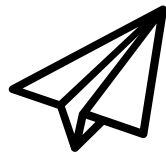


Essen, den 30. April 2021

Schriftliche Dokumentation des Messengers „Disputatio“

GFOS Innovationsaward 2021
Programmierung eines Browser-basierten Messengers



Von
Lukas Baginski
Simon Engel
& Jan Thomas

Gymnasium Essen Werden

INHALT	
EINLEITUNG.....	2
Vorwort der Autoren:	2
Aufgabenstellung.....	2
Umfang der Dokumentation	2
Modalitäten	3
PLANUNGEN DES PROJEKTS	3
Idee	3
Umsetzung und Strukturierung	3
DURCHFÜHRUNG.....	4
Design	4
IMPLEMENTIERUNG.....	6
Datenbank	6
Backend	7
Frontend	14
VORSTELLUNG DER APP.....	19
Anleitung.....	19
ZUSAMMENFASSUNG/FAZIT	39
BEMERKUNGEN	39
ANHANG	40

Einleitung

Vorwort der Autoren:

Liebe Leser*innen,

das Dokument, welches Sie gerade in den Händen halten, ist die Dokumentation zu unserem Ergebnis des GFOS Innovationsawards 2021. Diese Dokumentation beschreibt die Arbeitsweise des von uns entwickelten Messenger-Dienstes. Vorneweg möchten wir zweierlei loswerden: Dieses Projekt ist zwar von Lukas Baginski, Simon Engel und Jan Thomas angefertigt worden, dennoch wäre es mit nur diesen drei Kräften an der Tastatur nicht möglich gewesen. Unser Informatik-Lehrer, Herr OstR Michael Albrecht, hat uns bei diesem Projekt- wie bei vielen anderen auch- wieder tatkräftig unterstützt. Mit viel Geduld und Kompetenz begleitete er uns bei den einen oder anderen Probleme und war stets bemüht, neben der Funktionalität auch für Verständnis zu sorgen. An ihn richtet sich ein großer Dank.

Des Weiteren ist uns wichtig zu erwähnen, dass uns die Bearbeitung der Aufgabe großen Spaß bereitet hat. Messenger werden von den meisten Menschen Tag aus, Tag ein benutzt, ohne größer über die Funktionalität nachzudenken. Ähnlich wie beim Strom hat man den Eindruck, die Nachrichten kämen einfach und überlegt sich nicht, welche Systeme dahinterstecken und welchen Aufwand die Entwicklung einer solchen Software beansprucht.

Gerade aber für uns als junge Menschen und Schüler, die seit klein auf mit digitalen Medien und Messengern aufgewachsen sind, ist es eine interessante Aufgabe, ein solches System selber zu entwerfen. In der Zeit, in der wir die Aufgabe bearbeitet haben, lernten wir somit einiges über modernen Webdesign (für die Oberfläche oder das Frontend), verarbeitende Strukturen und Webserver, wie sie teilweise im Frontend beziehungsweise im Server oder Backend zum Einsatz kommen, und über die Verwaltung einer Datenbank, die zur Sicherung von Nutzerdaten, Einstellungen und Nachrichten notwendig wurde. Somit hat sich der diesjährige GFOS-Innovationsaward schon bis zum jetzigen Zeitpunkt für uns als Gruppe mehr als bezahlt gemacht.

Aufgabenstellung

Die Aufgabenstellung des diesjährigen GFOS-Innovationsawards stellte die Implementierung eines Browser-basierten Messenger-Dienstes dar, mit dem es möglich sein sollte, Nachrichten zwischen verschiedenen Kontakten zu versenden und entsprechend zu empfangen. Die Browser-Unterstützung sollte gewährleisten, dass es möglich ist, die App auf verschiedensten Arten von Geräten (Tablet, Handy, etc.) zu nutzen. Weitere Anforderungen waren:

Umfang der Dokumentation

Unsere Dokumentation ist auf eine bestimmte Weise strukturiert. Grundsätzlich orientieren wir uns hierbei an der Art und Weise, wie anfänglich unsere Planungen und schließlich auch die Implementierung verlief. Zunächst gliedert sich die Dokumentation in einen Einleitungsteil, der ein Vorwort sowie die Aufgabenstellung und diesen Absatz umfasst. Der Hauptteil der Dokumentation beginnt mit den „Planungen des Projekts“ in denen die Herangehensweise an die Aufgabenstellung geschildert wird. Diese beginnt mit einer Erklärung der eigentlichen Idee und mündet schließlich im Absatz „Implementierung“, welcher den Quelltext aller Bestandteile des Projekts beschreibt. Hinzu kommt eine Beschreibung der fertigen App, an die sich auch eine genaue Bedienungsanleitung anschließt. Diese Beschreibung kann als Handbuch verwendet werden und mit ihr ist es möglich, jeden Bedienschritt im Messenger nachzuvollziehen und selber durchzuführen. Der Absatz „Ergänzungen“ beschreibt Ergänzungen an der App, die wir hinlänglich

der reellen Betreibbarkeit der App vornehmen. Diese umfassen zum Beispiel eine eigene Datenschutzerklärung und eigene Nutzungsbedingungen.

Modalitäten

Unsere Dokumentation umfasst 39 Seiten und einen Anhang mit 3 Seiten. Außerdem umfasst die Dokumentation 8037 Wörter (949 Wörter im Anhang) und 51559 Schriftzeichen (6216 im Anhang). Es finden sich außerdem 4 Codezeilen-Abbildungen und 59 „normale“ Abbildungen.

Planungen des Projekts

Idee

Unsere Idee war es, einen Messenger zu entwickeln, der einfach zu handhaben ist und sehr eng an modernen Messenger-Standards orientiert ist, um die Bedienung intuitiver zu machen. Dazu sollten Bedienelemente an ähnlichen Stellen zu finden sein und Icons ähnliche Formen aufweisen. Darüber hinaus war es uns wichtig, den Messenger in seinen Funktionen zu begrenzen. Viele Zusatzfunktionen haben zwar Vorteile für Menschen, die sich gerne mit den Programmen beschäftigen und auseinandersetzen, sind aber für den gewöhnlichen Nutzer, der einfach chatten möchte, eher lästig. Somit lag unser Fokus genau auf diesem Gleichgewicht zwischen einem hohen Funktionsumfang und gleichzeitig einer einfachen und intuitiven Bedienung der Anwendung.

Hinzu kommt, dass der Messenger „sicher“ sein sollte. Abgesehen von raffinierten Hack-Methoden war es uns wichtig, dass Datenschutz zunächst für gewöhnliche Nutzer gewährleistet ist. Es sollte also nicht möglich sein, Chats von anderen Leuten einzusehen und grundsätzliche und bekannte Datenschutz-Einstellungen sollten vorzunehmen sein. Zu denen zählt zum Beispiel die Auswahl der Kontaktliste, welche wir durch die Funktion „blockieren“ ermöglichen wollten.

Auch weitergehend planten wir einen Schutz der Daten, der auf der einfachsten Ebene durch das „hashen“ von Passwörtern gewährleistet werden sollte. Weitere Sicherungsfunktionen sollten sich dann während der Implementierung herausstellen, da wir annahmen, in einer direkten Konfrontation wäre dies einfacher.

Umsetzung und Strukturierung

Nach der ersten Idee kam es dazu, diese zu strukturieren und in eine Form zu bringen, welche später tatsächlich umsetzbar sein sollte. Hierzu war es wichtig sich Gedanken darüber zu machen, welche Aspekte der Idee einfacher umzusetzen sind- diese sollten dann weiter vorne implementiert oder umgesetzt werden- und welche Aspekte schwieriger sind und längere Überlegungen benötigen.

Bei den Überlegungen zur Umsetzung kamen wir zu dem Schluss, dass es sinnvoll wäre, unsere App von einer kleinen Startversion aus aufzubauen. Das meint, dass man zunächst eine einfache Version mit grundlegenden Funktionen erstellt, die im Nachhinein erweitert werden kann. Das hatte für uns den Vorteil, dass auch während der eigentlichen Programmierung weitere Ideen eingearbeitet werden konnten. Darüber hinaus stellt diese Methode sicher, dass am Ende der Bearbeitungszeit eine lauffähige Lösung vorliegt und man sich nicht selber überschätzt hat.

Daraufhin begannen wir mit der Planung und Strukturierung der Datenbank. Dazu stellten wir uns Fragen zu den Speicheranforderungen der App.

Danach planten wir in einem weiteren Schritt das sogenannte Backend, also den Java basierten Webserver unserer Anwendung. Hierbei war es besonders wichtig, eine Kommunikation zwischen der Datenbank und dem Server herzustellen, damit dieser Daten aus der Datenbank an die Benutzeroberfläche weitergeben konnte beziehungsweise Daten von der Benutzeroberfläche in die Datenbank einschreiben konnte.

Letztlich nutzten wir viele Skizzen, um das Frontend zu planen. Hierbei schauten wir uns erstmal in modernen Messengern und Chat-Funktionen anderer Anwendungen um. Dabei richteten wir unser Augenmerk besonders auf WhatsApp, Telegram, Teams und die Instagram-Message-Funktion. Schließlich adaptierten wir Gemeinsamkeiten- wie zum Beispiel die Platzierung einzelner Steuerungselemente- für unsere App. Ein weiterer wichtiger Punkt war die Ausgestaltung eines Designs im Hinblick auf die Farben. Hierbei wählten wir ein dunkles Design, welches aus Blautönen bestand. Zugleich war aber auch in Planung, das Design durch einen „Bright-Mode“ zu ergänzen, der dann von helleren Farben geprägt ist. Diese Funktionen sind bei den meisten modernen Apps Standard.

Nach diesen groben inhaltlichen Planungen machten wir uns Gedanken über die Programmiersprachen, in denen wir diese App programmieren wollten.

Für das Frontend war die Benutzung eines JavaScript-Frameworks vorgeschrieben. Da wir durch den Informatik-Unterricht schon in Kontakt mit dem Framework „vue.js“ gekommen waren, planten wir unsere Anwendung damit. Dieses Framework hatte für uns - neben dem Bekanntheitsgrad - noch weitere Vorteile.

Zum einen ist dieses Framework sehr nah am HTML-Code, welchen wir bereits auch im Informatik-Unterricht bereits erlernt hatten, orientiert. Das schafft eine übersichtliche Programmierung des Frontends. Zum anderen hatte dieses Framework für uns noch einen weiteren Vorteil. In der Design-Bibliothek vuetify.js gibt es für vue.js einen reichhaltigen Fundus an modernen Designs, die gut übernommen und adaptiert werden können. Somit war sichergestellt, dass wir unserer Idee der Benutzerfreundlichkeit und einfachen Bedienung gerecht würden und gleichzeitig die Konzentration stärker auf die technischen denn auf die Design-Aspekte richten können.

