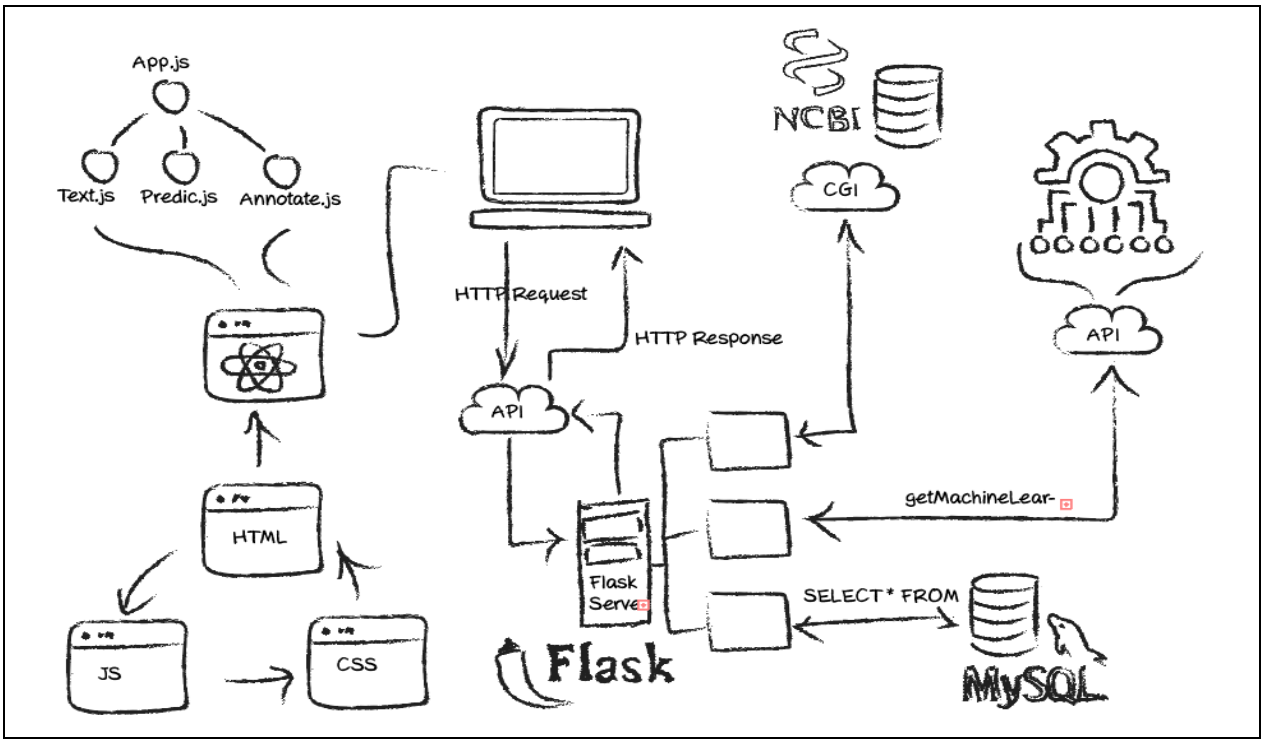
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

# Bachelorarbeit 2020

# Das Bereitstellen von Daten zur Analyse von Texten mittels Natural Language Processing und maschinelles Lernen




Student-in : Agron / Asani

Dozent : Henning / Müller

# Zusammenfassung

In der evidenzbasierenden Medizin ist das Formulieren einer Forschungsfrage eines der wichtigsten Teile in der Forschung. Zur Formulierung solcher Fragen soll der PICO Ansatz helfen. Veröffentlichte Literaturen werden anhand dieses Paradigmas evaluiert.

Das Begutachten von Dokumenten aus bibliografischen Datenbanken beansprucht sehr viel Zeit. Dieser Prozess soll mithilfe von Natural Language Processing (NLP) und Machine Learning (ML) automatisiert werden. Damit sich der Benutzer nicht noch mit der Auswertung der aus dem ML resultierenden Daten befassen muss, soll eine einfache und übersichtliche Benutzeroberfläche implementiert werden, die die gewonnenen Daten mittels komplexer Algorithmen verständlich veranschaulicht. Es stellt sich nun die Frage, auf welche Art die Daten am besten visualisiert werden können. Diese Bachelorarbeit befasst sich mit dem Thema des Erstellens einer Web-Anwendung von Grund auf. Dabei werden sämtliche Methoden der Softwareentwicklung berücksichtig. Am Ende wird die bestmögliche Art präsentiert, die folglich zur Beantwortung der Frage führt.

# Danksagung

Die vorliegende Bachelorarbeit entstand während meines Wirtschaftsinformatik-Studiums an der Hes-so Siders. Ein besonderer Dank gilt an Herr Müller Henning, den Betreuer dieser Arbeit, der mir bei Fragen stehts zur Seite stand sowie an Herr Schaer Roger, bei dem ich jeder Zeit während der Implementierungsphase um Rat fragen durfte.

Des Weiteren möchte ich mich bei Frau Dhrangadharya Anjani bedanken, die für das Machine Learning zuständig war und an alle anderen Personen, die mich während dieser Arbeit unterstützten.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

**PICO:** Population, Intervention, Calculation und Outcome

**ML:** Machine Learning

**NLP**: Natural Language Processing

**NCBI**: National Center for Biotechnology Information

**HTML**: Hyper Text Markup Language

**CSS**: Cascading Style Sheet

**JS**: JavaScript

**DOM**: Document Object Model

**HTTP:** Hyper Text Transfer Protocol

**SOAP**: Simple Object Access Protocol

**XML**: Extensible Markup Language

**REST:** Representational state transfer

**URI**: Unique Resource Identifier

**URL**: Uniform Resource Locator

**JSON**: JavaScript Object Notation

**API**: Application Programming Interface

**SDLC**: Software Development Lifecycle

**IDE**: Integrated Development Environment

**ID**: Identifikationsnummer

**SQL**: Structured Query Language

# 1 Einleitung

Die Webentwicklung ist eine schnell-lebige Welt, die für Web-Entwickler stets neue Herausforderungen bereithält. Wie Thomas Steglich bereits sagte: «Es liegt in der Natur des Webentwicklers, sich über neue technologische Errungenschaften auf dem Gebiet der Hard- und Software zu informieren» (2009, S. 190). Wo früher das Internet hauptsächlich als Informationsquelle diente, wird es heute immer mehr zu einem Instrument der Arbeitswelt. Es werden vermehrt Anwendungen im Internet gelagert. Solche Anwendungen werden Web-Anwendungen oder auch Web-Applikationen genannt. Dadurch entsteht der Vorteil, dass Benutzer die Anwendung ganz einfach von einem Browser aus bedienen können, wobei sämtliche Funktionalitäten, die eine lokale Anwendung bietet, auch angeboten werden. Während der Entwicklung durchläuft die Software sechs Phasen: die Voruntersuchungs-, die Konzeptionierungs-, die Design-, die Implementierungs-, die Einführungs- und die Wartungsphase.

## 1.1 Motivation

Die eHealth Forschungsgruppe des Informationssystem Institut der Hes-so Wallis ist bereits an der Entwicklung des Machine Learnings tätig, die durch das Anlernen von Daten die Elemente von PICO (Population, Intervention, Calculation und Outcome) in Texten prognostizieren kann. Nun wird eine Schnittstelle benötigt, die die Daten aus dem Machine Learning dem Benutzer visualisieren kann. Dazu wird eine Web-Anwendung implementiert die als Kommunikationsbrücke zwischen dem Benutzer und dem ML dient.

## 1.2 Problemstellung

Wenn der Benutzer die Ergebnisse, die das Machine-Learning hergibt, nicht verwerten kann, wird das ganze System keinen Nutzen haben. Damit der Benutzer sich nicht noch mit der Auswertung der Ergebnisse befassen muss, soll eine benutzerfreundliche Oberfläche erstellt werden. Sämtliche Auswertungen sollen auf einem Blick klar und deutlich visualisiert werden. Damit soll die Barriere zwischen dem Nutzer und der Komplexität der Daten überwunden werden.

Zurzeit existieren sehr wenig solcher Systeme was dazu führt, dass keine Massgebende Benutzeroberflächen verwendet werden können. Nichtsdestotrotz soll diese Herausforderung durch den Fortschritt der Technologie verknüpft mit der eigenen Kreativität überwältigt werden.

## 1.3 Zielsetzung

Diese Bachelorarbeit befasst sich mit der Realisierung einer Web-Anwendung von Grund auf neu. Dabei werden die Gedanken sowie die gefassten Entschlüsse während den einzelnen Phasen widerspiegelt.

Zu den Hauptfunktionalitäten der Webanwendung gehört das Hochladen von Artikeln aus der Pubmed-Datenbank, das Generieren und Visualisieren der PICO Elemente sowie das Korrigieren der Prognosen und das Anlernen des Machine Learning. In dieser Arbeit wurden die Elemente Calculation und Output zusammengeführt.

# 2 Allgemeine Informationen

Dieser Teil der Arbeit beschreibt das benötigte Hintergrundwissen, um die Entschlüsse und Ziele der Arbeit zu verstehen.

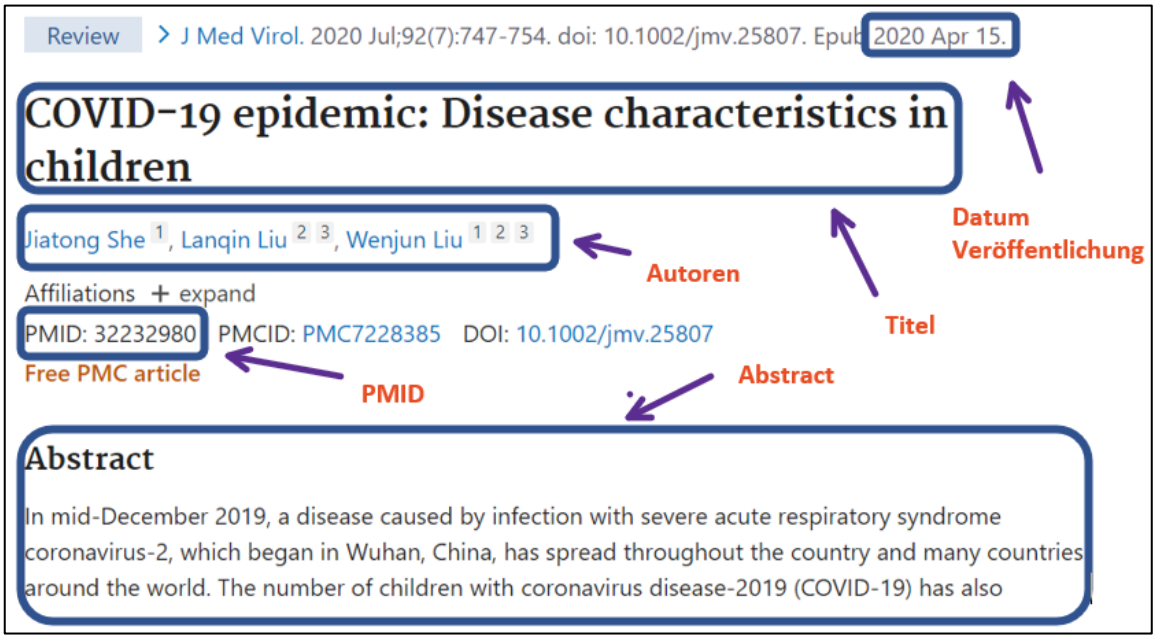
## 2.1 Auftraggeber

Das Forschungsinstitut für Informationssystemen des Hes-so Wallis spezialisiert sich für die Entwicklung von Informationssystemen für Unternehmen aller Branchen. Dazu gehören die Bereiche wie eHealth, eService, eGovernment, eEnergy und ERP. Zudem betreut das Institut Weiterbildungsprogramme in der Informationstechnologie.

## 2.2 National Center for Biotechnology Information

Das *National Center for Biotechnology Information* (NCBI) ist eine öffentliche Institution, die Informationen über biologische Sequenzen besitzt. Sie dient hauptsächlich für die Datenverarbeitung und Datenspeicherung in der Molekularbiologie. Die *Entrez Gene* ist die Datenbank der NCBI, in der die ganzen Informationen gespeichert werden. Zusätzlich bietet Entrez einen integrierten Zugang zu der *Medline* Datenbank. Die *Medline* Datenbank ist weltweit die wichtigste bibliographische Datenbank für Medizin und Biomedizin. Sie enthält über 22 Millionen Zitate aus ca. 5600 Zeitschriften. Der Zugriff auf diese Datenbank ist aus lizenzrechtlichen Gründen beschränkt. Eine freie zugängliche Version bietet Pubmed. Folgende Abbildung zeigt, welche Informationen ein Eintrag aus der Pubmed-Datenbank beinhaltet. Dabei wurden die wesentlichen Informationen, die für diese Arbeit benötigt wurden, umrandet.

**Abbildung 1: Pubmed-Dokument**




**Quelle: Eigene Darstellung**

## 2.3 Machine Learning

*Machine Learning*, zu Deutsch maschinelles Lernen, ist eine Art *künstliche Intelligenz* (KI), die von Daten anstelle von Programmen lernt. Mueller und Massaron (2017, S. 353) erläutern, dass oft die Begriffe maschinelles Lernen und künstliche Intelligenz synonym verwendet, was nicht so ganz korrekt ist. Die KI kann das maschinelle Lernen Umfeld umfassen jedoch umgekehrt ist das maschinelle Lernen mit der KI nicht deckungsgleich. Die KI kann Probleme selbstständig lösen, wobei das ML nur den Ansatz bietet, von Daten zu lernen. Die Algorithmen im Hintergrund identifizieren Mustern in den Daten und sind so in der Lage, auf sehr hohem Tempo prädikative Analysen zu verrichten. Damit das ML selbständig handeln kann, muss zuerst der Mensch die Algorithmen implementieren sowie das ML mit Daten versorgen.

Als Natural Language Processing (NLP) wird der Bereich in der KI bezeichnet, dass sich mir der menschlichen Sprache befasst. Dadurch sollen die Texte verstanden werden und anhand des ML die richtigen Antworten liefern.

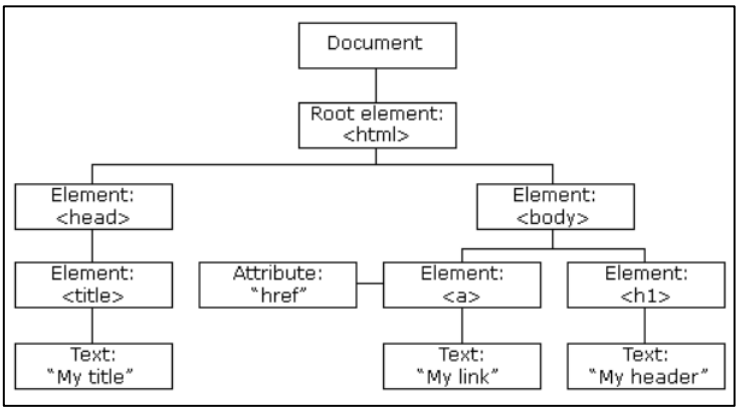
# 3 Technisches Grundwissen

Dieser Teil der Arbeit beschreibt sämtliche verwendetet Theorien, die im Verlaufe nicht erklärt werden.

## 3.1 Front- und Backend

Als Frontend wird der Teil einer Anwendung bezeichnet, welches der Benutzer in Form einer grafischen Benutzeroberfläche zu sehen bekommt, meistens durch einen Browser. Solche Benutzeroberflächen werden mit Programmiersprachen wie *Hyper Text Markup Language* (HTML), *Cascading Style Sheets* (CSS) und *JavaScript* (JS) erstellt. Alle drei Technologien sind eng miteinander verknüpft. Ein HTML Dokument enthält nur Text ohne Formatierung und endet mit .html. Alle Webseiten sowie Webanwendungen im Internet sind HTML Dokumente. Die Hauptaufgabe von HTML ist es, den Text in Elementen zu definieren, den der Benutzer sehen möchte. Um das Aussehen eines HTML-Dokuments zu verändern, wird CSS benötigt. Mit CSS ist es möglich, die einzelnen Elemente der HTML-Datei beliebig zu gestalten. Wenn eine Webseite geladen wird, wird gleichzeitig das *Document Object Model* (DOM) erstellt. Beim DOM handelt es sich um die Darstellung des HTML-Dokuments in einer Baumstruktur. Das nachfolgende Abbild repräsentiert ein DOM.

**Abbildung 2: Das Document Object Model**




**Quelle: W3Schools, URL: https://www.w3schools.com/js/js_htmldom.asp**

Durch JavaScript ist es möglich, Manipulationen am DOM durchzuführen. Die einzelnen Knoten in der Baumstruktur können dann verändert, hinzugefügt oder gelöscht werden. Diese

Aktionen führen dazu, dass das gesamte DOM aktualisiert werden muss. Folglich sieht der Benutzer am Browser die getätigten Änderungen.

Das Backend ist, im Gegensatz zum Frontend, der Teil der Anwendung, welches der Benutzer nicht zu sehen bekommt. Sie beinhaltet die Logik der Anwendung, mit der auf die Daten in einer Datenbank zugegriffen sowie verarbeitet werden. Die Interaktion zwischen dem Frontend und dem Backend der Anwendung kann durch das Client Server Modell realisiert werden, welches nachfolgend erklärt wird.

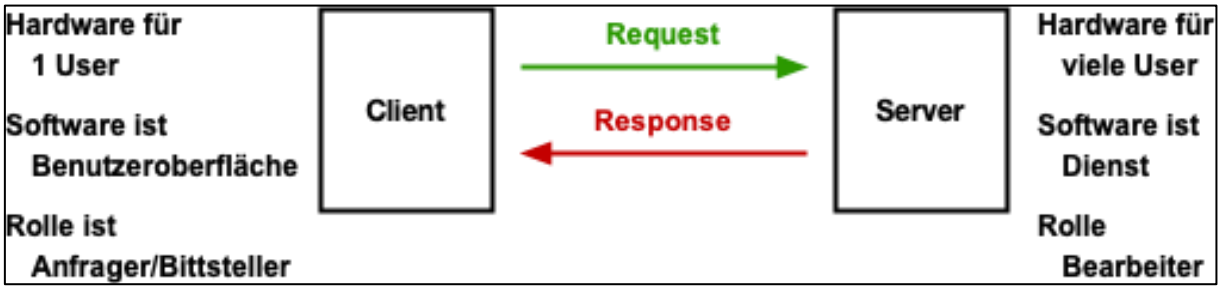
## 3.2 Das Client-Server Model

Beim Client-Server Modell handelt es sich um die Beziehung zwischen zwei Computer-Programmen: dem *Client*[1] Programm, oder auch Frontend, und dem *Server*[2] Programm, oder auch Backend genannt. Der Client fordert einen Dienst des Servers und der Server erfüllt dann die Anforderung. Der Server wartet permanent darauf, dass eine Anfrage eintrifft. Fordert der Client Daten vom Server, wird dies als Anfrage (Request[3]) bezeichnet. Reagiert der Server auf die Anfrage des Clients und sendet ihm Daten zurück, wird das als Antwort (Response[4]) bezeichnet. Beide Programme können an einem oder an verschiedenen Orten existieren.

