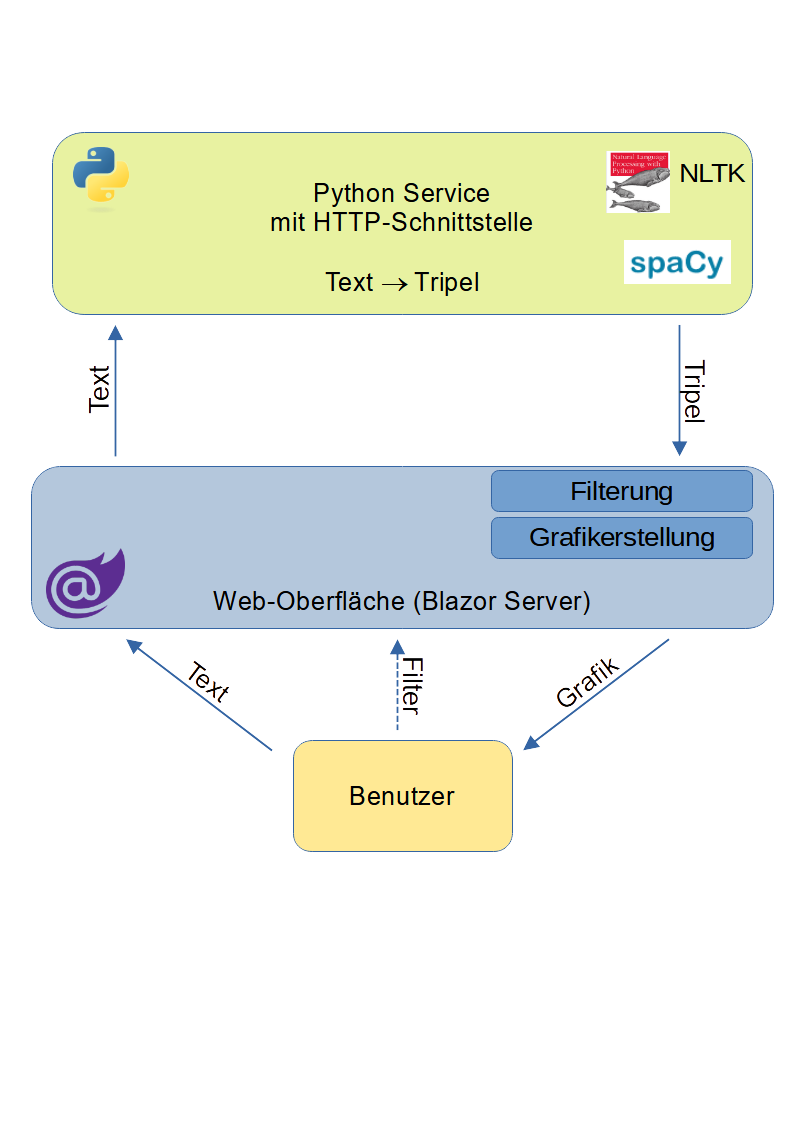
1. **Grobe Lösungsskizze**

Es soll eine Anwendung zur automatischen Generierung von *Concept Maps* aus einem vom Benutzer eingegebenen Text entwickelt werden.

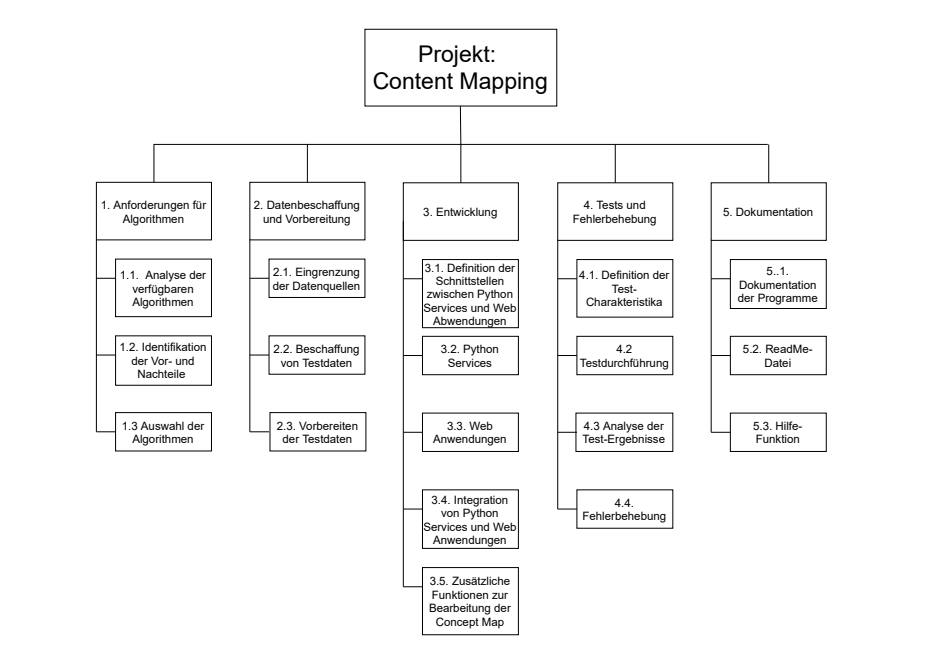
Die Benutzeroberfläche bildet eine Blazor (Server) Webanwendung, welche in C# geschrieben wird. Dies erleichtert die Entwicklung enorm, weil kein JavaScript-Code und keine separaten APIs für die Schnittstelle *Browser → Server* implementiert werden müssen.

