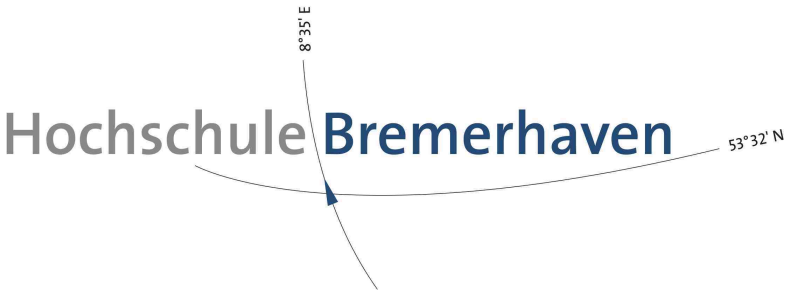
# Bachrlorprojekt: Pepper




**Eingereicht von:**

Benjamin Thomas Schwertfeger 36036

Kristian Kellermann 35751

Jacob Menge xxxxx

**Kursleitung:** Prof. Dr. Nadja Petram

Technik für Wirtschaftsinformatik

**Hochschule Bremerhaven**

# Eigenständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

Alle sinngemäß und wörtlich übernommenen Textstellen aus fremden Quellen wurden kenntlich gemacht.

Name: _____ Unterschrift: _____
Ort, Datum: _____

Name: _____ Unterschrift: _____
Ort, Datum: _____

Name: _____ Unterschrift: _____
Ort, Datum: _____

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Abstrakt

Diese Dokumentation ist im Rahmen des Bachelorprojektes von Benjamin T. Schwertfeger, Jacob Menge und Kristian Kellermann entstanden und dient neben der Dokumentation des selbst gewählten Projektes ebenfalls als Anforderung für das Bestehen des Bachelorstudiums.

Die Studenten haben sich dazu entschieden, eine Anwendung für den humanoiden Roboter Pepper, welcher sich derzeit im Besitz der Hochschule Bremerhaven befindet zu entwickeln. Hierzu werden verscheidene Anwendungsfälle implementiert, welche durch eine selbst implementierte Web-Schnittstelle in Form einer Backend Webanwendung unterstützt wird.

Ziel ist es, dem Roboter das Interagieren mit Menschen beizubringen, sodass sinnvolle Interaktionen zwischen Mensch und Maschine zustande kommen. Zudem sollen mit Hilfe der Webanwendung zusätzlich dynamisch Informationen an Pepper wetergegeben und von ihm abgerufen werden, sodass Informationen zu Interaktionen, wie Dauer, Intensität und Erfolg des Gespräches gespeichert und ausgewertet werden können.

Dieses Projekt wird von Frau Prof. Dr. Nadja Petram begleitet. Es wurdne keine Rahmenbedingungen festgelegt.

# Kapitel 1

# Dokumentation und Struktur

Diese Dokumentation ist in XXX Kapitel gegliedert, welche sich mit den einzelnen Aspekten unseres Projektes auseinandersetzen. Die Stuktur sieht wie folgt aus:

- *Kapitel 1*: Dokumentation und Struktur
- *Kapitel 2*: Motivation und Zielsetzung
- *Kapitel 3*: Erste Schritte und Installation
- *Kapitel 4*: Anwendungsfall - Hochschule
- *Kapitel 5*: Anwendungsfall - XXX
- *Kapitel 6*: Das Backend
- *Kapitel 7*: Möglichkeiten der Erweiterung
- *Kapitel 12*: Abschließende Wort

Zu sehen ist, dass wir nach der Erläuterung unserer Struktur und des Grundes für diese Arbeit von vorn bis hinten durch unser Projekt gehen. Nachdem wir unser Grundgerüst und unsere ersten Schritte einschließlich Installation dargelegt haben, werden wir zwei Anwendungsfälle zum Einsatz von Pepper Aufzeigen, sowie die von uns Implementierten Funktionen aufzeigen.

Ebenfalls werden wir die Herangehensweise, sowie die technische Umsetzung und Hürden, mit denen wir zu kämpfen gehabt haben erläutern, um alle Aspekte der Implementierung von Software für den Roboter Pepper abzudecken.

Nachfolgend haben wir die Fragestellungen festgehalten, auf welche wir in den jeweiligen Abschnitten eingehen werden.

| Kapitel | Fragestellung |
|---|---|
| 1. Dokumentation und Strktur | Wie ist diese Arbeit aufgebaut? |
| 2. Motivation und Zielsetzung | Was wollen wir hiermit erreichen? |
| 3. Erste Schritte und Installation | Was ist das Fundament unseres Softwareprojektes? |
| 4. Anwendungsfall - Hochschule | Wie kann Pepper der Hochschule helfen? |
| 5. Anwendungsfall - Hochschule | Was kann Pepper als Stadtführer? |
| 6. Das Backend | Welche Prozesse laufen im Hintergrund? |
| 7. Möglichkeiten der Erweiterung | Wie kann es weiter gehen? |
| 12. Abschließende Worte | Was haben wir aus diesem Projekt mitgenommen? |

Im Anschluss dieser Dokumenation ist das Literaturverzeichnis mit allen Referenzen zu finden.

# Kapitel 2

# Motivation und Zielsetzung

Wie alle Bachelorstudierende kommt irgendwann die Zeit, in der es darum geht, ein Projekt zu finden, mit welchem man sich über einen längeren Zeitraum beschäftigt, um dies im Rahmen des Studiums schriftlich niederzulegen. Wir drei sind von der Welt der Programmierung begeistert und haben von Frau Dr. Petram das Angebot erhalten, bei ihr das Bachelorprojekt durchzuführen. Da wir im Laufe unserers Studiums viele Kurse bei ihr besucht haben, unter anderem “KI - maschinelles Lernen“ und “Big Data“ waren wir sofort begeistert, als wir gehört haben, mit dem humanoiden Roboter Pepper arbeiten zu dürfen.

Wir haben von ehemaligen Masterstudierenden eine Vorführung einer laufenden Anwendung bekommen und haben uns natürlich direkt gefragt, was wir denn tolles entwickeln können, damit nicht nur wir, sondern auch die Hochschule den meisten Mehrwert davon erhalten.

Aufgrund dieser Fragestellung haben wir uns dazu entschieden, dass wir zwei Anwendungsfälle zum Einsatz des Peppers implementieren wollen, um diese auch mit Hinblick auf den Tag der Informatik vorstellen zu können.

Neben der Entwicklung dieser Anwendungsfälle wollen ebenfalls unser Verständnis für die Thematik KI und Big Data ausbauen, indem wir ein Backend schaffen, welches mit Hilfe von Pepper Daten sammelt, um diese in einem späteren Prozess anderweitig zu verwenden.

# Kapitel 3

# Erste Schritte: Das Grundgerüst

## 3.1 Allgemein

Wir haben bisher noch kaum Berührung mit Robotern im Alltag erlebt und daher ist unsere Erfahrung im Bereicht der Roboterprogrammierung sehr begrenzt. Zum Glück haben uns ehemalige Masterstudierende einen schnellen Einstieg ermöglicht, denn diese habe eine Anwendung für das Unternehmen “Erlebnis Bremerhaven“ entwickelt und uns deren Quellcode zur Verfügung gestellt.

