

Dokumentation Projekt questMe

Pavithra Sureshkumar, Kevin Sautner, Ralf Zeller

Geändert 17.11.2021

Inhaltsverzeichnis

Abbildungsverzeichnis	4
Tabellenverzeichnis	5
1 Vorwort	6
2 About Us	6
3 UI Designs	7
3.1 Recherche UI Designs	7
3.2 Version 1 von UI-Konzept	9
3.3 Version 2 von UI-Konzept	11
3.4 Bisheriger Prototyp vom UI-Design	16
4 Usability Test	18
4.1 Kriterien für Usability	18
4.1.1 Responsive Webdesign	18
4.1.2 Gute Lesbarkeit	18
4.1.3 Gute Navigation	18
4.1.4 Schnelle Ladezeiten	18
4.1.5 Interessantes Design	18
4.2 Unser Usability Test	19
4.2.1 Remote User Testing	19
4.2.2 Durchführung der Remote Usability Testsitzung	19
5 Use Cases	21
5.1 Struktur	21
5.2 Use Cases Version 1	21

6	User Story	24
6.1	Struktur der User Story	24
6.2	Student	24
6.3	unregistrierter Nutzer	25
6.4	Professor (Admin)	25
7	Zielgruppen	26
7.1	Zielgruppe: Nicht registrierte Nutzer	26
7.2	Zielgruppe: Student	27
7.3	Zielgruppe: Professor	28
8	Technologien	29
8.1	Kriterien für die Technologien	29
8.2	Technologie Versionen	29
8.2.1	Node.js 16.13.0 LTS	30
8.2.2	Angular 13.0.1	30
8.2.3	Socket.io 4.3.2	30
8.2.4	MongoDB Community Edition 5.0.3	30
8.2.5	KeyCloak 15.0.2	30
8.2.6	RegEx	31
8.2.7	NLP.js 4.22.2	31
8.3	Technologie Vergleich	32
8.3.1	Vergleich zwischen Angular und Vue.js	32
8.3.2	Vergleich zwischen MongoDB und PostgreSQL	33
8.4	Technologie Diagramme	34
8.4.1	UML Komponentendiagramme	34
8.4.2	UML Komponentendiagramm: Client	34
8.4.3	UML Komponentendiagramm: Server	35
8.4.4	UML Komponentendiagramm	36
8.4.5	UML Verteilungsdiagramm	37
8.5	Datenbank	38
8.5.1	Datenhaltung	38
8.5.2	ER Diagramme	38
9	Meilensteine	40
9.1	Recherche 13.10.21	40
9.2	Zwischenpräsentation 22.10.21	40
9.3	Implementation 25.10.21	41
9.4	MVP 02.12.21	41
9.5	Endpräsentation und Enddokumentation 14.01.22	41
10	Zeitmanagement	42
10.1	Gantt-Diagramm	42

11	Risikoanalyse	43
12	Appendix	46
12.1	Ältere Versionen des Komponentendiagramms	46
12.1.1	Komponentendiagramme	46
Literatur		50

Abbildungsverzeichnis

1	UML Komponentendiagramm Client	34
2	UML Komponentendiagramm Server	35
3	UML Komponentendiagramm	36
4	UML Verteilungsdiagramm	37
5	ER Diagramm Korpus	38
6	ER Diagramm Professor	39
7	ER Diagramm Student	39
8	ER Diagramm Unregistrierter Nutzer	39
9	Gantt-Diagramm	42
10	Komponentendiagramm Client	46
11	Komponentendiagramm Server	47
12	Komponentendiagramm v1.0	48
13	Komponentendiagramm v1.1	48
14	Komponentendiagramm v1.2	49

Tabellenverzeichnis

1	Technologie Liste	29
2	Vergleich zwischen Angular und Vue.js	32
3	Vergleich zwischen MongoDB und PostgreSQL	33
4	Meilenstein Liste	40
5	Risikoanalyse Tabelle Teil 1	43
6	Risikoanalyse Tabelle Teil 2	44

1 Vorwort

In dieser Dokumentation möchten wir alle relevanten Informationen zum Projekt sammeln.

2 About Us

Hier stellen wir uns kurz vor.

Wer sind wir ?



Wir sind questMe und wir bearbeiten das Thema ChatBot. In unserem Projekt möchten wir nicht nur den ChatBot programmieren oder erstellen, sondern auch neue Technologien kennenlernen. Wir möchten neue Erfahrungen sammeln, Lernen wie man organisiert mit Konflikten umgeht und auch bestimmte Probleme angeht. Wir sind als Team immer offen für Neues.

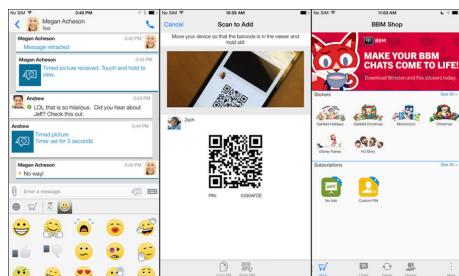
Wir sind das Team questMe.

3 UI Designs

In diesem Kapitel sollen die verschiedenen Versionen unseres UI Designs geführt werden

3.1 Recherche UI Designs

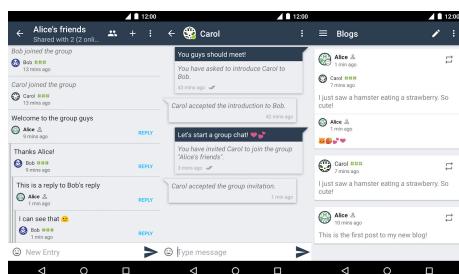
Zuerst haben wir uns mehrere UI Designs von verschiedenen Quellen angeschaut, um einen besseren Überblick über die Designmöglichkeiten zu bekommen.



Bildquelle: *Blackberry-messenger-live-free*, [o. D.]

Blackberry Messenger Live

Zuerst haben wir uns umgeschaut und nach Chatfenstern gesucht. Hier haben wir uns die Austauschung von Sprechblasen angeschaut und deren Ausrichtung.



Bildquelle: *Briar*, [o. D.]

Briar

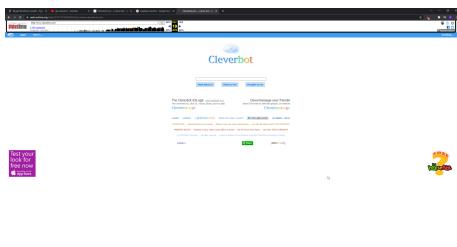
In diesem Beispiel haben wir uns wieder das Chatfenster und die verschiedenen Symbole angeschaut. Wie das Editierbutton oder das Hinzufügebutton.



Bildquelle: *Chatbot-cost-calculator*, [o. D.]

CHATBOT Cost Calculator

Hier haben wir uns ein ChatBot-Fenster angeschaut und dabei die Icons und die verschiedenen Elemente angeschaut, wie den Balken an der Oberseite oder das Sendebutton.



Cleverbot

Diesen Bot haben wir uns genauer angeschaut, weil dieser auch uns bekannt war.

Bildquelle: *Cleverbot*, [o. D.]



Telegram

Telegram haben wir uns wiederrum das Chatfenster und die Icons angeschaut.

Bildquelle: *Telegram Bild*, [o. D.]



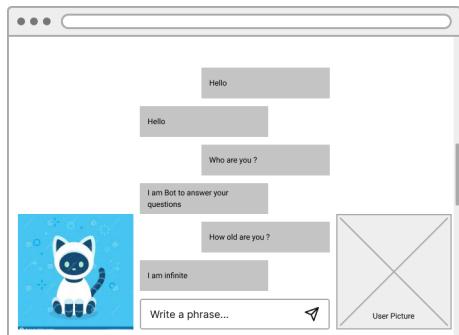
WhatsApp

Hier haben wir uns mehr auf die Ausrichtungen des Chatfensters angeschaut und wie die Sprechblasen ausgerichtet sind. Fast jeder benutzt WhatsApp deswegen haben wir uns die Struktur angeschaut, weil diese viele Benutzer bekannt ist.

Bildquelle: *Tim WhatsApp*, [o. D.]

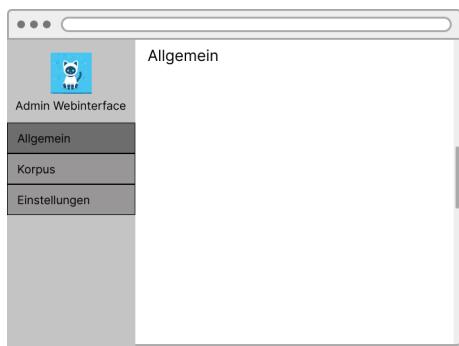
3.2 Version 1 von UI-Konzept

Das sind die älteren Entwürfe des Designs. Bei diesen Entwürfen haben wir uns nur auf die Struktur konzentriert und eine grobe Version erstellt. Uns war es hier hauptsächlich wichtig die Ideen, die wir durch die Recherche aufgenommen haben zu projektiere.



Webchat

In der alten Version unseres Webchat designs haben wir gedacht, auch für den Nutzer ein Profilbild hinzufügen. Diese Idee aber schien zu aufwendig zu sein und wir versuchen zuerst ein Minimal Viable Product zu erschaffen.



Admin Interface: Allgemein

Im Admin Interface kann man auf der linken Seite das Menü sehen. Was aber hier fehlt ist der Logout Button. Wir hatten auch keine richtigen Vorstellungen, was wir einführen möchten.



Admin Interface: Korpus 00

Im Korpus wollten wir schon von Anfang an das Hinzufügen darstellen, wussten aber nicht wie und haben herumprobiert.

The image displays three vertically stacked screenshots of a web-based administration interface for a chatbot system. The interface has a sidebar on the left with a logo and navigation links: 'Admin Webinterface' (selected), 'Allgemein', 'Korpus' (selected), and 'Einstellungen'. The main content area is titled 'Korpus' and shows a table with two rows. Each row contains a 'Frage' (Question) and an 'Antwort' (Answer). Row 1: Frage 'Wer bist du ?' and Antwort 'Ich bin ein Chatbot' with a plus sign icon. Row 2: Frage 'Mir geht es gut' and Antwort 'Ich fühle mit dir' with a plus sign icon. Below the table, there is explanatory text: 'Ich wurde von questMe erfunden und werde dir Fragen beantworten'.

Admin Interface: Korpus 01

In diesem Beispiel sieht man ein Basisbeispiel mit einer Frage die im Alltag gestellt wird und die dazugehörenden Antworten. Wie wir diese aber editieren haben wir hier noch nicht gezeigt.

Admin Interface: Korpus 02

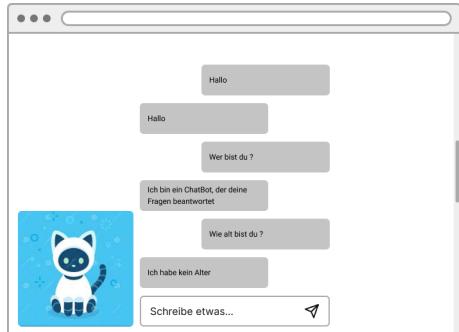
Im nächsten Beispiel sieht man eine weitere Frage und die dazugehörigen zwei Antworten. Es ist immer noch nicht bekannt, wie man Antworten und Fragen editiert oder Fragen und Antworten von verschiedenen Domänen bearbeitet.

Admin Interface: Admin Login

Hier haben wir nur einen klassischen Login Eingabefeld dargestellt, weil wir uns nicht klar waren, wie es mit der Authentifizierung und dem Admin Login funktioniert.