Durchführung

Design

Aufbau einer modernen Website

Die Unterschiede, die über die Jahre im Webdesign auftraten, sind beinahe unzählbar. Immer neue Strömungen verändern die Art, wie Websites gebaut werden und auch die Designs einzelner Komponenten. Die Aufgabenstellung forderte ein Design der App nach heute gültigen Standards. In diesem Absatz soll kurz zusammengefasst werden, wie moderne Websites aufgebaut sind und wie wir dementsprechend unsere Seite versuchten zu strukturieren.

Laut der Website webdesign-journal.de (siehe Quellenverzeichnis), wird das Design von Websites maßgeblich durch technische Neuerungen beeinflusst. In den letzten Jahren wurden immer mehr JavaScript-Frameworks für das Design von Websites eingesetzt. Diese bieten den Vorteil- anders als herkömmliche HTML-Seiten- komplett "Single-Page-Websites" zu sein.

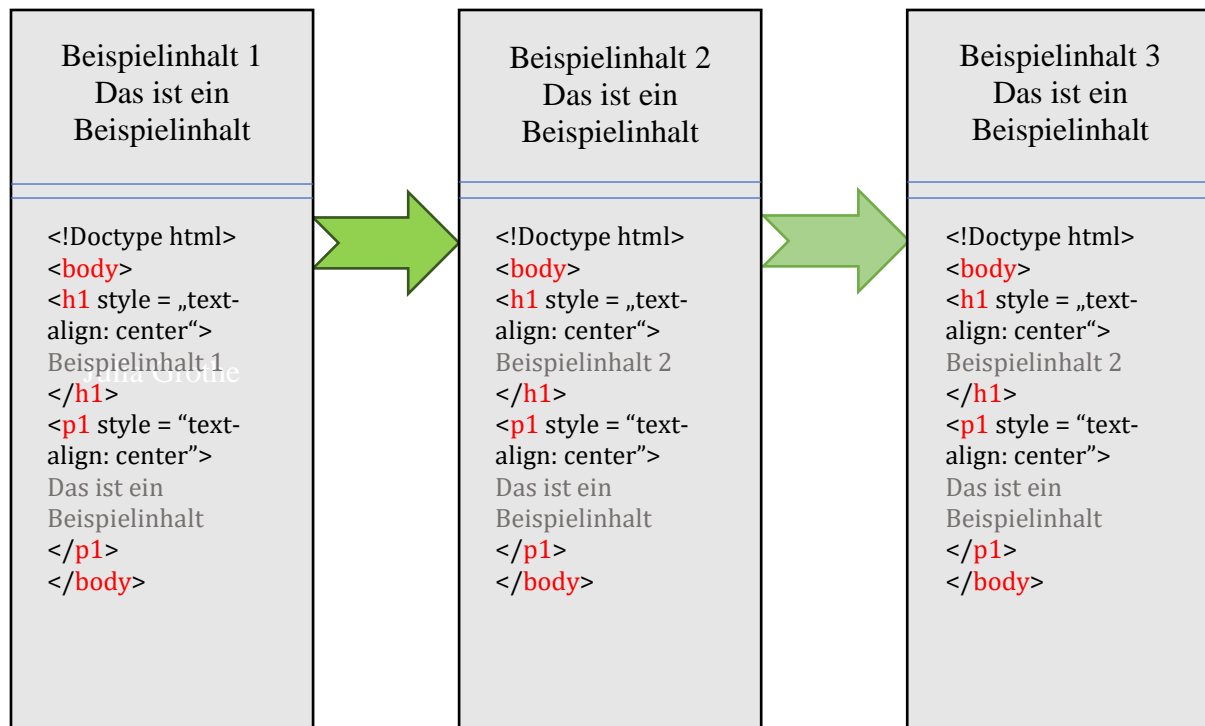


Abbildung 1: Darstellung des technischen Designs einer traditionellen HTML-Website

Die Abbildung veranschaulicht den Aufbau traditioneller HTML-Websites mit ihrem Frontend und dem dahinter stehenden Programm Quelltext. Zu erkennen ist, dass jede neue Seite, jeder Beispielinhalt, tatsächlich eine eigene Website ist, zwischen denen dann gewechselt wird. Unter Buttons finden sich schlussendlich nur Links, die einzelne Seiten austauschen. Das hat einen entscheidenden Nachteil: Bei jedem Aufruf einer neuen Seite, muss diese neu geladen werden. Bei langsamen Internetverbindungen kann das mitunter zu einer sehr langsamen Nutzungsgeschwindigkeit führen, da bei jeder Aktion eine neue Seite geladen werden muss.

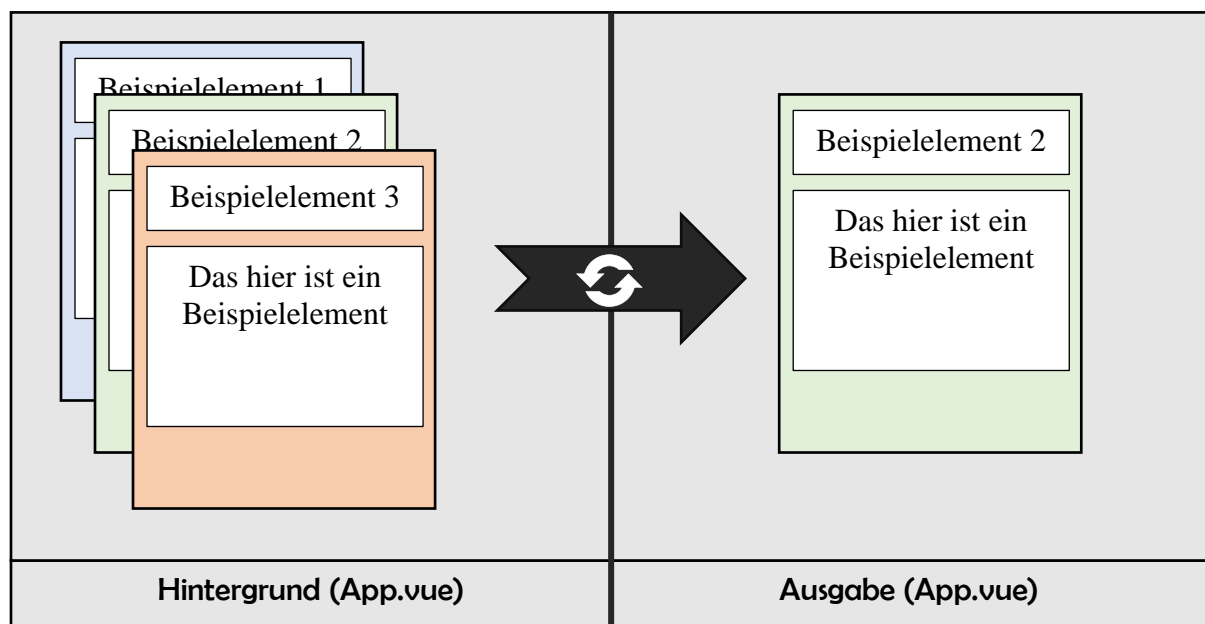


Abbildung 2: technische Darstellung einer Website, die mit vue.js programmiert wurde.

Bei modernen Websites, den sogenannten „Single-Page-Websites“ verläuft die Strategie etwas anders. Wie der Name bereits ahnen lässt, existiert nur eine Seite, die allerdings mehrere Komponenten hat. Diese Komponenten können nun die Funktion der einzelnen Websites der traditionellen HTML-Implementierung übernehmen, wodurch ein entscheidender Vorteil entsteht. Beim Laden der einen Website (App.vue) werden alle Komponenten geladen und stehen nun zur Verfügung. Während der Benutzung der Website können diese dann getauscht werden, ohne, dass neue Wartezeiten entstehen. Der Nutzer bekommt den Eindruck, im Browser mit einem Desktop-Programm zu arbeiten.

Diesen technischen Voraussetzungen zufolge werden moderne Websites designt. Besonders aktuell ist die Gestaltung der Website mit sogenannten „Card“-Elementen. Solche „Card“-Elemente stellen den Inhalt der Website dar, indem sie diesen auf Karten anzeigen. Diese Karten liegen auf dem Hintergrund der Website auf und werden somit auch klar durch Farbe und Seitenbegrenzung abgetrennt.

Die Kartenelemente unterteilen sich schließlich noch in Teile, die wiederum farblich voneinander getrennt werden. Unter diesen Teilen findet sich ein Titel am oberen Rand, der häufig etwas dunkler hinterlegt wird und in der Schriftgröße größer ist. Darunter befindet sich der Inhalt. Am unteren Teil der Karte werden schließlich die Aktionen angezeigt, die meistens in Form von Buttons oder Kästchen zum Ankreuzen vorliegen.

Generell ist das Design stark von Icons geprägt, die eine einfachere und sprachlich unabhängige Bedienung ermöglichen sollen. Dazu ist es wichtig, dass die Icons einen Wiedererkennungswert haben, Icons in bekannten Apps also ähnlich sehen. Häufig werden die Icons auch dadurch ergänzt, dass bei einem sogenannten „hover“ oder „Mouseover“ eine Information in Form einer Benennung angezeigt wird.

Implementierung

Datenbank

Eine Datenbank ist eine Sammlung von Informationen, die sich in Form eines Tabellenschemas organisiert. Daten in einer Datenbank können einfach geöffnet, abgerufen, verwaltet, verändert, aktualisiert und organisiert werden. Datenbanken werden in der Regel zur Speicherung großer Datensätze benutzt, die allerdings dennoch noch verwaltet werden müssen. Dabei finden sich Datenbanken in allen modernen Vertrieben und Verwaltungsagenturen und -Programmen. Datenbanken sind also das gängige Mittel zur Speicherung und Verwaltung von Daten.

In unserem Beispiel ist es ebenfalls wichtig, eine eventuell große Menge an Daten zu verwalten. Je nach Größe des späteren Messengers, muss eine ziemlich große Anzahl an Nutzern gespeichert werden, die wiederum eine noch größere Zahl an Nachrichten versenden. Aus der eigenen Erfahrung von Messenger-Diensten wissen wir, dass – vor allem bei einer längeren kontinuierlichen Nutzung- gut mehrere 10- oder 100tausend Nachrichten gesendet und empfangen werden können. Diese müssen alle gut organisiert abgelegt werden, damit sie den einzelnen Chats zugeordnet werden können.