Wir haben sehr schnell gemerkt, dass uns dies zwar eine gute Hilfe für den Anfang ist, jedoch gibt es mittlerweile deutlich effizientere Methoden einzelne Komponenten miteinander zu verknüpfen.

## 3.2 Wer ist dieser Pepper?

## 3.3 Projektmanagement

Um dieses Projekt erfolgreich durchzuführen und zielorientiert zur arbeiten, treffen wir uns wöchentlich in der Hochschule und Besprechen unsere neuesten Ideen und Implementierungen. Auch zwischendurch stehen wir im Kontakt, um Probleme und Schwierigkeiten schnell zu beheben.

Damit wir gemeinsam an einer Codebasis arbeiten können, haben wir zu Anfang unseres Projektes die Organisation HBV-Pepper auf GitHub angelegt. Jegliche Beteiligung, sowie verschiedene Versionen können dort eingesehen werden. Zum Ende des Projektes haben sich hier 4 Repositories von uns angesammelt, welche folgende Themen beinhalten:

- Pepper Anwendung
- Backend Serveranwendung
- Dokumentation

- Sonstiges

Das erste Repo beinhaltet unsere lauffähige Anwendung, welche die von uns implementierten Anwendungsfälle, sowie sonstige Funktionalitäten für den Roboter Pepper beinhaltet.

Der Backend Serveranwendung auf welchem wir in Kapitel ¡HIER LINK SETZEN¿ eingehen, ist unsere Schnittstelle, mit welcher wir Informationen, die Pepper durch seine Interaktionen mit der Umgebung sammelt, abfangen und speichern. Zudem Versorgen wir Pepper mit Informationen, Beispiel hierfür sind Informationen zum Stunden- und Mensaplan.

Das Repositorie zur Dokumenation beinhaltet alle zu diesem Dokument gehörenden Dateien. Da wir diese mit LaTeX anfertigen, lohnt es sich auch dies zu versionieren.

Für kleine Skripte, Tests und andere zu keinem festen Thema dazugehörigen Dokumente haben wir das Repositorie “Sonstiges“ angelegt.

Wir werden in diesem Dokument neben der Implementierung auch auf das Herunterladen, Installieren und Einrichten unserer Software eingehen, damit Studierende, die dieses Projekt weiterführen wollen, die Möglichkeit haben, mit einem etwas fortgeschrittenem Projekt zu starten. Darüber hinaus befindet sich in jedem Stammverzeichnis eine Readme, welche Anforderungen und erste Schritte aufzeigen, um einen schnellen Einstieg zu ermöglichen.

## 3.4 Persona

Damit wir Pepper mit allen möglichen Antworten auf alle möglichen Fragen vorbereiten können, müssen wir uns zuerst überlegen, welche Personengruppe mit Pepper sprechen wird. Auf dieser Basis können wir verschiedene Gesprächsthemen entwickeln, die für die Personengruppe interessant sein könnten.

Da wir Pepper als Hochschul Assistenzen einsetzten wollen, haben wir es mit folgenden Personas zu tun:

- Student
- Dozent
- Angestellter der Hochschule
- Besucher (Zukünftige Studenten, Interessenten)

Hauptsächlich soll Pepper für Studenten und insbesondere für Erstsemestler eingesetzt werden, die zum Beispiel nicht wissen, wo sich ein bestimmter Raum oder ein Gebäude befindet oder wann und wo die nächste Vorlesung stattfindet. Hierbei soll es auch die Möglichkeit geben, den kompletten Stundenplan für jeden Studiengang und jedes (zur

Zeit verfügbare) Semester anzeigen zu lassen, um auf jede Situation eine Antwort parat zu haben.

Ein weiterer Faktor ist der Altersintervall mit dem Pepper zu tun haben wird. Ältere Studenten, die sich auch in einem höheren Semester befinden, sind Themen wie Raumfindung oder Stundenplan eher uninteressant. Dafür könnten Themen wie der Mensaplan oder Informationen über die Hochschule und über Pepper selber interessant sein. Außerdem wird Pepper in der Lage sein, mit dem User ein kurzes Smalltalk Gespräch zu führen. Somit ist Pepper auch für alle Altersgruppen interessant. Diese Themen könnten für alle weitere Personas ebenfalls interessant sein. Vor allem Personen, die technikaffin sind oder das Thema Roboter einfach spannend finden, werden Pepper ebenfalls verwenden, weswegen wir für diese Personengruppe passende Gesprächsthemen und lustige Animationen vorbereiten.

Für Dozenten und Angestellte der Hochschule sind Themen wie der Stundenplan und die Raumfindung eher uninteressant und der Mensaplan wieder interessanter. Diese beiden Personas werden Pepper sehr wahrscheinlich am wenigsten verwenden.

Für Besucher sind Themen wie Informationen über Studiengänge und allgemein zur Hochschule eher interessanter als alle anderen Themen.

Um genau sagen zu können, welche Personengruppe sich mit Pepper unterhalten und über welche Themen sie sprechen, müssen wir Pepper zuerst sehr oft in der Praxis einsetzten und verschiedene Daten sammeln. Dazu werden wir im Punkt “Datensammlung” näher eingehen.

## 3.5 Datensammlung

Sobald alle Anwendungsfälle in Pepper integriert sind und er für den Praktischen Nutzen einsatzbereit ist, haben wir uns überlegt, verschiedene Daten zu sammeln um eine Art Feedback zu erhalten und diese graphisch darzustellen.

Folgende Daten sollen gesammelt werden:

- Distanz
- Alter
- Geschlecht
- Emotion/Stimmung
- Mimic
- Dialog Zeit
- ...

Diese Daten sollen während des Gesprächs zwischen Pepper und dem User gesammelt werden und zu unserem Node Server gesendet werden. Auf dem Node Server werden diese anschließend weiter verarbeitet und vorerst in einer MySQL Datenbank abgespeichert.

Um die Daten sammeln zu können, bietet Softbanks praktischerweiße verschiedene KI orientierte Funktionen an, wie zum Beispiel die Gesichts,- oder Alterserkennung.

Diese Daten sollen dafür eingesetzt werden, um Pepper in Zukunft weiter zu optimieren und folgende Fragen zu beantworten:

- War das Gespräch erfolgreich und konnte Pepper helfen?
- Hatte der User freude bei der Benutzung des Roboters?
- Hatte der User Schwierigkeiten, sich mit dem Pepper zu unterhalten oder verlief alles reibungslos?
- Welche Personengruppe hat mit Pepper am meisten gesprochen?
- Wie lange verläuft ein Gespräch im Durchschnitt?
- Gab es Fragen, die nicht beantwortet werden konnten?
- Welcher Anwendungsfall wurde wie oft verwendet?
- Gab es Systemfehler?
- Was ist die optimale Ansprech Distanz zwischen User und Roboter?
- Welches Semester und/oder Studiengang ist der User

# Kapitel 4

# Einrichtung des Laptops

## 4.1 Technische Details

- Intel Core i9-11900K Prozessor (bis zu 5,3 GHz), Octa-Core
- 43,9 cm (17,3") Full HD 16:9 Display (entspiegelt), Webcam
- 32 GB RAM, 1.500 GB SSD, Fingerprint
- 1,5TB SSD
- NVIDIA GeForce RTX3080 Grafik (16384 MB), HDMI, Thunderbolt 4, MiniDP
- Windows 10 Professional 64 Bit, 4,6 kg

Link: https://www.cyberport.de/?DEEP=1C10-3HP&APID=276&gclid=CjwKCAjwhaaKBhBcEiwA8acsHI
gHRYDqBoC2FoQAvD_BwE

## 4.2 Allgemein

Zu Beginn des Projekts haben wir einen XMG Ultra 17,3 Laptop erhalten, den wir beliebig verwenden durften. Wir haben uns Gedanken darüber gemacht, wie wir den Laptop am Sinnvollsten verwenden können und welches Betriebssystem wir verwenden.