Die Kommunikation zwischen beiden Programmen wird in Protokollen definiert, um den Ablauf der Anfragen und Antworten zu regeln. Eines dieser Protokolle ist das *Hypertext Transfer Protocol* (HTTP). Vom Client aus wird ein HTTP-Request an den Server versendet. Der Server verarbeitet die Anfrage und sendet die generierten Daten an den Client zurück. Das Zurücksenden wird HTTP-Response genannt. In der Abbildung 3 wird der Austausch der Informationen mittels HTTP veranschaulicht.

[1] Englisch für Auftraggeber
[2] Englisch für Zusteller
[3] Englisch für Anfrage
[4] Englisch für Antwort

**Abbildung 3: Das Client-Server Modell**




**Quelle: Elektronik Kompendium - Client-Server-Architektur, URL: https://www.elektronik-kompendium.de/sites/net/2101151.htm**

Anhand dieses Modells wird das Frontend nicht mit der Funktion der Datenverarbeitung belastet. Das heisst, das Frontend kann sich nur auf die Benutzeroberfläche konzentrieren, um so eine optimale Benutzerfreundlichkeit zu erlangen. Sämtliche Verarbeitungen und Datenbankzugriffe werden dann im Backend ausgeführt. Ein Vorteil, der dabei rausspringt, ist die Verbindung zwischen mehreren Clients an einem Server.

## 3.3 Representational state transfer

Neben dem Menschen wollen auch Anwendungen als Benutzer im Internet auf Daten zugreifen. Gewisse Dienste können im Internet für andere Anwendungen zur Verfügung gestellt werden. Solche Dienste werden Webservices genannt.

Bis vor einigen Jahren war das Protokoll *Simple Object Access Protocol* (SOAP) der Standard für Zugriffe auf solche Webservices. Gustavo, Fabio, Harumi und Vijay (2004, S. 156) erklären, dass anhand von SOAP die Organisation von Informationen in *Extensible Markup Language* (XML) definiert wird, um so den Austausch zwischen den Endpunkten einer Kommunikation zu ermöglichen. Die Kommunikation zwischen der Anwendung und dem Webservice fand unter recht komplexen XML-Nachrichten statt, welche über das HTTP transferiert wurden. XML ist ein Datenformat das Text so speichert, dass der Computer den verarbeiten kann. Die Kombination aus XML und HTTP machte das ganze plattformunabhängig[5] jedoch wurden die komplexen XML-Nachrichten dem ganzen System zum Verhängnis.

[5] Kompatibilität eines Programms mit unterschiedlichster Hard- und Software

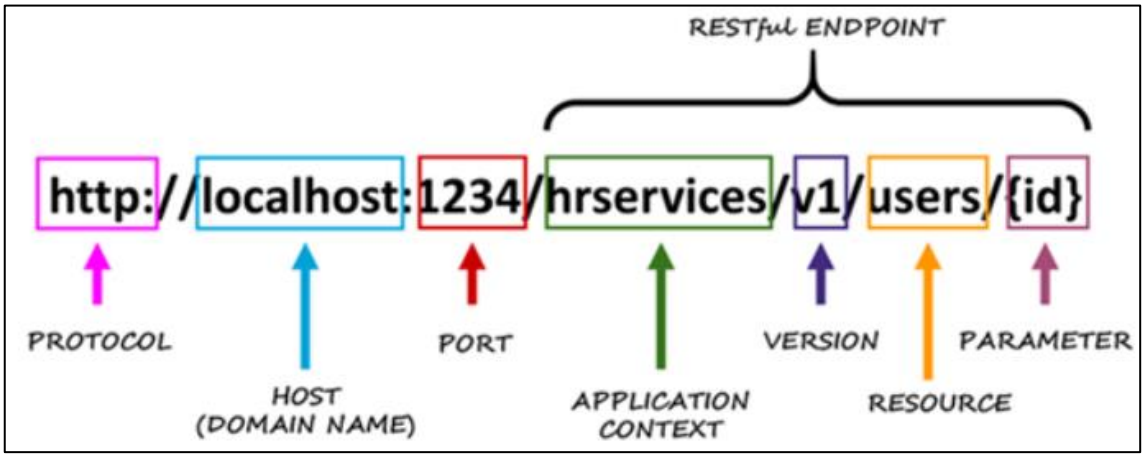
Die Entwickler suchten nach einer einfacheren Lösung und so wurde dann im Jahr 2000 der Begriff REST (*Representational state transfer*) von Roy Fielding geprägt. Die Idee: Die Architektur soll unabhängig jeglicher Protokolle funktionieren. Dabei wird dies dadurch erreicht, dass einzelne Komponente wie der Benutzer, als Ressourcen angesehen werden. Helmich (2013) beschreibt, welche Anforderungen, laut Roy Fielding, eine Ressource erfüllen muss (www.mittwald.de):

- Adressierbarkeit: Ressourcen müssen über eine bestimmte *Unique Resource Identifier*[6] (URI) identifiziert werden können.
- Zustandslosigkeit: Bei jeder Anfrage werden alle Informationen wieder neu mitgeschickt.
- Einheitliche Schnittstelle: Ressourcen müssen über die Standard Methoden von HTTP aufgerufen werden können.
- Entkoppelung von Ressourcen und Repräsentationen: Ressourcen können in verschiedenen Formaten wie XML oder JSON geholt werden.

Der Zugriff auf REST-Webservices geschieht über das HTTP Protokoll. Die Kommunikation findet nach dem Client-Server Prinzip statt. Um die Aufgabe des Servers zu erleichtern, adressiert der Client die Datei, auf die er Zugreifen will. Dazu übermittelt er dem Server im HTTP-Header eine *Uniform Resource Locator*[7] (URL). Das nachfolgende Abbild zeigt, wie eine URL aufgebaut ist.

[6] kompakte Abfolge von Zeichen, die eine abstrakte oder physische Ressource identifizieren
[7] Eine URI die zusätzlich auch ein Mittel zur Lokalisierung der Ressource angibt

**Abbildung 4: Aufbau einer URL**




**Quelle: Lansa – URL Anatomy, URL: http://apps.lansa.com/LearnLANSARESTAPI/index.html#!Documents/urlanatomy.htm**

In der URL ist klar ersichtlich, welches Transport-Protokoll verwendet wurde, auf welcher Domäne[8] unter welchem Port[9] zugegriffen wird und in welches Unterverzeichnis sich die Datei befindet.

Im Header wird auch die Methode definiert, welche bestimmt, wie die Daten verarbeitet werden sollen. Das Protokoll bietet bereits Methoden an, wie auf solche Ressourcen zugegriffen werden soll. Im Folgenden werden die Methoden erklärt, welche für diese Arbeit wichtig waren:

- GET: Dient dazu, Informationen aus dem Server zu holen. Darf nicht für die Verarbeitung von Daten im Server gebraucht werden.
- POST: Mit Post werden neue Daten im Server erstellt.
- DELETE: Durch DELETE werden Daten im Server gelöscht.

## 3.4 Application Programming Interface

Das *Application Programming Interface* (API) ist eine Schnittstelle, die den Austausch von Informationen zwischen einer Anwendung und einzelnen Programmteilen ermöglicht. Die API bietet somit den Entwicklern ein Mittel zur Zerlegung eines Systems in kleineren Modulen[10].

[8] Bezeichnet eine Gruppierung von Netzwerken
[9] Verbidungspunkt einer Domäne oder Anwendungsprogrammen
[10]

Dadurch werden komplexe Programme einfacher gewartet. Zudem definiert die API, wie die Informationen und Daten von den Modulen entgegengenommen und wieder zurückgegeben werden. Wo die Benutzeroberfläche die Schnittstelle zwischen dem Benutzer und der Anwendung darstellt, agiert die API als Schnittstelle zwischen Modulen. Jeder Web-Service ist eine API, aber nicht jede API ist ein Web-Service. Web-Services sind an Protokollen wie SOAP oder Ansätzen wie REST verbunden. APIs hingegen können über weitere Protokolle kommunizieren.

## 3.5 JSON

JSON steht für *JavaScript Object Notation* und dient als alternative zu XML. Mit JSON ist es um einiges einfacher Daten für das Speichern und den Austausch vorzubereiten. Den Vorteil den JSON mit sich bringt ist die Sprachenunabhängigkeit. Heisst, JSON-Daten können auch von anderen Programmiersprachen generiert werden. Die Syntax in JSON wurde sehr schlicht gehalten. Sie enthält zwei Strukturen: Objekte und Arrays. Ein Objekt ist eine Sammlung von Daten und ein Array ist eine geordnete Liste von Daten. In der Nachfolgenden Abbildung ist ein Array «Benutzer» zu sehen welches ein Datenobjekt mit den Daten «Name» und «Vorname» enthält. Eckige Klammern zeichnen ein Array aus und geschweifte Klammern ein Objekt.

**Abbildung 5: Aufbau eines JSON Dokuments**

```
{
    "Benutzer": [
        {
            "Name": "Asani",
            "Vorname": "Agron"
        }
    ]
}
```

**Quelle: Eigene Darstellung in Anlehnung an W3Schools**

# 4 Methodologie der Softwareentwicklung

Die Entwicklung einer Software beinhaltet viele Teilschritte, welche am Ende zum fertigen Produkt führen. Diese Teilschritte werden auch Phasen genannt. Um sich einen Überblick zu

diesen Phasen zu verschaffen, soll das Framework *Software Development Lifecycle* (SDLC) den Entwicklern behilflich sein.

## 4.1 Software Development Lifecycle

Wie es der Name bereits sagt, geht es beim SDLC um den Lebenszyklus der Entwicklung einer Anwendung. Damit soll sichergestellt werden, dass eine qualitativ hohe Software entwickelt wird. SDLC kann als ein Prozess angesehen werden, welches folgende Phasen beinhaltet:

- **Konzeptionierung**: In der ersten Phase geht es darum, die Probleme identifizieren zu können. Stakeholders[11] wie Kunden und Programmierer sorgen für den angemessenen Input. Diese Phase soll zudem einen Überblick über das gesamte Projekt liefern können sowie über die möglichen Probleme, die das Projekt auslösen kann
- **Analyse**: In dieser Phase werden die Kosten sowie die benötigten Ressourcen definiert. Zudem werden die möglichen Risiken des Projekts in Betracht gezogen, um Pläne zu erstellen, welche diese Risiken vermeiden sollen.
- **Design**: Bei der Umsetzung werden die Software Spezifikationen in einem Plan umgewandelt, welches dann die Stakeholder betrachten und kommentieren können.
- **Implementierung**: Zu diesem Zeitpunkt startet die Entwicklung der Software. Sämtliche Dokumente werden erstellt, die den Entwicklern während der Implementierung helfen sollen.
- **Veröffentlichung**: Nachdem das Programm geschrieben wurde, wird der code getestet, um sicherzustellen, dass auch wirklich das erreicht wurde, was man erreichen wollte.
- **Instandhaltung**: Nun wird die Software veröffentlicht damit sie auch für Dritte zugänglich ist. Meisten läuft nicht alles nach Plan. Während der Benutzung der Software können einige Bugs auftreten.

[11] Personen, die vom Projekt direkt oder indirekt betroffen sind

## 4.2 Agile Methodologie

In der agilen Methodologie der Softwareentwicklung wird der Ablauf der SDLC Phasen immer wieder und wieder durchgeführt. Das Wiederholen der Phasen ermöglicht beispielsweise einen mehrfachen Kontakt mit dem Kunden, was wiederum für mehr Feedback vom Kunden aus sorgt. Somit wird die Software schrittweise verbessert. Das Ziel ist es, nach jeder Iteration ein fertiges Teilprodukt zu implementieren, das bereits an den Kunden geliefert werden kann. Durch die Flexibilität haben die agilen Methoden in den letzten Jahren für einen bemerkenswerten Aufsehen in der Softwareentwicklung sowie im Projektmanagement gesorgt.

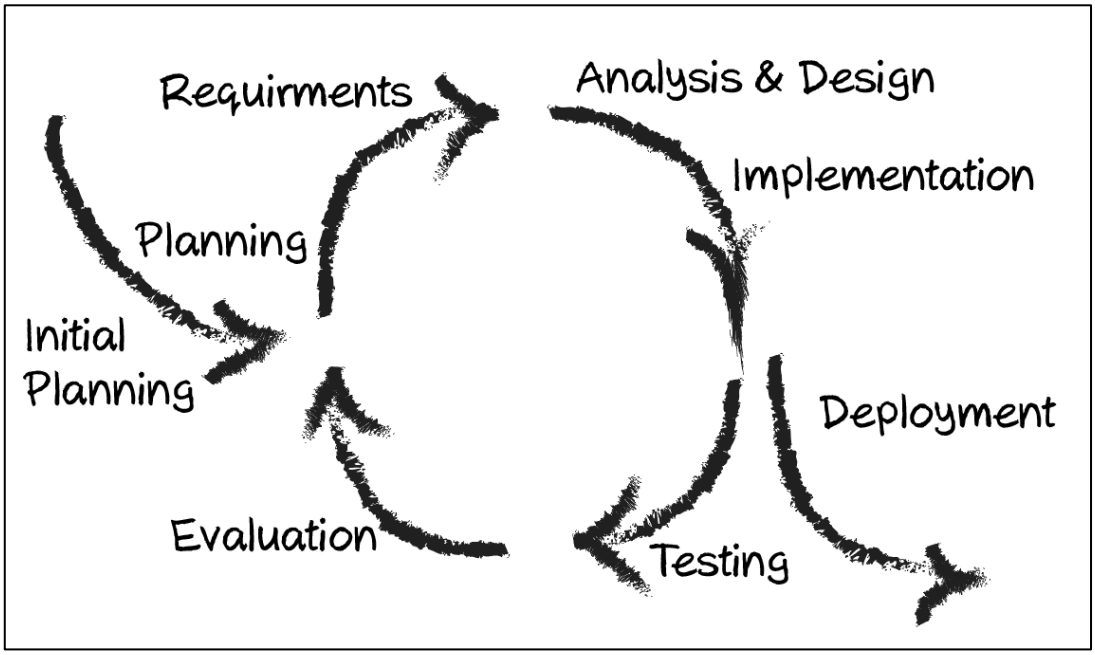
### 4.2.1 Scrum

Ein Framework, welches die agile Methodologie benutzt ist Scrum. Die Idee hinter Scrum liegt darin, dass das Team komplexe Prozesse bewältigt und dennoch produktiv unterwegs sein kann. Dies wird dadurch erreicht, dass Prozesse so klein wie möglich gehalten werden.

Die einzelnen Funktionalitäten der Anwendung werden als User Stories angesehen. Eine User Story beschreibt genau und in einer bestimmten Syntax die Funktionen der Anwendung aus Sicht des Benutzers. Jede User Story verfügt zusätzlich über ein Akzeptanzkriterium, die den Entwicklern mitteilt, ob eine User Story erfolgreich umgesetzt wurde. Der Aufbau einer User Story sieht wie folgt aus: «Als ein <*Benutzer typ*> möchte ich <*Tätigkeit verüben*> so dass ich <*gewünschtes Ergebnis*>». In den eckigen Klammern wird jeweils der Wert angepasst. Alle User Stories sind dann im *Porduct-Backlog* zu finden. Das *Product-Backlog* ist eine Liste mit priorisierten User Stories welche in der Entwicklung implementiert werden müssen. Die Einschätzung der Implementierungsschwierigkeit wird in Zahlen angegeben. Der Fachbegriff für diese Zahlen lautet *Story Points*. Je grösser die *Story Points*, desto schwieriger ist die Implementierung der *User Story*. Die Einschätzung der *Story Points* erfolgt im Team. Das ganze Team einigt sich, welche *User Story* am schnellsten und am einfachsten implementiert werden kann. Aufbauend auf die *User Story*, werden dann die Story Points den restlichen User Stories verteilt. Für gewöhnlich bestehen *Story Points* aus Zahlen der Fibonacci-Folge (0 ,1 ,1, 2, 3, 5, 8, 13, …).

Eine Iteration der Phasen wird als *Sprint* bezeichnet. Im *Sprint Planning* entschiedet das Team mit der Unterstützung des *Product Owners*, welche der *User Stories* innerhalb eines *Sprints* vollständig realisiert werden können. Die ausgewählten *User Stories* werden dann in das *Sprint Backlog* übernommen. Somit sind die zu einem *Sprint* gehörenden *User Stories* im *Sprint-Backlog* zu finden. Im nachfolgenden Abbild werden die einzelnen Phasen einer Iteration dargestellt.

**Abbildung 6: Ablauf der Iterationen in Scrum**




**Quelle: Eigene Darstellung in Anlehnung an Cotting, 2019, S. 29**

Während der Entwicklung mit Scrum stösst man auf weitere verschiedene Standartbegriffen sowie auf Begriffen aus der agilen Methodologie. Nachfolgend werden alle Begriffe gegliedert erklärt.

#### *4.2.1.1 Rollen*

Am Prozess sind drei Rollen beteiligt:

- **Product Owner**: Der Produkt Owner erhält die Rolle des Auftraggebers. Er sorgt für die Priorisierung der Anforderungen.
- **Scrum Master**: Er sorgt dafür, dass die Scrum-Regeln gehalten werden. Er ist die einzige Ansprechperson für den Product Owner und sorgt im Team für eine hohe Produktivität.

- **Team**: Das Team besteht meistens aus Mitarbeitern mit unterschiedlichen Stärken. Es soll alle Qualifikationen besitzen, um das Projekt erfolgreich beenden zu können.