Um die *Concept Map* aus einem Text zu generieren, wird zunächst ein Python-Service im Hintergrund aufgerufen:

* Zunächst werden aus dem Text die Tripel extrahiert. Jedes Tripel stellt die Beziehung von Wörtern untereinander dar. Dies soll mit Hilfe von Toolkits wie NTLK und spaCy geschehen.
* Anhand diesen Tripel kann dann die Grafik für die *Concept Map* erstellt werden. Der Benutzer soll jedoch die Möglichkeit haben, die Erstellung, z.B. mittels Filtern zu beeinflussen. Die vorher ermittelten Tripel könnten beispielsweise mittels Blocklist, Allowlist o.ä. gefiltert werden, bevor daraus die Grafik erstellt wird.
* Anschließend könnten die Elemente der Grafik automatisch und manuell angeordnet werden.

Die Bereitstellung (Deployment) soll mittels Docker-Containern ermöglicht werden. Mittels Docker Compose wäre das Deployment aller Container gemeinsam mit einem Konsolenbefehl machbar.

1. **Definition Work Breakdown Structure**



1. Analyse der Anforderungen (Klärung welcher Algorithmus)
2. Datenbeschaffung und -vorbereitung
3. Definieren der Schnittstellen zwischen Python und der Webanwendung
4. Entwicklung der Anwendungen
   1. Entwicklung der Python-Anwendung
   2. Entwicklung der Webanwendung
5. Integration der Python-Anwendung in die Webanwendung unter Nutzung der Schnittstellen
6. Implementierung von Funktionen zur Bearbeitung und Visualisierung von Concept Maps
7. Testing und Fehlerbehebung
8. Dokumentation
9. **Team** 
   1. **Wer kümmert sich um was?**

|  |  |  |
| --- | --- | --- |
| **Block** | **Detail** | **Hauptverantwortlich** |
| Analyse der Anforderungen | Literaturrecherche | alle |
| Aufbereitung der Ergebnisse in Vor- und Nachteilen |  |
| Auswertung und Auswahl der Algorithmen |  |
| Datenbeschaffung und -vorbereitung | Eingrenzung/Beschaffung/Vorbereitung | Adriana/Korvin |
| Definieren der Schnittstellen zwischen Python und der Webanwendung | gemeinsame Festlegung (da Grundlage der Webanwendung/Phython Anwendung | alle |
| Entwicklung der Python-Anwendung | NLTK | Adriana |
| PyTorch | Korvin |
| Entwicklung der Webanwendung |  | Sven |
| Integration der Schnittstellen | Integration der Python-Anwendung in die Webanwendung unter Nutzung der Schnittstellen | Sven |
| Visualisierung von Concept Maps | Erstellen von Concept Maps in Python (NLTK) in svg (zur Übermittlung in die Webanwendung) | Adriana |
| Erstellen von Concept Maps in Python (PyTorch ) in svg (zur Übermittlung in die Webanwendung) | Korvin |
| Implementierung von Funktionen zur Bearbeitung und Visualisierung von Concept Maps [Nice to have] | Sven |
| Testing und Fehlerbehebung | Festlegen von geeigneten Charakteristiken (Katalog) | Adriana oder Korvin |
| Planung und Durchführung von Testcases | Adriana oder Korvin |
| Dokumentation | Webanwendung | Sven |
| Python | Adriana oder Korvin |
| Statusupdate (Berichte) |  |
| Präsentation |  |
| Democase |  |