Wir haben uns dazu entschieden, den Laptop als Speicherort für die Daten zu benutzten, die wir mit Pepper generieren und speichern wollen. Außerdem soll der Laptop als gemeinsames System zur Kommunikation und Datentransfer dienen. Dafür war die Überlegung, ein Nextcloud Server zu implementieren und Open SSH zu verwenden um auch von außerhalb Zugriff zu gewährleisten. Des Weiteren hatten wir auch die Idee, den Laptop als Workstation zu benutzten, um gemeinsam in Android Studio zu arbeiten oder weitere Applikationen wie Python Anaconda zu nutzen.

Die Wahl des Betriebssystems fiel uns schwer, da jeder seine eigenen Vorlieben besaß. Am Ende konnten wir uns allerdings eindeutig entscheiden. Für den Nextcloud Server, sowie das Speichern der Daten in eine Datenbank haben wir Ubuntu Server verwendet.

Hier war auch die Überlegung die Desktop version zu verwenden, allerding bräuchten wir diesen für unser Vorhaben nicht unbedingt und zur Not hätte man sich den Desktop später immer noch einrichten können. Der Vorteil an Ubuntu Server ist, dass man von Null anfängt und keine unnötigen Applikationen bereits Installiert sind, so wie es mit der Ubuntu Desktop Version der Fall ist.

Da wir Ubuntu Server nicht als Betriebssystem für die Workstation verwenden konnten haben wir uns entschieden eine zweite Partition zu erstellen und Windows 10 einzurichten.

**Einrichtung der Betriebssysteme** ...

muss das wirklich sein? - das hier ist doch keine Bedienungsanleitung

# Kapitel 5

# Pepper App; Anwendungsfall - Hochschule

## 5.1 Die Vorüberlegungen

Nachdem wir uns darauf geeinigt haben, mindestens einen Anwendungsfall für den Roboter Pepper zu entwickeln, und sich dieser auf den Alltag an der Hochschule Bremerhaven beziehen soll, haben wir uns damit konfrontiert gesehen, uns Gedanken darüber zu machen, wie genau eine Interaktion zwischen einem Studenten oder einer Lehrkraft und dem Roboter stattfinden kann. Aufgrund der den aktuellen Beschlüssen der Regierung ist das Leben auf dem Campus fast zum Stillstand gekommen, wodurch sich für Pepper wenige Einsatzgebiete ergeben. Wir gehen jedoch davon aus, zu einer neuen Normalität zurück zu gekangen, und werden Pepper dahingehend vorbereiten.

Folgende Fragen sind für uns zentral:

- Wie sieht der Alltag auf dem Campus aus?
- Wann soll Pepper mit wem interagieren?
- Welche Funktionalitäten wollen wir implementieren?
- Wo setzen wir welche Werkzeuge ein?
- Wo fangen wir an?

Wir sind dann ziemlich schnell darauf gekommen, dass wir eine klare Struktur benötigen, um unsere Aufgaben und Ziele klar zu definieren. Daraufhin ist folgende Skizze entstanden, welche grob die Hauptfunktionalitäten unseres Projektes festhält:

**HIER WIRKLICH EINE SKIZZE ANFERTIGEN**

In Abb. ist zu sehen, dass wir folgende Grundfunktionalitäten definiert haben:

- Stundenplan
- Raum- und Lageplan
- Speisekarte der Mensa
- Smalltalk

Zudem soll die Interaktion mit Pepper für alle möglich sein. Studierende und Lehrende sollen erfragen können, wo welche Vorlesung stattfindet, in der Kaffeteria oder Mensa soll sich Pepper mit Gästen unterhalten, Smalltalk führen oder Empfehlungen für bestimmte Speisen ausgeben. Darüber hinaus wollen wir Pepper das lückenlose Interagieren beibringen, damit er an besonderen Veranstaltungen, wie dem Tag der offenen Tür oder dem Tag der Informatik, Interessenten Informationen über die Hochschule, das Leben auf dem Campus, sowie allgemeine Empfehlungen für die Stadt Bremerhaven ausgeben kann.

## 5.2 Hard- und Softwarespezifikationen

Da Peppers Tablet auf Android basiert, liegt es auf der Hand, Android Studio als Entwicklungsumgebung zu benutzen. Der Einfachheit halber, ist nachfolgend eine Tabelle, welche die von uns für die Pepper Anwendung benutzte Software mit den entsprechenden Versionsnummern auflsitet.

Tabelle 5.1: Hard- und Softwarespezifikationen

| Software / Tool | Version / Spezifikation | Beschreibung |
|---|---|---|
| Android Studio | Arctic Fox 2020.3.1 Patch 4 | IDE der IntelliJ Plattform |
| Gradle | 7.0.3 | Build Tool |
| Android SDK | 12 | Android Framework |
| Pepper API | v1.9 API 23 | Entwicklungstools für Pepper mit Emulator, Deploy und Debug Funktion |
| Android API SDK | 31 | Framework zum Verbinden und Installieren von Software auf Geräten mit Android OS |
| Java SDK | 1.8 | Basis der Programmiersprache Java und dessen Grundfunktionalitäten |

Mit abwichenden Versionen kann es zu Kompatibilitätsproblemen kommen.

Der Roboter Pepper selbst, ist XXX groß, wurde am XXX von XXX gekauft und ist seit XXX im Besitz der Hochschule und läuft derzeit mit der Version XXX.

## 5.3 Implementierung

# Kapitel 6

# Node - Express Webanwendung

## 6.1 Allgemein

Wir als Studenten haben durch das Studium, aber auch durch Nebenjobs und private Projekte Erfahrungen mit verschiedenen Programmierkonzepten sammeln können. Da wir bei der Entwicklung der Pepper Anwendung jedoch nur zwischen den Sprachen Kotlin und Java entscheiden konnten, haben wir zu Beginn unseres Projektes, das Gefühl gehabt, in unserer Entwicklung, was die Nutzung und Implementierung verschiedener Methoden angeht, sehr eingeschränkt zu sein.

Daher haben wir im Verlauf unseres Bachelorprojektes immer wieder gemerkt, dass mit dem Pepper allein nicht viel anzufangen ist, wenn es um Flexibilität in der Entwicklung geht. Zugriff auf Berührungssensoren, sowie die Kamera und die Mikrophone sind nicht direkt möglich, sondern nur über vordefinierte Funktionen der Aldebaran Bibliotheken. Somit haben wir keine Möglichkeit, selbst Gesichter zu erkennen oder Sprache zu analysieren, geschweige denn mehr als nur mit Pepper reden zu können.