#### *4.2.1.2 Artefakten*

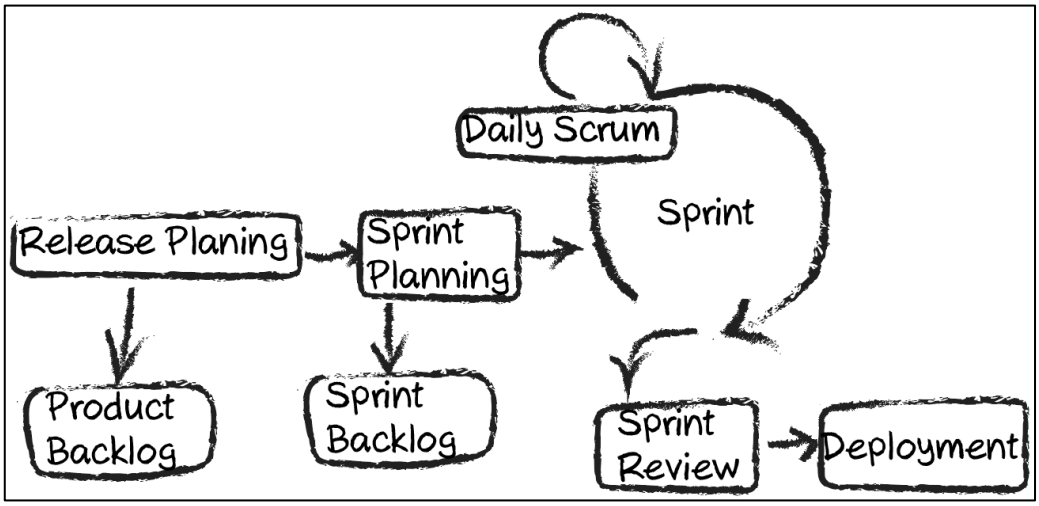
- **Product Backlog**: Im Product Backlog sind alle Funktionen enthalten, die das Entwicklerteam implementieren muss. Auch Bugs und Verbesserungen sind darin zu finden. Auch wenn es am Anfang sehr aufwendig ist das Product Backlog zu erstellen, darf die Wartung nach dem Projekt Start nicht vergessen werden.
- **Sprint Backlog**: Im Sprint Backlog sind alle Funktionen enthalten, die das Entwicklerteam innerhalb des Sprints implementieren müssen. Jeder Sprint verfügt über einen eigenen Sprint Backlog.
- **Produktinkrement**: Das Produktinkrement stellt die Anwendung dar, welches über die definierten Anforderungen im Produktbacklog verfügt. Nach jedem Sprint sollte die Anwendung mehr Funktionen bieten können.

#### *4.2.1.3 Ereignisse*

- **Sprint Planning**: In der Sprint Planning wird das Zieledefiniert, das am Ende eines Sprints erreicht werden sollen. Zu Beginn jedes Sprints findet ein Sprint Planning statt.
- **Daily Scrum**: Täglich findet eine 15 Minuten lange Besprechung im Team. Jedes Teammitglied sollte folgende Fragen beantworten können:
    - Welche Aktivitäten habe ich seit dem letzten Daily Scrum abgeschlossen?
    - Woran werde ich bis zum nächsten Daily Scrum arbeiten?
    - Wo könnten die Probleme liegen?
- **Sprint Review**: Am Ende jedes Sprints findet das Sprint Review statt. Der Sprint Review ist eine Besprechung, mit dem Ziel, das entstandene Produkt zu werten. Der Product Owner prüft, ob wirklich alle Funktionen enthalten sind.
- **Sprint Retrospektive**: Das Ziel in dieser Besprechung ist, die Zusammenarbeit sowie die Prozesse zu verbessern.

Die Nachfolgende Abbildung zeigt den Zusammenhang der einzelnen Begriffe in Scrum.

**Abbildung 7: Ablauf Scrum**




**Quelle: Eigene Darstellung in Anlehnung an Seifried, 2012**

# 5 Design und Konzeption der Anwendung

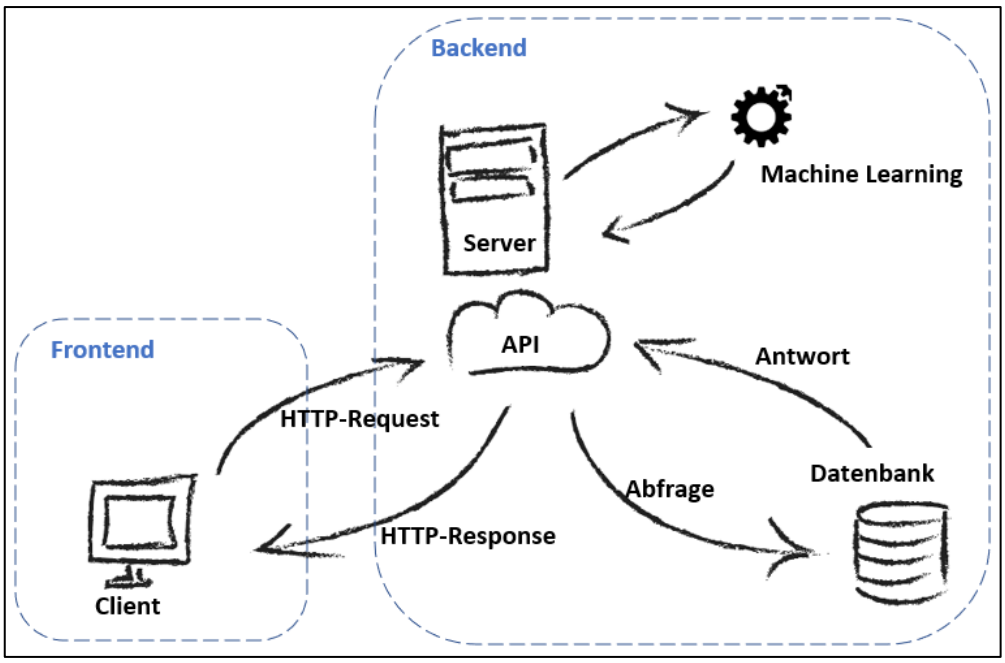
Das Ziel der Arbeit ist eine Web-Anwendung zu implementieren, welches dem Benutzer die generierten Ergebnisse der ML Algorithmen vereinfacht darstellt. Dabei soll geachtet werden, dass die Daten so einfach und verständlich wie möglich dargestellt werden. Dafür wird ein Frontend, ein Backend sowie eine Datenbank benötigt. Im Backend befindet sich die API, die für die Anfragen vom Frontend aus zuständig ist. Zudem leitet sie weitere Operationen wie die Verarbeitung der Daten oder den Zugriff auf die Datenbank aus. Im Frontend wird die URL des Pubmed-Dokuments eingegeben. Aus dem ML erstellte Prognosen werden im Text mit Hilfe von Farben gekennzeichnet. Zudem soll der Benutzer die Möglichkeit haben, erstellte Prognosen verändern zu können. Das Prinzip der Benutzeroberfläche beruht auf eine Single-Page-Webanwendung. Damit werden Web-Anwendungen bezeichnet, die sämtliche Informationen auf eine Seite darstellen. Folglich wird sich die Web-Anwendung auf einer Seite in drei Teilen gliedern. Der obere Teil befasst sich mit dem Hochladen von Dokumenten, der mittlere Teil mit der Visualisierung der Prognosen in Texten, und der letzte Teil mit der Verrichtung von Korrekturen um das ML neu anzulernen.

## 5.1 Architektur

Das Grundgerüst dieses System basiert auf das Client-Server Prinzip. Das Frontend verfügt hauptsächlich über Funktionen zur Visualisierung der Daten. Daten werden in Strukturen

gespeichert, welche dann den Funktionen im Frontend dienen. Vom Client aus gehen die Anfragen zum Server. Die Anfragen werden von der API im Server verarbeitet. Im Backend befindet sich die Logik und der Zugriff auf die Datenbank. In der Abbildung 12 ist ein Grobkonzept der einzelnen Komponenten und deren Kommunikation zu sehen.

**Abbildung 8: Grobkonzept der Web-Anwendung**




**Quelle: Eigene Darstellung**

## 5.2 Funktionen

Die Web-Anwendung verfügt über folgende Funktionen:

- **Dokumente hochladen**: In der Pubmed Datenbank sind alle Dokumente gespeichert. Der Benutzer kann die URL kopieren und in der Anwendung einfügen. Beim Hochladen wird die URL über eine HTTP-Anfrage in Form einer JSON-Datei an den Server versendet. Von dort aus wird das Dokument von der Pubmed Datenbank heruntergeladen. Da nicht alle Informationen benötigt werden, werden nur die wichtigsten Informationen ausgefiltert und in der Datenbank gespeichert
- **Dokumente anzeigen**: Der Benutzer hat die Möglichkeit, sämtliche Dokumente, die er hochgeladen hat, zu sehen. Aktualisiert er die Web-Anwendung, wird über eine HTTP-Anfrage der gesamte Inhalt der Datenbank vom Server verlangt. Der Server holt die

Daten, speichert die in Form einer JSON-Datei und sendet sie über eine HTTP-Antwort dem Client zu.

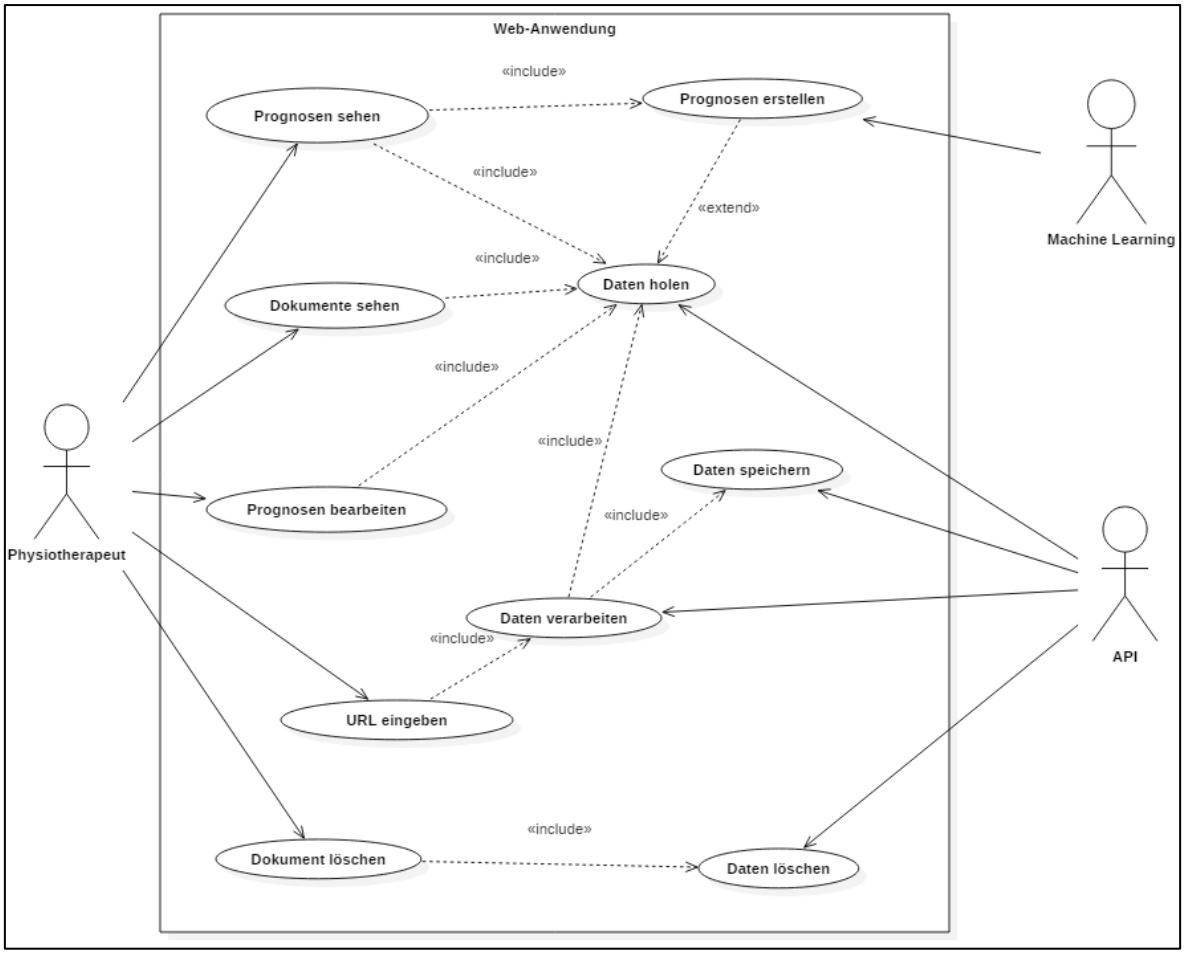
- **Dokumente löschen**: Der Benutzer hat die Möglichkeit, bereits hochgeladene Dokumente zu löschen. Hinter jedes Dokument befindet sich der *Button*[12] welches dies veranlasst. Über eine HTTP-Anfrage an den Server wird die *Identifikationsnummer* (ID) des Dokuments versendet. Von dort aus wird das gewünschte Dokument von der Datenbank gelöscht
- **Prognosen visualisieren**: Nach dem ein Dokument hochgeladen wurde, kann der Benutzer eine Prognose erstellen. Über eine HTTP-Anfrage wird die ID des Dokuments dem Server zugeteilt. Der Server holt lediglich den Abstrakt des Dokuments und leitet es an die ML Algorithmen weiter. Nach dem die Prognose erstellt wurde, werden die erstellten Daten in der Datenbank gespeichert. Über die HTTP-Antwort werden die Daten sowie der Abstrakt des Dokuments in Form eines JSON-Dokuments übermittelt. Im Frontend werden die Daten zusammengefügt. Anhand von CSS werden die Prognosen visualisiert
- **Prognosen kommentieren**: Sollten einige Prognosen nicht stimmen, kann der Benutzer die Prognose eines einzelnen Wortes ändern. Dabei klickt er auf das Wort im Text und kann dann im Dropdown Menü die richtige Prognose auswählen und speichern.
- **Modell anlernen**: Nach dem der Benutzer die Prognosen korrigiert hat, hat er die Möglichkeit, die Algorithmen neu anzulernen. Dabei werden über eine HTTP-Anfrage die veränderten Daten in Form einer JSON-Datei an den Server geleitet. Danach gelangen die Daten im ML. Der Algorithmus wird neu trainiert.
- **Veränderungen speichern**: Das ML kann mit seinen Prognosen auch falsch liegen. Dafür steht die Funktion *Kommentare* dem Benutzer zur Seite. Die einzelnen Fehlprognosen können ganz einfach korrigiert und das ML neu angelernt werden. Wurden alle Prognosen korrigiert, kann der Benutzer die Prognose speichern. Im Frontend wird die zusammengefügte Liste in drei Listen unterteilt. Über eine HTTP-

[12] Englisch für Schaltfläche, die der Benutzer auf der Benutzeroberfläche betätigen kann

Anfrage werden die veränderten Daten in Form einer JSON-Datei an den Server geleitet. Von dort aus werden die Daten in der Datenbank gespeichert.

Nachfolgende Abbildung veranschaulicht sämtliche Funktionen anhand eines Use Case Diagramms.

**Abbildung 9: Use Case Diagramm mit den Funktionalitäten**




**Quelle: Eigene Darstellung**

# 6 Analyse

Die Erstellung von Web-Anwendungen erfordert ein hohes Mass an Konzeptionierung. Das wird mit der Frage, was eine Web-Anwendung genau ist, verdeutlicht. Eine Web-Anwendung ist eine Dynamische Seite auf einem Browser. In Kombination mit einem Backend erlaubt eine Web-Anwendung dem Benutzer eingegebene Daten im Backend zu verarbeiten oder eine Verbindung zur Datenbank herzustellen und somit die generierten Ergebnisse im Browser zu

veranschaulichen. Neben dem Frontend müssen sich Web-Entwickler auch mit dem Backend befassen. Daher muss die Erstellung der Web-Anwendungsarchitektur genau geplant werden. Daten müssen vom Server wie auch vom Client verstanden werden, Datensätze können in einer Datenbank verändert, erstellt und gelöscht werden etc. Dem Entwickler stehen dafür eine Menge Tools zur Verfügung. Die Auswahl der richtigen Programme erfordert eine genaue Untersuchung. Die Programme müssen sämtliche Funktionalitäten bieten können, um das gewünschte Ziel zu erreichen. Vergleichsportale sowie die Erfahrung der Entwickler, die bereits mit den Programmen gearbeitet haben, beispielswiese durch Foren, sind ein wichtiger Faktor, der bei der Entscheidung mitberücksichtig werden sollte. Im Nachfolgenden werden die Tools erklärt, die aus der Analyse als bestmögliche Optionen resultierten und während der Implementierungsphase dieser Arbeit genutzt wurden.

## 6.1 IDE

IDE steht für *Integrated Development Environment*, zu Deutsch; integrierte Entwicklungsumgebung. Sie bietet den Entwicklern die wichtigsten Elemente für die Softwareentwicklung. Der Entwickler kann unter einer Benutzeroberfläche Tools wie Quelltextformatierung, Compiler, Debugger etc. finden. Es herrschen eine Vielzahl von IDEs mit unterschiedlichsten Funktionen.

### 6.1.1 Webstorm

Webstorm ist eine IDE für die Programmiersprache JavaScript welches von dem Unternehmen JetBrains entwickelt wurde. Sie unterstützt zudem andere Technologien wie HTML5, Node.js, Bootstrap und React.js. Dank der Unterstützung vieler Web-Technologien, wird Webstorm vorwiegend für die Entwicklung von Web-Anwendungen benutzt.

## 6.2 Frontend

Das Frontend ist die einzige Interaktion, die der Benutzer zur Anwendung hat. Daher sollte bei der Gestaltung der Struktur (HTML), Aussehen (CSS) und Logik (JavaScript) sehr viel Wert auf Klarheit und Übersichtlichkeit gelegt werden. Ein Benutzer greift auf die Web-Anwendung mit unterschiedlichsten Bildschirmauflösungen zu, was dazu verursacht, dass Web-Entwickler auch auf solche Aspekte achten müssen. Bevor es mit der Implementierung beginnen kann, sollte die Benutzeroberfläche von Hand oder anhand von Tools skizziert werden.