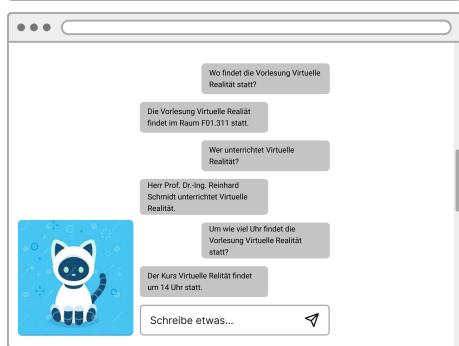
3.3 Version 2 von UI-Konzept

Das sind die neuen Entwürfe des UI Designs. In der neuen Version haben wir uns auf die feineren Elemente konzentriert und uns sogar eine mobile Versionen ausgedacht, weil wir unbedingt "mobilefirst" Programmieren möchten.



Webchat

Hier haben wir einen Beispielchat mit dem Bot dargestellt. In diesem Beispiel haben wir ein Basisgespräch geführt.



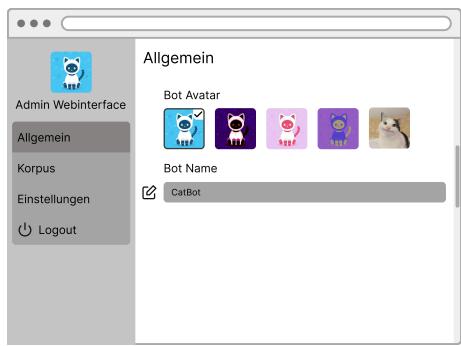
Webchat mit der Hochschuldomäne

Diesmal haben wir konkrete Hochschulfragen gestellt und die Hochschuldomäne dargestellt.

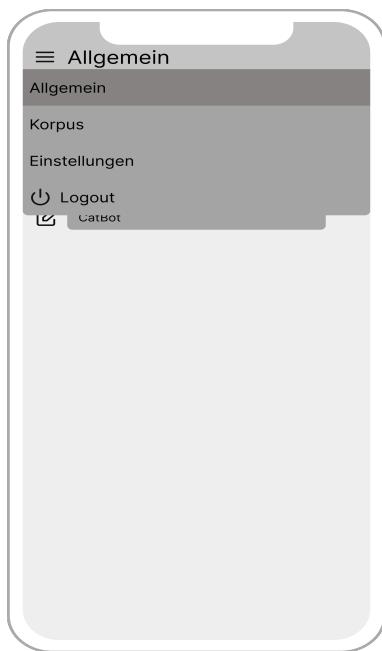


Webchat als mobile Version

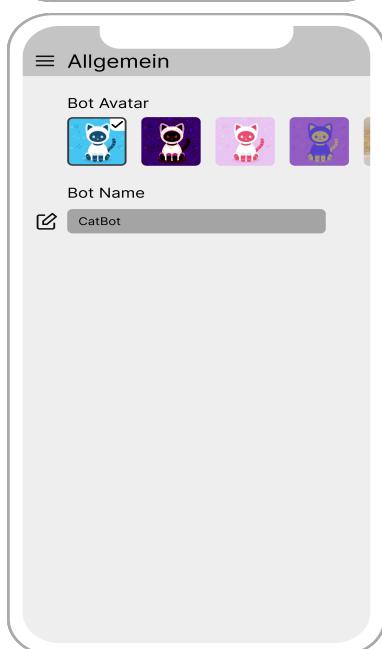
Dies ist die mobile Version des Webchats. Wir haben die Darstellung so einfach wie möglich dargestellt. Hier haben wir auch die Hochschulbezogenen Fragen gestellt.



Admin Webinterface: Allgemein
Auf der linken Seite sieht man die Kategorien, die der Admin bearbeiten kann. Im Allgemeinen kann der Admin den Bot Avatar wechseln, diese wird dann mit einem Haken gekennzeichnet. Außerdem kann der Admin den Bot Namen ändern, indem er den Editierbutton drückt.



Admin Webinterface mobil: Allgemein dropdown Menü
In der mobilen Version haben wir die Kategorien, die der Admin bearbeiten kann im dropdown Menü dargestellt.



Admin Webinterface mobil: Allgemein
Die mobile Variante funktioniert genauso wie die Webvariante. Man kann den Bot Avatar wechseln und den ChatBot Namen frei bestimmen.

The screenshot shows the Admin Webinterface with the 'Korpus' section selected. The interface includes a sidebar with 'Admin Webinterface', 'Allgemein', 'Korpus' (which is highlighted in grey), 'Einstellungen', and 'Logout'. The main area is titled 'Korpus' and 'Hochschule'. It lists three questions with their corresponding answers:

- Frage: Wo findet die Vorlesung Virtuelle Realität statt? Antwort: Die Vorlesung Virtuelle Realität findet im Raum F01.311 statt.
- Frage: Wer unterrichtet Virtuelle Realität? Antwort: Herr Prof. Dr.-Ing. Reinhard Schmidt unterrichtet Virtuelle Realität.
- Frage: Um wie viel Uhr findet die Vorlesung Virtuelle Realität statt? Antwort: Der Kurs Virtuelle Realität findet um 14 Uhr statt.

Admin Webinterface: Korpus 00

Im Korpus kann der Admin weitere Domänen, Fragen und Antworten einsehen.

This screenshot shows the same Admin Webinterface as the previous one, but with a new answer added to the first question. The 'Frage' row for 'Wo...' now has two 'Antwort' rows: the original one and a new one below it: 'Die Vorlesung findet im Raum F01.311 statt.'

Admin Webinterface: Korpus 01

Mit dem Editierbutton kann er Domänen, Fragen und Antworten hinzufügen und entfernen.

In this screenshot, a new answer is added to the first question ('Wo...') and a new answer is added to the second question ('Wer...'). The 'Frage' row for 'Wo...' now has three 'Antwort' rows. The 'Frage' row for 'Wer...' now has two 'Antwort' rows: the original one and a new one below it: 'Herr Prof. Dr.-Ing. Reinhard Schmidt unterrichtet Virtuelle Realität.'

Admin Webinterface: Korpus 02

So kann man wie im Beispiel eine weitere Antwort zu der Frage: "Wo findet die Vorlesung Virtuelle Realität statt?", hinzufügen.

Admin Webinterface: Korpus 03

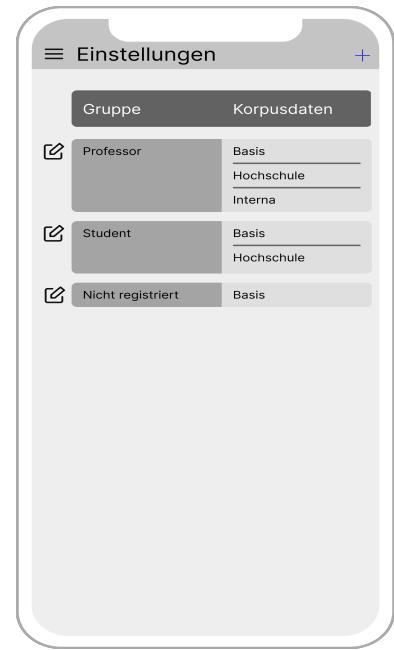
Natürlich kann man auch die hinzugefügte Antwort entfernen. Man klickt auf das Feld und das Feld erscheint rötlich und ein Müllimersymbol entsteht. Wenn man jetzt auf das Eimer-Symbol klickt kann man die Antwort löschen.

Admin Webinterface mobil: Korpus

In der mobilen Version des Korpuses kann man die Elemente genauso editieren wie im Webbrower.

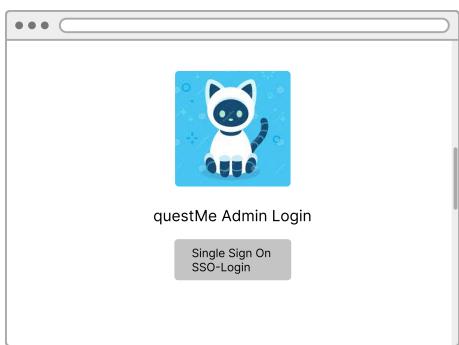
Admin Webinterface: Einstellungen

In Einstellungen kann der Admin seine Gruppen einsehen, hinzufügen und entfernen. Er kann auch die dazugehörigen Korpusdaten sehen und bearbeiten. In den Korpusdaten sind die Domänen der jeweiligen Gruppe eingetragen.



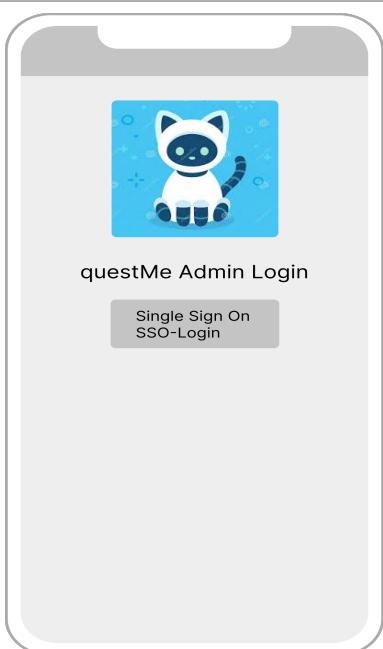
Admin Webinterface mobil: Einstellungen

In der mobilen Version kann man ebenso die gleichen Features nutzen.



Admin Webinterface: Single Sign On

Mit einem Link gelangt der Admin zu der Admin Login Seite, wo er mit Shibboleth sich einloggen kann.



Admin Webinterface mobil: Single Sign On

In der mobilen Version wird die gleiche Prozedur benutzt.

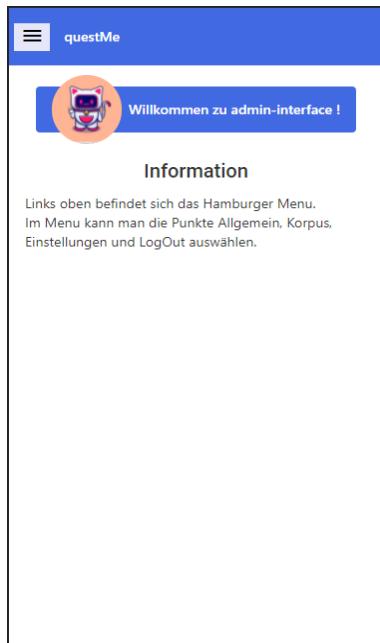
3.4 Bisheriger Prototyp vom UI-Design

Hier wird der bisherige Prototyp des UI-Designs, welche wir mit Angular bis jetzt programmiert haben, dargestellt.



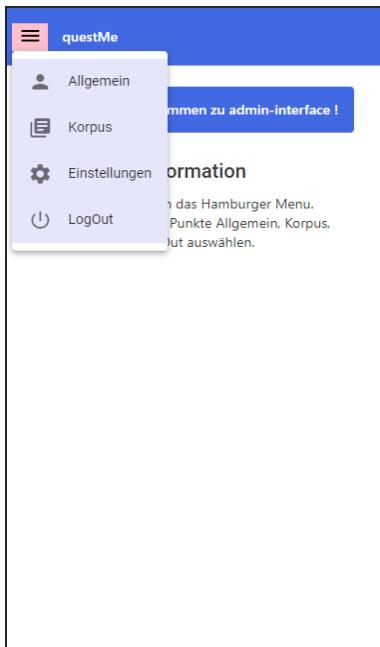
Chat Interface

Dies ist der bisherige Prototyp von dem Chat Interface. Wie man hier sieht haben wir schon auf eine gute Lesbarkeit des Chates geachtet und auch mobile-first entwickelt.