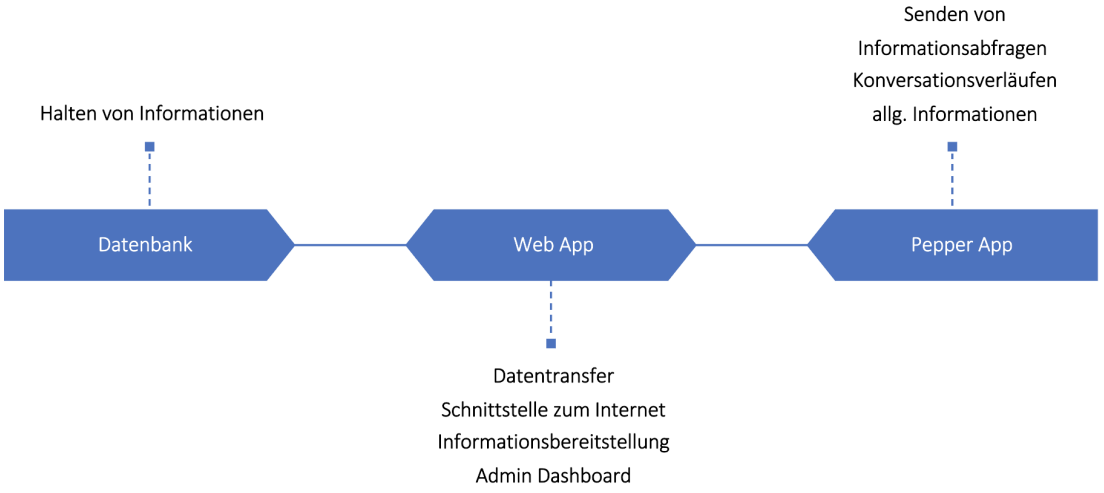



Abbildung 6.1: Zusammenhang: Pepper App und Backend

Bei der Implementierung der Stundenpläne etwa, werden verschiedene Anfragen an die Hochschulserver geschickt und ausgewertet, jedoch ist auch dies in Java nicht wirklich entwicklerfreundlich. Daher sind wir auf die Idee gekommen, einen Webserver aufzusetzen, den Pepper kontaktieren kann, um weiterführende Informationen zu erhalten.

Genau hier war es uns von Vorteil, schon Erfahrungen mit Node, Datenbanken und API's gesammelt zu haben. Daher ist es ein leichtes Unterfangen gewesen, einen Express Webserver mit Hilfe der Laufzeitumgebung Node aufzusetzen, auf welchem wir vielfältige Möglichkeiten zur Implementierung von komplexen Vorgängen haben.

So hat es sich ergeben, dass wir nicht nur eine Anwendung für Pepper entwickeln wollen, sondern einen zweiten Schwerpunkt festlegen, welcher sich auf das Sammeln und Auswerten von Daten konzentriert. Diese Daten sollen natürlich von Pepper über unsere Anwendung gesammelt werden.

Das Repository dieser Webanwendung ist öffentlich und unter folgender Adresse erreichbar:

https://github.com/ProjectPepperHSB/NodeJS_Server4Pepper

## 6.2 Überblick: Webanwendung

Bevor wir im Folgenden auf die Implementierung und Spezifikationen unserer Webanwendung eingehen, wollen wir hier kurz die Hauptfunktionalitäten, welche wir in drei Kategorien aufgeteilt haben, aufzeigen.

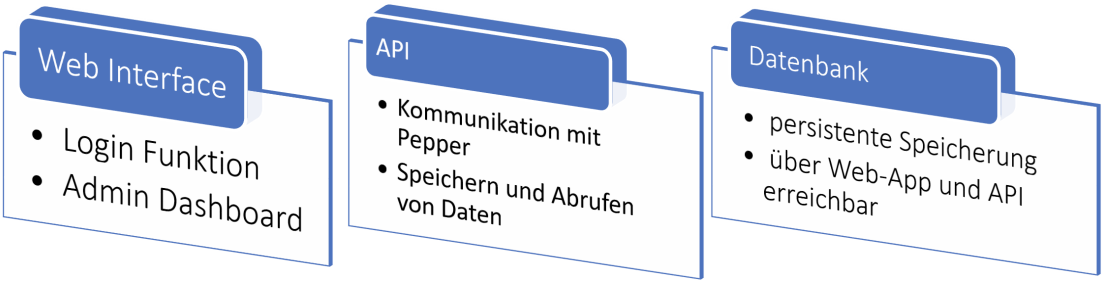



Abbildung 6.2: Komponenten der Webanwendung

Diese Webanwendung, sowie die dazugehörige Datebank laufen derzeit auf dem Hochschulserver Hopper in einem abgeschirmten Docker Kontainer, welcher einem Benutzer mit Namen "hbv-kms" gehört. Dies ist ein Benutzer, auf welchen wir gemeinsam als Team zugreifen können. Da die Anwendung über den HTTP Port des Dockers läuft, findet sich der Pfad "docker-hbv-kms-http" in allen Endpunkten dieser Webanwendung wieder.

Sie ist jedoch auch auf fast jedem lokalen Rechner ausführbar, sofern eine funktionsfähige MySQL Instanz aktiv ist. (vgl. Abschnitt 6.4)

### 6.2.1 Web Interface

Das Web Interface dient dazu, den Entwicklern und Betreibern der Webanwendung Informationen über die gesammelten Daten aufzuzeigen. Da wir es für sinnvoll halten, nicht jedem den kompletten Zugriff auf die gesammelten Daten zu gewähren, haben wir uns dazu entschieden, nur einen Benutzer in unserer Webanwendung anzulegen, welcher Zugriff auf das Admin Dashboard hat. Somit sparen wir uns die Implementierungen zur Registration und Verwaltung von Nutzern. (weiteres in Abschnitt 6.4.

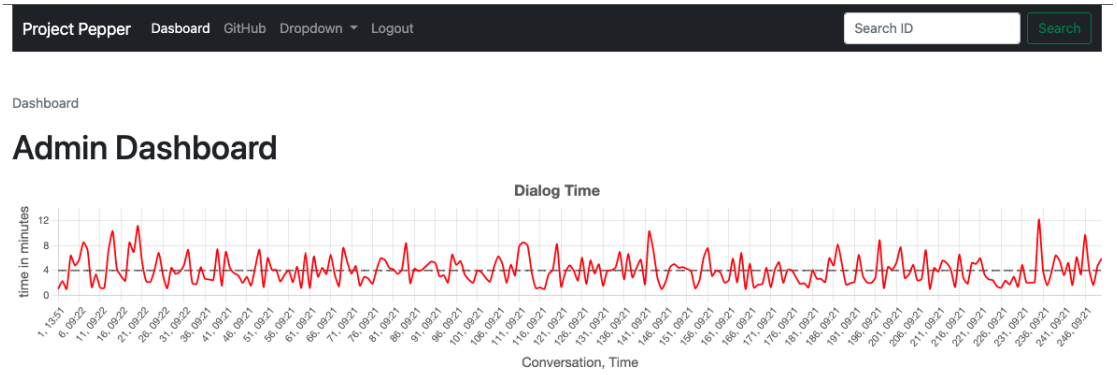
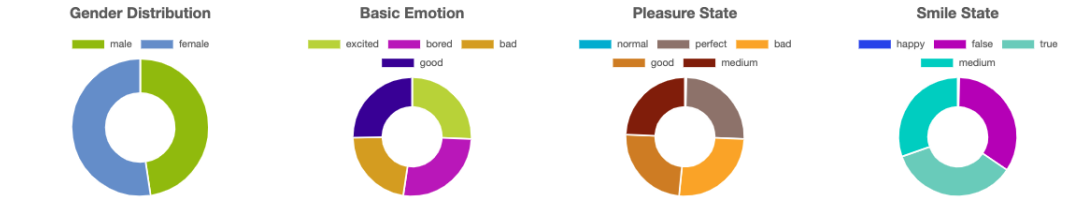