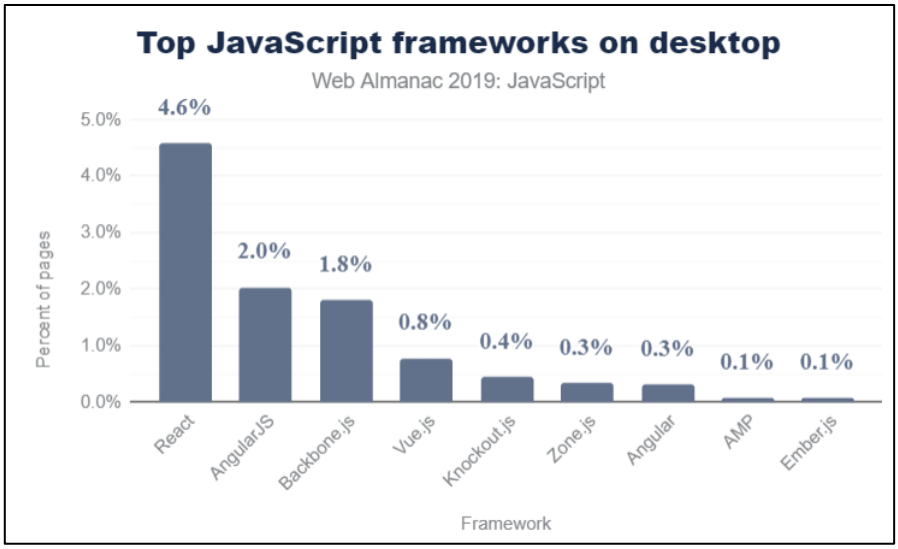
### 6.2.1 Moqups

Mockups sind Skizzen der Benutzeroberfläche, die zu Beginn erstellt werden. Sie dienen hauptsächlich als ein Leitfaden während der Implementierung. Moqups ist eine Web-Anwendung, die das Erstellen solcher Skizzen digital ermöglicht. Sie bietet verschiedene Werkzeuge, die das Gestalten der Skizzen so Realitätsnah wie möglich machen.

### 6.2.2 React JS

React ist ein von Facebook entwickelte Open-Source JavaScript Framework und wurde erstmals 2013 vorgestellt. Seit der Veröffentlichung wurde React stetig mit der Community weiterentwickelt. Wegen ihrer Einfachheit wurde React über die Jahre immer beliebter bei den Entwicklern. Das Nachfolgende Abbild zeigt eine Statistik, die von Almanac erstellt wurde. Darin ist die Verwendung von React in Webseiten im Vergleich zu anderen Frameworks zu sehen.

**Abbildung 10: Statistik über der Anzahl der Webseiten in Prozent die das Framework nutzen**




**Quelle: Almanac – JavaScript, URL: https://almanac.httparchive.org/en/2019/javascript#fig-12**

Die Stärken von React sind die Einfachheit und die Anpassungsfähigkeit. Zusätzlich enthält React einige bemerkenswerte Merkmale wie *Komponente* und *Virtuelles Dokument Objekt*, die das Framework so beliebt bei den Entwicklern macht.

#### *6.2.2.1 Komponente:*

Die Benutzeroberfläche wird in React in einzelnen Komponenten unterteilt. Sie werden zum Teil ineinander verschachtelt, um so in Form einer Baumstruktur die App aufzubauen.

Jede Komponente kann eine *Rendering* Methode aufweisen, die Daten entgegennimmt und zurückgibt. Eine Komponente, die als eine Art Container für andere Komponenten dient, wird Parent-Komponente genannt und alle darin befindlichen Komponenten heissen Child-Komponente. Komponente sind zudem in der Lage über ihren eigenen *state* zu verfügen.

#### *6.2.2.2 State*

In React ist *state* ein Objekt, dass Teile der Anwendung, welche sich verändern können, repräsentiert. Jedes Mal, wenn sich das *state* Objekt verändert, verändert sich das virtuelle DOM.

#### *6.2.2.3 Virtuelles DOM*

DOM ist eine API für HTML und XML-Dokumente, die dem Benutzer die Möglichkeit gibt, den Inhalt und die visuelle Darstellung verändern zu können. Der Nachteil liegt darin, dass bei jeder Veränderung, das gesamte DOM neu generiert wird. Das unnötige Verändern des gesamten DOMs zeichnet sich dann in der verlangsamten Geschwindigkeit aus. Hingegen wird im virtuellen Dom nur ein Abbild des «echten» DOMs dargestellt. Die deklarative API von React sorgt dann, dass der Benutzer den Zustand der Benutzeroberfläche bestimmen kann.

#### *6.2.2.4 Axios*

Mittels AXIOS kann React auf die API im Backend zugreifen. AXIOS ist eine Programmierbibliothek[13] (Library) die den HTTP-Client für den Browser darstellt. Damit wird der Austausch von Anfragen und Antworten mit dem Server ermöglicht.

## 6.3 Backend

React ist eine Frontend-Bibliothek und daher eine reine clientseitige Bibliothek. Für die Verarbeitung von Daten wird ein Backend benötigt. Damit React auf das Backend zugreifen kann wird eine REST-API im Backend erstellt die auf einem Server läuft. Mittels HTTP erhält die API Daten und delegiert sie für die Verarbeitung oder Datenbankzugriffe an die anderen Klassen weiter. Die komplette Logik im Backend wurde in Python geschrieben. Die gewonnenen Informationen aus der Logik werden dann wieder durch die API an das Frontend zurück versendet.

[13] Sammlung von Klassen und Funktionen

### 6.3.1 Flask Server

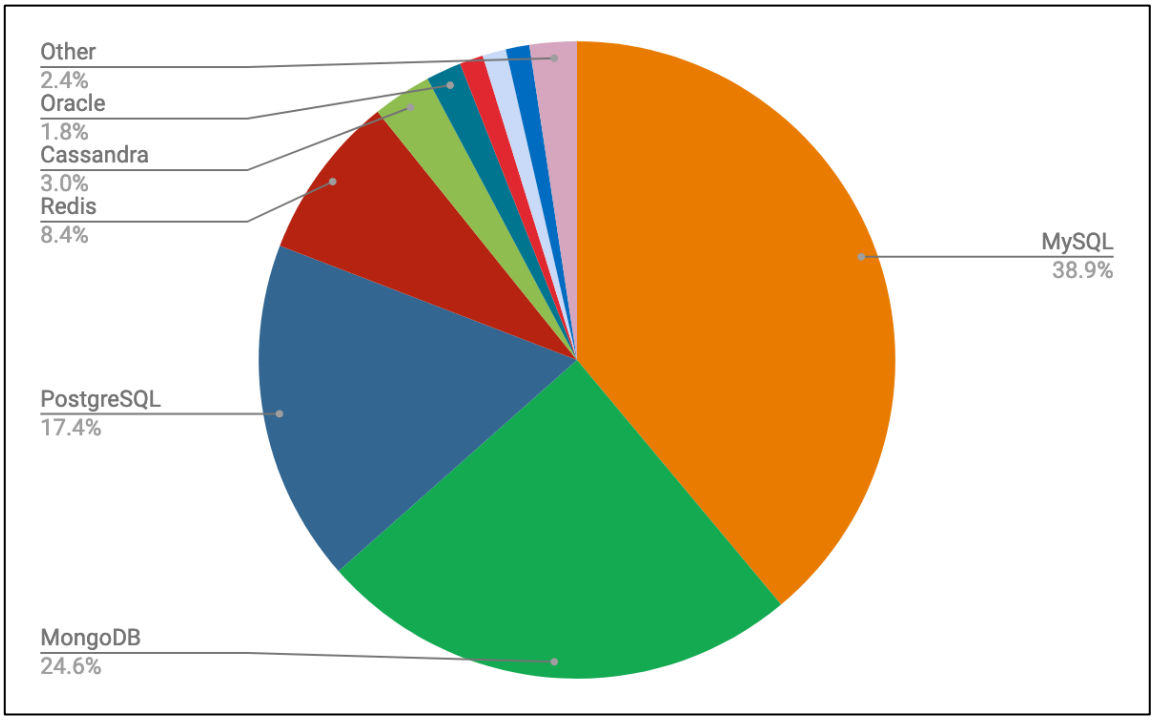
Flask ist ein Python Framework das dem Entwickler Tools wie Bibliotheken und Technologien bietet, um eine Web-Anwendung erstellen zu können. Das besondere an Flask ist, es gehört zu den Mikro-Framworks. Mikro-Frameworks sind dafür bekannt, dass sie keine Abhängigkeiten zu externen Bibliotheken aufweisen. Der Vorteil ist, durch geringe Abhängigkeiten müssen die externen Bibliotheken nicht aktualisiert werden. Flask wird durch die Einfachheit ausgezeichnet. Die Installation umfasst nur einige Schritte.

### 6.3.2 MySQL Datenbank

MySQL ist eine Open-Source[14] Datenbankverwaltungssystem, das 1995 zum ersten Mal eingeführt wurde. 2010 übernahm Oracle das System, behielt es dennoch als eine Open-Source. MySQL ist in der Lage, Daten schnell und performant zu speichern. Zudem ist die Software eine sehr populäre Lösung für die Implementierung von Datenbanken. Die Abbildung 10 zeigt in Form eines Kuchendiagramms die Beliebtheit der einzelnen Open-Source Datenbanksystemen.

[14] Der Quellcode einer Software ist für jeden zugänglich

**Abbildung 11: Die Beliebtheit von Datenbanksystemanbietern**




**Quelle: ScaleGrid - 2019 Database Trends – SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use, URL: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/**

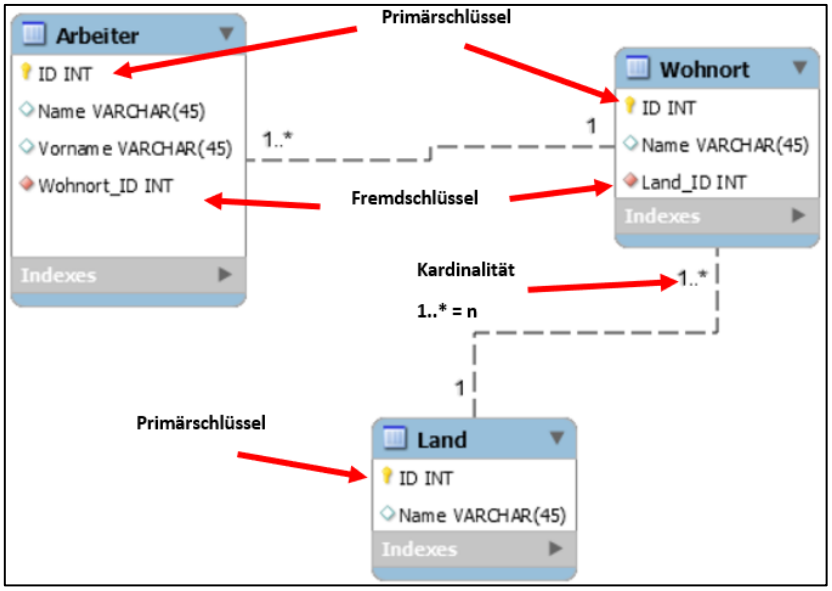
Daten werden auf unterschiedlichster Art und Weise in einer Datenbank gespeichert. MySQL verwendet den Ansatz einer Relationalen Datenbank. Anstelle die Daten einfach in einem Bereich zu speichern bietet eine relationale Datenbank mehrere separate Bereiche. Diese Bereiche werden Tabellen genannt. Ein Vorteil einer relationalen Datenbank ist, dass durch die Tabellen für weniger Redundanzen[15] gesorgt wird.

Mit Hilfe eines Schlüssels werden die Daten aus den Tabellen miteinander verknüpft. In der Fachsprache werden diese Schlüssel Primärschlüssel genannt. Ein Primärschlüssel wird für die eindeutige Identifizierung eines Eintrags in einer Tabelle verwendet. Das ermöglicht das Arbeiten mit verschiedenen Datensätzen aus verschiedenen Tabellen. Das Referenzieren auf andere Tabellen geschieht auch anhand eines Schlüssels. Dieser Schlüssel wird Fremdschlüssel genannt.

[15] Das mehrfach Vorkommen eines gleichen Datensatzeintrages

Unter den Tabellen herrschen gewisse Kardinalitäten. Kardinalitäten definieren die Anzahl Beziehungen, die eine Tabelle zur anderen hat. Kann ein Eintrag einer Tabelle nur einem Eintrag einer anderen Tabelle zugehören, spricht man von einer 1:1 Verbindung. Kann der Eintrag mehreren Einträgen der anderen Tabelle angehören, spricht man von einer 1:n Verbindung. Kann der Eintrag mehreren Einträgen aus der anderen Tabelle angehören und umgekehrt, ist von einer n:m Beziehung die Rede. In der Abbildung 11 sind die verschiedenen Kardinalitäten zwischen den Tabellen sowie die Primär- und Fremdschlüssel zu sehen.

**Abbildung 12: Die Kardinalitäten zwischen Tabellen**




**Quelle: Eigene Darstellung**

Die Daten einer MySQL Datenbank werden über Abfragen abgerufen, verarbeitet oder gelöscht. Solche Abfragen werden auch *queries* genannt und sind in der Datenbanksprache SQL (Structured Query Language) geschrieben. SQL wurde entwickelt, um mit relationalen Datenbanken kommunizieren zu können. Die Datenbanksprache verlangt eine einfache Syntax und vereinfacht dadurch das Verarbeiten der Daten in einer Datenbank.

### 6.3.3 Python

Python ist eine Programmiersprache und wurde in den 90er von Guido van Rossum entwickelt. Trotz ihrer einfachen Syntax und Lesbarkeit lässt sich die Sprache sehr vielseitig einsetzen und unterstützt objektorientierte Programmierung[16].

### 6.3.4 GitHub

Git allein ist eine Versionsverwaltungssoftware. Solche Softwares ermöglichen den Entwicklern zusammen an einem Projekt zu arbeiten ohne, dass dabei Konflikte entstehen. Jeder Entwickler arbeitet an einem bestimmten Teil an einer Software. Ohne eine Versionsverwaltungssoftware, müssten die Entwickler den Quellcode an eine Person zukommen lassen damit er am Ende alle Teile zusammenfügen kann. Git automatisiert diesen Prozess. Mit Git werden alle Teile der Software zusammengefügt und bei jeder Veränderung im Cod erstellt Git eine neue Version. Das Ermöglicht den Entwickler jeder Zeit auf die vorherige Version der Software zuzugreifen.

GitHub ist die Webseite, die diesen Dienst ermöglicht. Dadurch können Projekte in Git grafisch in einem Browser verwaltet werden.

# 7 Implementierung

Dieser Teil der Arbeit beschreibt, wie die Implementierung der Web-Anwendung stattgefunden hat sowie die genaue Funktionsweise der einzelnen Komponenten des Systems. Sämtliche Tools, die in der Analysephase beschrieben wurden, werden hier zum Einsatz kommen. Beginnend mit der Softwareentwicklungsmethode.

## 7.1 Scrum

Die gesamte Projektverfolgung beruht auf den Ansätzen von Scrum. Die Ereignisse in Scrum wie Sprint-Planning oder Sprint-Review erfolgten meist über Videokonferenzen mit Microsoft Teams.

[16] Objektorientierte Programmierung ist ein Paradigma, welches die Grundstruktur einer Software an die Wirklichkeit ausgerichtet.

Das Product-Backlog und das Sprint-Backlog sowie weitere wichtige Artefakte in Scrum wurden mittels Excel gepflegt. Das Product-Backlog wurde zu Beginn mit Frau Dhrangadhariya erstellt. Die Beurteilung des Product-Backlogs ergab eine Länge von 5 Sprints.

Im ersten Sprint ging es um die Bereitstellung der Entwicklungsumgebung. Dazu wurde zunächst eine Untersuchung der Tools durchgeführt. Danach wurden alle notwendigen Tools installiert und für die Implementierung der Anwendung vorbereitet sowie eine Datenbank Schema erstellt. Die Dauer betrug eine Woche.

Im zweiten Sprint wurde das Backend der Anwendung erstellt sowie der erste Bereich der Benutzeroberfläche. Begonnen wurde mit der Datenbank. Die Implementierung der Datenbank erfolgte anhand des Schemas. Als nächstens wurde die API entwickelt, die für die Zugriffe auf die Datenbank zuständig ist. Das Veranlasst die Erstellung der React-Applikation in der IDE. Danach stand das Frontend im Vordergrund. Und zwar der Dokument Abschnitt der Benutzeroberfläche. Da das Projekt in der IDE bereits im vorherigen Sprint kreiert wurde, wurde die Struktur der gesamten Benutzeroberfläche erstellt. Folglich konnte direkt mit der Implementierung begonnen werden.

Im dritten Sprint wurde der Prognose Abschnitt kodiert. Durch das Betätigen des Buttons im ersten Teilbereich werden Prognosen erstellt und in diesem Abschnitt ersichtlich.

Im vierten und letzten Sprint wurde der letzte Abschnitt der Benutzeroberfläche implementiert. Prognosen können verändert und gespeichert werden.

## 7.2 Versionskontrolle

Bei der Verwaltung von Versionskontrollen können verschiedene Verzweigungsstrategien (englisch: Branching strategies) eingesetzt werden. Verzweigungsstrategien ermöglichen eine Trennung der Arbeit. Das ermöglicht eine kohärente und parallele Entwicklung im Team. Auch wenn das Entwicklerteam aus einer einzigen Person besteht, lohnt es sich Versionskontrollen durchzuführen. Man erhält einen Überblick über sämtliche Änderungen, die im Projekt für Auswirkungen gesorgt haben. Sollten falsche Entscheidungen bei der Programmierung getroffen worden sein, kann mittels der Versionskontrolle wieder auf eine frühere Version des

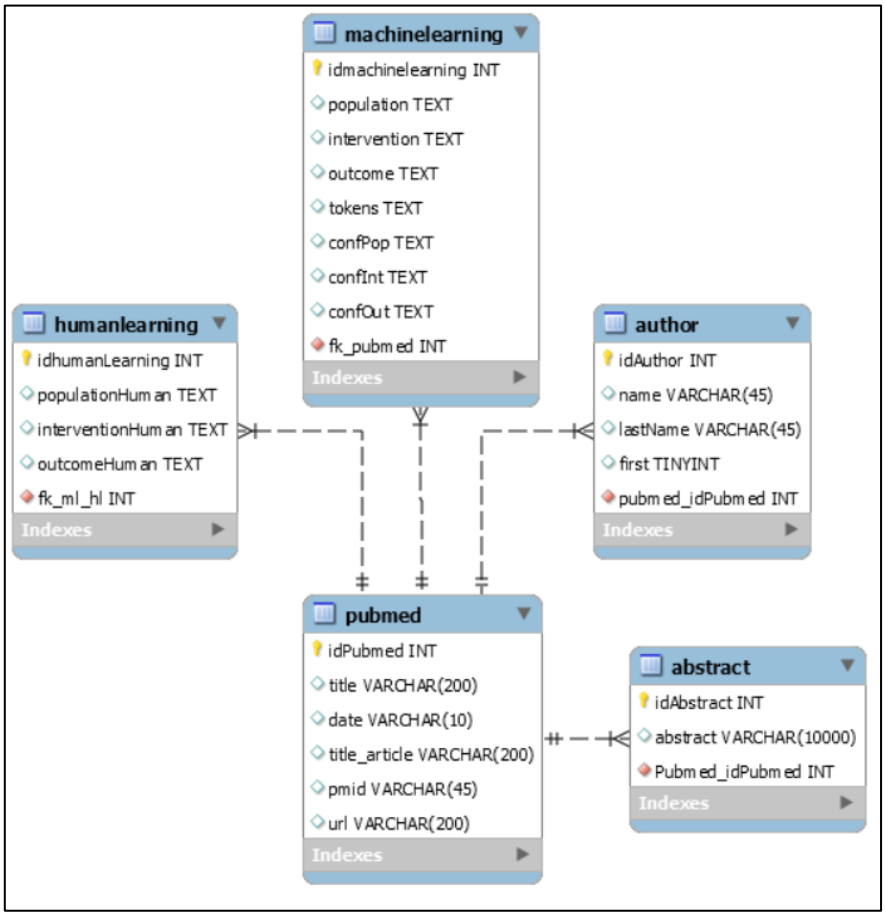
Codes zugegriffen werden. Ausserdem, sollte sich das Entwicklerteam erweitern, steht dadurch bereits ein *Repository* zur Verfügung.

