht tiefere Einblicke in die Daten.

Projekt-Scope:

- Entwicklung einer benutzerfreundlichen Webanwendung
- Implementierung der Crawling- und PDF-Extraktionsfunktionen
- Erstellung von Wortstatistiken und Wortwolken

Nicht-Ziele:

- Verarbeitung anderer Dokumenttypen außer PDF
- Entwicklung einer mobilen Anwendung

Umsetzungsansatz:

- Verwendung von Flask als Webframework
- Einsatz von PDFMiner und PyPDF2 zur PDF-Verarbeitung
- Nutzung von JavaScript, HTML, CSS und Bootstrap f
 ür die Frontend-Entwicklung
- Integration von Flask-Login für die Benutzerauthentifizierung

3. Spezifikation der Lösung

Systemumfeld:

Die Lösung ist eine Webanwendung, die in einem standardmäßigen Webbrowser betrieben wird. Sie setzt auf Flask als Backend-Framework und verwendet eine relationale Datenbank für die Benutzerdaten und Crawling-Informationen.

Features (Funktionale Anforderungen):

- **Benutzerregistrierung und Login**: Sichere Registrierung und Authentifizierung der Nutzer.
- URL-Eingabe und Crawling: Eingabe von URLs und Definition der Crawling-Tiefe.
- **PDF-Extraktion**: Finden und Herunterladen von PDF-Dokumenten.
- Wortstatistiken: Berechnung der häufigsten Wörter in den PDF-Dokumenten.
- Wortwolken: Erstellung von Wortwolken basierend auf den Wortstatistiken.

Schnittstellen:

- **PDFMiner und PyPDF2**: Zur Extraktion von Text aus PDF-Dokumenten.
- **Flask-Mail**: Für die Passwort-Wiederherstellungsfunktion.

Qualitätseigenschaften, technische Anforderungen (Nicht-Funktionale Anforderungen):

- **Performance**: Effiziente Verarbeitung und Anzeige der Daten.
- Usability: Benutzerfreundliche Oberfläche und einfache Navigation.
- **Sicherheit**: Sichere Speicherung und Verarbeitung der Benutzerdaten.

Mockups und Visualisierungen:

- Login-Seite: Einfaches Formular zur Eingabe von E-Mail und Passwort.
- **Crawl-Seite**: Eingabeformular für URL und Crawling-Tiefe, Anzeige der gefundenen PDF-Dokumente.
- **Profil-Seite**: Anzeige der Crawling-Historie und Wortstatistiken.

4. Aufwandschätzung

Aufwandsschätzung für dieses Semester:

- **Sprint 1**: 35 Stunden
- **Sprint 2**: 30 Stunden
- Sprint 3: 25 Stunden
- **Sprint 4**: 36 Stunden

Ergebnis der Aufwandsschätzung:

Insgesamt ca. 126 Stunden für die Umsetzung der geplanten Funktionen und Features.

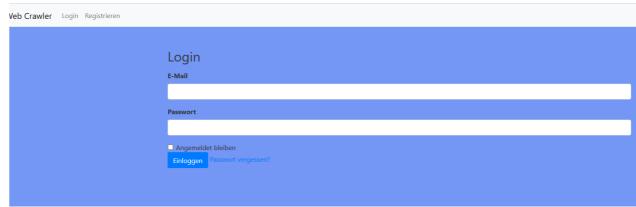
5. Auslieferung

Ein lauffägie Version des Projects wurde in einem Docker Container bereitgestellt um das programm zu auszuführen sollen Sie das Docker ausführen und das Project_ Directory in einem Terminal öffen und ein Image für das project "docker build -t web_crawler:latest . " und dann ein Container für den Server laufen lassen durch "docker run -p 5000:5000 web crawler:latest"

6. Unser Projekt-Tagebuch

Sprint 1 (Woche 1):

- **Kickoff-Meeting**: Einführung und Planung des Projekts.
- **Setup**: Einrichtung der Entwicklungsumgebung und Tools.
- **Implementierung**: Benutzerregistrierung und Login-System.
- Challenges: Sicherheitsaspekte bei der Benutzerauthentifizierung.
- Screenshots:



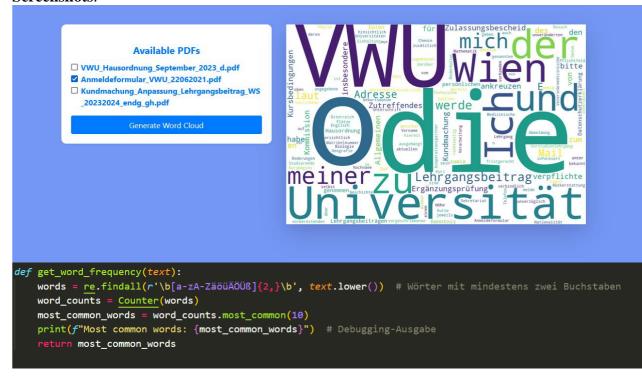
Sprint 2 (Woche 2):

- **Feature-Entwicklung**: URL-Eingabe und Crawling-Logik.
- **Integration**: PDFMiner und PyPDF2 zur PDF-Extraktion.
- **Testing**: Erste Tests und Fehlerbehebungen.
- Review: Rückblick auf den Fortschritt und Anpassungen der Roadmap.
- Screenshots:

Sprint 3 (Woche 3):

- Wortstatistiken: Implementierung der Wortstatistiken.
- Wortwolken: Erstellung der Wortwolken.
- **Suchfunktion**: Integration der Suchfunktion.
- Optimierung: Verbesserung der Benutzeroberfläche.

• Screenshots:



Sprint 4 (Woche 4):

- Finale Tests: Durchführung umfassender Tests.
- Fehlerbehebungen: Behebung aller verbleibenden Fehler.
- **Dokumentation**: Erstellung der Projektdokumentation.
- **Präsentation**: Vorbereitung der Abschlusspräsentation.

• Screenshots:

```
@app.route('/search_word', methods=['POST'])
@login required
def search_word():
   search_word = request.form.get('search_word').lower()
   matching_pdfs = []
    crawl_records = CrawlData.query.filter_by(user_id=current_user.id).all()
    for record in crawl_records:
       word_stats_list = []
       pdf_url_to_stats = {}
        for stat in record.word_stats.split(';'):
            if '|' in stat:
                pdf_url, word_counts = stat.split('|', 1)
                word_count_pairs = word_counts.split(',')
                word_counts_dict = {}
                for word_count in word_count_pairs:
                    parts = word_count.split(':')
                    if len(parts) == 2:
                        word, count = parts
                            word_counts_dict[word] = int(count)
                        except ValueError:
                pdf_url_to_stats[pdf_url] = word_counts_dict
        record.parsed_word_stats = pdf_url_to_stats
    for record in crawl_records:
```