Abbildung 6.3: Webansicht des Admin Dashboards Teil 1

(Abb. 6.3) Sofern sich ein Administrator angemeldet hat, gelangt er in die Ansicht des Admin Dashboards. Dieses besteht aus mehreren Komponenten. Zu sehen ist, dass sich am oberen Bereich der Abbildung eine Menüleiste befindet. Diese Beinhaltet eine Verlinkung zu unseren öffentlichen GitHub Repositories, sowie weitere Dropdowns, welche bisher noch keine Funktionen besitzen. Dies ist je nach Anwendungsfall und Einsatzgebiet individuell erweiterbar. Die dargestellte Suchleiste funktioniert jedoch. Uber diese können nach speziellen Identifikationsnummern einzelner Konversationen und Datenreihen gesucht werden. (vgl. Abschnitt 6.3.1.3 f.)

Des Weiteren ist ein Liniendiagramm abgebildet. Dies zeigt die Dauer der letzten 250 Konversationen, welche von Pepper geführt und anschließend über unserer Webanwendung in die Datenbank gespeichert wurden. Dieses Diagramm ist jedoch nicht aussagekräftig, da wir zur Füllung unserer Datenbank, Skripte zur randomisierten Generierung von Konversationen erstellt und ausgeführt haben. Mehr dazu in **HIER VERLINKUNG ZUM SKRIPTE SUBSECTION**.

Wir liefern immer nur die letzen 250 Konversationen aus, da Pepper, falls er denn mal zum Einsatz kommt, gar nicht so sehr viele Konversationen führen wird, da diese auch eine gewisse Zeit in Anspruch nehmen. Somit ist es nur sinnvoll, sich die nur letzen 250 anzeigen zu lassen, um einen aktuellen Überblick über die vergangenen Interaktionen zu erhalten. (Über die Suchleiste kann dennoch jede Konversation gefunden werden.)




| # | Distance | Age | Gender | Basic Emotion | Pleasure State | Excitement State | Smile State | Dialog Time | TimeStamp |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.823 meters | 24 | male | excited | normal | medium | happy | 1m 1s | 2022-01-28T13:51:42.000Z |
| 2 | 0.5363 meters | 66 | male | bored | perfect | excited | false | 2m 17s | 2022-01-25T09:22:07.000Z |

Abbildung 6.4: Webansicht des Admin Dashboards Teil 2

Zusätzlich zur Darstellung der Konversationsverläufe haben wir uns entschieden, weitere interessante Informationen zu den zur Grunde liegenden Gesprächen visuell hervorzuheben. Somit ist es möglich, sich einen schnellen Überblick über die letzen Interaktionen mit Pepper zu gelangen.

(Abb. 6.4) Sprechen mehr männliche oder weibliche Personen mit Pepper? Wie ist die Stimmung unserer Akteure und wie verhalten sie sich? Wird gelacht oder sind sie gelangweilt? Diese Informationen können anhand der Donut-Diagrammen abgelesen und interpretiert werden. So hat man beispielsweise Auskunft darüber, ob ein neues Update bei den Kunden und Anwendern gut ankommt, oder ob man nicht doch noch etwas ausbessern sollte. Zudem ist auf unseren Anwendungsfall Hochschule bezogen, auch nachvollziehbar, wie die allgemeine Stimmung der Studenten und Interessierten ist.

Unterhalb den Diagrammen begint eine Tabelle, welche die ausgewerteten Datenreihen beinhaltet. Jede dieser Reihen führt per Klick auf eine Detailansicht der jeweiligen Konversation und bietet somit einen genaueren Einblick in das geführte Gespärch.

Dashboard / View a8d22954abe14f33970142baae657056

**Conversation: a8d22954abe14f33970142baae657056**

Fri Jan 28 2022 14:51:42 GMT+0100 (Central European Standard Time)

**General Data**

| Distance | Age | Gender | Basic Emotion | Pleasure State | Excitement State | Smile State | Dialog Time |
|---|---|---|---|---|---|---|---|
| 0.823 meters | 24 | male | excited | normal | medium | happy | 1m 1s |

**Use-Case History**

| # | Use Case | Timestamp |
|---|---|---|
| 1 | Small Talk | Fri Jan 28 2022 14:48:43 GMT+0100 (Central European Standard Time) |
| 2 | Mensa | Fri Jan 28 2022 14:49:01 GMT+0100 (Central European Standard Time) |
| 3 | RouteFinder | Fri Jan 28 2022 14:50:00 GMT+0100 (Central European Standard Time) |

**Not understand phrases**

ieso

Abbildung 6.5: Ansicht einer bestimmten Konversation

(Abb. 6.5) Jeder Konversation, die Pepper führt, wird eine Identifikationsnummer zugewisen. Dies ermöglicht die dynamische Zuordnung von Inhalten zu einem bestimmten Gespräch. Somit ist eine neben der Einsicht in die allgemeinen Informationen wie dem Alter und Geschlecht des Gespärchspartners möglich, auch Informationen über verwendete Anwendungsfälle und deren chronologischer Reihenfolge zu erhalten. Auch Phrasen, welche Pepper nicht verstanden hat, sei es aufgrund von Akzenten oder falscher Aussprache, werden der Konversation zugewiesen.

Dies alles ermöglicht es uns und dem Anwender der Software, mehr über seine Kunden und Anwender zu erfahren - oder auf die Hochschule bezogen: Die Stimmung der Studenten, sowie des Personals und Interessierten kann somit erfasst werden.

### 6.2.2 API

Die Kommunikation mit unserer Webanwendung verläuft nicht nur über das Web Interface, sondern über verschiedene Endpunkte, auch Routes genannt. Über diese können je nach Grad der Authentifizierung verschiedene Inhalte gesendet und empfangen werden.

Um hier nicht zu viele Informationen aus den Bereichen der Endpunkte (6.3) und Implementierung (6.4) vorweg zu nehmen, wollen wir es bei der Erwähnug und Beschreibung der wichtigsten Endpunkte belassen.

- /docker-hbv-kms-http/collector
- /docker-hbv-kms-http/getData
- /docker-hbv-kms-http/api/v1

Der Endpunkt “collector” dient der Speicherung verschiedener Informationen, welche Pepper an unserern Server über die von ihm geführten Konversation sammelt (vgl. Abschnitt 6.3.2.1). Dieser ist sowohl als GET, als auch über eine POST Anfrage erreichbar und nimmt verschiedene Parameter entgegen. Bisher ist dieser Endpunk ohne Authentifizierung, das heißt, jeder der weiß, in welcher Form die Daten übermittelt werden, kann Informationen in unsere Tabellen speichern.