Admin Interface Infoseite

Hier haben wir die Admin Interface Infoseite, wo der Admin begrüßt wird.



Hamburgermenu für das Navigieren

Das Hamburgermenü befindet sich links oben auf der Toolbarleiste. Im Menü kann der Admin durch die einzelnen Seiten navigieren und sich auch ausloggen.

4 Usability Test

Hier werden wir unser Vorgehen des Usability Tests beschreiben und die ausführliche Durchführung des Tests.

4.1 Kriterien für Usability

Unsere Kriterien für Usability, die uns wichtig sind und für unser User Interface essentiell sind, werden hier kurz erläutern.

4.1.1 Responsive Webdesign

Wir möchten unseren ChatBot auf allen Geräten ohne Probleme ausführen lassen. Das heißt, dass wir auch auf mobilen Geräten unsere Software im Browser laufen lassen wollen. Die mobile Seite sollte dann auch auf die wichtigsten Elemente beschränkt werden, damit der Benutzer es leichter hat die Icons treffsicher mit ihren Fingern zu benutzen.

4.1.2 Gute Lesbarkeit

Unsere Anwendung sollte leicht zu lesen sein, weil das Lesen auf dem Bildschirm grundsätzlich schwieriger ist. Deswegen sollten wir auf jeden Fall auf Textgröße und Kontrastreiche Farben achten. Die Textgröße sollte mindestens 12pt haben. Auch sollten wir lange Textblöcke und Schachtelsätze vermeiden.

4.1.3 Gute Navigation

Eine übersichtliche und eine verständliche Navigation ist das wichtigste bei einer Webseite. Eine gute Navigation verhindert Verwirrung und unterstützt den Benutzer zu seinem Ziel zu gelangen. Bei der Navigation sollte man darauf achten, dass alle Verlinkungen funktionieren.

4.1.4 Schnelle Ladezeiten

Die Webseite sollte ihre Inhalte schnell laden und keine großen Verzögerungen aufzeigen. Sie sollte bei ungefähr drei Sekunden liegen, um keine User zu verlieren.

4.1.5 Interessantes Design

Eine Konsistente Einhaltung von bestimmten Farben ist sehr wichtig. Der erste Eindruck von einer Webseite zeigt schon, ob die User die Webseite benutzen möchten.

oder nicht. So können Benutzer durch Bilder mit schlechter Qualität oder zu grellen Farben abgeschreckt werden. Pop-Ups sollten in Grenzen gehalten werden oder vermieden werden.

4.2 Unser Usability Test

Hier beschreiben wir welche Methode wir zum Testen unserer Usability Kriterien benutzen möchten.

4.2.1 Remote User Testing

Zuerst plannen wir was wir testen möchten und dann wie wir testen möchten. In unserem Fall haben wir uns für remote testing entschieden. Als nächstes überlegen wir uns besondere Tasks, die der User absolvieren muss, um zu erkennen, ob unsere Kriterien eingehalten werden und was wir Verbessern sollten. Die Szenarien sollten so einfach und realistisch sein, dass der Benutzer es leicht durchführen kann. Dann müssen wir auch Tester finden, welche in unserer Zielgruppe passen.

4.2.2 Durchführung der Remote Usability Testsitzung

Der erste Schritt beinhaltet den Usability-Testplan. Dieser beinhaltet den Zweck des Tests, die Kosten und Zeiteinschätzung zur Durchführung des Tests, das Testskript mit den Usability-Testaufgaben und die Rekrutierung der Testteilnehmer.

Bei der Rekrutierung werden die User ausgesucht am besten werden alle Zielgruppen gedeckt. Jeder Testteilnehmer/in wird einzeln durch das Usability Test durchgeführt. Geplant sind insgesamt vier bis sechs Teilnehmer. Nach dem Aussuchen der Testteilnehmer wird am Tag des Tests die Einverständniserklärung für die Teilnahme und Datenschutz von den Usern unterschrieben. Die Einverständniserklärung sollte also schon vorbereitet sein. Auch die Tasks werden von vorneherein mitbestimmt. Nach dem Unterschreiben werden die Testteilnehmer mittels eines Briefings benachrichtigt, wie der Test abläuft und was zu beachten ist. Außerdem wird ausdrücklich auch vermittelt, dass es kein Test ist, um ihre Leistungsfähigkeit zu beurteilen, sondern dient nur für die Weiterentwicklung und zur Bewertung unserer Software.

Als nächstes findet die Pre-Session statt. In dieser Session finden wir heraus, welche Erfahrungen die Testteilnehmer mit der zu testenden System hat und welche Interessen er vertritt und was er von Chatbots hält. Nach der Pre-Session werden die Testteilnehmer gebeten, laut-denkend ihre Testaufgaben durchzuführen. Der Moderator sollte still zuhören und Beobachten. Ganz wichtig ist es nicht bei Schwierigkeiten einzugreifen, weil es sonst den Test manipuliert.

Eine Aufgabe könnte sein den Testteilnehmer zu bitten, dass Sie den Chatbot dazu bringen eine Information zu vermitteln, welche die Teilnehmer haben möchten. Der Testteilnehmer könnte Fragen stellen, welche der Chatbot kennt oder auch nicht. Der Moderator schreibt sich auf, wie der Testteilnehmer auf sein Ergebnis kommt, oder auch gewisse Schwierigkeiten zeigt und bei Aufgaben stockt. Es könnte sein, dass man dann den Korpus Erweitern oder Anpassen müsste. Alles wird gründlich dokumentiert, während der Testteilnehmer laut-denkend seine Aufgaben erledigt.

Bei uns ist der Moderator der Protokollant, da es remote abläuft. Wir überlegen außerdem die Session aufzunehmen.

Abschließend, in der Post-Session, werden dann die Teilnehmer auf den Gesamteindruck des zum testenden System befragt. Es ist sehr wichtig eine ausführliche Dokumentation zu schreiben, um jeden Eindruck und jedes Vorgehen festzuhalten.

Nach der Usability-Testsitzung werden die Befunde zusammengeschrieben und die Ergebnisse werden ausgewertet. Anschließend wird der Usability-Testbericht geschrieben. Der Usability-Testbericht beinhaltet nicht nur die Usability-Probleme, sondern auch die gelungenen Usability-Befunde.

5 Use Cases

In diesem Kapitel haben wir unsere Use Cases gesammelt.

5.1 Struktur

Als <Akteur> möchte ich <Funktion>, um <Nutzen> zu erreichen.

5.2 Use Cases Version 1

Hier listen wir unsere ersten Ideen auf, die wir erfüllen möchten.

Chatfenster

- u10001. Als Nutzer möchte ich einen Sendebutton betätigen, um eine Nachricht abzuschicken.
- u10002. Als Nutzer möchte ich ein Eingabefeld für meine Fragen haben, um eine Antwort zu erhalten.
- u10003. Als Nutzer möchte ich einen Chatfenster haben, um eine Nachricht zu erhalten.
- u10004. Als Nutzer möchte ich das Icon vom Bot sehen, um die Nachricht des Bots von meiner zu unterscheiden.
- u10005. Als Nutzer möchte ich verschiedene farbige Sprechblasen sehen, um die Nachricht des Bots von meiner zu unterscheiden.
- u10006. Als Nutzer möchte ich hingewiesen werden, wo ich zu Schreiben habe, um eine Nachricht verfassen zu können.

Hinweis: Ein Teil der Professoren übernimmt administrative Tätigkeiten des Chatbots. Dadurch ist mit dem erwähnten Administator immer ein administrativ tätiger Professor gemeint.

Admin Interface: Allgemein

- u20001. Als Admin möchte ich ein dropdown Menü oder etwas gleichwertiges haben, um auf mein Allgemein zu wechseln.
- u20002. Als Admin möchte ich verschiedene Icons zur Auswahl haben, um mein Bot Avatar zu wechseln.
- u20003. Als Admin möchte ich ein Haken als Bestätigung sehen, um zu sehen, welchen Bot Avatar ich gewählt habe.
- u20004. Als Admin möchte ich ein Editierbutton haben, um mein ChatBot Namen zu ändern.
- u20005. Als Admin möchte ich ein Eingabefeld benutzen können, um mein Chat-Bot Namen eingeben zu können.
- u20006. Als Admin möchte ich eine gefärbte Fläche sehen, um ein Eingabefeld erkennen zu können.
- u20007. Als Admin möchte ich die ausgewählte Fläche in einer anderen Farbe sehen, um zu erkennen ob ich im Allgemein bin.

Admin Interface: Korpus

- u30001. Als Administrator möchte ich eine Liste mit allen Fragen und Antworten, um einen Überblick über den Korpus zu haben.
- u30002. Als Administrator möchte ich einen Button, der einen neuen Eintrag hinzufügt, um neue Fragen und Antworten hinzuzufügen zu können.
- u30003. Als Administrator möchte ich eine Möglichkeit zum Hinzufügen von Fragen, um neue Fragen hinzuzufügen zu können.
- u30004. Als Administrator möchte ich eine Möglichkeit zum Hinzufügen von Fragen, um neue Fragen hinzuzufügen zu können.
- u30005. Als Administrator möchte ich eine Möglichkeit zum Hinzufügen von Antworten, um neue Antworten hinzuzufügen zu können.
- u30006. Als Administrator möchte ich eine Möglichkeit zum Entfernen von Fragen, um Fragen entfernen zu können.
- u30007. Als Administrator möchte ich eine Möglichkeit zum Entfernen von Antworten, um Antworten entfernen zu können.
- u30008. Als Administrator möchte ich eine Möglichkeit einen Eintrag „Fragen und Antworten“ bearbeiten zu können, um den Eintrag zu ändern.

Node.js Allgemein

- u40001. Als Nutzer möchte ich eine bidirektionale Kommunikation zwischen dem Client und Server, um direkt mit dem Bot kommunizieren zu können.
- u40002. Als Nutzer möchte ich, dass der ChatBot meinen Kontext versteht, um mit dem Bot nach Kontext zu chatten.
- u40003. Als Administrator möchte ich die Möglichkeit den Korpus des ChatBots persistent zu speichern, um auf den Korpus zuzugreifen zu können.
- u40004. Als Nutzer möchte ich, dass der ChatBot über eine Webadresse erreichbar ist, um mit dem ChatBot online zu kommunizieren.

KeyCloak

- u50001. Als Admin möchte ich mich in KeyCloak einloggen können, um es zu verwalten.
- u50002. Als Admin möchte ich mich in das Admin-Interface einloggen können, um die Einstellungen des Chatbots zu verwalten.
- u50003. Als Hochschulangehöriger möchte ich mich mit dem Shibboleth SSO der Hochschule einloggen, um relevante Daten mitzuteilen.
- u50004. Als Admin möchte ich schnellen Zugriff auf das KeyCloak-Webinterface über das Admin-Interface, um Zeit zu sparen.
- u50005. Als Admin möchte ich neue Nutzergruppen erstellen, um zielgerichteter Fragen beantworten zu können.
- u50006. Als Admin möchte ich einer Nutzergruppe einen neuen Fragensatz zuweisen, um die möglichen Fragen für diese Gruppe zu erweitern.
- u50007. Als Admin möchte ich einen, zu einer Nutzergruppe zugewiesenen, Fragensatz entfernen, um möglichen Fragen für diese Gruppe einzuschränken.
- u50008. Als Admin möchte ich Nutzer verwalten, um bei Bedarf Änderungen vorzunehmen.
- u50009. Als Admin möchte ich die Login-Seite anpassen, um sie nach meinen Vorstellungen zu ändern.