In GIT wird als Repository ein Projekt bezeichnet. Dort sind alle Dateien zu finden, aus denen das Projekt besteht. Jedes Teammitglied, der am Projekt beteiligt ist, kann sich das Repository herunterladen und anfangen, Änderungen zu tätigen. Mittels *Pull* werden sämtliche Daten aus dem Repository geholt. Nachdem der Code verändert wurde, wird mittels *Commit* ein Kommentar zu den Veränderungen abgelegt. Zuletzt muss durch *Push* der geänderte Code im Repository hochgeladen werden.

## 7.3 Datenbank Modell

Die Anwendung muss Dokumente aus der Pubmed Datenbank in ihrer eigenen Datenbank speichern können. Zusätzlich müssen die generierten Prognosen sowie erstellten Kommentare vom Benutzer zu dem Dokument gespeichert werden. Da dies die einzigen Daten sind, die verarbeitet werden, ist kein komplexes Datenbank Modell nötig.

**Abbildung 13: Das Datenbank-Modell**




**Quelle: Eigene Darstellung**

Von jedem Pubmed Dokument werden nur die benötigten Daten gespeichert. Zu diesen Daten gehören der Titel des Dokuments, das Erscheinungsdatum, der Titel des Artikels, die Pubmed ID, die URL des Dokuments aus der Pubmed Datenbank, von den Autoren der Vor- und Nachname sowie die einzelnen Texte. Da ein Dokument über mehrere Autoren und Texten verfügen kann wurden die in der Datenbank in einzelnen Tabellen gelagert.

- **Abstract**: Ein Pubmed Dokument kann über mehreren Texten verfügen jedoch gehört ein Text ausschliesslich einem Dokument.
- **Autor**: Bei den Autoren könnte es sein, dass ein Autor zu mehreren Dokumenten gehören kann. In dieser Web-Anwendung ist diese Information nicht nötig. Da bei der Darstellung der Dokumente in der Tabelle lediglich der erste Autor angezeigt wird, verfügt die Tabelle über einen Schalter namens «first». Die einzige Funktion, die der

Schalter hat, ist bei der Datenverarbeitung zu informieren, ob es sich um den ersten Autor handelt (1) oder nicht (0).

- **Machinelearning**: Nach dem eine Prognose erstellt wurde, werden folgende Daten generiert: Prognosen der Population, Prognosen der Intervention, Prognosen der Outcome, der Text, Koeffizient der Population, Koeffizient der Intervention und Koeffizient der Outcome
- **Humanlearning**: In dieser Tabelle werden die Kommentare des Benutzers zum Dokument gespeichert. Dazu gehören Daten wie Prognose Population, Prognose Intervention und Prognose Outcome.

## 7.4 React Projekt

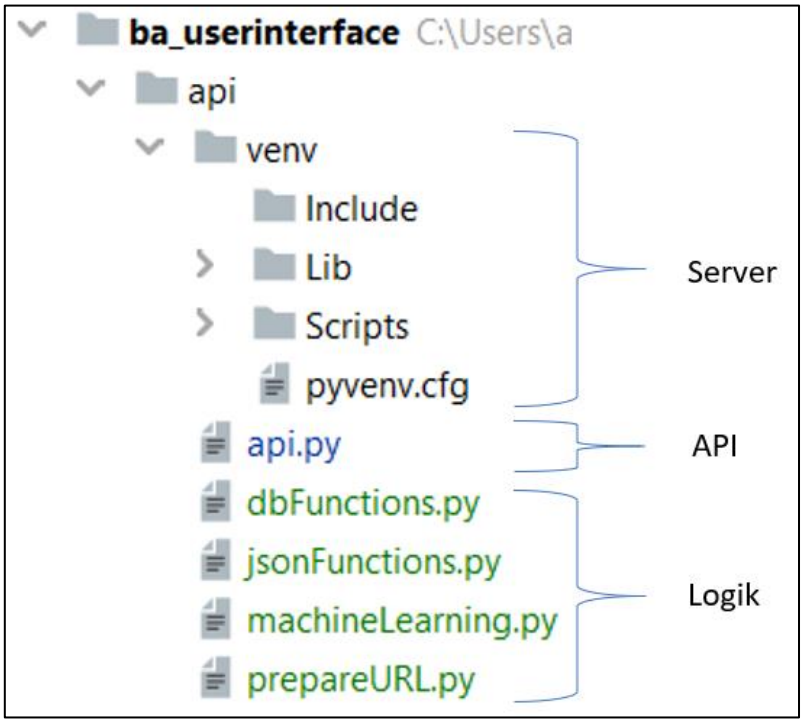
Da das Backend um einiges komplexer als das Frontend ist, wurde bereits zu Beginn eine React-Anwendung erstellt, die das Backend im Projektordner beinhaltet. Bei der Erstellung der React-Anwendung werden einige Dateien automatisch kreiert. Im Folgenden werden einige der wichtigsten Dateien erklärt:

- **Package.json**: Alle Dependecies sind in dieser Datei vorhanden. Als Dependency wird die Abhängigkeit eines Programms bezeichnet, welches Code von anderen Quellen wie zum Beispiel Frameworks oder Bibliotheken benötigt, um ausgeführt zu werden.
- **Index.html**: Beim Ausführen der Anwendung ist diese Datei die erste Seite, die geladen wird.
- **Index.js**: Dies ist die zu index.html gehörende JavaScript-Datei. In der Datei wird bestimmt, dass die App Komponente in einem HTML Element geladen wird.
- **Index.css**: Bestimmt die Darstellung der index.js Datei.
- **App.js**: Die App.js ist die Hauptkomponente in React. Von hier aus werden weitere Komponenten geleitet.
- **App.css**: Bestimmt die Darstellung der App.js Datei.

## 7.5 Flask Server

Das Front- und Backend befindet sich im selben Projekt. Um dennoch eine gewisse Struktur liefern zu können, wurde der Order namens «api» im Projekt erstellt. In diesem Ordner befindet sich die Logik sowie der Server.

**Abbildung 14: Die Datei-Struktur des Backend im Projekt**




**Quelle: Eigene Darstellung**

## 7.6 API

Die api.py Datei sorgt für die Verarbeitung von Anfragen und Antworten. Im Grunde genommen sorgt das Routing dafür. Routing ist ein Mechanismus, welches eine URL direkt dem code zuordnet, welches die Webseite generiert. Das Routing ist in Python bereits implementiert. Flask verwendet dafür den *rout()*-Dekorator. Wenn die URL «http://localhost:3000/api/pubmed/getPubmed» im Browser erwähnt wird, wird die Methode getPubmed() ausgeführt.

**Abbildung 15: Das Routing in Python**

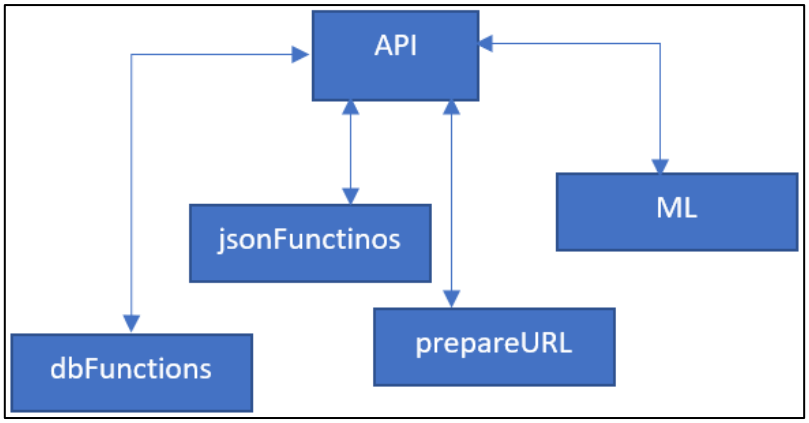
```
@app.route('/api/pubmed/getPubmed', methods=['GET'])
def getPubmed():
```

**Quelle: Eigene Darstellung**

## 7.7 Logik

Die Daten in der Datenbank werden mittels INSERT, DELETE und SELECT abfragen bearbeitet. Wird die gesamte Logik in der API-Klasse implementiert, kann das schnell zu einem unübersichtlichen Code führen. Daher wurde die Logik in 5 verschiedenen Klassen unterteilt.

**Abbildung 16: Die Zusammenhänge der Dateien in der Logik**




**Quelle: Eigene Darstellung**

Dokument hochladen: Vom Frontend aus gelangt eine GET HTTP-Anfrage mit der URL als Parameter an die API. Der nächste Schritt wäre, das Dokument aus der Pubmed Datenbank zu holen. Dafür ist die Datei *prepareURL* zuständig. Die URL wird an die Methoden dieser Datei weitergeleitet, um aus der URL das vollständige Pubmed Dokument zu holen und es der *API-Datei* im JSON Format zurückzugeben. Das Vollständige Dokument enthält eine Menge an Informationen, für die das ML keine Verwendung hat. Diese unnötigen Informationen werden in der Datei *jsonFunctions* beseitigt. Nachdem das Dokument bereinigt wurde, leitet die API nun das Dokument in die *dbFunctions* Datei, die dafür sorgt, dass das Dokument in der Datenbank gespeichert werden kann. Das Dokument wurde hochgeladen und der Benutzer bekommt es im Frontend zu sehen.

Prognosen erstellen: In der Tabelle werden alle Pubmed Dokumente angezeigt, die bereits hochgeladen wurden. Zu jedem Eintrag befindet sich ein Button «Get Prediction». Dieser Button ermöglicht dem Benutzer Prognosen über diesen Eintrag zu erstellen. Wurde der Button betätigt, gelangt die ID dieses Eintrags über eine GET HTTP-Anfrage an die API. Mit Hilfe der ID wird das richtige Dokument in der *dbFunctions* Datei geholt und der API zurückgegeben. Die API leitet die gewonnenen Informationen an das ML weiter. Hier werden

die Prognosen generiert und an die API wieder zurückgeleitet. Die API leitet die Daten weiter, um sie dann in der Datenbank zu speichern und gleichzeitig werden die Daten wieder an das Frontend geschickt, um die Prognosen visualisieren zu können.

Kommentieren: Mit dem Button «Annotate» werden die korrigierten Prognosen mittels einer POST HTTP-Anfrage an die API versendet. Die API übergibt die Daten an die Datei *dbFunctions* um sie dann in der Datenbank zu speichern.

Anlernen: Beim Anlernen der Algorithmen wird die ID der Prognose durch eine POST HTTP-Anfrage an die API versendet. Die API holt mittels der Datei *dbFunctions* die korrigierten Prognosen aus der Datenbank und leitet sie weiter an das ML.

### 7.7.1 Verbindung zur Datenbank

Damit auf die Datenbank zugegriffen werden kann, braucht Python eine MySQL Bibliothek. Dazu muss der der *MySQL-Connector* installiert werden. Der MySQL Connector ermöglicht Python Programmen den Zugriff auf die Datenbank. Bevor Abfragen erstellt werden können, muss eine Verbindung zur Datenbank hergestellt werden. Die Methode *connect()* erzeugt eine Verbindung zum MySQL Server und gibt dabei ein *MySQLConnection-Objekt* zurück.

**Abbildung 17: Verbindung zur Datenbank mittels MySQL Connector in Python**

```
db = mysql.connector.connect(host = "localhost",
                             user = "root",
                             passwd = "tearca.A8",
                             database = "mydb3",
                             auth_plugin = "mysql_native_password")
```

**Quelle: Eigene Darstellung**

Sämtliche Informationen wie Host, Benutzer, Passwort, Datenbank und Autorisation müssen definiert werden. Nachdem die Verbindung hergestellt wurde, wird das Verbindungsobjekt in der Variablen *db* gespeichert.

Anhand der *cursor()* Methode werden Datensätze in der Datenbank abgefragt. Die *cursor()* Methode wird mit der Verbindungsmethode *connect()* benutzt. Die Abfrage, welche in der Datenbank ausgeführt werden soll, wird in der Variablen *query* gespeichert. Anschliessend wird die Abfrage mit der Methode *execute()* ausgeführt die die Variable *query* übergibt.

Nachdem die Abfrage ausgeführt wurde, werden alle Ergebnisse der *execute()* Methode anhand der *fetchall()* Methode in der Variable *results* gespeichert.

**Abbildung 18: Ausführen von Abfragen an die Datenbank in Python**

```
mycursor = db.cursor()
query = "SELECT * FROM mydb3.pubmed WHERE idPubmed = " + id
mycursor.execute(query)
```

**Quelle: Eigene Darstellung**

### 7.7.2 Jsonify

Die Vermittlung der Daten erfolgt im JSON Format. Das Umwandelt der Daten in JSON geschieht in Python durch die *jsonify()* Methode. Nun kann die Variable *result*, welche die Ergebnisse der Abfrage beinhaltet, in JSON umgewandelt werden, indem die *result* Variable der Methode *jsonify()* übergeben wird. Die Methode *getPubmed()* gibt nun die Ergebnisse der Abfrage an die Datenbank als JSON zurück.

### 7.7.3 Dokumente aus der Pubmed Datenbank holen

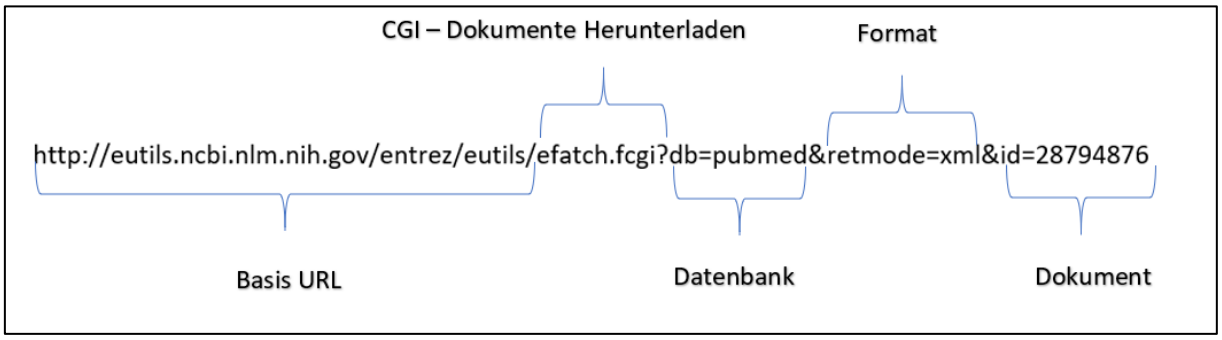
Eine weitere wichtige Funktion ist das Holen der Dokumente aus der Pubmed Datenbank anhand einer URL. Dem Benutzer der Anwendung soll es möglich sein, allein durch die Eingabe der URL das gesamte Dokument aus der Pubmed Datenbank beziehen zu können, die notwendigen Informationen zu holen und schlussendlich, die gesamten Daten in der Datenbank zu speichern.

Zunächst muss auf die Datenbank von Pubmed zugegriffen werden. Dafür bietet E-Utilities eine Lösung. E-Utilities ist eine API welches alle Hauptfunktionen von Entrez bietet wie das Herunterladen von Dokumenten in verschiedenen Formaten. Diese Funktion von Entrez ist eine sogenannte *Common Gateway Interface* (CGI). Das Common Gateway Interface ist eine Schnittstelle zwischen einem Webserver und einer Software. Diese Schnittstelle ermöglicht einen Datenaustausch zwischen den einzelnen Komponenten. Alle CGIs von Entrez teilen dieselbe Basis URL.

Damit Dokumente heruntergeladen werden, wird die *efatch.fcgi* CGI benötigt. Mit Hilfe von Parametern kann definiert werden, welches Dokument von welcher Datenbank und in

welchem Format heruntergeladen werden soll. Die Nachfolgende Abbildung zeigt, wie diese Informationen in einer URL geordnet sind.

**Abbildung 19: Beispiel einer URL für das Holen der Dokumente**




**Quelle: Eigene Darstellung in Anlehnung an NCBI, E-Utilities Introduction, 13.04.2012, URL: https://www.youtube.com/watch?v=BCG-M5k-gvE&t=4s**

Nach dem die URL definiert wurde, liegt es Python, die URL abzurufen. Python bietet dafür die urllib Bibliothek an welches dies ermöglicht. Die Methode *urlopen()* erlaubt, Objekte, die die URL nach dem Abrufen beinhaltet, zu holen. Der Parameter *pmid*, welches eine Pubmed ID ist, wird der URL hinzugefügt. Mit Hilfe der *urlopen()* Methode wird das Objekt, das Pubmed Dokument, in der Variablen *data* im XML-Format gespeichert. Das Konvertieren in ein JSON Objekt geschieht anhand der *json.dumps()* Methode. Die Variable *data* wird zunächst in ein Python Dictionary umgewandelt und anschliessend mit *json.dumps()* in ein JSON Objekt

**Abbildung 20: Methode, die eine URL entgegennimmt und ein Pubmed-Dokument Objekt in JSON hergibt in Python**

```
def fetch_PubMed_document(pmid):
    base = 'http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed&retmode=xml&id='
    fetchPMID = base + pmid
    with urllib.request.urlopen(fetchPMID) as url:
        data = url.read()
        url.close()

        article = json.dumps( xmltodict.parse(data) )
        article = json.loads(article)

    return article
```

**Quelle: Eigene Darstellung**

Die gesamte Methode verlangt einen einzigen Parameter, und zwar eine Pubmed ID. Die Funktion «Dokumente Hochladen» verlangt vom Benutzer die URL der Pubmed Webseite, in der sich das Dokument befindet. In der URL befindet sich die Pubmed ID, welches für die Anwendung wichtig ist, um das richtige Dokument herunterladen zu können. Der nächste Schritt ist es, die Pubmed ID von der URL zu trennen. Dafür ist die Methode *tasks()* zuständig.