Wir haben überlegt dies zu Umgehen, indem wir Pepper ein Token generieren lassen, welches nötig ist, um die Daten zu speichern, jedoch würde dies das Problem nur verlagern, da unser gesamter Quellcode öffentlich einsehbar ist. Dies kann nicht über Umgebungsvariablen umgangen werden, da dies in der Android Studio Umgebung zur Entwicklung von Android Anwendungen nicht so zur Verfügung steht, wie wir es benötigen.

Der Endpunkt “getData” wird von dem Web Interface verwendet, um die Daten, welche sich in unserer Datenbank befindet abzurufen. Dieser setzt voraus, dass der Benutzer als Admin angemeldet ist und ein gültiges JSON Web Token in seinem Cookie besitzt. (vgl. Abschnitt 6.3.1.4)

Der letzte Endpunkt in der Liste “api/v1’ ist die Schnittstelle über welche man extern auf Daten in der Datenbank zugreifen kann. Hier ist es ebenfalls erforderlich, authentifiziert zu sein. Dafür muss ein Auth Token bei jeder Abfrage übermittelt werden, welches mit dem in unserer Vebanwendung hinterlegten verglichen wird. Hierbei haben wir nich das Problem des öffentlichen Einsehens in das Token, da wir dieses nicht über GitHub speichern. (vgl. Abschnitt 6.3.2.2)

Es gibt natürlich noch viele weitere Endpunkte in unserer Webanwendung, wie zum Beispiel den zur Abfrage von Stunden- oder Mensaplänen. Diese bekommen auch Parameter übergeben, woraufhin diese mit Hilfe verschiedener Methoden die gewünschten Informationen bereitstellen. (vgl. Abschnitt 6.3)

### 6.2.3 Datenbank

Um die Informationen, welche unser Pepper während seiner Gespräche sammelt auch persistent speichern zu könenn, haben wir uns dazu entschieden, eine MySQL Datenbank zu integrieren. Zunächst haben wir es als fortschrittlicher gehalten, MongoDB zu verwenden, da dies eine JSON-Objekt basierte Speicherung von Daten ermöglicht und durch seine skalierbare und einfache Integration in Expres und JavaScript vielfältige Möglichkeiten bietet.

MongoDB ist jedoch nicht auf dem Hochschulserver installiert und da wir durch verschiedene Kurse schon sehr gute Kenntnisse und Erfahrungen mit MySQL gesammelt haben, war es dann doch nicht so schlimm, auf MongoDB zu verzichten.

Die von der Webanwendung verwendete Datenbank läuft in dem zuvor erwähnten Docker Kontainers des geteilten Benutzers.

Weitere Informationen und Spezifikationen sind der Tabelle im Abschnitt 6.4.3 zu entnehmen.

## 6.3 Endpunkte und Responses der Webanwendung

Unsere Webanwendung ist über viele Endpunkte ansprechbar. Nachfolgend sind diese in zwei Gruppen Aufgeteilt. Endpunkte, welche für das Admin Dashboard benötigt werden, sowie als Resultat eine HTML Seite rendern, sind unter dem Abschnitt 6.3.1 zu finden. Alle weiteren Endpunkte, welche zumeist nur Daten entgegen nehmen oder Informationen in Form von JSON zurück geben, sind in Abschnitt 6.3.2 aufgelistet.

### 6.3.1 Web Interface Endpunkte / Routes

#### 6.3.1.1 Startseite

**Beschreibung:** Liefert die Startseite der Webanwendung wieder. Auf dieser ist nur der Titel der Anwendung, sowie einen Button, der zur Login Seite führt. (siehe 6.3.1.5)

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/

**Parameter:** keine

#### 6.3.1.2 Dashboard

**Beschreibung:** Liefert einem angemeldetem Admin die Ansicht des Dashboards. (vgl. Abb. 6.3 und 6.4)

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/dashboard

**Parameter:** keine

#### 6.3.1.3 Dashboard: Detailansicht

**Beschreibung:** Dient der visuellen Übermittlung der Detailansicht einer Konversation. (vgl. Abb. 6.5) Dies ist nur über den Browser erreichbar, sofern ein gültiges JSON Web Token im Cookie hinterlegt ist.

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/dashboard/view

**Parameter:**

| Param | Typ | Beschreibung |
|---|---|---|
| conversation_id | String | ID der zu abzurufenden Konversation |

#### 6.3.1.4 Dashboard: Datenabfrage

**Beschreibung:** Endpunkt, welcher das JSON Web Token überprüft und bei Erfolg den übermittelten Query-String ausführt. Die Antwort wird als JSON Objekt zurückgegeben. Diese Methode kann zum Beispiel im Admin Dashboard nach Eingabe eines Query Strings in der Suchleiste abgerufen werden.

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/dashboard/getData

**Parameter:**

| Param | Typ | Beschreibung |
|---|---|---|
| query_string | String | MySQL Query String |
| n | String | [optional] Anzahl der abzurufenden Zeilen |

#### 6.3.1.5 Login

**Beschreibung:** Liefert bei einer GET Anfrage die Login Seite aus. Der POST Endpunkt wird nach Abschicken des Login-Forumulars angesprochen, validiert den Login und leitet den Client bei Erfolg an das Dashboard weiter. Gleichzeitig wird das JSON Web Token im Cookie des Clients hinterlegt.

**Methode:** GET, POST

**Endpunkt:** /docker-hbv-kms-http/dashboard/login

**Parameter GET:** keine

**Parameter POST:**

| Param | Typ | Beschreibung |
|---|---|---|
| username_input | String | Benutzername |
| password_input | String | Passwort |

#### 6.3.1.6 Logout

**Beschreibung:** Löscht das JSON Web Token aus dem Cookie des Benutzers und meldet ihn somit ab. Anschließend wird nach /docker-hbv-kms-http/ weiter geleitet. (vgl. Abschnitt 6.3.1.1)

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/dashboard/logout

**Parameter:** keine

### 6.3.2 API Endpunkte / Routes

#### 6.3.2.1 Datenspeicherung

**Beschreibung:** Dieser Endpunkt kann über GET und POST angesprochen werden und dient der Speicherung von Daten. Diese Daten werden von Pepper oder externen Skripten an die Webanwendung gesendet, welche anschließend die Speicherung realisiert.

**Methode:** GET, POST

**Endpunkt:** /docker-hbv-kms-http/dashboard/collector

Der Parameter "subject" gibt an, welche Art von Daten gespeichert werden sollen. Folgende Strings sind valide Wete für den Parameter "subject" bei einer GET Anfrage:

- emotion_data
- use_case_data
- did_not_understand_data

Der GET Endpunkt nimmt nur Key-Value Paare entgegen, hingegen nimmt der POST Endpunkt auch JSON Objekte an.