* 1. **Way-of-Working**
* Hauptverantwortliche benannt, aber gleichzeitig ist jedes Teammitglied in der Lage, den aktuellen Stand der Dinge präzise zu kommunizieren.
* Agile Entwicklungsmethode mit zweiwöchigen Statusreports (1 Seite)
* Folgende Tools werden genutzt: GitHub, Python, Blazor

1. **Grobe Zeitplanung (auf Milestones achten) -> alle**

Woche 1 (04.05. - 17.05.):

Analyse der Anforderungen und Klärung des geeigneten Algorithmus

Festlegung der erforderlichen Daten für die Anwendung

Erstellung eines detaillierten Projektplans mit Meilensteinen

Woche 2 (11.05. - 17.05.):

Datenbeschaffung und -vorbereitung (z. B. Daten sammeln, bereinigen, formatieren)

Definition der Schnittstellen zwischen Python und der Webanwendung

Woche 3-4 (18.05. - 12.06):

Entwicklung der Python-Anwendung

Entwicklung der Webanwendung

Woche 5-6 (01.06. - 14.06.):

Integration der Python-Anwendung in die Webanwendung unter Verwendung der definierten Schnittstellen (Parallel zur Entwicklung der Anwendungen)

Woche 7-8 (15.06. - 28.06.):

Durchführung von umfangreichen Tests und Fehlerbehebung

Erstellung der Projektdokumentation (z. B. technische Dokumentation, Benutzerhandbuch)

Milestones:

Abschluss der Anforderungsanalyse und Klärung des Algorithmus (17.05.)

Abgeschlossene Datenbeschaffung und -vorbereitung sowie definierte Schnittstellen (17.05.)

Entwicklung der Python-Anwendung abgeschlossen (12.06)

Entwicklung der Webanwendung abgeschlossen (12.06)

Erfolgreiche Integration der Python-Anwendung in die Webanwendung (14.06.)

Prototyp Pitches “Durchstich” (15.06)

Durchgeführte Tests und behobene Fehler (28.06.)

Vollständige Dokumentation des Projekts (11.07.)

Abschlussdemo (12.07)

1. **Beschreibung Datensammlung** (Was genau versteht er hier runter?) -> alle, Frage per Mail stellen

* Datensammlung aus einer bestimmten Domäne (z.B. Medizin)
* ~~Verwendung von RDF-Daten zur Generierung der Concept Maps (WIE?, ich kenne nur die Art über Texte und dann Extraktion etc.)~~
  1. Umfang (nur Datensammlung oder das Projekt?)
* Generierung von Concept Maps aus Text-Datei (bzw. RDF)
* User kann Concept Maps in der Webanwendung anzeigen und bearbeiten

5.2 Charakteristik (nur Datensammlung oder das Projekt?)

- englisch

5.3. Methode: (nur Datensammlung oder das Projekt?)

- Algorithmen

1. Evaluation & Test -> alle

Für den Test und die Evaluation der Concept Maps des fertigen Programms sollen quantitative und qualitative Kriterien verwendet werden, um den Erfolg des Projekts zu verifizieren. Hierdurch sollen auch die zwei Resultate der beiden Python Services (NLTK und PyTorch) mit einander verglichen werden, um mögliche Vor- und Nachteile der beiden Implementierungen identifizieren. Hierzu wird parallel zum laufenden Projekt ein Kriterien-Katalog entwickelt und ein Set von Testcases erstellt. Quantitative Kriterien könnten z.B. die korrekte Anzahl von identifizierten Objekten und Relationen zu einem Test-Text oder eine Verteilungsanalyse der Relationen sein. Qualitative Bewertungen können durch die subjektive Bewertungen der Maps z.B. in Hinsicht auf Übersichtlichkeit erfolgen.

Des Weiteren wird auch die Web-Anwendung mittels qualitativer Kriterien auf die Benutzerfreundlichkeit und mittels der Verwendung von Testcases auf ihre Zuverlässigkeit getestet.

1. Aufbereitung & Visualisierung ->alle

Die Dokumentation inklusive Readme-Datei wird bereits parallel zur Entwicklung der Programmteile geschrieben. Sowohl für die Zwischenpräsentation als auch der Abschlusspräsentation werden Demo-Cases vorbereitet, anhand derer die Funktionalitäten des Programms gezeigt werden können. Die quantitativen und qualitativen Ergebnisse der abschließenden Programm-Evaluation werden für die Abschlusspräsentation übersichtlich dargestellt.