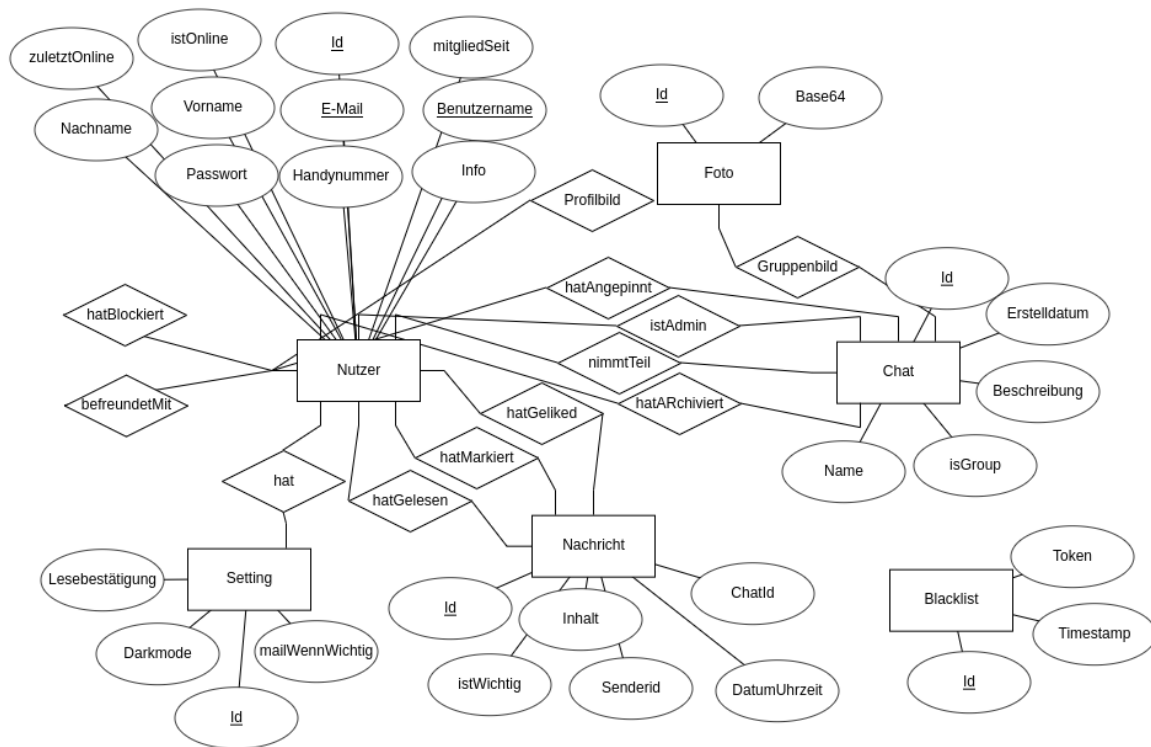


Abbildung 3: Darstellung des relationalen Datenbankschemas

In dieser Abbildung kann man eine Entitäts-Beschreibung in Form eines Entity-Relationship-Diagramms erkennen.

Backend

Im Folgenden findet sich die Dokumentation des Backends. Das Backend in der App hat die Aufgabe, als Server alle Aktivitäten zu verwalten und für den Datenaustausch primär zwischen Frontend und Datenbank zu sorgen.

Unser Backend ist nach dem System eines Restful-Webservices aufgebaut. Dabei hat das Backend die Aufgabe, die von den Nutzern eingegebenen Daten zu verarbeiten und in der Datenbank zu speichern. Außerdem müssen auf Anfrage des Clients Daten aus der Datenbank gezogen, verarbeitet und zurückgegeben werden. Die Datenbankzugriffe werden in unserem Fall durch die Datenbankzugriffsschicht und die zugehörigen Entitätsklassen realisiert, die Verarbeitung und Rückgabe der Clienteingaben geschieht durch die Webservice-Schicht. Dazu gibt es noch Serviceklassen, die einzelne Aufgaben, die sehr häufig ausgeführt werden müssen, auslagern.

Das auf der nächsten Seite abgebildete Modell, das unser Backend in diverse Schichten einteilt, bietet einige Vorteile. Zuerst ist die Übersichtlichkeit beim Entwickeln gesichert, da sich jede Aufgabe, die ausgeführt werden muss, in der dafür vorgesehen Klasse oder dem zugehörigen Paket befindet. Außerdem arbeitet der Service schneller, da sich wiederholende Aufgaben ausgelagert sind.

Anmerkung: Das komplette Javadoc mit sämtlichen Klassen- und Methodendokumentationen befindet sich im Anhang.

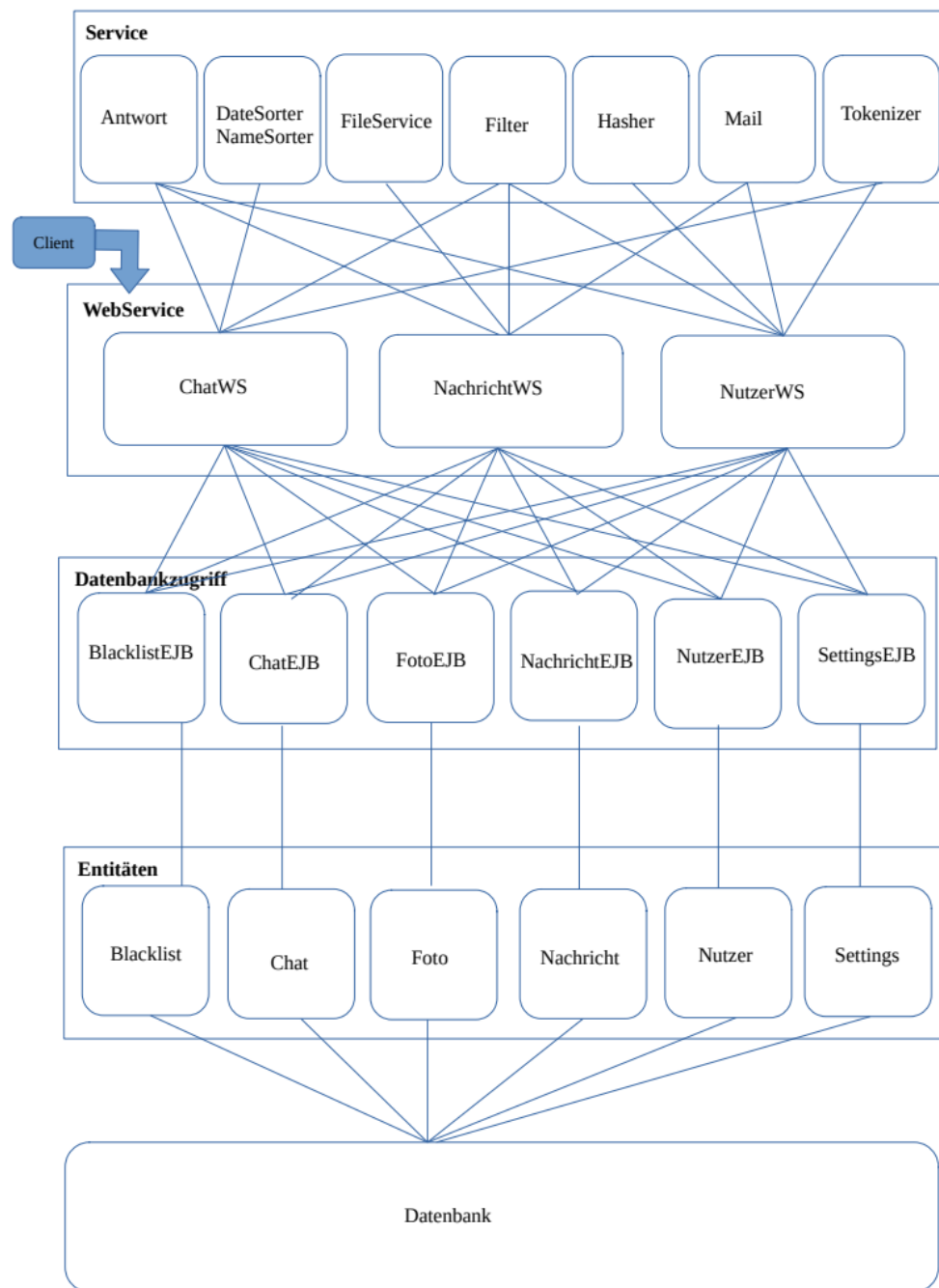


Abbildung 4: Darstellung des Schichtenmodells

Paketübersicht

Packages	
Package	Description
Application	Der Startpunkt der Anwendung.
EJB	Die Datenbankzugriffsschicht.
Entity	Die aus der Datenbank generierten Entity-Klassen.
Filter	Die Klassen, die für das Finden von Schimpfwörtern in Texten zuständig sind.
Service	Die Serviceklassen der Anwendung.
WebService	Der Webservice der Anwendung.

Abbildung 5: Die Paketübersicht des Java-Docs

Jeder Schicht aus dem Schichtenmodell, abgesehen von der Datenbank, ist ein eigenes Paket zugewiesen. Darin befinden sich alle Klassen, die zu dieser Schicht gehören.

Package Webservice

Package Webservice

Der Webservice der Anwendung.

See: [Description](#)

Class Summary	
Class	Description
ChatWS	Der Webserver für die Datenverarbeitung, bezogen auf die Chats.
NachrichtWS	Der Webserver für die Datenverarbeitung, bezogen auf die Nachrichten.
NutzerWS	Der Webserver für die Datenverarbeitung, bezogen auf die Nutzer.

Abbildung 6: Der Package-Webservice

Der Webservice verarbeitet die Nutzereingaben, die vom Client über das HTTP-Protokoll als Plain-Text-Objekte im JSON-Format übergeben werden. Der Webservice greift dann auf die darunterliegende Schicht, die Datenbankzugriffsschicht zu. Um die Verarbeitung der Daten einfacher zu machen, werden zusätzlich Serviceklassen verwendet. Die generierten Antworten beziehungsweise aus der Datenbank gezogene Daten werden als Response-Objekt zurückgegeben, um die CORS-Problematik (Cross-Origin Resource Sharing) zu umgehen. Der Webservice ist wegen der Übersicht nach den einzelnen Datenbankklassen aufgeteilt (Nutzer, Chats, Nachrichten und Einstellungen). Die zu den Aufgaben des Webservices gehörenden Routen folgen dabei einem strikten Prinzip. So erhält jede Klasse einen Pfad, der die Entitätsklasse angibt, zum Beispiel „/nutzer“, und eine Kurzbeschreibung der angefragten Aufgabe, wie „/getFriendList“. Am Ende jeder Route, werden optionale Daten angeben und zwingend der Token des Nutzers.

Package EJB

Diese Klassen sind für den Datenbankzugriff zuständig. Sie ziehen mithilfe des javaeigenen EntityManagers die Daten aus der Datenbank. Das geschieht mithilfe von sogenannten NamedQueries, die in den Entity-Klassen definiert sind und wie SQL-Statements aufgebaut sind, zum Beispiel „@NamedQuery(name="Nachricht.findByNachrichtid", query="SELECT n FROM Nachricht n WHERE n.nachrichtid = :nachrichtid)",“. Der EntityManager kann nun auf diese Suchanfragen zugreifen und erhält so die geforderten Datensätze aus der Datenbank. Die

Rückgaben der Queries werden dann in die zugehörigen Entity-Klasse umgewandelt und an die Webservernschicht weitergegeben.