**Parameter für GET mit subject = emotion_data:**

| Param | Typ | Beschreibung |
|---|---|---|
| subject | String | Art des zu speichernden Sachverhaltes |
| identifier | String | Identifikationsnummer der Konversation |
| distance | String / Float | [optional] Distanz zwischen Pepper und dem Sprecher |
| age | String / Float | [optional] Alter des Sprechers |
| gender | String | [optional] Geschlecht des Sprechers |
| basic_emotion | String | [optional] Generelle Stimmung des Sprechers |
| pleasure_state | String | [optional] Motivation des Sprechers |
| excitement_state | String | [optional] Begeisterung des Sprechers |
| smile_state | String | [optional] Einschätzung der Glücklichkeit des Sprechers |
| dialog_time | String / Float | [optional] Zeit des Gespräches |

**Parameter für GET mit subject = use_case_data:**

| Param | Typ | Beschreibung |
|---|---|---|
| subject | String | Art des zu speichernden Sachverhaltes |
| identifier | String | Identifikationsnummer der Konversation |
| use_case | String | Name des Use-Cases |

**Parameter für GET mit subject = did_not_understand_data:**

| Param | Typ | Beschreibung |
|---|---|---|
| subject | String | Art des zu speichernden Sachverhaltes |
| identifier | String | Identifikationsnummer der Konversation |
| phrase | String | Nicht verstandene Phrase |

Für eine POST Anfrage ist nur "conversation_data" ein gültiger Wert für den Parameter "subject".

**Parameter für POST mit subject = conversation_data:**

| Param | Typ | Beschreibung |
|---|---|---|
| subject | String | Art des zu speichernden Sachverhaltes |
| identifier | String | Identifikationsnummer der Konversation |
| data | String / JSON | variable Daten der Konversation |

#### 6.3.2.2 Abfrage von Daten aus der DB mittels SQL Query String

**Beschreibung:** Endpunkt zur Interaktion mit der Datenbank. Ein API-Token wird benötigt. SQL Befehle können übermittelt werden. Aus sicherheitsgründen werden Anfragen, welche folgende Wörter beinhalen, abgelehnt: "drop", "delete", "show", "users", "insert", "into", "create". Bei erfolgreicher Abfrage wird das Ergebnis des SQL-Statements als JSON zurückgeliefert.

**Methode:** POST

**Endpunkt:** /docker-hbv-kms-http/api/v1

**Parameter:**

| Param | Typ | Beschreibung |
|---|---|---|
| auth_key | String | API-Token |
| subject | String | Art der Abfrage ("test" oder "sql_query") |
| sql_query | String | Auszuführender SQL Befehl |

#### 6.3.2.3 Abfrage des Stundenplans

**Beschreibung:** Dient der Abfrage von Stundenplänen eines bestimmten Studiengangs. Bei erfolgreicher Abfrage wird ein JSON Objekt, oder einen HTML String zurückgegeben.

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/timetable

**Parameter:**

| Param | Typ | Beschreibung |
|---|---|---|
| course | String | Studiengang (Kürzel wie: "WI", "BWL" etc.) |
| kw | String / Int | [optional] Kalenderwoche |
| semester | String / Int | [optional] Semester |
| htmlOnly | Boolean | [optional] HTML oder JSON Response (Default: false) |

#### 6.3.2.4 Abfrage des Mensaplans als JSON

**Beschreibung:** Liefert den Mensaplan für die aktuelle Woche als JSON Objekt zurück.
**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/timetable

**Parameter:** keine

#### 6.3.2.5 Abfrage des Mensaplans als Bild

**Beschreibung:** Liefert den Mensaplan fur die aktuelle Woche als Bild zurück.

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/timetable/img

**Parameter:** keine

#### 6.3.2.6 File Server

**Beschreibung:** Dieser Endpunkt wird verwendet, Skripte und Stylesheets dynamisch aus dem "static" Verzeichnis abzurufen.

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/fileserver

**Parameter:**

| Param | Typ | Beschreibung |
|---|---|---|
| name | String | Name der angeforderten Datei |

#### 6.3.2.7 Abfrage des Kurspreises einer Kryptowährung

**Beschreibung:** Dieser Endpunkt kann einem den Kurspreis und weitere relevante Informationen zu einem Kryptowährungs-Handelspaar zurückliefern.

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/crypto

**Parameter:**

| Param | Typ | Beschreibung |
|---|---|---|
| subject | String | Anforderung (nur "price" ist implementiert) |
| ksymbolw | String | Symbol (Bsp.: "BTC-USDT") |

#### 6.3.2.8 Abfrage der IP-Adresse des Clients

**Beschreibung:** Liefert die IP Adresse des Clients zurück.

**Methode:** GET

**Endpunkt:** /docker-hbv-kms-http/ip

**Parameter:** keine

### 6.3.3 Responses und Status Codes

Alle An- und Abfragen an unsere Webanwendung liefern verschiedene Antworten / Responses zurück. In der nachfolgenden Tabelle sind die Geläufigsten aufgelistet. Da gewisse Error-Handling Mechanismen automatisch eine entsprecheende Antwort liefern, kann es zu abweichenden Beschreibungen und weiteren Status Codes kommen.

| Status Code | Bedeutung |
|---|---|
| 200 | OK |
| 401 | ungültige Parameter |
| 404 | Datei oder Endpunkt nicht gefunden |
| 500 | Interner Server Error |

## 6.4 Implementierung

Bei unserer Web App handelt es sich um eine Anwendung, welche mit dem Framework Express für die Laufzeitumgebung Node entwickelt wurde. Express zeichnet sich durch die simple Konfiguration, sowie einfache Implementierung verschiedenster Funktionalitäten aus. Express wird meist für Backendanwendungen verwendet, wir benutzen es jedoch auch, um serverseitiges Rendering zu realisieren. Dies bedeutet, dass Ansichten (Views), welche an den Client ausgeliefert werden, zuvor auf dem Server gerendert worden sind.

Von außen ist unsere Anwendung über die URL https://informatik.hs-bremerhaven.de/docker-hbv-kms-http erreichbar. Sollte sie lokal ausgeführt werden, muss die Domain auf Localhost geändert werden - hier ist dann auch die Angabe des Ports nötig. (mehr dazu in Abschnitt 6.4.2)

Unsere Anwendung besitzt jedoch nicht all zu viele Views, bis auf das zuvor gezeigte Admin Dashboard (vgl. Abschnitt 6.3.1.2), sowie die Detailansicht der einzelnen Daten der Konversationen (vgl. Abschnitt 6.3.1.3). Darüber hinaus gibt es nur die Startseite (vgl. ABschnitt 6.3.1.1) und die Login-View. Diese Login-View ist über den Endpunkt "/docker-hbv-kms-http/login" (vgl. Abschnitt 6.3.1.5) erreichbar und beinhaltet nur ein Formular zur Eingabe des Benutzernamens, sowie des dazugehörigen Passwortes.

Abbildung 6.6 zeigt den Inhalt des Root-Verzeichnisses unserer Webanwendung. Im Folgenden wird auf die einzelnen Inhalte eingegangen.