Nachdem die Anfrage durch die API an die Methode tasks() weitergeleitet wurde, wird vom JSON Objekt die URL geholt. Mit der Methode get_json() [‘title] wird im Objekt nach der Überschrift url gesucht um somit das dazugehörige Datenobjekt holen zu können. Das Datenobjekt wird als String in der Variablen docURL gespeichert.

**Abbildung 21: ID aus der Anfrage holen in Python**

```
url = request.get_json()['title']
docURL = str(url)
```

**Quelle: Eigene Darstellung**

Die URL muss überprüft werden, ob es sich auch um eine Pubmed URL handelt. Dafür ist die Methode *verifyURL_PubMed()* zuständig. Von der eingegebenen URL wird die Basis der URL verglichen. Sollte die Basis stimmen, dann ist die URL korrekt.

**Abbildung 22: Methode die die URL überprüft in Python**

```
def verifyURL_PubMed(url_string):

    url_string = url_string.rstrip()
    url_split = url_string.split('/')
    url_base = '/'.join(url_split[0:-2])
    assert url_base == 'https://pubmed.ncbi.nlm.nih.gov'
```

**Quelle: Eigene Darstellung**

Handelt es sich um eine Pubmed URL, wird die Pubmed ID von der URL getrennt. Zunächst werden Leerschläge entfernt. Danach wird vom letzten Backslash der Letzte Teil der URL entnommen.

**Abbildung 23: Methode, die die Pubmed ID von der URL extrahiert in Python**

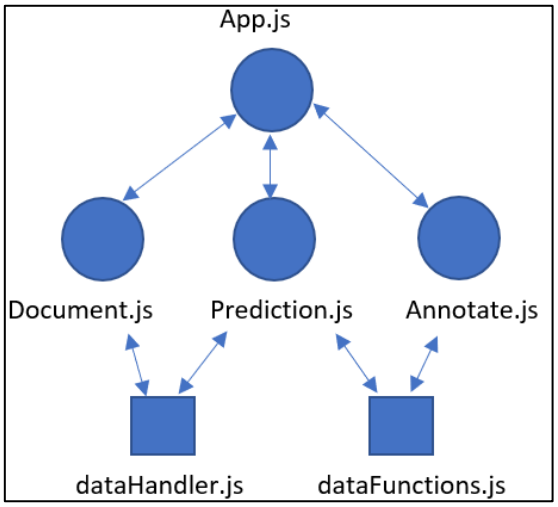
```python
def extractID(url_string):

    url_string = url_string.rstrip()
    url_split = url_string.split('/')
    pubmed_id = url_split[-2]

    return pubmed_id
```

**Quelle: Eigene Darstellung**

## 7.8 Frontend

Zu Beginn der Implementation wurde die React-App bereits erstellt. Im Ordner «components» befinden sich die benötigten Komponenten die von der App.js aus aufgerufen werden. Ausschliesslich der App.js ist das Frontend auf drei Komponenten aufgebaut: die Dokument-, Prognose- und Korrekturkomponente. Zusätzlich sind zwei weitere Klassen zu finden die für die Verarbeitung der Daten notwendig sind. Die dataHandler.js ist für das Holen und Speichern von Daten zuständig und die dataFunctions.js für die Verarbeitung der Daten

**Abbildung 24: Struktur der JavaScript Dateien in React JS**

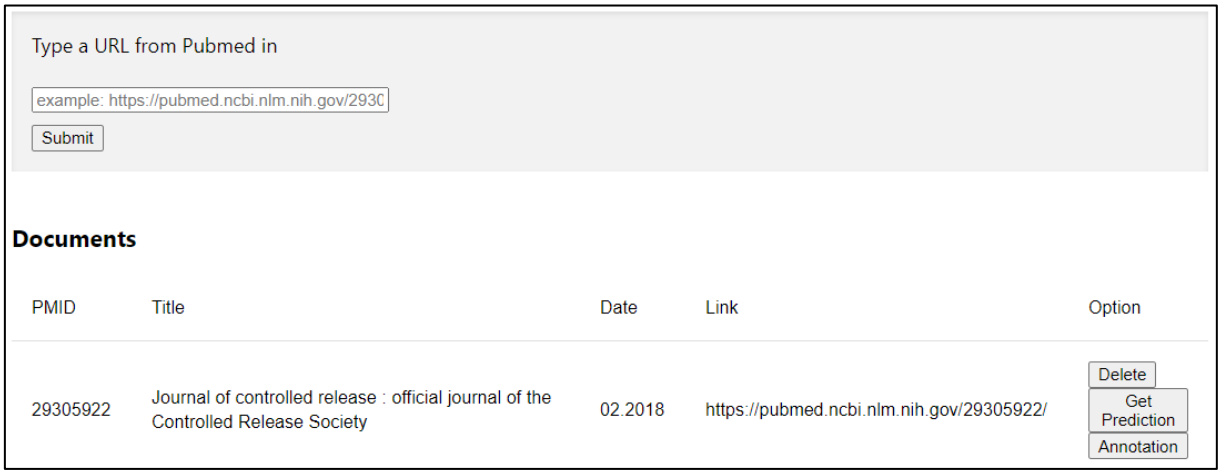



**Quelle: Eigene Darstellung**

Die gesamte Anwendung lässt sich in 3 Teilen gliedern. Jedes Teil der Anwendung beruht auf eine ähnliche Struktur. Der obere Teil verlangt die Eingaben vom Benutzer und der Untere Teil dient zur Visualisierung.

Der erste Teil ist für das Hochladen und Darstellen der Dokumente in einer Tabelle zuständig. Im oberen Abschnitt der Komponente kann der Benutzer die URL eingeben und mittels eines Buttons das Dokument speichern. Im unteren Abschnitt befindet sich eine Tabelle. Die Tabelle besitzt einen Tabellenkopf die als Überschrift der einzelnen Einträge dient. Ein Eintrag besteht aus der PMID, dem Titel, dem Datum der Veröffentlichung und dem Link eines Pubmed-Dokuments. Der letzte Teil des Eintrags beinhaltet die Optionen. Das Dokument kann gelöscht werden, zur Erstellung von Prognosen sowie für die Korrektur von Prognosen benutzt werden. Die nachfolgende Abbildung zeigt den ersten Teil der Benutzeroberfläche.

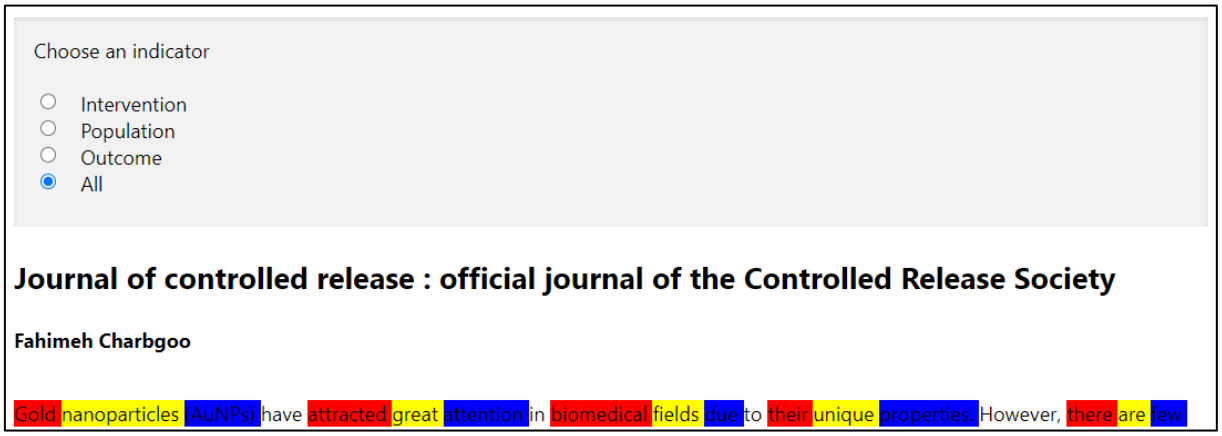
**Abbildung 25: Der obere Teil der Benutzeroberfläche**




**Quelle: Eigene Darstellung**

Im zweiten Teil werden die gewonnen Daten visualisiert. Im oberen Bereich kann der Benutzer jeweils eine Option von vier gleichzeitig auswählen. Anhand der Auswahl wird das gewünschte Abbild angezeigt. Im Unteren Bereich wird gross der Titel des Dokuments angezeigt, der Autor mit etwas kleineren Schrift und der Abstract mit den PICO Elementen in den verschiedenen Farben. Die nachfolgende Abbildung zeigt den zweiten Teil der Benutzeroberfläche.

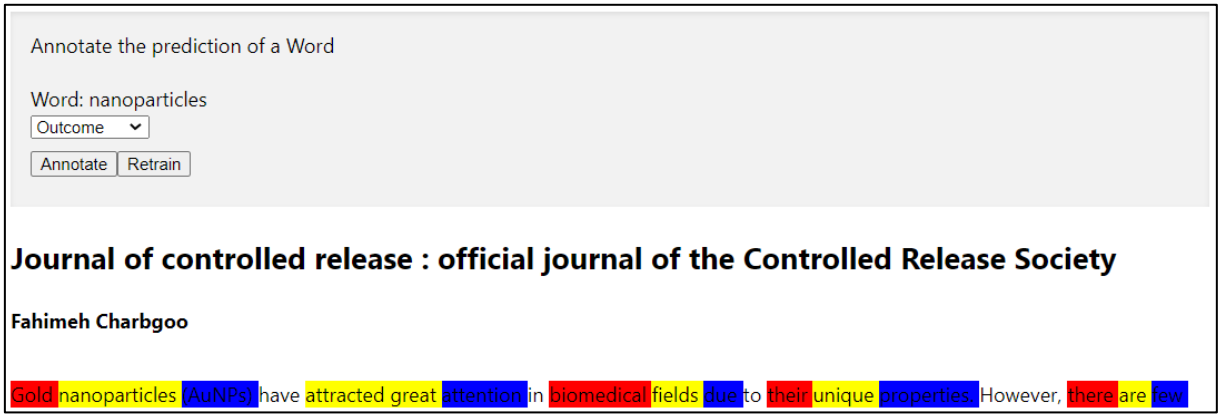
**Abbildung 26: Der mittlere Teil der Benutzeroberfläche**




**Quelle: Eigene Darstellung**

Im dritten Teil erhält der Benutzer die Möglichkeit falsch prognostizierte Werte zu evaluieren und korrigieren. Im oberen Bereich wird das Ausgewählte Wort vom unteren Bereich angezeigt. Mittels einer Dropdown-Schaltfläche[17] kann der richtige Wert ausgewählt werden. Mit dem Button «Annotate» werden die Korrekturen in der Datenbank gespeichert. Mit «Retrain» wird das gesamte Model des ML neu angelernt. Im Unteren Bereich ist das gleiche wie im zweiten Teil der Anwendung zu finden. Nur besteht hier die Möglichkeit, die einzelnen Wörter anzuklicken.

**Abbildung 27: Der Untere Teil der Benutzeroberfläche**




**Quelle: Eigene Darstellung**

[17] Eine Schaltfläche, die beim Anklicken eine Liste anzeigt

### 7.8.1 Ansicht Dokumente

Für die Darstellung der Dokumente in einer Tabelle ist die Komponente List.js zuständig. Sie enthält einen Bereich, in der der Benutzer die URL eingeben kann sowie eine Tabelle, die dem Benutzer alle Pubmed Dokumenten anzeigt, die sich in der Datenbank befinden. Der erste Schritt, den der Benutzer ausführt, ist die Eingabe der URL. Mit dem Button «Submit» speichert er das Dokument in der Datenbank und gleichzeitig wird die Tabelle aktualisiert.

Wurde der Button «Submit» betätigt, gelangt die URL mittels einer GET HTTP-Anfrage an das Backend. Im Backend wird das Dokument verarbeitet und gespeichert. Beim Hochladen von Dokumenten wird zusätzlich die Methode getAll() ausgeführt. Anhand dieser Methode werden alle Pubmed Daten von der Datenbank geholt. In React wird der state der Komponente durch die Daten angepasst, um so ein *rendering* zu erzeugen. Dadurch wird erreicht, dass der Benutzer nach jedem Hochladen eines Dokuments stehts ein aktualisiertes Abbild der Datenbankeinträgen zu sehen bekommt.

### 7.8.2 Ansicht Prognosen

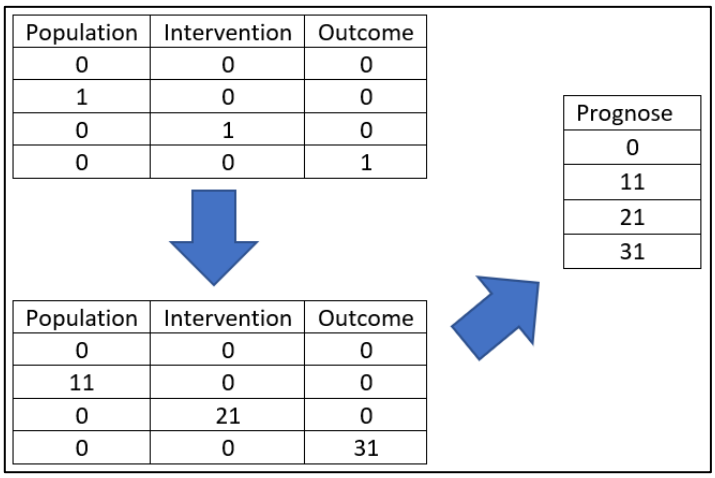
Mit dem Button «Get Prediction» gelangt der Abstract mittels einer GET HTTP-Anfrage an das Backend. Im Backend wird eine Prognose des Abstracts erstellt und an das Frontend zurückgegeben. Im Frontend werden sechs verschiedene Daten erwartet: Prognosen über Population, Interventen und Outcome. Für jede einzelne Prognose werden auch Daten übergeben, die die Wahrscheinlichkeit der Korrektheit in Prozent einer Prognose angibt.

Eine Prognose besteht aus lauter Einsen und Nullen. Jede Zahl der Prognose steht für ein Wort aus dem Abstract. Eins, die Prognose trifft zu und Null, die Prognose trifft nicht zu. Für die Visualisierung der Prognosen im Abstract wird die Hilfe von CSS verwendet. Bis es zur Visualisierung kommt, werden erst alle drei Prognosen zusammengefügt. Bevor die einzelnen Daten zusammengefügt werden, wird zunächst dafür gesorgt, dass sich die Einsen der verschiedenen Prognosen voneinander unterscheiden lassen.

Nachdem die Daten vom Backend geholt wurden, wird durch jede Prognose durch iteriert. Trifft Die Prognose zu, wird zusätzlich eine zweite Zahl vor der Eins angehängt. Das geschieht mit allen drei Daten. Eine Eins für Population, eine Zwei für Intervention und eine Drei für

Outcome. Da nur eine Prognose für ein Wort zutrifft, werden nun die Daten zu einer einzigen Liste vereint. Die Liste beinhaltet nun Daten wie 0, 11, 21 und 31.

**Abbildung 28: Das Zusammenfügen der Prognosen bildlich erklärt**




**Quelle: Eigene Darstellung**

Die gesamte Liste wird für die Visualisierung verwendet. Wie bereits erwähnt wird unter Einfluss von CSS der Abstract mit den dazugehörigen Prognosen abgebildet. Der gesamte Abstract wird in einzelnen Wörtern zerlegt. Jedes Wort wird als ein eigenes Objekt in einer Liste angesehen. Bei der Darstellung werden beide Listen benötigt. Mittels einer Schleife wird durch jede Iteration einmal durch die Liste mit den Wörtern und einmal durch die Liste mit den Prognosen iteriert. Das Wort wird in einem Paragraf Element ( ) von HTML gespeichert. Ein Paragraf in HTML dient nur zur Veranschaulichung von Texten. Das besondere bei HTML Elementen ist, dass sie mit Attributen verseht werden können. Attribute bieten zusätzliche Informationen über ein HTML Element an. Mittels dieser Attribute wird das Paragraf Element mit einer ID versehen. Dadurch wird erreicht, dass CSS den Paragrafen formatieren kann. In CSS wird für diese ID eine Formatierung deklariert, damit jedes Paragraf Element, die diese ID besitzt, von der Formatierung profitieren kann. Insgesamt werden vier Formate von CSS benötigt, je eines für eine Prognose. Die ID stellt sich wie folgt zusammen: aus dem Wort «prediction» und dem Prognosewert. Daraus resultieren folgende IDs: Prediction0, prediction11, prediction21 und prediction31. Nun bei der Iteration der Beiden Listen wird das Wort im Paragraf-Element gespeichert und die Prognose wird dem Wort «prediction» in der ID beigefügt.