6 User Story

Hier werden wir unsere User Stories zu den einzelnen Zielgruppen auflisten. Wir haben die Zielgruppen nicht registrierte Nutzer, Professor und Student.

6.1 Struktur der User Story

User Story:

Als <Akteur> möchte ich <Funktion>, um <Nutzen> zu erreichen.

Akzeptanzkriterien:

Szenario:

kurze Beschreibung des Szenarios

Wenn ich ...

und ...

Dann ...

6.2 Student

User Story:

Als Student möchte ich meine Informationen von meinem Stundenplan abrufen können, um meine Vorlesungen einsehen zu können.

Akzeptanzkriterien:

Szenario:

Der angemeldete Student fragt nach, wann die Vorlesung Virtuelle Realität stattfindet.

Wenn ich nach der Uhrzeit von der Vorlesung Virtuelle Realität nachfrage und mich im Chatfenster befindet.

Dann bekomme ich meine Vorlesungsinformationen zu Virtuelle Realität.

6.3 unregistrierter Nutzer

User Story:

Als unregistrierter Nutzer, möchte ich Zugang zur Chatbot Chat Seite haben, damit ich einen Smalltalk mit dem Chatbot führen kann

Akzeptanzkriterium:

Szenario:

Als unregistrierter Nutzer, möchte ich Zugang zur Chatbot Chat Seite haben, damit ich einen Smalltalk mit dem Chatbot führen kann

Wenn ich ein unregistrierter Nutzer bin und mich auf der Chatbot Chatseite befinde und in das Chatfenster Fragen schreibe und diese abschicke.

Dann antwortet der Chatbot auf meine Fragen mit allgemeinen Antworten.

6.4 Professor (Admin)

User Story:

Als Professor möchte ich den Korpus des Chatbots um eine Frage und Antwort Möglichkeit erweitern, um den Nutzern ein größeres Repertoire zu bieten.

Akzeptanzkriterium:

Szenario:

Der angemeldete Professor möchte eine neue Frage-Antwort Möglichkeit hinzufügen.

Wenn ich mich auf der Korpus-Seite des Admin-Interfaces befinde und auf den Button zum hinzufügen einer Frage-Antwort Möglichkeit drücke. Dann kann ich eine neue Frage und eine dazugehörige Antwort eintippen.

7 Zielgruppen

In diesem Abschnitt Listen wir unsere drei Zielgruppen: Nicht registrierte Nutzer, Student und Professor. Zu den Zielgruppen werden wir jeweils, ihre Probleme, was ihre geforderten Eigenschaften sind beschreiben und was bei unserer Software für Sie einzigartig ist.

7.1 Zielgruppe: Nicht registrierte Nutzer

Probleme:

- Interesse an einem Smalltalk mit einem Chatbot
- Möchten einfache Fragen beantwortet bekommen
- Haben bisher mit sehr monoton antworteten Chatbots gechattet

Eigenschaften

- Nicht unbedingt technikaffin
- Möchten eine leichte Bedienung
- Haben sehr wenig Erfahrung mit Chatbots
- Interesse an einem Smalltalk
- Möchten unterhalten werden
- Kennen Whatsapp, Skype, etc.

Alleinstellungsmerkmal / Einzigartigkeit:

- Möchten sich sehr schnell in die Chatoberfläche einfinden
- Möchte ein einfaches und simples Userinterface
- Möchten, dass der Chatbot wie ein Mensch mit ihnen chattet

7.2 Zielgruppe: Student

Probleme:

- Studenten finden ihre Raumnummer nicht mehr
- Studenten wissen nicht, wann Ihre Vorlesung stattfindet
- Studenten müssen immer ihren Stundenplan einsehen und verlieren dabei Zeit
- Studenten müssen immer manuell nach Informationen zu ihren Kursen suchen

Eigenschaften

- Studenten möchten alles einfacher
- Keine komplizierten Umwege
- Einfache Bedienung
- Gutes bzw. ansprechendes Design
- Gezielte Antworten auf bestimmte Fragen

Alleinstellungsmerkmal / Einzigartigkeit:

- Studenten können mehrere Fragen stellen
- Eine leichte, aber ansprechende Bedienung des Chats
- Bekannte Chatoberfläche
- Keine komplizierten Umwege

7.3 Zielgruppe: Professor

Probleme:

- Professoren finden ihre Raumnummer nicht mehr
- Professoren wissen nicht, wann Ihre Vorlesung stattfindet
- Professoren müssen immer ihren Stundenplan einsehen und verlieren dabei Zeit
- Professoren wissen nicht, ob ein Raum frei ist, wenn sie sich z.B. einen größeren suchen müssen
- Professoren wissen nicht, wann der Prüfungstermin für ihre Vorlesung ist
- Professoren wollen ihren Studenten Informationen übermitteln
- Professoren wollen ihren Studenten bei Problemen helfen

Eigenschaften

- Professoren möchten alles einfacher
- Keine komplizierten Umwege
- Einfache Bedienung
- Gutes bzw. ansprechendes Design
- Gezielte Antworten auf bestimmte Fragen

Alleinstellungsmerkmal / Einzigartigkeit:

- Professoren können mehrere Fragen stellen
- Eine leichte, aber ansprechende Bedienung des Chats
- Bekannte Chatoberfläche
- Keine komplizierten Umwege
- Professoren übernehmen die Rolle des Admins
- Professoren können die Einstellungen des Chatbots verwalten
- Professoren können den Korpus um Fragen und Antworten erweitern

8 Technologien

In diesem Kapitel soll alles zum Thema Technologien zusammengefasst sein. Zusätzlich sollen Schaubilder und Diagramme das Verständies für die Technologien erleichtern. Sowie Beschreibungen und Erklärungen zu der Wahl der einzelnen Technologien.

8.1 Kriterien für die Technologien

Bei der Wahl der Technologien haben wir nach Möglichkeit LTS Versionen gesucht. Damit wir eine Applikation erstellen können, die für lange Zeit Sicherheitsupdates bekommt. Zusätzlich haben wir als Ziel, dass die Software als ingesamtes Paket sehr lange stabil läuft und dadurch Ausfälle minimiert werden.

8.2 Technologie Versionen

Hier sollen die einzelnen Technologien im Einzelnen beschrieben werden. Sowie ihre geplante Funktion in unserem Projekt. In der unten angegebenen Liste soll ein Überblick über die im Projekt verwendeten Technologien bereitgestellt werden.

	Node.js 16.13.0 LTS
	Angular 13.0.1
	Socket.io 4.3.2
	MongoDB Community Edition 5.0.3
	KeyCloak 15.0.2
	NLP.js 4.22.2

Tabelle 1: Technologie Liste

8.2.1 Node.js 16.13.0 LTS

Als Basis für Angular, den Chatbot und Keycloak nutzen wir eine LTS Version von Node.js. Auf dem Node.js Server verwenden wir zusätzlich die Express Erweiterung, um eine Kommunikation zu den Servern herstellen zu können. Für uns war sehr wichtig, dass wir eine sehr stabile Basis für die Anwendung, die wir entwickeln haben. Deswegen haben wir uns ebenfalls informiert, ob alle geplanten oder in Frage kommenden Technologien mit der LTS Version kompatibel sind. Nach unserem derzeitigen Stand empfehlen die Entwickler der Bibliotheken und Frameworks eine LTS Version von Node.js zu verwenden. Bildquelle:*Node.js Icon*, [o. D.]

8.2.2 Angular 13.0.1

Wir verwenden Angular für die Frontend-Entwicklung. Angular ist die Basis, um das Chat- und das Admin-Interface zu erstellen. Um unsere UIs noch effektiver zu gestalten werden wir vereinzelt auf Angular Material zurückgreifen. Dort gibt es zu häufig genutzten UI Komponenten "Code Snippets", die ausführlich getestet wurden. Bildquelle:*Angular Icon*, [o. D.]

8.2.3 Socket.io 4.3.2

In unserem Projekt nutzen wir Socket.io, um eine bidirektionale Kommunikation zwischen dem Bot und der Benutzeroberfläche herzustellen. Es ermöglicht uns, dass ein User mit dem Chatbot ohne merkbare Verzögerung chatten kann. Bildquelle:*Socket.io Icon*, [o. D.]

8.2.4 MongoDB Community Edition 5.0.3

Wir haben uns für die Datenbank mongoDB entschieden, da es ein sehr einfaches Datenbank Modell bereitstellt. MongoDB ist eine NoSql Datenbank wodurch auch der Verwaltungsaufwand der Datenbank minimiert wird. In der Datenbank wird der Korpus des Chatbots gespeichert. Bildquelle:*MongoDB Icon*, [o. D.]

8.2.5 KeyCloak 15.0.2

In unserem Projekt verwenden wir Keycloak, damit sich der Admin sicher in das Admin-Webinterface einloggen kann. Zusätzlich werden wir das Shibboleth Single-Sign-On Verfahren von der Hochschule integrieren. Damit Hochschulangehörige die Möglichkeit haben mit ihrem Hochschul Account sich einloggen zu können. Dadurch können wir zusätzlich begrenzen welche Personengruppen auf welche Daten Zugriff haben. Bildquelle:*Keycloak Icon*, [o. D.]

8.2.6 RegEx

Wir haben in unserem Projekt anfangs aus Sicherheitsgründen RegEx eingeplant. RegEx dient in unserem Projekt als Fallback. Um ein Scheitern des Projektes zu verhindern, falls NLP.js sich nicht in unser Projekt integrieren lassen sollte. Damit der Chatbot dennoch Begriffe und Sätze verstehen kann. Nach unserem derzeitigen Stand brauchen wir nicht auf eine RegEx Implementation zurückzugreifen, da NLP.js sich sehr gut in unser Projekt integrieren lässt.

Als Beispiel für den Satz "Wie geht es dir?". Müsste ein User oder Admin folgenden RegEx Befehl integrieren:

(Wie|wie)\s*geht\s*es\s*dir\s*\?

Damit man nach "Wie/wie geht es dir ?" mit und ohne Leerzeichen prüfen kann.

Bildquelle:*Regex Icon*, [o. D.]

8.2.7 NLP.js 4.22.2

Damit wir NLP.js nutzen können benötigen wir eine LTS Version von Node.js. Diese Begrenzung haben wir zusätzlich in die Technologie Entscheidung einbezogen. Indem NLP.js ab der Version 4 und höher modular ist. Wird uns ermöglicht leichter eigene Module/Plugins für das Framework zu schreiben. Dadurch können wir einen ChatBot mit verbesserter Spracherkennung gestalten. Bildquelle:*NLP Icon*, [o. D.]

8.3 Technologie Vergleich

In diesem Bereich wollen wir unsere Design Entscheidung für eine Technologie durch einen Vergleich darstellen. Dafür haben wir für die Themen "Vergleich zwischen Angular und Vue.js" und "Vergleich zwischen MongoDB und PostgreSQL" eine Tabelle erstellt.

8.3.1 Vergleich zwischen Angular und Vue.js

Als wir uns für ein Framework zur Entwicklung des Frontends entscheiden mussten, kamen Angular und Vue.js in unsere engere Auswahl. Damit wir uns einen besseren Überblick über die Eigenschaften dieser Kandidaten verschaffen und ihre Vor- und Nachteile besser abwägen können, haben wir uns diese Tabelle erstellt.