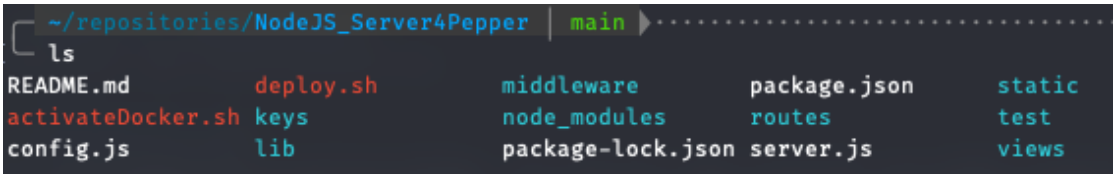



Abbildung 6.6: Hierarchie des Stammverzeichnisses der Webanwendung

### server.js

Der Einstiegspunkt unserer Webanwendung ist das Skript "server.js". Dieses beinhaltet die Einbindung aller benötigten Dateien und Skripte, sowie die Anwendung der in den Konfigurationsdateien festgelegten Einstellungen (mehr dazu in Abschnitt 6.4.2).

### node_modules, package-lock.json und package.json

Da wir Node als Laufzeitumgebung gewählt haben, ist es uns möglich den Node Package Manager (npm) zur Installation verschiedener Module anzuwenden. Alle installierten Module sind in dem Verzeichnis "node_modules" zu finden. Deren Spezifikationen sind in den Konfigurationsdateien "package.json", sowie "package-lock.json" festgehalten. package-lock.json ist eine automatisch generierte Datei, welche Node verwendet um Dependencies und Versionen festzuhalten. Aufgrund dieser Konfigurationsdateien ist es

nicht nötig, alle im Verzeichnis "node_modules' befindlichen Erweiterungen in das jeweilige GitHub Repository zu laden, denn diese können, nach einer neuen Initialisierung über den Befehl "npm init" anhand der Definitionen in package-lock.json, automatisch nachinstalliert werden.

**views**

In diesem Verzeichnis befinden sich Templates der Seiten, welche vom Server gerendert und an den Client ausgeliefert werden.

**routes**

Der Ordner "routes" beinhaltet die verschiedene Skripte, welche die unterschiedlichen Endpunkte unserer Webanwendung definieren. Innerhalb dieser Skritpe werden auch die Ansichten aus dem "views"-Verzeichnis gerendert und an den Client geschickt. Hier sind alle in Abschnitt 6.3 behandelten Endpunkte definiert.

**static**

Hier finden sich statische Dateien, hierzu gehören Bilder und unabhängige Skripte, sowie Stylesheets.

**middleware**

Bei Middlewares handelt es sich um Skripte, welche Abläufe beinhalten, die zwischen anderen Prozessen statt finden. Hierzu gehört bei uns die Authentifizierung von Nutzern, indem das sich dort befindliche Skript "auth.js" eingebunden wird, während ein Nutzer einen beschränkten Endpunkt anspricht. Dies sorgt dafür, dass nur ausgewählte Benutzer bestimmte Endpunkte erfolreich ansprechen können. Alle anderen bekommen die Mitteilung, dass sie nicht die nötigen Berechtigunen besitzen.

**lib**

Im Ordner lib (aka Libraries) befinden sich alle von uns für die Webanwendung verwendeten Skripte, für welche wir keine genaue Unterkategorie finden konnten. Hierbei handelt es sich um Skripte, welche Anfragen (Requests) synchron und asynchron durchführen können, sowie Funktionen zur Generierung und Validierung von Passwörtern und deren Hash.

**keys**

Da wir beschränkte Endpunkte haben und unsere Authentifizierung mittels JSON Web Token im Cookie des Clients realisiert ist, muss unsere Anwendung dieses Token auch auf Richtigkeit überprüfen können. Daher haben wir uns dazu entschieden ein Schlüsselpaar zu generieren, welches für die Ver-und Entschlüsselung der JSON Web Token verwendet wird.

#### Sonstige Dateien

Im Hauptverzeichnis befinden sich weitere Dateien, wie die README.md, welche eine Quick-Start Anleitung bietet, die in dem Repository der Webanwendung dargestellt ist.

Des weiteren befindet sich ein Skript zum Aktivieren des Docker-Kontainers und ein Deploy-Skript, welches für die Aktivierung des Dockers, das Übertragen der Daten in das entsprechende Verzeichnis, sowie für den Start der Webanwendung sorgt.

Im Repository ist auch die Datei ".example.env" zu finden, welche ein Beispiel dafür bietet, wie die ".env" Datei aussehen muss. Diese .env Datei legt Umgebungsvariablen fest, sowie beinhaltet es sensible Daten, wie die Informationen zum Login in die Datenbank, sowie den API-Key zur Nutzung der API über externe Programme / Skripte (vgl. Abschnitt 6.3.2.2).

### 6.4.1 Error-Handling

### 6.4.2 Konfiguration

### 6.4.3 Versionen und Spezifikationen

MySQL: Ver 15.1 Distrib 10.6.5-MariaDB, for debian-linux-gnu

## 6.5 Installation

erst dies, dann dass und ananas ggf. muss das Deploy Skript angepasst werden

## 6.6 Möglichkeiten der Erweiterung

# Kapitel 7

# Skripte und Erweiterungen

# Kapitel 8

# Big Data und Ausblick

Durch die Anbindung von Pepper an unsere Webanwendung ist es möglich, sämtliche Informationen, während einer Konversation zu speichern und nachzuvollziehen. Hiermit können Unternehmen herusfinden, was ihre Kunden bewegt und in welchen Bereichen man noch an seinem Geschäftsmodell arbeiten muss. Es wäre Denkbar, mehrere Pepper an verschiedene Unternehmen zu vermieten, die den selben Datensammlungsablauf haben, womit man als Vermiter ein großes Kontingent an Informationen zu verschiedenen Branchen sammeln und auswerten kann, welche man dann an die Firmen und Unternehmen übermitteln kann.

# Kapitel 9

# Abschließende Worte

Wir haben in diesem Projekt sehr viele neue Methoden und Möglichkeiten zur Entwicklung verteilter Anwendungen kennen gelernt. Der Entwurf und die Implementierung einer solch umfangreichen Webseite bzw. Webanwendung hat uns sehr gut gefallen, da wir hier im Vergleich zu vielen anderen Kursen unseres Studiums, Methoden und Werkzeuge aus der Vorlesung direkt anwenden und ausbauen konnten.

Auch die Arbeit im Team war sehr angenehm. Wir haben zum Teil täglich miteinander den Diskurs gesucht, neue Ideen ausgetauscht miteinander gearbeitet. Hierbei konnte jeder etwas von dem Anderen lernen, sodass nach einem Treffen alle etwas schlauer und mit einem guten Gefühl den Jitsi Raum verlassen konnten.

Wir nehmen aus diesem Projekt viele neue Eindrücke und vor Allem neues Wissen mit. Webentwicklung war für uns drei vor diesem Kurs, ein bisher ≫nur≪ interessnates Thema, von dem wir mal gehört haben, mit dem wir aber keine praktischen Erfahrungen teilen konnten. Die intensive Auseinandersetzung mit JavaScript und php hat bisher jedoch bei jedem von uns gefehlt, daher sind wir sehr froh auch dies endlich nachgeholt zu haben.

# Literaturverzeichnis

[1] Leonhard Thomas Schwertfeger. Bild: Mygain - trading logo, Juli 2021.

[2] Investopia Tim Smith. Brokers. https://www.investopedia.com/terms/b/broker.asp, letzter Zugriff am 05. Juli 2021.