Jede Entity-Klasse besitzt eine eigene EJB-Klasse, die die zugehörige Datenbank Tabelle verwaltet.

Package EJB

Die Datenbankzugriffsschicht.

See: [Description](#)

Class Summary	
Class	Description
BlacklistEJB	Die Klasse zum Verwalten der Blacklist-Tokens in der Datenbank.
ChatEJB	Die Klasse zum Verwalten der Chats in der Datenbank.
FotoEJB	Die Klasse zum Verwalten der Fotos in der Datenbank.
NachrichteEJB	Die Klasse zum Verwalten der Nachrichten in der Datenbank.
NutzerEJB	Die Klasse zum Verwalten der Nutzer in der Datenbank.
SettingsEJB	Die Klasse zum Verwalten der Einstellungen in der Datenbank.

Abbildung 7

Package Entity

Die Entity-Klassen repräsentieren jeweils eine Tabelle der Datenbank, also in unserem Fall Nutzer, Chat, Nachricht, Einstellungen, Blacklist und Fotos. Sie werden mithilfe von Netbeans automatisch generiert, sodass jede Klasse die Attribute aus der zugehörigen Datenbanktabelle erhält. Die Datenbank enthält auch Tabellen, die keine eigene Klasse abbilden. Sie sind nur für Beziehungen untereinander zuständig, wie zum Beispiel die Nutzerliste eines Chats. Diese Beziehungen sind in den Entity-Klassen mit sogenannten „Join-Tables“ realisiert. Durch diese erhält zum Beispiel der Chat eine Liste aus Nutzern als Attribut. Die einzelnen Klassen und ihre Zuordnung untereinander sind Abbildung 1 zu entnehmen (siehe oben).

Package Entity

Die aus der Datenbank generierten Entity-Klassen.

See: [Description](#)

Class Summary	
Class	Description
Blacklist	Die Klasse, die ein Token in der Datenbank darstellt.
Chat	Die Klasse, die einen Chat in der Datenbank darstellt.
Foto	Die Klasse, die ein Foto in der Datenbank darstellt.
Nachricht	Die Klasse, die eine Nachricht in der Datenbank darstellt.
Nutzer	Die Klasse, die einen Nutzer in der Datenbank darstellt.
Setting	Die Klasse, die die Einstellungen eines Nutzers in der Datenbank darstellt.

Abbildung 8

Package Service

Package Service

Die Serviceklassen der Anwendung.

See: [Description](#)

Class Summary	
Class	Description
Antwort	Die Klasse zum Erstellen von Response-Objekten.
DateSorterChat	Die Klasse zum Sortieren der Chats.
FileService	Die Klasse zum Verwalten von Dateien, die von Nutzern gesendet werden.
Filter	Die Klasse zum Filtern von Nachrichten.
Hasher	Die Java-Klasse zum Hashen der Passwörter.
Mail	Die Java-Klasse zum Versenden von E-Mails.
NameSorter	Die Klasse zum Sortieren der Chats.
Tokenizer	Die Java-Klasse zur Identifizierung und Authentifizierung der Nutzer.

Abbildung 9

Die folgenden Klassen übernehmen Aufgaben, die nicht klar in die anderen Schichten einzuordnen sind. Um auch hier Redundanz zu vermeiden und die Organisation zu vereinfachen, sind die Klassen der Aufgaben nach voneinander getrennt.

Klasse Antwort

Beim Verbinden des Front- und Backends ist das Problem des Cross Origin Ressource Sharing, kurz CORS, aufgetreten. Es tritt immer dann auf, wenn zwei Serveranwendung Daten untereinander austauschen, da es sich um eine massive Sicherheitslücke handeln kann. In unserem Anwendungsfall muss CORS aber explizit erlaubt werden, da Front- und Backend voneinander getrennt sind. Als Lösungsansatz haben wir uns dafür entschieden, alle Antworten des Backend-Servers in ein HTTP-Response-Objekt zu kapseln, das heißt der Server liefert nicht direkt den JSON-String, sondern ein Response-Objekt, in welchem dann der JSON-String steckt.

Klasse DateSorter

Da wir uns beim Erstellen unseres Messengers an der Gestaltung und Nutzungsweise aktueller Software orientiert haben, wird die einheitliche Chat- und Gruppenliste eines Nutzers des Datums der letzten Nachricht nach absteigend sortiert. Diese Klasse erstellt dafür einen Comparator, der die Daten der letzten Nachrichten miteinander vergleicht. Der Webserver wiederum nutzt diesen dann zur Sortierung der Chats und Gruppen.

Klasse Hasher

Um die Sicherheit der Nutzer zu gewährleisten und um der neuen Datenschutzgrundverordnung zu entsprechen, werden die in der Datenbank gespeicherten Nutzerpasswörter gehasht. Es wird eine kryptografisch sichere Hashfunktion, hier PBKDF2 mit HMAC SHA512, eingesetzt. Dabei handelt es sich um eine Einwegfunktion, das heißt aus dem Hashwert kann das Passwort nicht mehr berechnet werden. Die Sicherheit wird weitergehend durch die Nutzung eines Salt-Wertes erhöht, welcher beim Hashen eines Passworts einfach an dieses gehängt wird. Dadurch wird der Suchraum bei zum Beispiel einem Rainbowtable-Angriff massiv vergrößert.

Klasse Mail

Um weitergehend mit den Nutzern kommunizieren zu können, haben wir uns für die Einrichtung eines E-Mail-Services entschieden. Die E-Mails können automatisiert an die Nutzer versendet werden, wobei es zwei Einsatzzwecke gibt. Der Erste kommt direkt beim Registrieren zu tragen. Damit das System vor Bots geschützt ist und die Nutzer sich auch mit einer validen E-Mail anmelden, wird bei der Erstellung eines Accounts ein zufälliger, vierstelliger Pin erstellt und per E-Mail an den Nutzer verschickt. Bevor dieser sich in das System einloggen kann, muss er den erhaltenen Pin erst bestätigen. Der zweite Einsatzzweck ist die Informierung des Nutzers über die Änderung seiner Accountdaten, damit dieser, falls jemand ohne Berechtigung die Daten verändert hat, darauf aufmerksam wird.

Im Folgenden sind Beispielmails abgebildet, die vom Server versendet wurden (hier am Beispiel der Mail-App von iOS-Geräten).

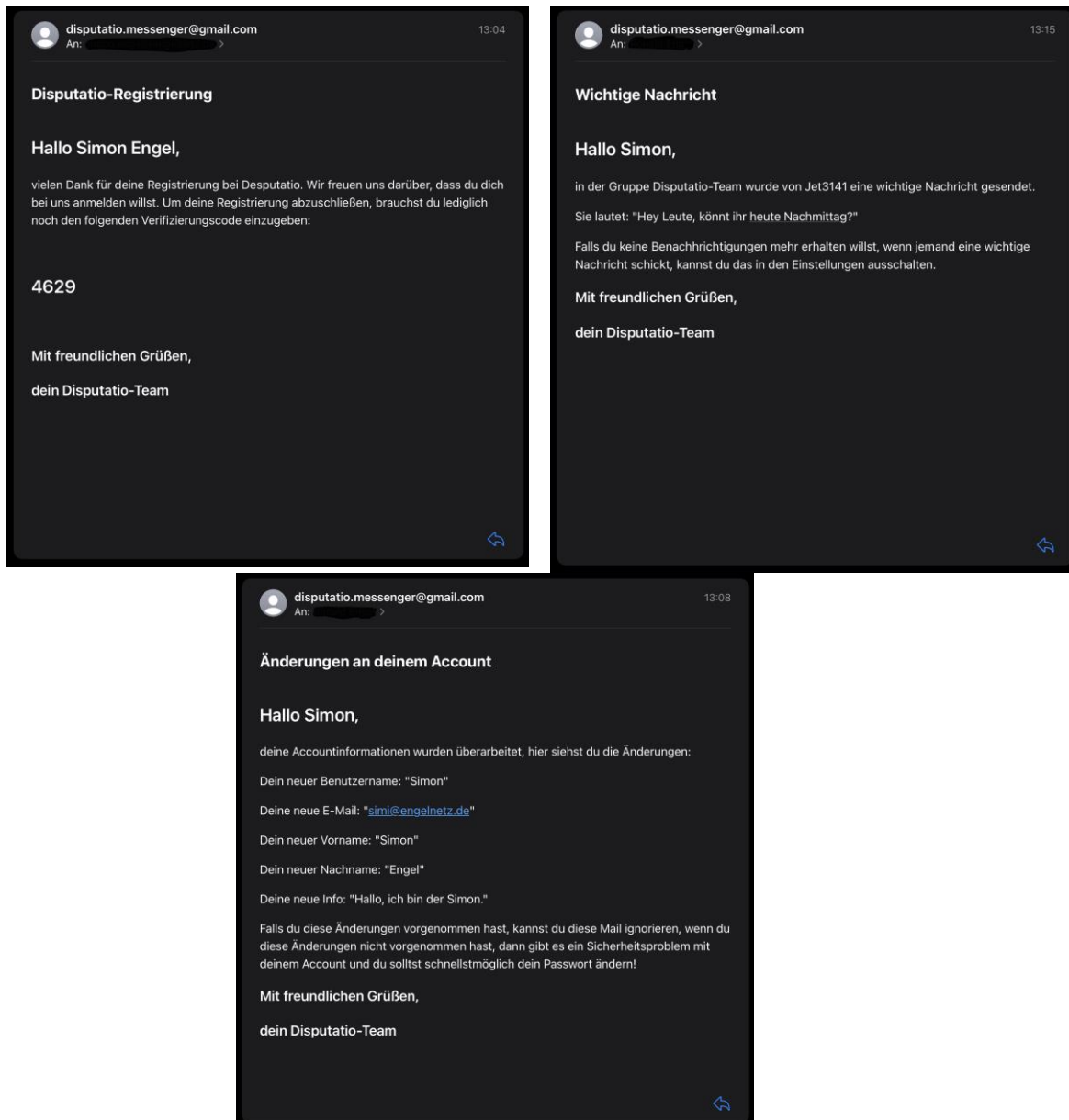


Abbildung 10: Beispiele für gesendete E-Mails der Klasse Mail

Klasse Tokenizer

Um die Sicherheit unseres Messengers zu gewährleisten, haben wir uns für die Nutzung von JWTs, also JSON-Web-Tokens entschieden. Diese haben eine frei konfigurierbare Ablaufzeit, nach der das Token nicht mehr gültig ist. Bei jeder Anfrage des Nutzers an den Server wird überprüft, ob es sich

bei dem gesendeten Token um ein noch gültiges handelt. Ist das der Fall, wird die Gültigkeitsdauer wieder um die festgelegte Zeit verlängert. Die Tokens werden über eine sichere Hashfunktion (HMAC256) aus dem Nutzernamen und einem Geheimnis, in unserem Fall die Variable SECRET in der Klasse Tokenizer, berechnet. Da es sich dabei um eine Einwegfunktion handelt, kann der Benutzername des Nutzers problemlos als Klartext in das Token kodiert werden. Das hat den großen Vorteil, dass die Tokens nicht alle in der Datenbank gespeichert werden müssen, was unnötigen Traffic bedeuten würde, sondern eine schnellere Überprüfung gewährleistet ist. Ein Nachteil an JWTs ist, dass ein klassisches Logout aus dem System nicht vorhanden ist. Ein Token wäre so auch nach einem Logout eine Eintrittskarte in das System. Dieses Problem haben wir über eine Blacklist gelöst. Das bedeutet, dass bei einem Logout das Token mit Zeitstempel in der Datenbank gespeichert wird. Bei der Überprüfung eines Tokens muss es dann noch zusätzlich gegen die Blacklist geprüft werden, die einmal pro Stunde geleert wird, um diesen Prozess zu verkürzen.