	Angular	Vue.js
Framework, Library, Platform	Entwicklungsplattform (Development platform)	Progressive Framework
Gründer	Google	ehemaliger Google Mitarbeiter
Technology Typ	MVC Framework	MVVM Framework
Programmiersprache	TypeScript	JavaScript
Performance	niedrig	hoch
Größe	500 kB	80 kB
Lernkurve	Eine steile Lernkurve	Eine geringe Lernkurve
Dokumentation	vorhanden	vorhanden
Datenbindung	Bi-directional	Bi-directional
Rendering	beim Client	beim Server
Code reuse	möglich	Ja, CSS und HTML
Skalierbarkeit	sehr hoch	hoch
Testbarkeit	mit einem Tool	mehrere Tools benötigt
Vollständige Web App	Kann als standalone Basis verwendet werden	benötigt Third Party Tools
Lizenz	MIT License	MIT License

Tabelle 2: Vergleich zwischen Angular und Vue.js

Unsere Entscheidung fiel schlussendlich auf Angular. Da es bereits länger auf dem Markt ist und auch einen höheren Marktanteil hat, ist es generell schon interessant es sich mal anzuschauen und damit zu arbeiten. Hinzu kommt die sehr gute Testbarkeit, für die man lediglich ein einziges Tool benötigt. Ein großer Nachteil, der beim Blick auf

die Tabelle sofort ins Auge sticht, ist aber die Performance. Allerdings haben wir sie nur auf niedrig eingestuft, da am Anfang viel mitgeladen werden muss. Bei größeren Projekten bietet Angular mehr Stabilität und Performance als seine Konkurrenz in diesem Vergleich.

8.3.2 Vergleich zwischen MongoDB und PostgreSQL

Ähnliche Gedanken wie für unser Frontend mussten wir uns auch für unser Backend stellen. Genauer, welche Datenbank wollen wir verwenden? Da wir in der Datenbank den Korpus unseres Chatbots in Form von JSON-Dateien speichern wollen, haben wir uns zwei der populärsten Lösungen rausgesucht und diese in einer Tabelle gegenübergestellt.

	MongoDB	PostgreSQL
Primäres Datenbankmodell	Dokumentenorientiert	Relationales DBMS
Entwickler	MongoDB. Inc	PostgreSQL Global Development Group
Datenschema	Schemafrei (NoSQL)	Ja
Programmiersprachen	JavaScript + 28 weitere	JavaScript + 9 weitere
Query Language	MQL	SQL
Maximale Dateigröße	16 MB	1 GB
Lernkurve	Eine geringe Lernkurve	Eine geringe Lernkurve
Dokumentation	vorhanden	vorhanden
Skalierbarkeit	sehr hoch	hoch
Lizenz	Open Source	Open Source

Tabelle 3: Vergleich zwischen MongoDB und PostgreSQL

Wir haben uns schlussendlich dafür entschieden, MongoDB zu nutzen. Da es bereits ein schemafreies und Dokumentenorientiertes Datenbankmodell ist, kommt uns das sehr gelegen. Ein Nachteil, der sofort auffällt, ist die Maximale Dateigröße von gerade einmal 16 MB. Da wir aber nicht davon ausgehen, dass unsere Dateien diese Größe erreichen werden, stellt das für uns kein Hindernis dar.

8.4 Technologie Diagramme

Hier werden die Zusammenhänge der Technologien in Form von verschiedenen Diagrammen vorgestellt.

8.4.1 UML Komponentendiagramme

Hier werden die UML Verteilungsdiagramme von der Server- und Client-Seite dargestellt.

8.4.2 UML Komponentendiagramm: Client

Hier wird das UML-Komponentendiagramm Client dargestellt.

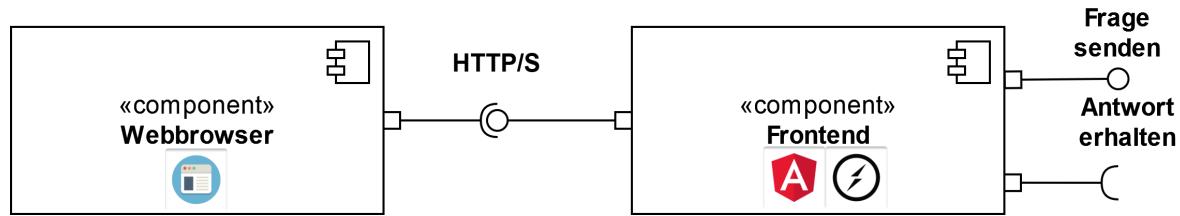


Abbildung 1: UML Komponentendiagramm Client

In der Abbildung 1 sieht man, dass der Webbrower und das Frontend auf der Client Seite. Das Frontend wird mit Angular entwickelt. Das besitzt aber selber auch einen node.js Server. Per Socket.io Client wird dann eine Verbindung zwischen den Backend produziert. Per HTTP/S wird eine Verbindung zwischen Webbrowser und dem Frontend entwickelt. Das Frontend component sendet eine Frage und erhält anschließend eine Antwort vom Backend.

8.4.3 UML Komponentendiagramm: Server

Hier wird der UML-Komponentendiagramm Server dargestellt.

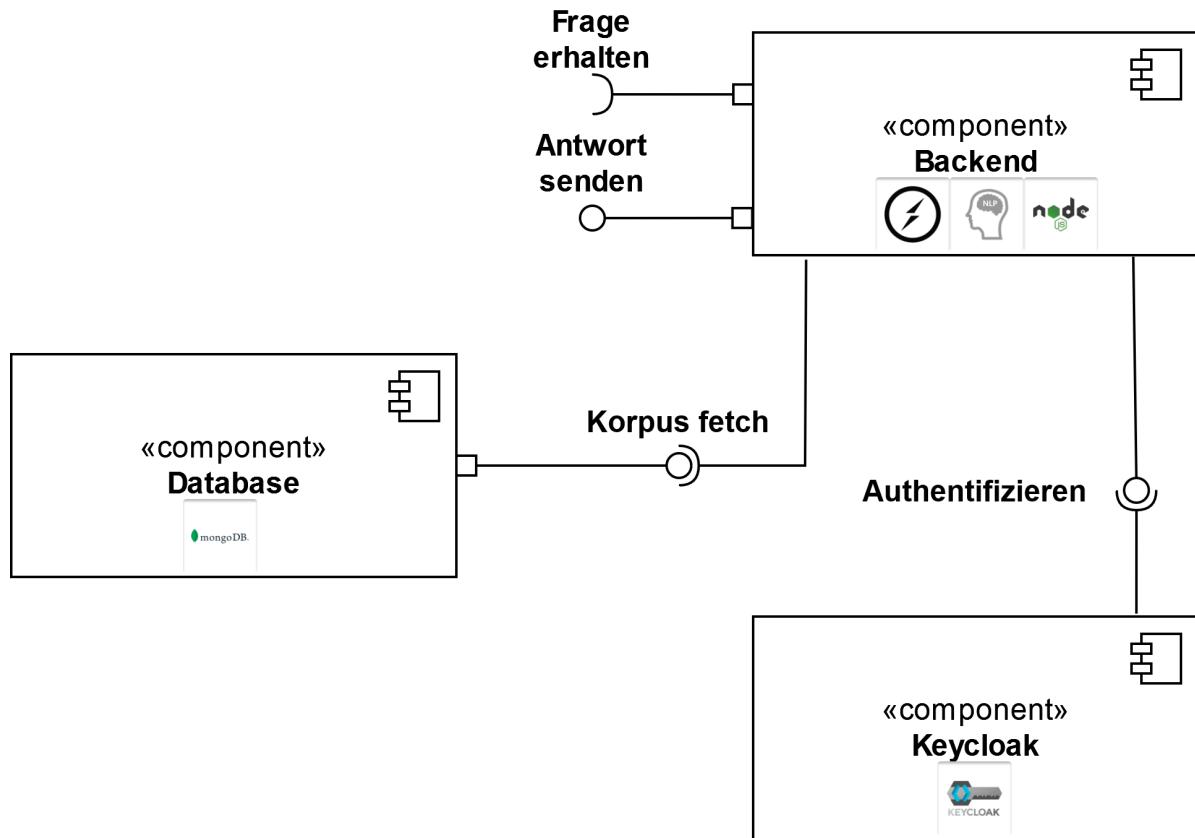


Abbildung 2: UML Komponentendiagramm Server

In der Server-Seite befindet sich das Backend, die Datenbank und das KeyCloak. Im Backend befindet sich der Socket.io Server, NLP und ein node.js Server. Das Backend erhält die Frage vom Frontend und schickt daraufhin eine Antwort mit Hilfe des Socket.io Servers zurück. Von der Datenbank wird der Korpus, und das Backend, vermittelt. KeyCloak dient zur Authentifizierung und hat ebenfalls einen node.js Server.

8.4.4 UML Komponentendiagramm

Hier sieht man die ganze Darstellung von dem Komponentendiagramm mit Server- und Client-Seite.

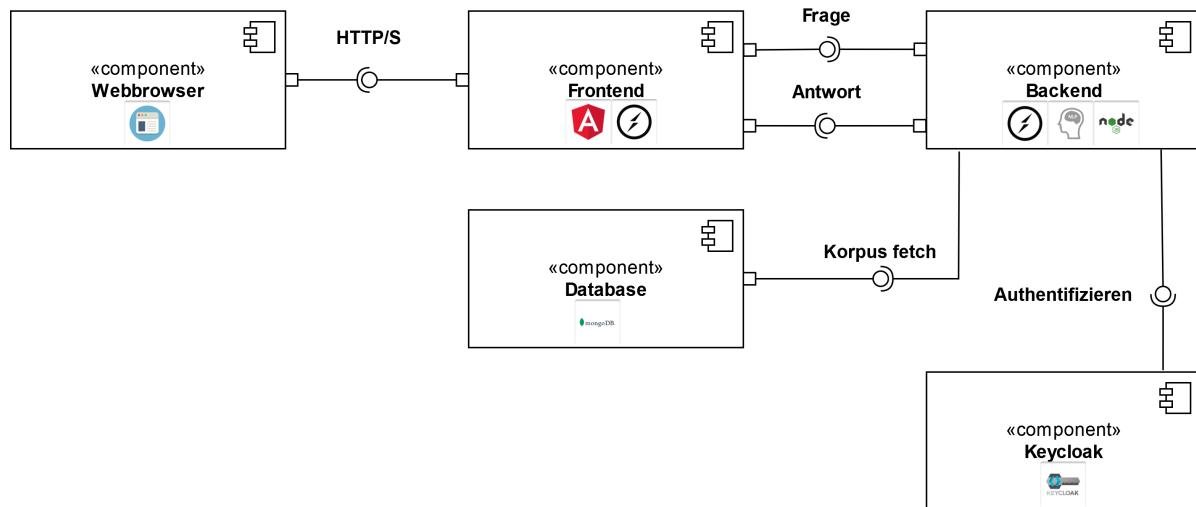


Abbildung 3: UML Komponentendiagramm

In dieser Abbildung 3 sieht man die Client- und Server-Seite. Als Austausch zwischen Frontend und Backend haben wir eine Frage und eine Antwort dargestellt. Des weiteren besitzt das Frontend, das Backend und KeyCloak einen node.js Server.