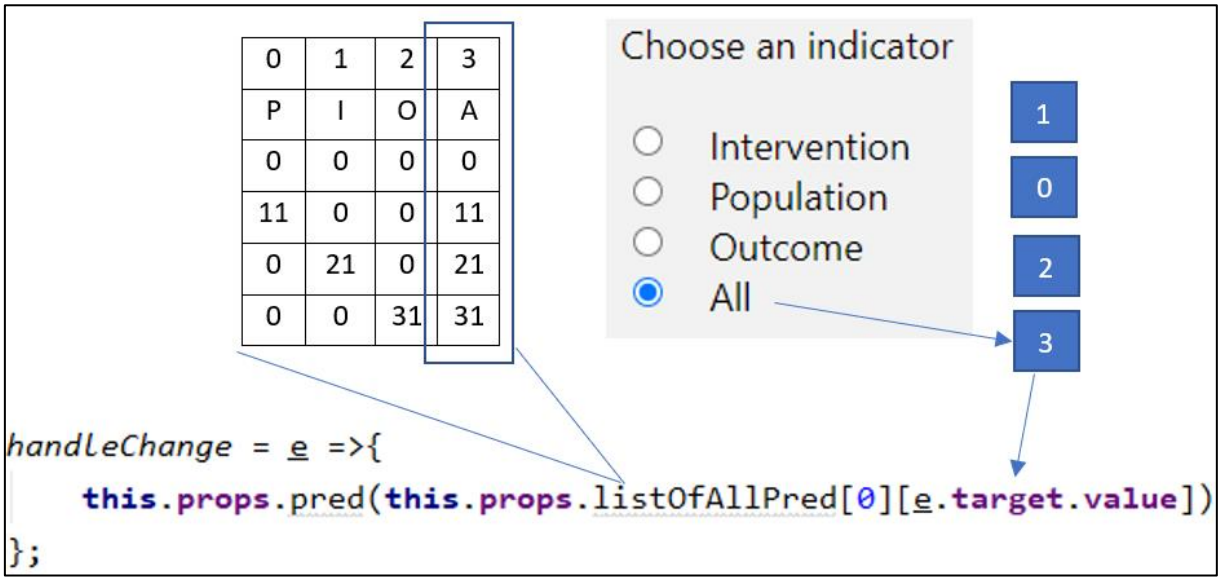
**Abbildung 29: Verwenden der verschiedenen Prognosen in React**

```
this.props.allWords.map((word, index) =>
    <p id={"prediction"  + this.props.concatPredMod[index]}>{word} </p>
)
```

**Quelle: Eigene Darstellung**

Somit wird jedes Wort anhand der erstellten CSS-Formatierung visualisiert. Mit Radio-Buttons kann der Benutzer die gewünschten PICO Elemente anzeigen lassen. Wie bereits vorhin erwähnt, durchlaufen die generierten Daten aus dem ML im Prognose Abschnitt einen Prozess, die die Daten verarbeitet und zusammenstellt. Während dem durchlauf wird an der Stelle, wo die Einsen von den jeweiligen Prognosen mit Zahlen markiert werden, die einzelnen Listen separat in einer einzigen Liste samt der Liste, die alle Prognosen beinhaltet im state gespeichert. Dieser Schritt erlaubt dem Benutzer zwischen verschiedenen PICO-Elementen zu wechseln.

**Abbildung 30: Visualisierung der Methode**




**Quelle: 1Eigene Darstellung**

## 7.8.3 Ansicht Kommentare

Jeder Eintrag in der Tabelle der Dokumente verfügt über einen «Annotate» Button. Nach dem der Button betätigt wurde, gelangt die ID des Eintrags mittels einer GET HTTP-Anfrage an das Backend. Das Backend holt die Daten aus der Datenbank und leitet sie weiter an das Frontend. Für die Darstellung wird das gleiche Prinzip angewendet wie bei der Ansicht der

Prognosen. Zusätzlich wird hier das Paragraf HTML-Element mit einem *onClick* Attribut ausgestattet. Dieses Attribut erlaub dem Element Aktionen durchzuführen, falls auf das Element angeklickt wurde. Wird auf das Element angeklickt, erscheint das Wort im oberen Bereich. Mit der Dropdown-Schaltfläche wird die Prognose in der Liste verändert. Durch das Betätigen des «Annotate» Buttons wird die Liste als eine POST HTTP-Anfrage an das Backend versendet.

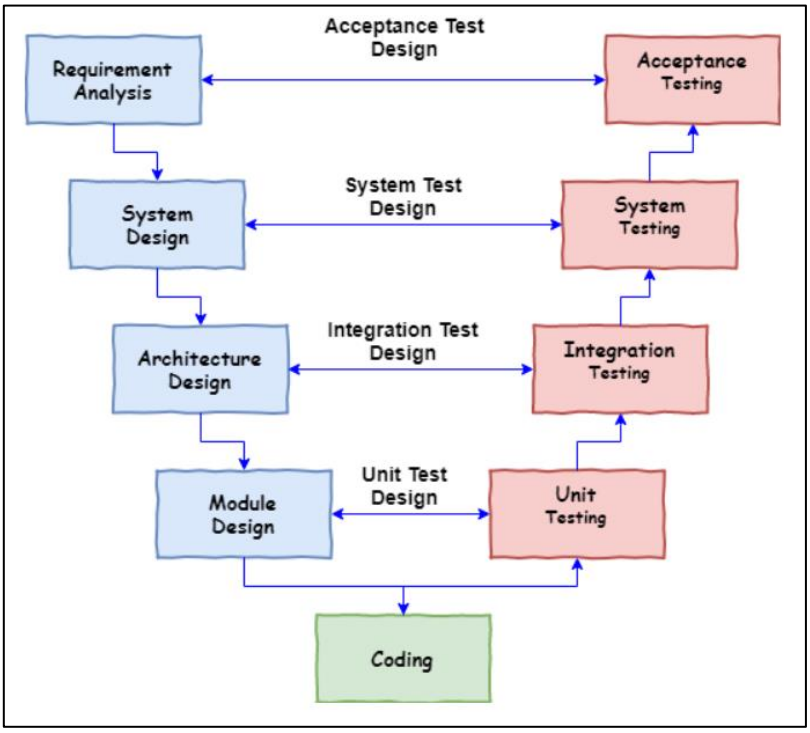
# 8 Test

In der Test Phase wird sichergestellt, dass die Funktionen der Anwendung einwandfrei funktionieren. Bugs [18] in einer Anwendung sorgen für schlechte Benutzererfahrung und können sogar die gesamte Anwendung unbrauchbar machen. Durch das Testen werden sämtliche Funktionen einer Anwendung anhand von spezifischen Anforderungen verifiziert sowie mittels der Erwartungen der Endbenutzer validiert. Daher wird die Testphase solang wiederholt, bis sämtliche Bugs bereinigt wurden.

Beim Testen verhilft der Ansatz des V Modells. Dadurch wird von Stufe zu Stufe in einer Anwendung getestet. Das Ziel ist es, dass zuallererst die kleinstmögliche Einheit im *Komponententest* (englisch: Unit Test) getestet wird. Falls die Anforderungen erfüllt worden sind, geht es in der nächsten Stufe, dem *Integrationstest* (Integration Test), weiter. Hier wird die Zusammenarbeit der einzelnen Einheiten getestet. Sollte auch die Kommunikation der einzelnen Komponenten einwandfrei funktionieren, wird dann in der nächsten Stufe, dem *Systemtest* (System Test), das gesamte System verifiziert und validiert. Zuletzt wird mit Hilfe vom Abnahmetest (Acceptance Test) die Akzeptanzkriterien vom Endnutzer abgeprüft und bestätigt. In der Nachfolgenden Abbildung ist das V-Modell zu sehen welches die Auszuführenden Tests den Phasen gegenüberstellt.

[18] In der IT-Welt werden Programmfehler als Bugs bezeichnet

**Abbildung 31: Das V Modell**




**Quelle: Meenakshi Agarwal, SDLC V Model – A Step by Step Guide for Beginners, TechBeamers.com, URL: https://www.techbeamers.com/v-model/**

## 8.1 Komponententest und Integrationstest in Python

In Python sind Komponenten- sowie Integrationstest möglich. Um einen Komponententest durchzuführen, wird in Python eine Methode mit der *assert* Funktion überprüft, ob das Resultat der Methode mit dem verlangten Resultat übereinstimmt.

Integrationstests werden in der gleichen Art wie Komponententests geschrieben. Der Unterschied liegt hier, dass ein Integrationstest mehrere Komponenten umfasst.

### 8.1.1 Jest

Um React Komponente zu Testen wird ein Framework benötigt. Facebook bietet ein solches Framework namens Jest an. Jest ist Open-Source und wurde 2014 speziell für das Testen von React-Anwendungen entwickelt. Es sind keine Konfigurationen notwendig, um eine React-Anwendung zu testen.

### 8.1.2 Postman

Postman ist ein sehr nützliches Tool für die Entwicklung und Testen von APIs. Eines der Hauptfunktionen von Postman ist das Testen der Aufrufe auf die API. Das ermöglicht, dass die

API bereits ohne eine Benutzeroberfläche getestet werden kann. Im Postman kann man die URL, auf die zugegriffen werden soll, die Parameter, die die API verlangt sowie die Methode, die angibt, wie auf die API zugegriffen werden soll, eingeben. Beim Ausführen des Tests erhält man dann im Response Fenster die Antwort der API.

## 8.2 Systemtest

Beim Systemtest wird meistens mit einem *End-to-End-Test* getestet. Das heisst, ein Prozess wird vom Anfang bis zum Ende durchgeführt und bewertet. Solche Tests macht Cypress in React JS möglich.

### 8.2.1 Cypress

Cypress ist eine Open-Source Lösung für End-to-End Tests. Zunächst muss die Library in React JS installiert werden. Nach der Installation kann Cypress ausgeführt werden. In der Konsole des Browsers werden sämtliche Resultate angezeigt.

## 8.3 Abnahmetest

Beim Abnahmetest wird das fertige System getestet ob die erwarteten Anforderungen des Kunden erfüllt worden sind. Einer dieser Tests ist der Benutzerabnahmetest.

### 8.3.1 Benutzerabnahmetest

Beim *Benutzerabnahmetest* (User Acceptance Testing) wird das System in der Betriebsumgebung getestet. Eine Auswahl von Endnutzern wird beauftragt, das System zu testen.

# 9 Schlussfolgerung

## 9.1 Ergebnisse

Die Implementierung der Web-Anwendung war ein Erfolg. Alle erforderlichen Funktionalitäten wurden implementiert und erfüllen ihren Zweck. Bei der Implementierung wurde stehts auf die Erweiterbarkeit der Anwendung geachtet, um so in Zukunft weitere Funktionen einfacher zu realisieren. Zusätzlich verhalf der geschickte Einsatz sämtlicher Tools dem Projekt zur Aneignung weiterer Eigenschaften wie Modifizierbarkeit und Struktur. Aufgrund der Entwicklung des ML, wurde die Web-Anwendung noch nicht veröffentlicht jedoch wird dieser Schritt in naher Zukunft realisiert. Zurzeit wird noch die Funktion im ML implementiert, welches das Modell neu anlernen soll.

## 9.2 Verbesserungsvorschläge

Die Hauptfunktionen wurden implementiert und dennoch können einige Verbesserungen an der Anwendung durchgeführt werden. Das Ergebnis führt zu folgenden Verbesserungsvorschlägen:

- Login: Jeder Benutzer kann sich in der Web-Anwendung authentifizieren, um so stets einen Überblick über seine eigenen Dokumente zu erhalten.
- Navigation Bar: Eine Navigation Bar sollte zusätzlich für mehr Benutzerfreundlichkeit sorgen. Durch das Anklicken der Menu-Punkte gelangt der Benutzer schnell an die Gewünschte Oberfläche.
- Hochladen von mehreren URLs: Es könnte zudem möglich gemacht werden, dass eine Liste von URLs übergeben werden kann, die sämtliche Dokumente separat speichert.
- Das ML mittels API mit der Web-Anwendung verknüpfen

## 9.3 Grenzen

Diese Arbeit musste innerhalb einer gewissen Zeit realisiert werden. Durch den Mangel an Zeit konnten einige Funktionen nicht ausführlich überdacht werden um die Verlangten Funktionalitäten implementieren zu können. Daher wurde sehr viel wert bei der Entwicklung auf die Erweiterbarkeit der Web-Anwendung gelegt. Nach der Abgabe dieser Bachelorarbeit wird weiterhin an der Verbesserung sowie an der Integration des ML gearbeitet.

# Literaturverzeichnis

Admin. (22. November 2019). *Was ist Softwareentwicklung? Dinge, die du wissen musst.* Verfügbar unter: https://vizah.ch/was-ist-softwareentwicklung-dinge-die-du-wissen-musst/

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web Services - Concepts, Architectures and Applications.* Heidelberg: Springer-Verlag.

Augsten, S. (08. März 2017). *Was ist eine API?* Dev-Insider. Verfügbar unter: https://www.dev-insider.de/was-ist-eine-api-a-583923/

Augsten, S. (20. April 2018). *Was ist JSON?* Dev-Insider. Verfügbar unter: https://www.dev-insider.de/was-ist-json-a-702243/

Begemann, O. (31. Oktober 2018). *Roy Fielding's REST dissertation*. Verfügbar unter: https://oleb.net/2018/rest/

Cotting, A. (2019). *Agile Methodologies* [Vorlesungsunterlagen]. Siders: Hes-so Valais

Dhrangadhariya, A., Hilfiker, R., Schaer, R. & Müller, H. (2020). *Machine Learning Assisted Citation Screening for Systematic Reviews.* DOI: 10.3233/SHTI200171.

Facebook. (kein Datum). *React - A JavaScript library for building user interfaces*. Verfügbar unter: https://reactjs.org/

Fielding, T. (2000). *Architectural Styles and the Design of Network-based Software Architectures (Dissertation).* University of California, Irvine. Verfügbar unter: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Flask. (kein Datum). *Welcome to Flask*. Verfügbar unter: https://flask.palletsprojects.com/en/1.1.x/ abgerufen

Frontned GmbH (kein Datum). *Was ist Frontend — was Backend?* Verfügbar unter: https://www.frontend-gmbh.de/blog/frontend-und-backend/

Heinle, N., Peña, B., & Speidel, U. (2006). *Webdesign mit JavaScript & Ajax.* Köln: O'Reilly Verlag.

Helmich, M. (12. März 2013). *RESTful Webservices (1): Was ist das überhaupt?* Verfügbar unter: https://www.mittwald.de/blog/webentwicklung-design/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt

Higgins, J. P. & Altman, D. G., (2008). *Assessing risk of bias in included studies*. DOI: 10.1002/9780470712184

JetBrains. (kein Datum). *Webstorm*. Verfügbar unter: https://www.jetbrains.com/webstorm/

Johnson, R. E., & Foote, B. (Juni 1988). Designing Reusable Classes. *Journal of Object-Oriented Programming*, S. 22-35.

Luber, S., & Augsten, S. (8. März 2017). Was ist eine API? *Dev Inseider*.

Maglott, D., Ostell, J., Pruitt, K. D., & Tatusova, T. (2006). *Entrez Gene: gene-centered information at NCBI.* Verfügbar unter: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1761442/#:~:text=Entrez%20Gene%20provides%20unique%20integer,E%2DUtilities%20(1).

Meenakshi, A. (kein Datum). *SDLC V Model – A Step by Step Guide for Beginners.* Verfügbar unter: https://www.techbeamers.com/v-model/

Maurice, F., & Rex, P. (2007). *jetzt lerne ich CSS - Standardkonformes Webdesign mit Cascading Style Sheets.* München: Markt+Technik Verlag.

Moqups. (kein Datum). *Moqups*. Verfügbar unter: Moqups: https://www.moqups.com/de/

Mueller, J. P., & Massaron, L. (2017). *Maschinelles Lernen mit Python und R.* Weinheim: WILEY-VCH Verlag GmbH & Co. KGaA.

Münz, S., & Nefzger, W. (2004). *HTML und Web-Publishing Handbuch.* Poing: Franzis' Verlag.

National Library of Medicine. *MEDLINE®: Description of the Database.* Verfügbar unter: https://www.nlm.nih.gov/bsd/medline.html

NCBI. *About NCBI.* Verfügbar unter: https://www.ncbi.nlm.nih.gov/home/about/

NCBI [Pseudonym]. (2012). *E-Utilities Introduction.* [Video]. Verfügbar unter: https://www.youtube.com/watch?v=BCG-M5k-gvE&t=4s

Nelson, H. D. (2014). *Systematic reviews to answer health care questions*. Lippincott Williams & Wilkins.

Nye, B., Jessy Li, J., Patel, R., Yang, Y., Marshall, I., Nenkova, A., & Wallace, B. (2018). *A Corpus with Multi-Level Annotations of Patients, Interventions and Outcomes to Support Language Processing for Medical Literature.* Melbourne: Association for Computational Linguistics.

Postman. (kein Datum). *The Collaboration Platform for API Development.* Verfügbar unter: https://www.postman.com/

Pröll, S., Zangerle, E., & Gassler, W. (2011). *MySQL - Das Handbuch für die Administratoren.* Bonn: Galileo Press.

Programmieren starten [Pseudonym]. (2018). *Was ist GitHub?* [Video].Verfügbar unter: https://www.youtube.com/watch?v=3ZlpJHZBbi8&t=253s

Python. (kein Datum). *urllib.request — Extensible library for opening URLs.* Verfügbar unter: https://docs.python.org/3/library/urllib.request.html

Python. (kein Datum). *Python 3.8.5 documentation.* Verfügbar unter: https://docs.python.org/3/

Rouse, M. (Oktober 2008). *Client-Server.* ComputerWeekly. Verfügbar unter: https://www.computerweekly.com/de/definition/Client-Server

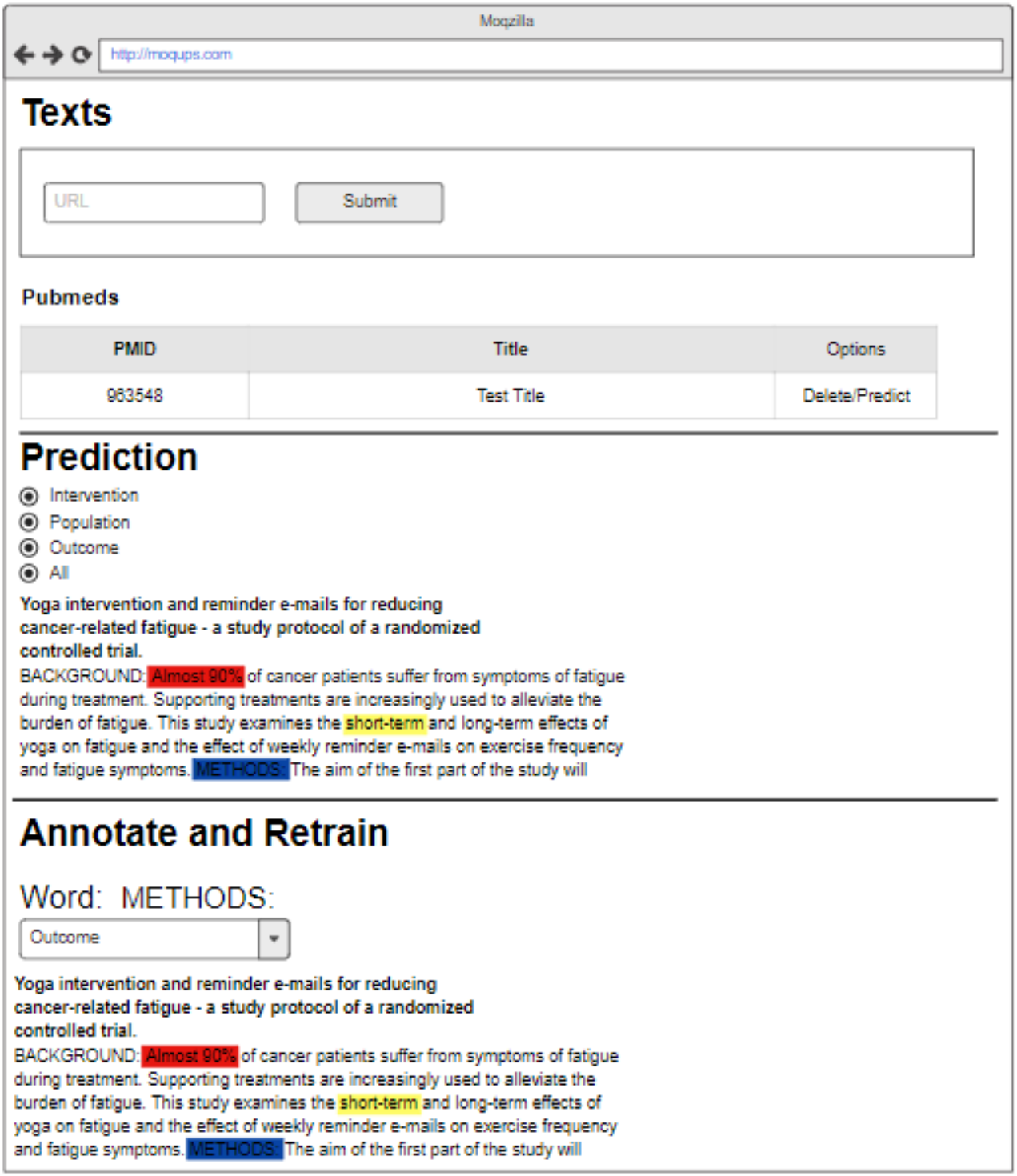
Schlich, M. (2019). *Softwaretesten nach ISTQB.* Weinheim: WILEY-VCH Verlag GmbH & Co. KGaA.