Klasse FileService

Beim Aufbau unseres Messengers haben wir uns dazu entschieden, unseren Nutzern die Möglichkeit geben zu wollen, Dateien zu verschicken. Das stellte uns vor ein paar Probleme. Das Größte war es, wie die Dateien in der Datenbank gespeichert werden sollen. Die Dateien werden vom Frontend in Base64-Strings umgewandelt. Da diese bei Textdokumenten auch mal mehrere Millionen Zeichen lang werden können, konnten diese nicht direkt in der Datenbank gespeichert werden. Aus diesem Grund werden die Base64-Strings in Dateien geschrieben, die gespeichert werden und einer festen Namensgebung folgen. So steht zuerst die Id des Senders der Nachricht, gefolgt von einem "|", dann das Sendedatum Uhrzeit der Nachricht in Millisekunden, wieder gefolgt von einem "|" und abschließend der Name der Datei. Ein Beispiel dafür ist: „2|1617210935178|test.txt“. Der Name der Datei wird in den Inhalt der Nachricht geschrieben, weswegen der eigentliche Dokumenteninhalt mit diesem aus der gespeicherten Datei gelesen werden kann. Diese Vorgehensweise bietet den Vorteil, dass kaum eine Begrenzung der Dateigröße gegeben ist, da das System nicht darauf angewiesen ist, etwas in die Datenbank zu schreiben, wobei es zu Größenbegrenzungen kommen kann.

Klasse Filter

Um den Nutzern unserer Anwendung ein möglichst angenehmes Erlebnis zu liefern, haben wir uns für die Nutzung eines Schimpfwortfilters entschieden. Da es bei der Implementierung eines solchen Systems viele Dinge zu beachten gibt, haben wir uns für den Einbau eines bereits bestehenden Filters entschieden. Dieser ist hier zu finden: „<https://github.com/Zabuzard/Grawlox>“ (Stand 27. April 2021). Er besitzt eine frei konfigurierbare Liste an nicht erlaubten Wörtern, prüft die Wörter unabhängig von Groß- und Kleinschreibung und ist in der Lage „LeetSpeak“ zu erkennen, also die Ersetzung von Buchstaben durch Zahlen (zum Beispiel: h3llo). Die unerlaubten Wörter können mit einem wählbaren Zeichen ersetzt werden, in unserem Fall ein „*“. Der Filter kann von den Nutzern auf Wunsch ausgestellt werden, ist standardmäßig aber eingeschaltet. Diese Klasse stellt die Verbindung zu den eigentlichen Filterklassen dar.

Aufbau weiterer Funktionen

Die Online-Funktion

Heutige Messenger-Dienste bieten in der Regel die Möglichkeit, für die Kontaktliste einen Status abzurufen. In der Regel hat dieser Status zwei Zustände: Entweder der Kontakt ist offline und seine letzte Online-Zeit wird angezeigt, oder der Kontakt ist online, was ebenfalls angezeigt wird.

Da wir (siehe Idee) uns an modernen Messengern orientieren, entschieden wir uns, diese Funktion auch für unseren Messenger zu implementieren. Dabei sind wir aber auf einige Probleme gestoßen. Den Status eines Nutzers auf „online“ zu setzen, war durch die Login-Route, die bei jedem Login angesteuert wird, einfach zu realisieren. Das Gegenteil, also das Setzen des Status auf „offline“, stellte sich als bedeutend schwieriger heraus, da ein Nutzer nicht nur durch das Aufrufen der Route zum Ausloggen nicht mehr im System sein kann, sondern auch sein Token einfach ablaufen kann und er deshalb keinen Zugriff mehr auf das System hat, also offline ist. Um diese Problematik zu lösen, haben wir uns dazu entschieden, dass bei jedem Aufruf jeglicher Methoden durch einen Nutzer sein Status auf „online“ gesetzt wird sowie das Datum, zu dem er zuletzt online war, auf das aktuelle Datum. Nun wird alle fünf Minuten eine Methode ausgeführt, die den Status aller Nutzer auf „offline“ setzt, sowie wieder das Zuletzt-Online-Datum auf das aktuelle Datum. So ist auch gegeben, dass auch die Nutzer, die sich nicht bewusst ausloggen, nicht durchgehend online sind.

Frontend

Das Programmieren mit vue.js

Die Script-Sprache vue.js ist eine Sprache zur Programmierung dynamischer Websites, die den Vorteil hat, dass sie durch Vorkenntnisse in ähnlichen Sprachen einfach zu erlernen ist. Dabei orientiert sich vue.js sehr stark an HTML und beinhaltet außerdem Elemente von CSS. Der Script-Teil einer solchen Sprache kann in einfachem JavaScript beschrieben werden. Die einzelnen Befehle werden dabei ähnlich wie bei HTML- in spitze Klammern geschrieben.

Vue.js bringt dabei einige Funktionen mit, die ein sogenanntes „conditional rendering“ zulassen. Damit ist gemeint, dass Elemente der App durch gewisse Ergänzungen nur angezeigt werden, wenn eine gewisse Bedingung erfüllt ist. Für dieses „conditional rendering“ bestehen zwei Möglichkeiten.

1. Der „v-show“ Block.

Die erste dieser Möglichkeiten ist der sogenannte „v-show“-Block. Diese Funktion ist sehr einfach und somit nicht vielseitig nutzbar, weil sie letztlich den binären Zweig zwischen anzeigen/nicht anzeigen darstellt. Dabei wird ein Objekt angezeigt, wenn die Bedingung des Blocks erfüllt ist. Wird diese nicht erfüllt, so bleibt das Objekt versteckt.

```
<v-bin v-show="loggedIn" @click="login()">Hier klicken</v-bin>
```

In
dieser

Codezeile 1

Abbildung kann man die Syntax eines Buttons erkennen, welcher nur in einem bestimmten Fall angezeigt wird. Dieser Fall wäre nun, wenn die Variable „loggedIn“ den Wert „true“ hätte. Wie zu sehen ist, erfolgt die Implementierung dieses Blocks direkt im Button-Bereich der spitzen Klammern. Dabei wird mit einem Gleichheitszeichen und doppelten Anführungszeichen die Variable gekennzeichnet, die den Wert „true“ haben soll. Wie bereits erwähnt, ist das v-show-Element recht primitiv, da es nur die beiden Möglichkeiten „anzeigen“ oder „nicht anzeigen“ beinhaltet.

Sollte bei dem vorliegenden Beispiel der Knopf nur angezeigt werden sollen, wenn die Variable „loggedIn“ auf „false“ steht, dann müsste die Implementierung folgendermaßen

```
<v-btn v-show="!loggedIn" @click="login()">Hier klicken</v-btn>
```

Codezeile 2

aussehen:

Das hier im prägnantem Grün markierte Ausrufezeichen stellt den Unterschied dar. Dieses Zeichen ist aus verschiedenen Programmiersprachen für die Verneinung bekannt. In diesem Falle würde der Knopf also nur im Falle dessen, dass „loggedIn“ den Wert „false“ hat, angezeigt.

2. Der „v-if“ Block.

Die zweite Möglichkeit für das „conditional rendering“ wird durch den sogenannten „v-if“-Block dargestellt. Wie die Bezeichnung vermuten lässt, werden mit diesem Element komplexe Überprüfungen möglich. Dabei besteht das komplette Element aus drei Anweisungen, die sich gegenseitig bedingen, von anderen Programmiersprachen aber durchaus bekannt sind.

Die „v-if“-Anweisung stellt anschaulich eine „Wenn, dann“ Bedingung dar. Sollte also eine bestimmte Bedingung erfüllt sein, dann wird etwas anderes ausgeführt. Um dies zu ergänzen, gibt es neben „v-if“ auch noch „v-else“, ein Fall, der auftritt, sollte die Bedingung nicht erfüllt werden. Zu guter Letzt stellt das Element „v-else-if“ die Möglichkeit dar, nach der Überprüfung über eine „v-if“-Anweisung, die nicht erfüllt wird, die „else“-Anweisung zu differenzieren.

Diese Anweisungen, die ebenfalls in den spitzen Klammern eingetragen werden, ermöglichen- ähnlich wie das „v-show“- das Anzeigen gewisser Objekte in einem bestimmten Fall.

```
<p v-if="angenommen === true && angegeben === true">
  Sie haben beide Bedingungen erfüllt.
</p>
<p v-else-if="angenommen === true && angegeben === false">
  Sie haben lediglich angenommen.
</p>
<p v-else>
  Sie erfüllen keine Bedingung.
</p>
```

Angenommen: true Angabegeben: true	Angenommen: true Angabegeben: false	Angenommen: false Angabegeben: false
Sie haben beide Bedingungen erfüllt.	Sie haben lediglich angenommen.	Sie erfüllen keine Bedingung.

Abbildung 11: Darstellungen eines <v-if>-Komplexes

In der Abbildung sieht man die Implementierung eines Beispiel-Blocks mit allen drei Anweisungen. Diese stehen in den spitzen Klammern des Teils des Quellcodes, der die Möglichkeit, Text zu verfassen, öffnet. Die vorliegenden Anweisungen überprüfen mit einem „und“ Attribut (&&) zwei Variablen. Unter dem Quelltext ist die jeweilige Ausgabe im Falle des grün hinterlegten Ausgabewertes der Variablen zu erkennen.