8.4.5 UML Verteilungsdiagramm

Hier sieht man die ganze Darstellung unseres Verteilungsdiagramms

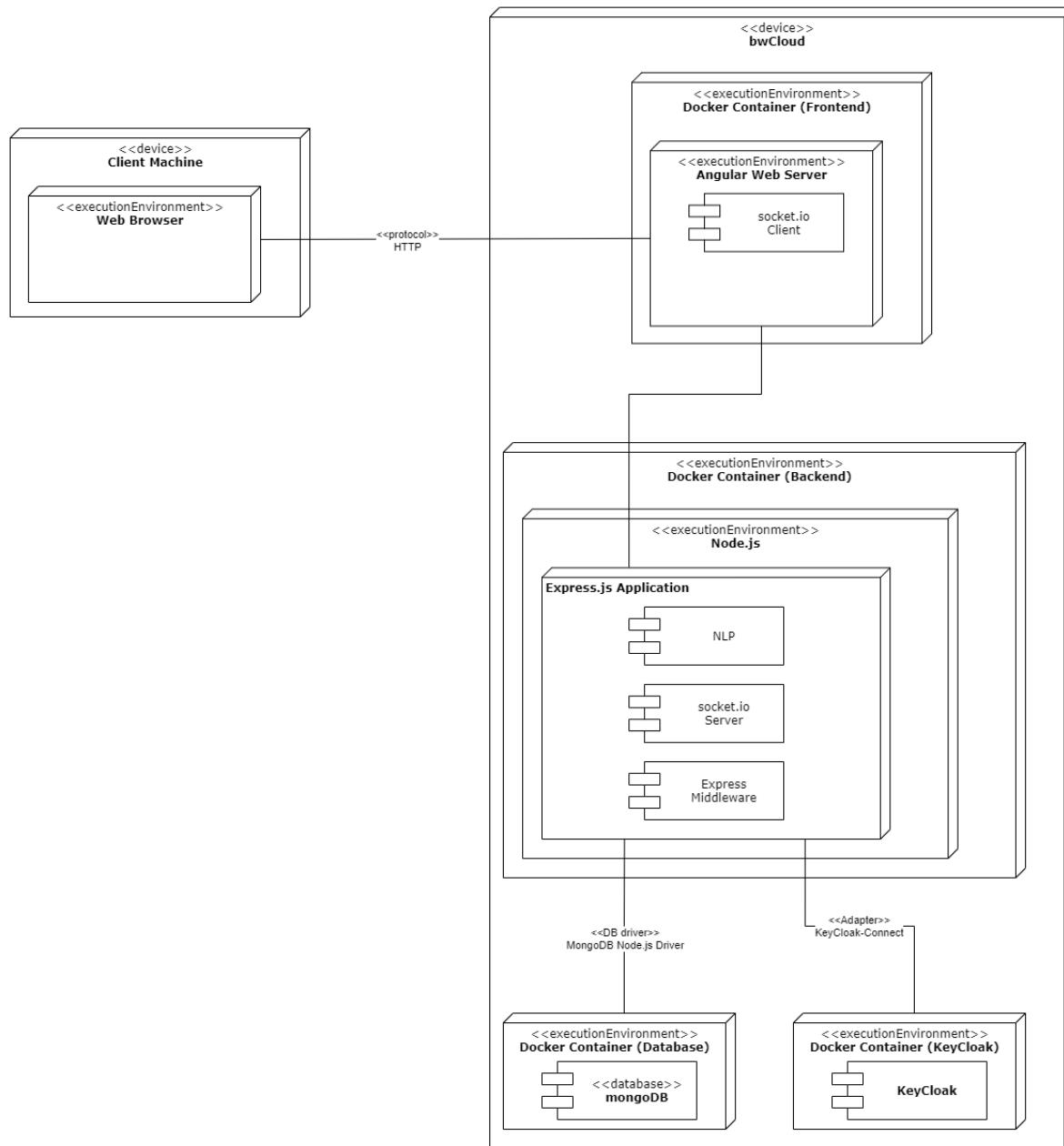


Abbildung 4: UML Verteilungsdiagramm

In unserem Verteilungsdiagramm kann man sehen, wie die verschiedenen Technologien verschachtelt und wie sie miteinander verbunden sind. Außerdem kann man sehen, was wir in unseren Dockercontainer hineinlegen.

8.5 Datenbank

In folgendem soll Aufschluss über die Struktur der Datenbank des ChatBots gegeben werden. Die Datenbank wird in einem NoSql Datenbank mongoDb gehalten.

8.5.1 Datenhaltung

Die Daten, die in der mongoDb Datenbank gespeichert sind sollen über ein Webinterface verändert werden können. Wenn der Chatbot gestartet wird lädt der Chatbot den Korpus aus der mongoDb. Sollten die Daten der Datenbank verändert worden sein, dann muss der Chatbot den Korpus erneut laden. Damit die Veränderungen der Datenbank auch beim Chatbot geändert werden.

8.5.2 ER Diagramme

Hier werden die ER Diagramme für die Datenbank aufgeführt. Zusätzlich wird erklärt welche Daten bei den Entities enthalten sind.

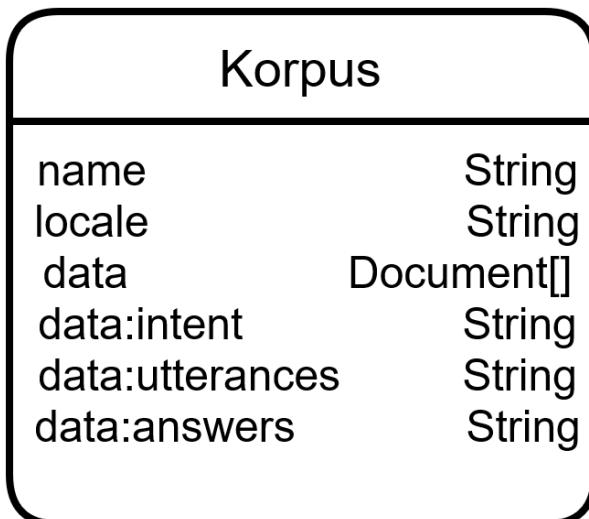


Abbildung 5: ER Diagramm Korpus

Der Korpus besteht aus name, locale und data. Der "name" ist der Name des Korpus. Das "locale" gibt an in welcher Sprache der Korpus verfasst wurde. Das "data" ist gefüllt mit Intents, die benötigt werden um einschätzen zu können in welchem Kontext der ChatBot antworten soll. Der Intent besteht aus Utterances (Äußerung/ Frage des Nutzers) und Antworten. Die Utterances sind die möglichen Fragen des Nutzers und die Antworten sind die möglichen Antwortmöglichkeiten des Chatbots. Ein Korpus hat sehr viele Intents, die den Wissenschatz des ChatBots abbilden.

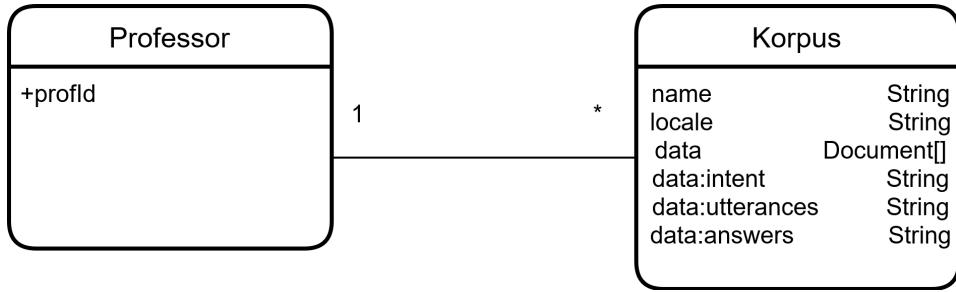


Abbildung 6: ER Diagramm Professor

In der Abbildung 6 soll dargestellt werden, dass ein Professor auf mehrere Korpusse zugreifen kann. Der Professor wird mit einer "profID" ausgestattet, damit man bestimmen kann welche Korpusse ihm zugeordnet sind.

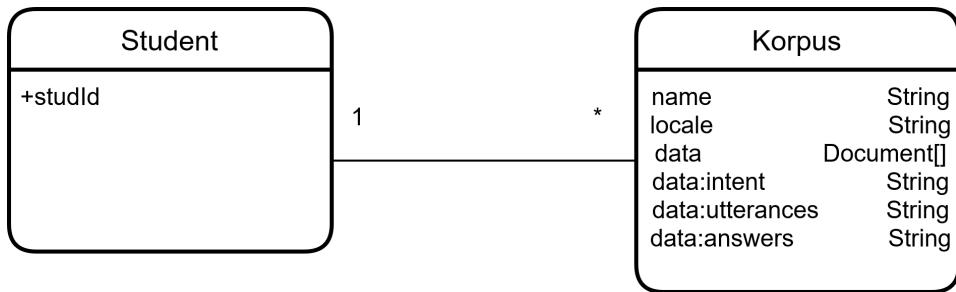


Abbildung 7: ER Diagramm Student

In der Abbildung 7 soll dargestellt werden, dass ein Student auf mehrere Korpusse zugreifen kann. Der Nutzer bekommt eine "studId" damit man bestimmen kann, welche Korpusse f�r den Studenten bereitgestellt werden m[us]sen.

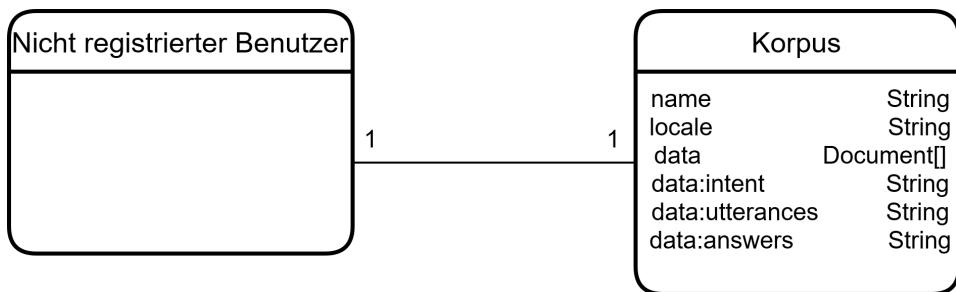


Abbildung 8: ER Diagramm Unregistrierter Nutzer

In der Abbildung 8 soll dargestellt werden, dass ein nicht registrierter Benutzer nur Zugriff auf einen Korpus hat. F[ur] unregistrierte Benutzer soll nur der Korpus "Allgemein" zur Verf[ug]ung stehen.

9 Meilensteine

In der nachfolgenden Tabelle 4 sind alle einzelnen Meilensteine aufgelistet. Jeder Meilenstein wird noch etwas genauer weiter unten erklärt.

Recherche	13.10.21
Zwischenpräsentation	22.10.21
Implementation	25.10.21
MVP	02.12.21
Endpräsentation und Enddokumentation	14.01.22

Tabelle 4: Meilenstein Liste

9.1 Recherche 13.10.21

Während der Recherche haben wir alle Informationen gesammelt, die wir für das Projekt benötigen. Die Themen Node.js, Keycloak, Angular, Socket.io und NLP waren vorrangige Themen, um Risiken zu minimieren. Deswegen haben wir uns in dieser Phase möglichst ausführlich informiert und Bücher, Webadressen und weitere Materialien besorgt. Außerdem haben wir nach Möglichkeit alle Unklarheiten geklärt.

9.2 Zwischenpräsentation 22.10.21

Wir haben eine Woche früher (12.10.21) angefangen alle relevanten Themen zu sammeln, um Materialien für die Präsentation zu haben. Damit die Präsentation sehr interessant für alle Teilnehmer ist, haben wir die wichtigsten Themen optisch ansehbar gestaltet. In unserer Zeitplanung ist auch die praktische Übung der Folien im Team eingeplant.