Seifried, P. (12. März 2012). *Was ist Scrum eigentlich?* Verfügbar unter: https://www.modusnext.ch/scrum/scrum-was-ist-das-eigentlich/

Steglich, T. (2009). *Zeitmanagement für Webentwickler.* Köln: O'Reilly Verlag.

Tiedemann, M. (16. Juni 2020). *Natural Language Processing (NLP): Natürliche Sprache für Maschinen*. Von alexanderthamm: https://www.alexanderthamm.com/de/blog/natural-language-processing-nlp-natuerliche-sprache-fuer-maschinen/ abgerufen

# Anhang I: Mockups



# Anhang II: Scrum – Product Backlog

US Nr.

| US Nr. | Theme | User Stories | | |
|---|---|---|---|---|
| | | As an/a ... | I want to ... | so that ... |
| 1 | Preparation | Developer | Prepare the work environment | I can start implementig the application |
| 2 | Research | Developer | Draw mockups | I can have a better overview of the application |
| 3 | Preparation | Developer | do researches about the tools | I can have a better understanding |
| 4 | Research | Developer | create the database | I can store datas |
| 5 | Preparation | Developer | have a Server | I can implement the Backend |
| 6 | Preparation | Developer | have a API | I can interrect with the Backend |
| 7 | Functional | User | upload documents | I can create predictions |
| 8 | Functional | User | see the uploadet documents | I can have an overview |
| 9 | Functional | User | remove documents | i can remove useless documents |
| 10 | Functional | User | get Prediction of a single Document | i can evaulate the abstract |
| 12 | Functional | User | see the whole abstract with the predictions | I can evaluate the whole document |
| 11 | Functional | User | see the prediction of each element separatly | I can evaluate each element separatly |
| 13 | Functional | User | annote a Prediction of a single Document | I can correct the results |
| 14 | Functional | User | correct wrong predictions compared with the abstract | I can evaluate wrong predictions |
| 15 | Functional | User | chose a word | I can do corrections |
| 16 | Functional | User | chose the right element for the word | I can correct wrong predictions |
| 17 | Functional | User | store the annotation | I can always check my annotation |
| 18 | Functional | User | remove annotations | I can delete my corrections |
| 19 | Functional | User | retrain the model with my annotation | I can imporve the machine learining |

| Acceptance Criteria | Status | Story Points | Sprint |
|---|---|---|---|
| The github is created, the excel sheet is ready to user, the drive is set up. | | 3 | 0 |
| All mockups are created and contain a good overview of what the application should look like. | | 3 | 0 |
| The product backlog is complete and provides informations about the tasks to be done. | | 3 | 0 |
| The database schema contains all informations to be saved and all relations are correct. | | 10 | 0 |
| Have a working Server | | 1 | 1 |
| Have a working API | | 5 | 1 |
| Have a Upload button which stores the document in the database | | 15 | 1 |
| See the panel on the left side | | 5 | 1 |
| Have a remove Button which updates the panel on the left side | | 2 | 1 |
| Have A Button which gets the predictions | | 1 | 2 |
| Have the abstract with the elements marked | | 10 | 2 |
| Have radio buttons to switch between results | | 6 | 2 |
| Have a Button which lets the user to annotate the results | | 2 | 3 |
| Have the whole abstract separated from the prediction abstract | | 2 | 3 |
| Have the possibility to cklick on words | | 11 | 3 |
| Have a dropdown menu with the elements | | 3 | 3 |
| Have a button which stores the corrections | | 2 | 3 |
| Have a button which deletes the corrections | | 2 | 3 |
| Have a retrain Button | | 2 | 3 |

| US accepted (done done) | MoSCoW |
|---|---|
| 18.05.2020 | Must Have |
| 18.05.2020 | Must Have |
| 18.05.2020 | Must Have |
| 18.05.2020 | Must Have |
| 02.06.2020 | Must Have |
| 02.06.2020 | Must Have |
| 02.06.2020 | Must Have |
| 02.06.2020 | Must Have |
| 02.06.2020 | Must Have |
| 17.06.2020 | Must Have |
| 01.07.2020 | Must Have |
| 17.06.2020 | Must Have |
| 01.07.2020 | Must Have |
| 01.07.2020 | Must Have |
| 01.07.2020 | Must Have |
| 01.07.2020 | Must Have |
| 01.07.2020 | Must Have |
| 01.07.2020 | Must Have |

# Anhang III: Sprint 0 - Sprint Backlog

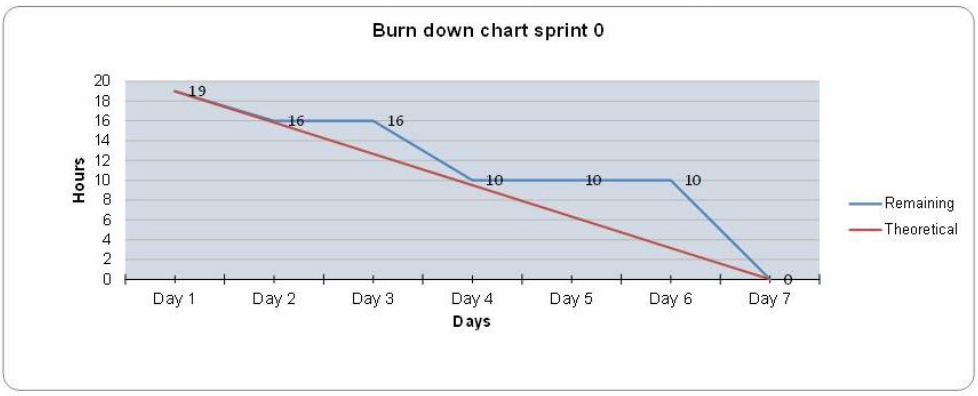
Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

## Sprint Backlog

| | |
|---|---|
| Sprint | 0 |
| Start date | 11.5.2020 |
| End date (included) | 18.5.2020 |
| Initial sprint estimation | 19 |
| Sprint goal | Get the working environment and the database ready |
| Nbre working days | 7 |

| ID US.task | Task Name | Resp. | Initial Estimate |
|---|---|---|---|
| 1.1 | Analysing the topic | Agron | 1 |
| 1.2 | Create Product Backlog | Agron | 1 |
| 1.3 | Prepare the Work Environment | Agron | 1 |
| 2.1 | Designing Mockups and System | Agron | 3 |
| 3.1 | Research React JS | Agron | 1 |
| 3.2 | Research Python | Agron | 1 |
| 3.3 | Research Flask | Agron | 1 |
| 4.1 | Create Database Schema | Agron | 5 |
| 4.2 | implement the Database | Agron | 5 |
| | | **Total** | 19 |
| | | **Remaining** | 19 |
| | | **Therical** | 19 |

| Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 3 | 3 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 5 | 5 | 5 | 5 | 5 | 5 | 0 |
| 5 | 5 | 5 | 5 | 5 | 5 | 0 |
| *19* | *16* | *16* | *10* | *10* | *10* | *0* |
| *19.0* | *15.8* | *12.7* | *9.5* | *6.3* | *3.2* | *0.0* |



# Anhang IV: Sprint 1 – Sprint Backlog

Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

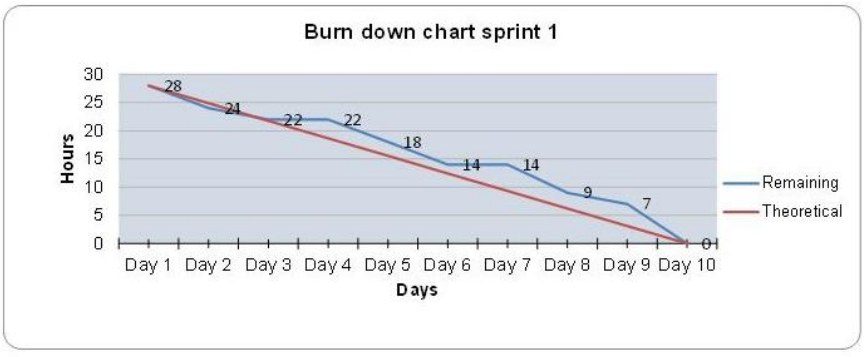
## Sprint Backlog

| | |
|---|---|
| Sprint | 0 |
| Start date | 19.05.2020 |
| End date (included) | 02.06.2020 |
| Initial Scrum effort | 24 |
| Sprint Objectives | Get the first part of the application implemented |
| Nbre working days | 10 |

| ID | Task Name | Resp. | Initial Estimate |
|---|---|---|---|
| 5.1 | Install the Flask Server | Agron | 1 |
| 6.1 | Install Python in the Project | Agron | 1 |
| 6.2 | Create API | Agron | 2 |
| 6.3 | Connect to the Database | Agron | 2 |
| 7.1 | Get Document from NCBI Database | Agron | 8 |
| 7.2 | Store the document in the Database | Agron | 5 |
| 7.3 | Create the Input field in the Frontend | Agron | 2 |
| 8.1 | Create a Table in the Frontend | Agron | 2 |
| 8.2 | Get the Data from the Backend | Agron | 3 |
| 9.1 | Create the Delete Button | Agron | 1 |
| 9.2 | Delete the document from the database | Agron | 1 |

| | |
|---|---|
| **Total** | 28 |
| **Remaining** | 28 |
| **Therical** | 24 |

| Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Day 8 | Day 9 | Day 10 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 8 | 8 | 8 | 4 | 0 | 0 | 0 | 0 | 0 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| *28* | *24* | *22* | *22* | *18* | *14* | *14* | *9* | *7* | *0* |
| *28.0* | *24.9* | *21.8* | *18.7* | *15.6* | *12.4* | *9.3* | *6.2* | *3.1* | *0.0* |



# Anhang V: Sprint 2 – Sprint Backlog

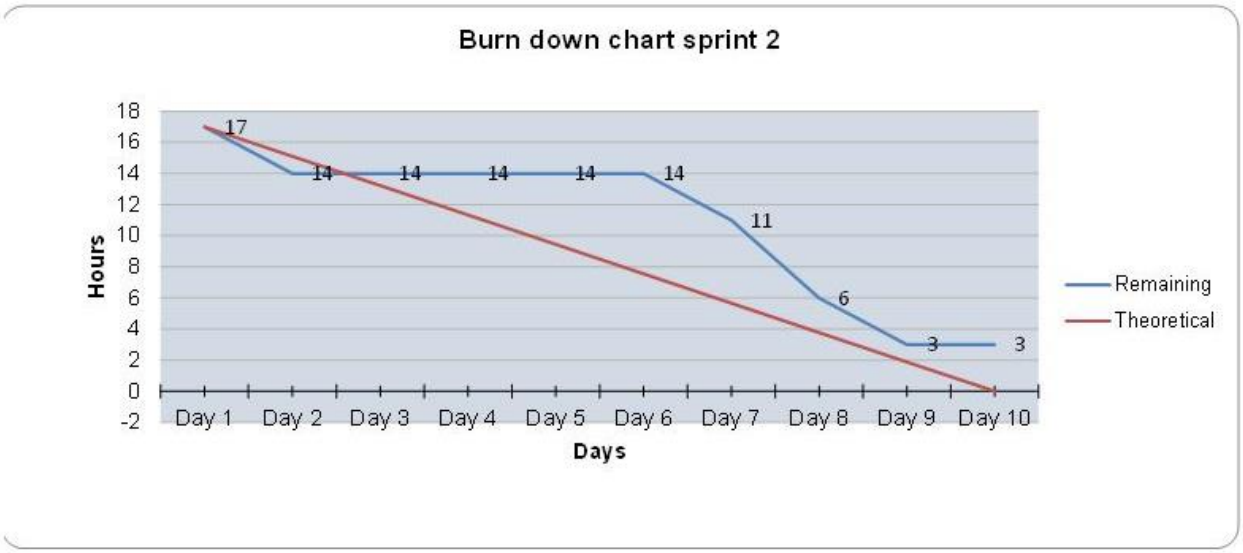
Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

## Sprint Backlog

| | |
|---|---|
| Sprint | 2 |
| Start date | 03.06.2020 |
| End date (included) | 17.06.2020 |
| Initial Scrum effort | 21 |
| Sprint Objectives | Get the second part of the application implemented |
| Nbre working days | 10 |

| ID | Task Name | Resp. | Initial Estimate |
|---|---|---|---|
| 10.1 | Hardcode the output of the Machine Learning in the backend | Agron | 1 |
| 11.1 | Get the Abstract of the document from the database | Agron | 1 |
| 11.2 | Join the predictions together | Agron | 1 |
| 11.3 | Visualize the Abstract in the frontend | Agron | 3 |
| 11.4 | Visualize the predictions in the frontend | Agron | 5 |
| 12.1 | Implement radio buttons to change the predictions | Agron | 3 |
| 12.2 | Separate the List of all the predictions | Agron | 3 |
| | | **Total** | *17* |
| | | **Remaining** | *17* |
| | | **Therical** | |

| Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Day 8 | Day 9 | Day 10 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 0 | 0 | 0 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 0 | 0 |
| *17* | *14* | *14* | *14* | *14* | *14* | *11* | *6* | *3* | *3* |
| *17.0* | *15.1* | *13.2* | *11.3* | *9.4* | *7.6* | *5.7* | *3.8* | *1.9* | *0.0* |



# Anhang VI: Sprint 3 – Sprint Backlog

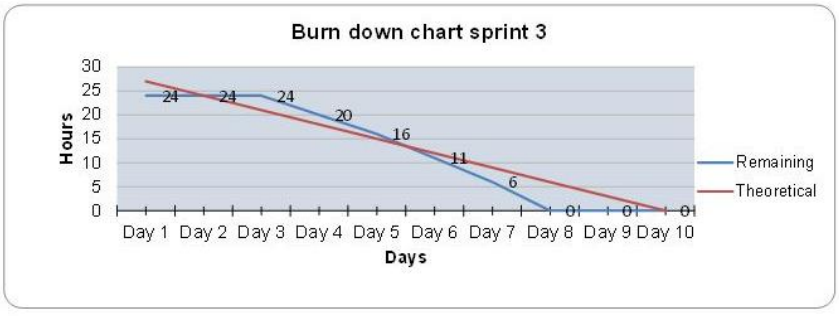
Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

## Sprint Backlog

| | |
|---|---|
| Sprint | 3 |
| Start date | 17.06.2020 |
| End date (included) | 01.07.2020 |
| Initial Scrum effort | 22 |
| Sprint Objectives | Get the third part of the application implemented |
| Nbre working days | 10 |

| ID | Task Name | Resp. | Initial Estimate |
|---|---|---|---|
| 12.1 | Implement radio buttons to change the predictions | Agron | 3 |
| 13.1 | Create the Annotate Button in the Table | Agron | 1 |
| 13.2 | Get the Abstract from the Database | Agron | 1 |
| 14.1 | Display the abstract under the Abstract with the predictions | Agron | 2 |
| 15.1 | get the idea about the function | Agron | 8 |
| 15.2 | implement each word of the abstract as an element of HTML | Agron | 1 |
| 15.3 | write the onClick Method to display the choosen word | Agron | 2 |
| 16.1 | implement a dropDown menu with the diffrent PICO elements | Agron | 1 |
| 16.2 | store the choosen PICO element to chang it in the prediction list | Agron | 2 |
| 17.1 | implement the Annotate Button in the Annotation part | Agron | 1 |
| 17.2 | store the Annotation in the DB | Agron | 1 |
| 18.1 | add a delete Button | Agron | 1 |
| 18.2 | delete the annotation from the database | Agron | 1 |
| 19.1 | Add a retrain Button | Agron | 1 |
| 19.2 | retrain the model of the machine learning | Agron | 1 |
| | | **Total** | 27 |
| | | **Remaining** | 27 |
| | | **Therical** | 22 |

| Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Day 8 | Day 9 | Day 10 |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 8 | 8 | 8 | 4 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| *24* | *24* | *24* | *20* | *16* | *11* | *6* | *0* | *0* | *0* |
| *27* | *24* | *21* | *18* | *15* | *12* | *9* | *6* | *3* | *0* |



# Selbstständigkeitserklärung des Verfassers

Ich bestätige hiermit, dass ich die vorliegende Bachelorarbeit alleine und nur mit den angegebenen Hilfsmitteln realisiert habe und ausschliesslich die erwähnten Quellen benutzt habe. Ohne Einverständnis des Studiengangsleiters und des für die Bachelorarbeit verantwortlichen Dozierenden sowie des Forschungspartners, mit dem ich zusammengearbeitet habe, werde ich diesen Bericht an niemanden verteilen, ausser an die Personen, die mir die wichtigsten Informationen für die Verfassung dieses Berichts geliefert haben und die ich nachstehend aufzähle: Herr Müller Henning und Frau Dhrangadhariya Anjani.

Agron Asani