Über das „conditional rendering“ hinaus bietet vue.js auch weitere bekannte Methoden aus anderen Programmiersprachen. Eine davon sind die sogenannten „for“-Schleifen.

Im Allgemeinen ist eine „for“-Schleife eine Wiederholung einer gewissen Anzahl an Programmzeilen, die für eine gewisse Menge an Objekten, bzw. solange eine gewisse Anzahl noch nicht erreicht wurde, durchgeführt wird. Als solche ist sie in jeder Programmiersprache zu finden und unerlässlich.

Bei vue.js findet sich die „v-for“-Anweisung ähnlich wie beim „conditional rendering“ - in den spitzen Klammern des Teils des Befehls, der den Befehl öffnet. Prinzipiell also wie folgt:

```
<div v-for="...">Ein Beispieltext</div>
```

Codezeile 3

Eine „for“-Anweisung wird im Programmquelltext vor allem für das Anzeigen von Listen in vue.js Objekten benutzt, die allerdings entweder dynamisch sind, oder alle in der gleichen Objektart wiedergegeben werden sollen. Ein Beispiel dafür wäre eine Liste von Anweisungen in einer Bedienungsanleitung.

Ausgabe (Frontend)	Quellcode (vue.js)
<p>Eine Bedienungsanleitung.</p> <ol style="list-style-type: none"> 1. Öffnen Sie das Programm. 2. Klicken Sie auf „Login“. 3. Bitte geben Sie Ihre Daten an. 	<pre><p>Eine Bedienungsanleitung</p> <li v-for="schritt in bedienungsanleitung"> {{schritt.anweisung}} <script> data:{ Bedienungsanleitung:[{ anweisung: ‚Öffnen Sie das Programm‘ } { anweisung: ‚Klicken Sie auf „Login“.‘ } { anweisung: ‚Bitte geben Sie Ihre Daten an.‘ }] } </script></pre>

Abbildung 12: Darstellung eines <v-for>-Komplexes

Wie in der Abbildung zu erkennen ist, lassen sich somit Listen darstellen, ohne jedes einzelne Element in den tatsächlichen Quelltext der Website eintragen zu müssen. Diese Funktion ist sehr komfortabel, da sie gerade bei aufwendigen Designs Arbeit erspart.

Besonderheiten der Programmierung mit vuetify.js

Am Anfang dieses Kapitels wurde erwähnt, dass wir für das Design unserer Website die UI-Elemente von vuetify.js und dessen Funktionen verwendeten. Im Folgenden soll auf die Programmierung mit diesen Komponenten näher eingegangen werden und die Besonderheiten des Aufbaus von Seiten der Implementierung verdeutlicht werden.

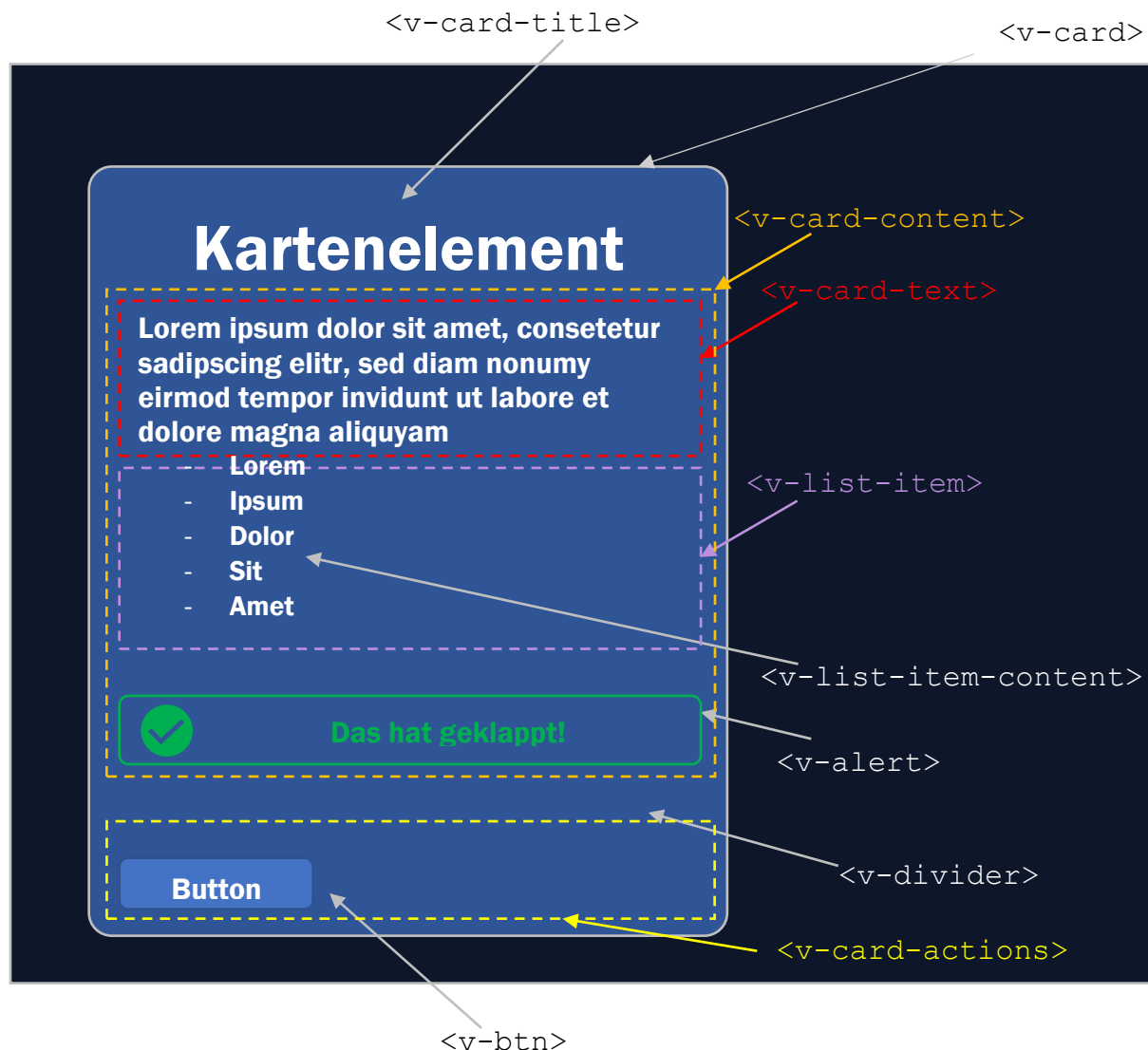


Abbildung 13: Darstellung der einzelnen vue (vuetify)-Elemente und ihrer Funktion in der Anwendung.

In der vorliegenden Abbildung ist der Aufbau einer Website zu erkennen, die mit vuetify.js-Elementen zusammengefügt wurde. Die moderne Aufbauweise heutiger Websites erfordert den Aufbau mit sogenannten Card-Elementen (näheres dazu findet sich unter „Design“)

```
<v-btn dense text outlined :dark="darkmode" @click="login()">Login</v-btn>
```

Codezeile 4

Die Programmierung im <script>-Teil

Zu Anfang wurde angesprochen, dass auch eine Programmierung im <script>-Teil möglich ist. Wie bei einer HTML-Seite hat man auch bei vue.js die Möglichkeit, in einem abgetrennten Script-Teil, welcher wie alle anderen Elemente mit „<script></script>“ umklammert wird, eine andere Programmiersprache zu verwenden: JavaScript. JavaScript auf einer Website zu verwenden hat viele Vorteile.

So können in unserem Beispiel gewisse Prozesse der Website auch auf dieser ausgeführt werden und die Kommunikation zum Server kann somit ausgelassen werden. Das kommt vor allem bei „Spielereien“ wie Animationen- von denen in der App einige verbaut sind- zum Einsatz. Gewisse Rechenprozesse werden dann einfach über die Website ausgeführt, wodurch diese dynamischer wird.

Darüber hinaus lässt sich mit der Möglichkeit, JavaScript zu verwenden, die Kommunikation zwischen dem Server und der Frontend besser realisieren. Die eingegebenen Daten können organisiert werden und in einem JSON-String (Erklärung folgt) verpackt an den Server verschickt werden.

Besonderheiten im Script-Teil:

1. Der JSON-String

Der JSON-String ist eine Möglichkeit zur Kommunikation zwischen dem Server und dem Frontend über das http-Protokoll.

2. Animationen

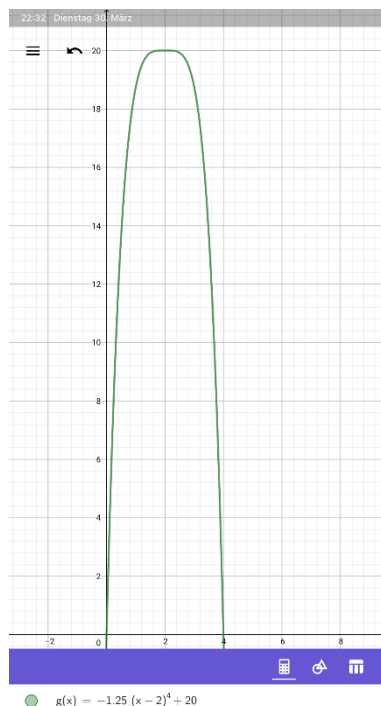


Abbildung 14: Der Graph der Animation

Nachrichten-Ladeanimation:

Nachdem die Nachrichten geladen wurden, werden sie in das HTML-Dokument eingefügt und sind zu Beginn nicht sichtbar.

Mit zwei unterschiedlichen Animationen werden die Nachrichten dem Nutzer angezeigt. Die erste Animation ist für das horizontale gleiten von links nach rechts. Dabei wird die Nachricht mit einer linearen Geschwindigkeit eingeblendet. Die zweite Funktion ist für das vertikale Scrollen. Der Vorgang besteht aus drei unterschiedlichen Funktionen, abhängig von der Anzahl an Nachrichten.

Die nebenstehende Funktion

$$f(x) = -1.25 * (x-2)^4 + 20$$

Ist die erste und scrollt durch alle geladenen Nachrichten durch, beginnend mit der letzten bis zur neusten. Sie scrollt für 4s bei bis zu 40 Nachrichten. Die zweite Funktion

$$f(x) = -0.078125 * (x - 4)^4 + 20$$

Gilt für eine Nachrichtenanzahl bis zu 59 Nachrichten. Ab 60+ Nachrichten wird eine endlose scroll-Animation abgespielt, bis das

Ende erreicht wird.

Die Funktion zum endlosen scrollen lautet wie folgt:

$$f(t, x) = 1.0625 * (t - (x / 18))^2 + x$$

Wobei „x“ die Anzahl an Nachrichten ist und „t“ für die vergangene Zeit seit Animationsbeginn steht.