9.3 Implementation 25.10.21

In der Implementierung wollen wir die recherchierten Materialien umsetzen und praktische Erfahrung sammeln. Während wir versuchen alle relevanten Informationen in einen MVP umzusetzen. Zusätzlich wird in dieser Phase ein Teil der Recherche in das L^AT_EX-Format übertragen.

9.4 MVP 02.12.21

Der MVP ist unser angestrebtes Ziel. Damit wir ein Produkt zum Präsentieren haben. Während wir unser angesammeltes Wissen in die Praxis umsetzen versuchen wir frühzeitig ein funktionierendes Produkt mit den Mindestanforderungen umzusetzen. Wir hatten vorerst geplant unseren MVP zum 03.12.21 zu liefern. Interessanterweise wurde später der Termin des MVP vom Professor auf den 02.12.21 gelegt. Wodurch wir unseren geplanten Zeitraum weiterhin nachverfolgen können und den zeitlichen Rahmen minimal korrigieren müssen. Demnach haben wir relativ gut eingeschätzt bis wann der MVP fertig sein sollte.

9.5 Endpräsentation und Enddokumentation 14.01.22

Für die Endpräsentation ist eine Woche früher (07.01.22) der Beginn der Erstellung der Präsentation eingeplant. Ziel ist hierbei, dass ein Prototyp mit interessanten Features und Funktionen vorgestellt werden kann. Für die Enddokumentation werden wir ab dem Start der Implementation anfangen, alle wichtigen Informationen zu dokumentieren. Sehr wichtig ist hierbei für unser Team, dass wir alle kontinuierlich wichtige Themen zu unserem Projekt dokumentieren. Für unsere Dokumentation wählen wir das L^AT_EX-Format, da es uns hilft besser kooperativ über Gitlab zu arbeiten.

10 Zeitmanagement

In diesem Abschnitt wollen unsere Pläne zum Zeitmanagement des Projektes vorstellen.

10.1 Gantt-Diagramm

Im nachfolgenden Gantt-Diagramm sieht man, welche Tasks wir eingeplant haben. Wann wir mit diesen beginnen wollen und wann wir wird mit ihnen fertig sein wollen.

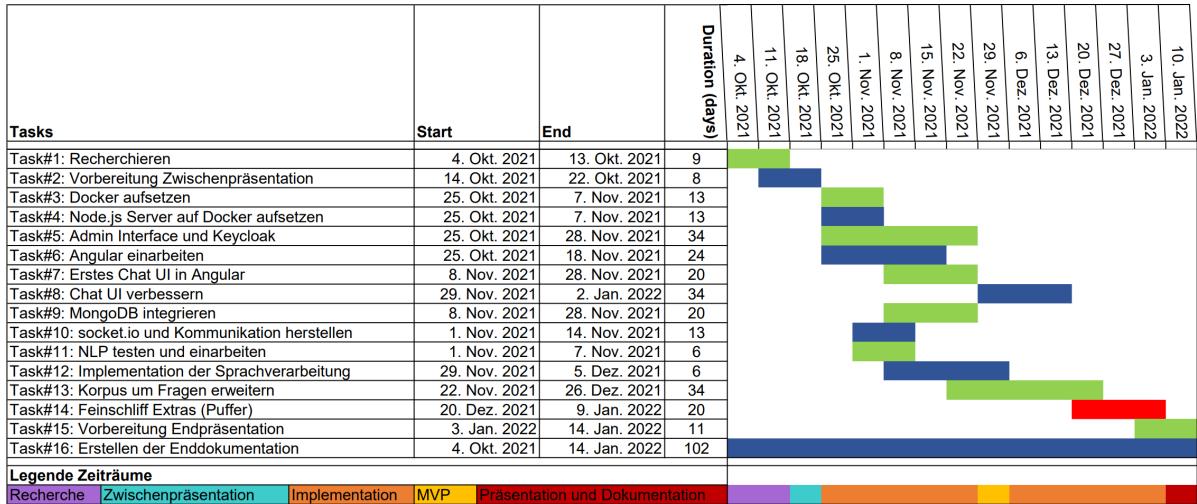


Abbildung 9: Gantt-Diagramm

In der Darstellung der Balken stellt eine Spalte, eine Woche dar, beginnend mit jedem Montag. Am unteren Ende des Diagramms sieht man, wie wir unsere Milestones eingeplant haben.

11 Risikoanalyse

Auf den folgenden Seiten haben wir in einer Tabelle möglicher Risiken für unser Projekt aufgelistet. Sowie Maßnahmen wie wir diese Risiken verringern möchten.

Risiko	Eintrittswahrscheinlichkeit	Auswirkung	Maßnahme
Team schafft es nicht schnell genug Typescript zu lernen	niedrig	hoch	Alle Teammitglieder beginnen frühzeitig sich mit Typescript zu beschäftigen
Team schafft es nicht Keycloak zu integrieren	niedrig	hoch	Alle Teammitglieder schauen sich rechtzeitig die Einführungsvideos von Herr Rößler zu Keycloak an
Das Team hat Schwierigkeiten das Userinterface in Angular zu entwickeln	niedrig	hoch	Das Team greift auf gut bewährte Designs zurück
Das Team hat Schwierigkeiten eine Verbindung mit socket.io herzustellen	niedrig	hoch	Das Team informiert sich rechtzeitig auf der Socket.io Website, wie eine Verbindung aufgebaut wird
Das Admin Interface lässt sich nicht flexibel genug anpassen	niedrig	niedrig	Das Team informiert sich rechtzeitig welche Möglichkeiten es in das UI einbauen möchte, um Flexibilität zu garantieren
Der ChatBot antwortet stark verzögert	niedrig	mittel	Das Team muss mit einplanen, dass der ChatBot in kleine saubere Module aufgeteilt wird

Tabelle 5: Risikoanalyse Tabelle Teil 1

Risiko	Eintritts-wahrscheinlichkeit	Auswirkung	Maßnahme
Hohe Latenz durch alle Komponenten	unwahr-scheinlich	mittel	Das Team muss den ChatBot testen, um z.B. Endlosschleifen zu verhindern
Der ChatBot hat Schwierigkeiten Sätze zu verstehen	mittel	mittel	Das Team muss bei einem Regex Ansatz mehrere Regex Befehle vordefinieren, um ein großes Spektrum abzudecken
Die Hardware des Kunden ist nicht kompatibel mit der Software	niedrig	hoch	Das Team muss frühzeitig mit dem Kunden klären für welche Hardware der ChatBot entwickelt werden soll
Das Team scheitert einen MVP zu entwickeln	niedrig	hoch	Das Team muss sehr früh mit der Implementierung beginnen und ausführlich genug recherchieren
Das Team scheitert rechtzeitig genug alle Technologien zu lernen	niedrig	hoch	Das Team recherchiert frühzeitig und versucht für alle möglichen Probleme Lösungen zu finden

Tabelle 6: Risikoanalyse Tabelle Teil 2

Fazit:

Bei Beginn unseres Projektes haben wir mögliche Risiken aufgelistet, die in unserem Projekt auftauchen könnten und wie wir diese nach Möglichkeit verhindern möchten. Wir haben dabei herausgefunden, dass wir sehr viele Risiken haben, die von niedriger Eintrittswahrscheinlichkeit sind. Dennoch sind die Mehrheit der Risiken von hoher Auswirkung im Projekt. Als Beispiel, das Team schafft es nicht schnell genug TypeScript zu lernen. Die wichtigsten Risiken für uns waren die oben genannten Risiken zu Angular, Keycloak und Socket.io. Wenn eines dieser Risiken von unserem Team nicht ausgleichbar wäre, dann würden wir gar nicht die Möglichkeit haben einen MVP oder später ein fertiges funktionierendes Produkt abzuliefern. Mit den Tabellen 5 und 6 möchten wir für unsere Gruppe festhalten, welche Gedanken wir uns über die möglichen Risiken in unserem Projekt gemacht haben. Damit wir besser und effizienter unser Projekt vorantreiben können und vorbereitet sind auf mögliche Schwierigkeiten.

12 Appendix

Hier wird der zusätzliche Content abgebildet.

12.1 Ältere Versionen des Komponentendiagramms

Hier werden die älteren Versionen und Entwürfe des Komponentendiagramms abgelegt.

12.1.1 Komponentendiagramme

Im folgenden sind die älteren Versionen unserer Komponentendiagramme

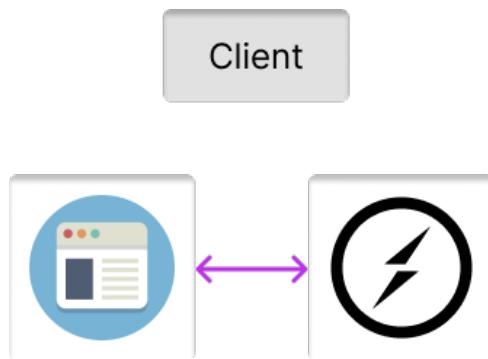


Abbildung 10: Komponentendiagramm Client

Hier wird die Client Seite bildlich dargestellt. Man sieht, dass der Webbrowseer durch die Socket.io Client deployed wird und dadurch wird dann eine Beziehung zur Serverseite aufgebaut. (Siehe Abbildung 10)

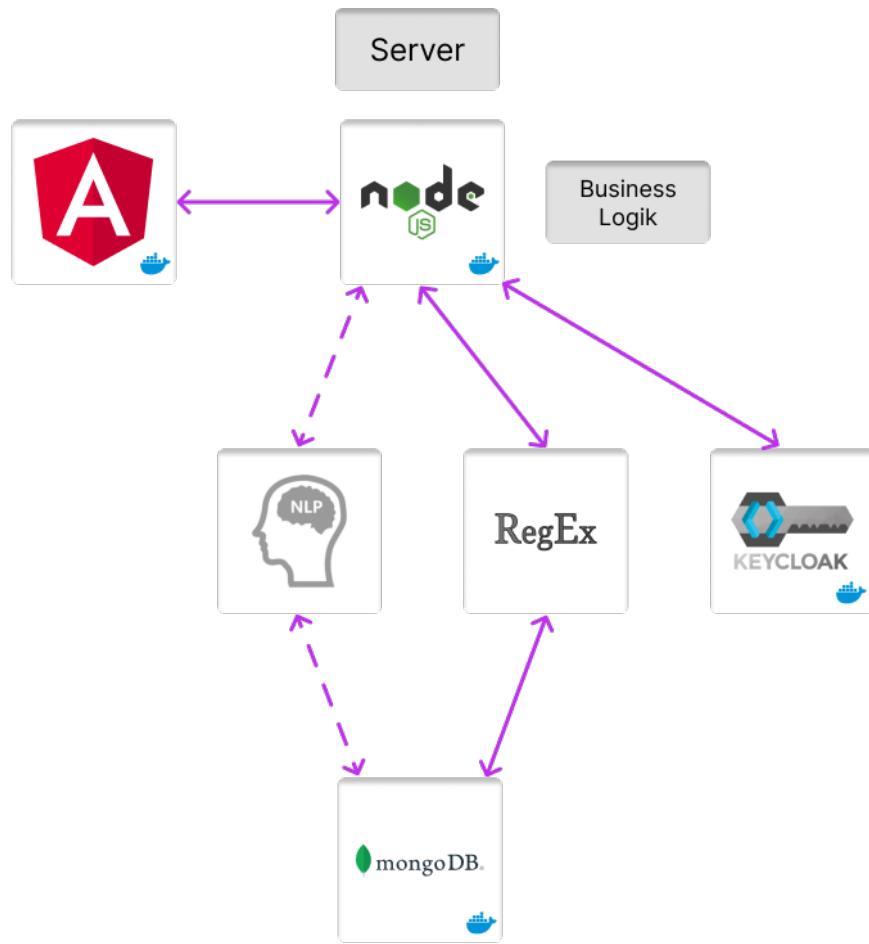


Abbildung 11: Komponentendiagramm Server

In der Serverseite wird dann durch den Socket.io Server die Verbindung zum Client aufrecht gehalten. Im Server befindet sich Angular, node.js, KeyCloak und die Datenbank mongoDB und haben jeweils einen eigenen Dockercontainer. Die Erkennung von der eingegebenen Sprache möchten wir zunächst mit RegEx ermöglichen, um das Minimal Viable Product hinzubekommen. Optional dann mit NLP (Natural Language Processing) erweitern.