Vorstellung der App

In diesem Kapitel möchten wir die fertige App vorstellen, die genau so programmiert ist, wie sie im Kapitel „Implementierung“ beschrieben wurde und wie wir sie geplant haben. Während der Implementierung hat es sich als sinnvoll erwiesen, von der Basis-App aus auszugehen und das System darauf aufzubauen. Im Folgenden wird der Funktionsumfang erklärt, sowie die Basis-Bedienung neben ein paar nützlichen Funktionen erläutert.

Anleitung

Es gibt einen Admin-Nutzer mit Beispieldaten, damit alle Funktionen angesehen werden können. Die Zugangsdaten sind: Admin (Benutzername), admin (Passwort).

1. Login/Registrieren

Bevor Sie den Messenger verwenden, müssen Sie sich einloggen. Dafür ist ein Benutzername, ein Passwort und die E-Mail-Adresse erforderlich. Alle anderen Felder müssen nicht zwingend ausgefüllt werden.

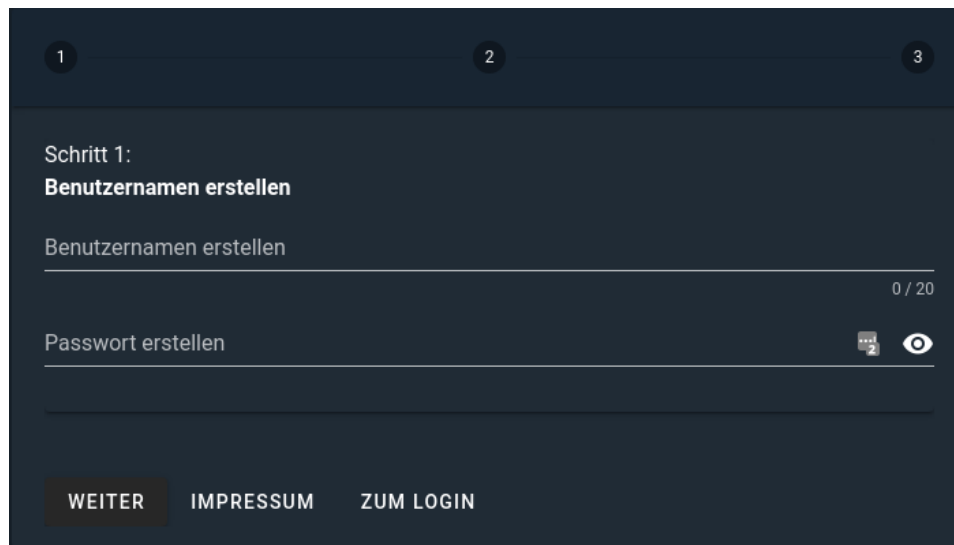


Abbildung 15: Desktop-Ansicht des Registrieren Fensters

Wenn Sie auf der ersten Karte (siehe Abbildung 7) das Feld „zur Anmeldung“ klicken, gelangen Sie zu dieser Ansicht. Hier müssen Sie den Benutzernamen und das Passwort erstellen. Wenn Sie Ihr Passwort unzensiert anzeigen wollen, dann klicken Sie auf das Auge.

The image shows a mobile application interface for the 'Disputatio' system. At the top, the word 'Disputatio' is displayed in a stylized font. Below it, a progress bar indicates three steps, with the first step (1) being active. The main content area is titled 'Schritt 1: Benutzernamen erstellen'. It contains two input fields: 'Benutzernamen erstellen' with a character count of '0 / 20', and 'Passwort erstellen' with a character count of '0 / 20'. A small icon of a person is visible next to the password field.

Abbildung 16: Mobile Version des Registrieren-Fensters

The image shows a desktop application interface for the 'Disputatio' system. At the top, a progress bar indicates three steps, with the second step (2) being active. The main content area is titled 'Schritt 2:'. It contains five input fields: 'Vornamen eingeben' (0 / 25), 'Nachnamen eingeben' (0 / 25), 'E-Mail eingeben' (0 / 45), 'Info eingeben' (0 / 256), and 'Profilbild hinzufügen' (0 / 256). Below the input fields, there are two checkboxes: 'Ich benötige Hilfe.' (unchecked) and 'Ich bin mit den Nutzungsbedingungen und der Datenschutzerklärung einverstanden.' (checked). At the bottom, there are two buttons: 'REGISTRIEREN & WEITER' and 'ZURÜCK'.

Abbildung 17: Desktop-Ansicht der zweiten Registrieren-Karte

Nachdem Sie ihr Passwort und den Benutzernamen erstellt haben, gelangen Sie mit dem Klick auf „Weiter“ zu dieser Karte. Hier können Sie weitergehende Informationen eingeben. Nur die E-Mail-Adresse ist verpflichtend, weil darüber die 2-Faktor-Authentifizierung stattfindet.

An den Zählern am Ende der Eingabefelder können Sie erkennen, wie viele Zeichen Sie noch einfügen dürfen. Die Zeichenlängen sind so gewählt, dass auch zweite Vornamen im Regelfall eingegeben werden können.

The image shows the mobile view of the registration form, specifically the second step (Schritt 2). The form is titled "Disputatio" at the top. It features a progress bar with three steps: 1 (checked), 2 (current), and 3. The form fields are: "Vornamen eingeben" (0 / 25), "Nachnamen eingeben" (0 / 25), and "E-Mail eingeben" (0 / 45).

Abbildung 18: Die mobile Ansicht der zweiten Registrieren-Karte

Bei dieser mobilen Ansicht muss im Zweifelsfall gescrollt werden, weil die ganze Karte (auch im Hochformat) bei den meisten Handys zu hoch ist.

The image shows the desktop view of the login form. It is titled "Login" and contains two input fields: "Benutzername" (0 / 20) and "Passwort" (with a password strength indicator and a toggle for visibility). Below the fields is a link "Noch nicht angemeldet?". At the bottom, there are two buttons: "ZUR ANMELDUNG" and "LOGIN".

Abbildung 19: Desktop-Ansicht der Login-Karte

Nachdem Sie alle Daten auf der zweiten Registrieren-Karte angegeben haben und bestätigt haben, dass Sie mit den Nutzungsbedingungen einverstanden sind, gelangen Sie zu dieser ersten Seite. Hier müssen Sie sich anmelden, das bedeutet, Sie geben Ihren neu erstellten Benutzernamen und Ihr Passwort an. Danach klicken Sie auf „Login“.

The image shows the mobile view of the login form. It is titled "Disputatio" at the top. The form is titled "Login" and contains two input fields: "Benutzername" (0 / 20) and "Passwort" (with a password strength indicator and a toggle for visibility). Below the fields is a link "Noch nicht angemeldet?". At the bottom, there are two buttons: "ZUR ANMELDUNG" and "LOGIN".

Abbildung 20: Die mobile Version der Login-Karte

2. Chats

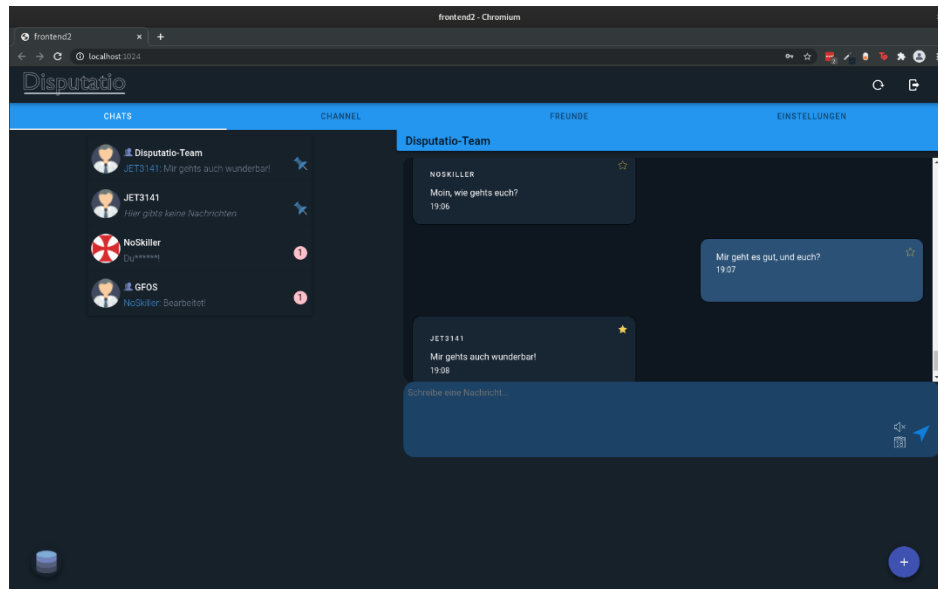


Abbildung 21: Die Chatansicht in der Desktop-Variante (Übersichtsbild)

2.1. Die Chatliste



Abbildung 22: Die Desktop-Ansicht der Chatliste

Unter Punkt 2 wird die gesamte Bildschirmansicht gezeigt, die diesen Bereich ebenfalls abdeckt. An dieser Stelle kann man nämlich die Chatliste sehen, eine Auflistung aller Chats, die man hat. Im Zweifel kann in dieser Liste gescrollt werden, wenn die Anzahl die Höhe des Fensters übersteigt. Die letzten Nachrichten werden als Schrift unter dem Namen der Gruppe oder des Chats angezeigt. Durch das Personen-Icon links neben den Namen ist zu erkennen, ob der Chat eine Gruppe oder ein Einzel-Chat mit nur einem anderen Nutzer ist. Eine Gruppe ist durch das vorhandene Icon gekennzeichnet.

Die Stecknadel am rechten Rand zeigt an, ob der Chat „angepinnt“ ist. Solche Chats werden durch neue Nachrichten in ihrer Position nicht verschoben und bleiben immer am oberen Ende der Liste, sodass man schnell auf sie zugreifen kann.

Erhält man eine Nachricht, wird links neben der Stecknadel oder ganz am rechten Rand ein blauer Kreis mit der entsprechenden Anzahl der Nachrichten angezeigt.

Grundsätzlich bietet die Liste mehrere Aufruf-Möglichkeiten, dazu findet sich mehr in den nächsten Abschnitten.