In dieser Version haben wir erst einmal die Struktur von unseren Komponenten gesucht und eine grobe Darstellung erstellt. Was wir hier aber nicht wussten ist, wie wir das NLP darstellen sollten. NLP war für uns vorher eine optionale Möglichkeit.

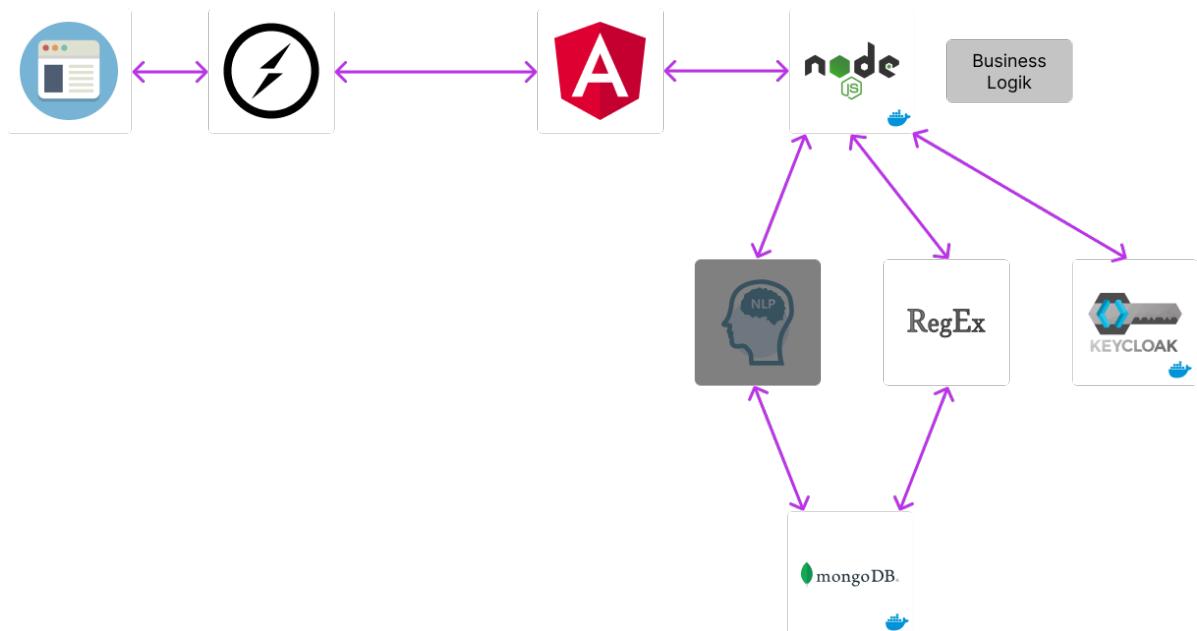


Abbildung 12: Komponentendiagramm v1.0

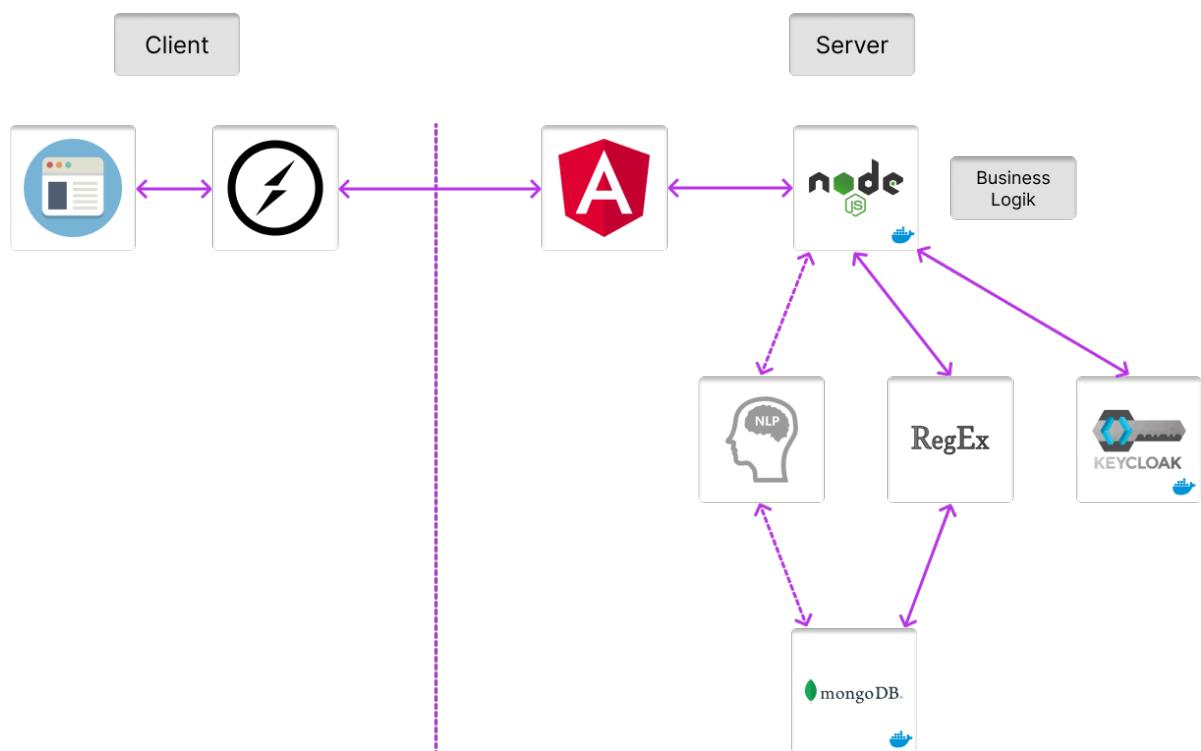


Abbildung 13: Komponentendiagramm v1.1

In dieser Darstellung haben wir die einzelnen Komponenten in Server und Client eingeteilt, um die Struktur besser zu verstehen. Wir haben aber die Abtrennung nicht

richtig anzeigen können und das NLP haben wir auch nicht klar als optional zeigen können.

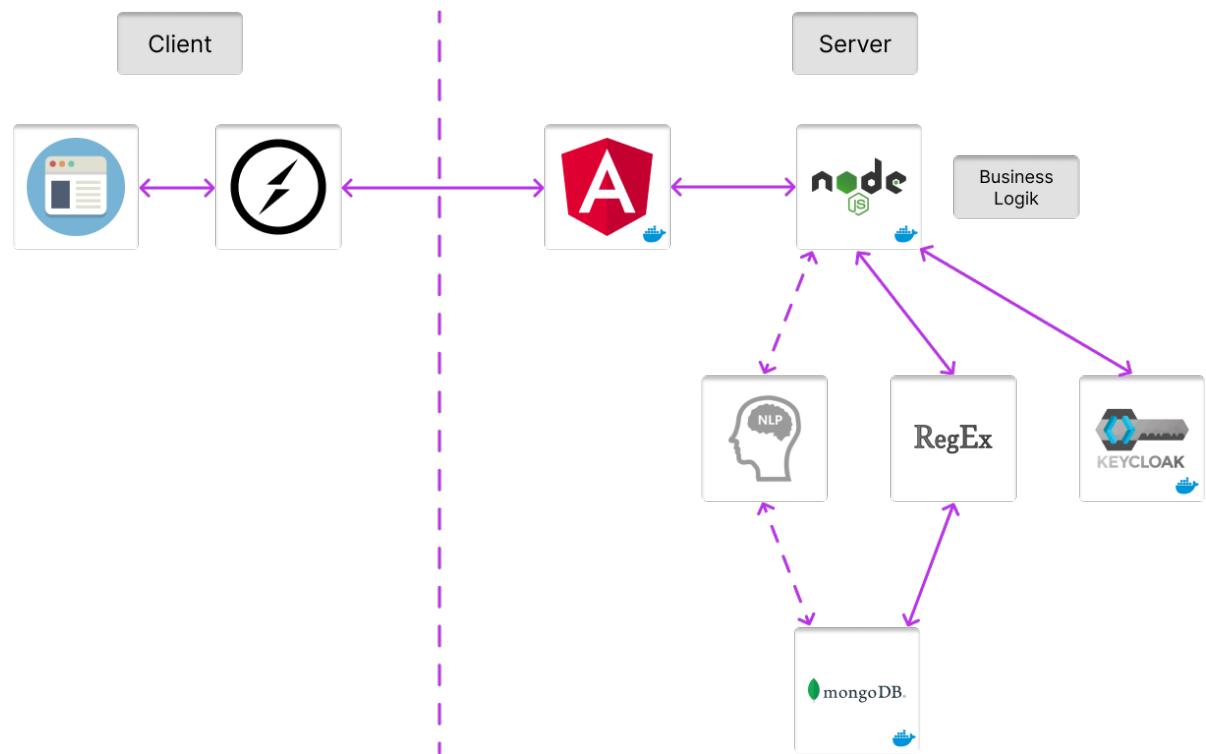


Abbildung 14: Komponentendiagramm v1.2

Literatur

Angular Icon, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://angular.io/assets/images/logos/angular/angular.svg#>.

Blackberry-messenger-live-free, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://cdn.geckoandfly.com/wp-content/uploads/2016/01/blackberry-messenger-live-free.jpg>.

Briar, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://cdn4.geckoandfly.com/wp-content/uploads/2018/08/briar.jpg>.

Chatbot-cost-calculator, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://risingmax.com/blog/ai-based-chatbot-cost-calculator/>.

Cleverbot, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://www.cleverbot.com/>.

Keycloak Icon, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://www.katacoda.com/sebastienblanc/avatar>.

MongoDB Icon, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: https://webimages.mongodb.com/_com_assets/cms/kpo5kblefbjq79065-Horizontal_Default.svg?auto=format%252Ccompress.

NLP Icon, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://ecomschool.uk/wp-content/uploads/2020/01/NLP.png>.

Node.js Icon, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://cdn.freebiesupply.com/logos/thumbs/2x/nodejs-1-logo.png>.

Regex Icon, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: https://upload.wikimedia.org/wikipedia/commons/thumb/d/d3/Toolbaricon_RegEx.svg/1280px-Toolbaricon_RegEx.svg.png.

Socket.io Icon, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://socket.io/images/logo.svg>.

Telegram Bild, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: <https://cdn2.geckoandfly.com/wp-content/uploads/2016/01/telegram-chat-secure.jpg>.

Tim WhatsApp, [o. D.] **online**[besucht am 2021-11-11]. Abger. unter: https://s3.amazonaws.com/cdn.freshdesk.com/data/helpdesk/attachments/production/42438944/original/Aw4-xHy7s_EILasM39zbaxogNI7PZdtvzw.png?1545221722.