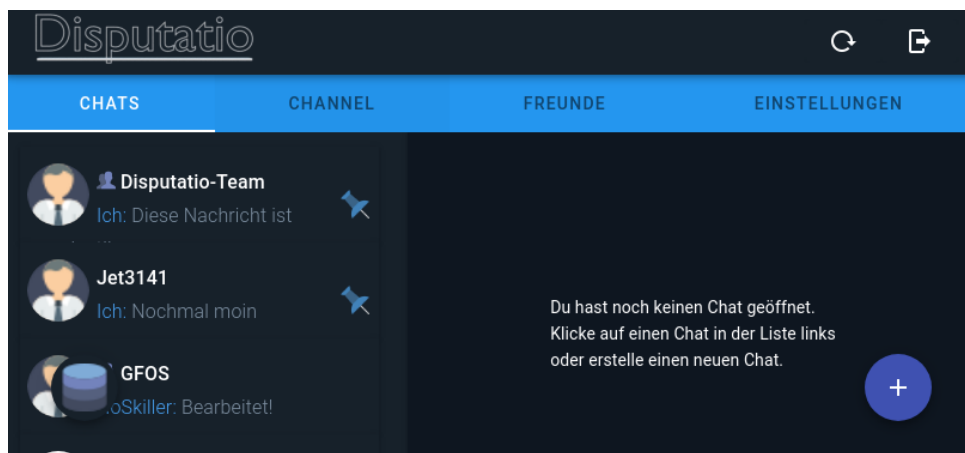


Abbildung 23: Die mobile Ansicht der Chatliste

2.1.1. Chatinfo öffnen



Abbildung 24: Die Ansicht der Chat-Info für Einzelchats (dieses Fenster ist bei der mobilen Version wie bei der Desktop-Version)

Durch einen Rechtsklick auf das Profilbild eines Chats in der Liste wird die Nutzerinfo angezeigt. Diese zeigt das Profilbild, den Benutzernamen und die E-Mail.

Sofern weitere Daten vorhanden sind, sind diese auch sichtbar. Gleiches gilt für die Beschreibung oder Info, die bei der Registrierung eingegeben und unter „Einstellungen“ geändert werden kann.

Auf dem roten Button „Blockieren“ lässt sich der Nutzer blockieren. Blockierte Nutzer können Ihnen keine Nachrichten mehr senden und können Ihre Daten in einer solchen Chatinfo nicht einsehen.

Mit einem Klick auf „Löschen“ kann man den Chat mit dem Nutzer löschen. Es erscheint ein Bestätigung-Dialog, sodass ein Chat nicht aus Versehen gelöscht wird, da dies eine endgültige Aktion ist.



Abbildung 25: Die Ansicht einer Gruppen-Chatinfo (diese Ansicht ist bei Mobil- und Desktop-Version gleich)

Mit einem Rechtsklick auf das Gruppenbild einer Gruppe, wird die Gruppeninfo geöffnet. Diese ähnelt der einfachen Chatinfo. Angezeigt wird das Profilbild der Gruppe, der Gruppenname und eine Beschreibung, sofern diese vorhanden ist. Darunter finden sich die Mitglieder der Gruppe. Sollten diese den Status eines „Gruppen-Administrators“ haben, wird dies mit „Admin“ (siehe Simon) angezeigt.

Darunter sind die Aktionen, die dem Nutzer zur Verfügung stehen. Über den Button „Verlassen“ kann die Gruppe verlassen werden. Man erhält damit keine Nachrichten aus der Gruppe mehr und diese wird aus der Chatliste entfernt.

Sofern man ein Gruppen-Administrator ist, wird neben dem Button das grüne Plus im Kreis angezeigt. Wenn man daraufklickt, ist es möglich, weitere Nutzer der Gruppe hinzuzufügen. Hinzugefügt werden kann auch, wer die Gruppe schon mal verlassen hat. Blockierte Nutzer können nicht hinzugefügt werden.

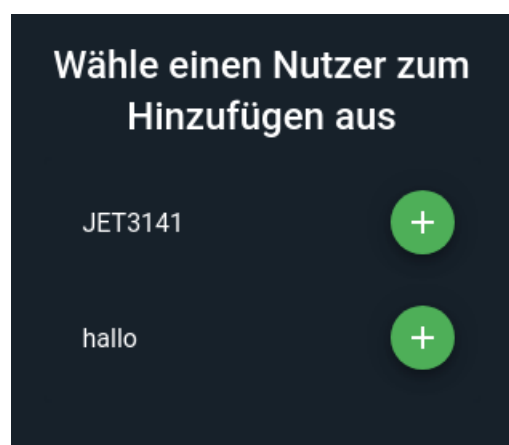


Abbildung 26: Die Ansicht des Gruppen-Administrators beim Hinzufügen eines neuen Gruppenmitglieds

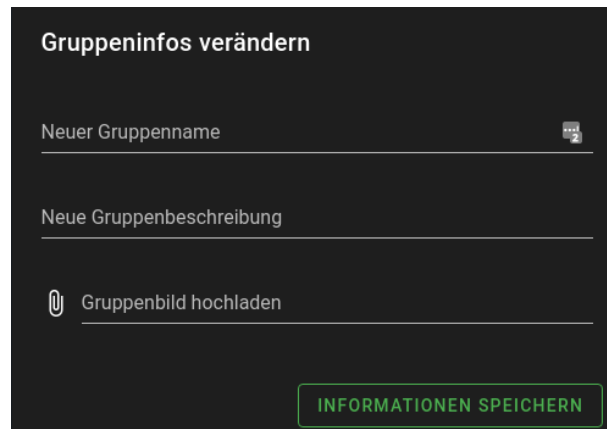


Abbildung 27: Die Karte zur Änderung der Gruppeninformationen (diese Ansicht ist in Mobil- wie Desktop-Version gleich)

Wenn der Nutzer selber ein Admin der Gruppe ist, so kann er in der Gruppeninfo mit einem Rechtsklick auf das Profilbild diese Karte öffnen. Hier lassen sich die Beschreibungen der Gruppe, sowie der Gruppenname und das Profilbild ändern.

Die Abbildung stellt diesen beschriebenen Vorgang dar. Die Karte, die sich öffnet, ist in der Mobil-Version wie in der Desktop-Version (Abbildung 13) gleich. Mit einem Klick auf das weiße Plus im grünen Kreis wird der entsprechende Nutzer hinzugefügt. Verwirrend kann hier sein, dass „JET3141“ angezeigt wird, obwohl ein Nutzer mit diesem Benutzernamen im Chat bereits existiert. Das liegt daran, dass die richtige Groß- und Kleinschreibung zu beachten ist, weshalb es zwei unterschiedliche Nutzer gibt: Der eine heißt „JET3141“, der andere „Jet3141“.

2.1.2. Pin/Archivieren Menü öffnen



Abbildung 28: Darstellung des Chatmenüs (diese Ansicht ist bei Mobil- und Desktop-Version gleich)

Zu sehen ist das Chatmenü. Dieses erreicht man aus der Chatliste, wenn man einen Rechtsklick auf den Namen des Chats oder der Gruppe durchgeföhrt. Das Menü stellt zwei Funktionen für den Chat zur Verfügung.

Unter „Entpinnen“ kann ein bereits angepinnter Nutzer gelöst werden. Dieser Nutzer würde sich dann wieder systematisch nach der Reihenfolge der letzten Nachrichten unter den anderen Chats und Gruppen einordnen. Wenn der Nutzer noch nicht angepinnt ist, lautet diese Funktion „Anpinnen“ und hat den genau gegenteiligen Effekt zum „Entpinnen“.

Die Funktion „Archivieren“ stellt die Möglichkeit dar, einen Chat zur archivieren. Nähere Informationen finden sich unter Punkt 2.4.

2.2. Das Chatfenster

Der Chat-Tab beinhaltet neben der ChatListe (links) auch den jeweils aktuellen Chat (rechts). Wenn anfangs noch kein Chat offen ist, wird ein Platzhalter angezeigt.

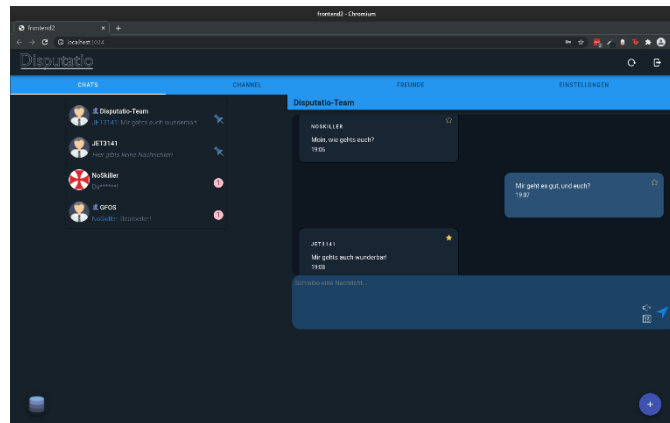


Abbildung 29: Das gesamte Chatfenster der Disputatio-Browser-Anwendung in Desktop-Version

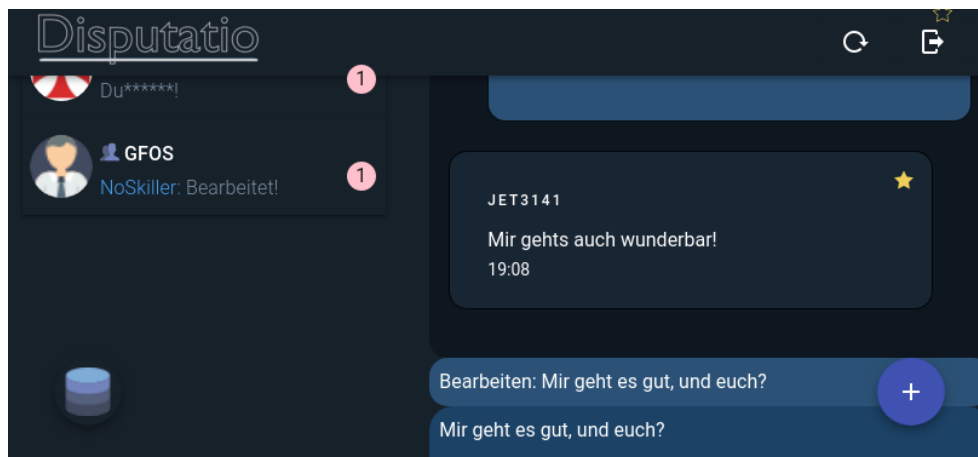


Abbildung 30: Das Chatfenster in der mobilen Version

In der mobilen Variante ist der Aufbau im Querformat gleich. Die „Floating Action Buttons (FABs)“ um archivierte Chats anzuzeigen und Gruppen zu erstellen werden unten in den Ecken angezeigt.

2.2.1. Nachrichten senden (geplant/wichtig)



Abbildung 31: Das Eingabe-Feld für Nachrichten

Beim Senden von Nachrichten haben Nutzer verschiedene Möglichkeiten: Die Nachricht kann erstens ganz normal gesendet werden. An besonderen Möglichkeiten kann die Nachricht entweder als laute Nachricht gesendet werden, was mit einem Klick auf den Lautsprecher erreicht werden kann, oder als geplante Nachricht. Dann kann das Sendedatum der Nachricht beliebig in die Zukunft verschoben werden (siehe Abbildung 32). Die Nachricht ist nur für den Sender selbst sichtbar.

2.2.2. Bearbeiten/Löschen/Antworten

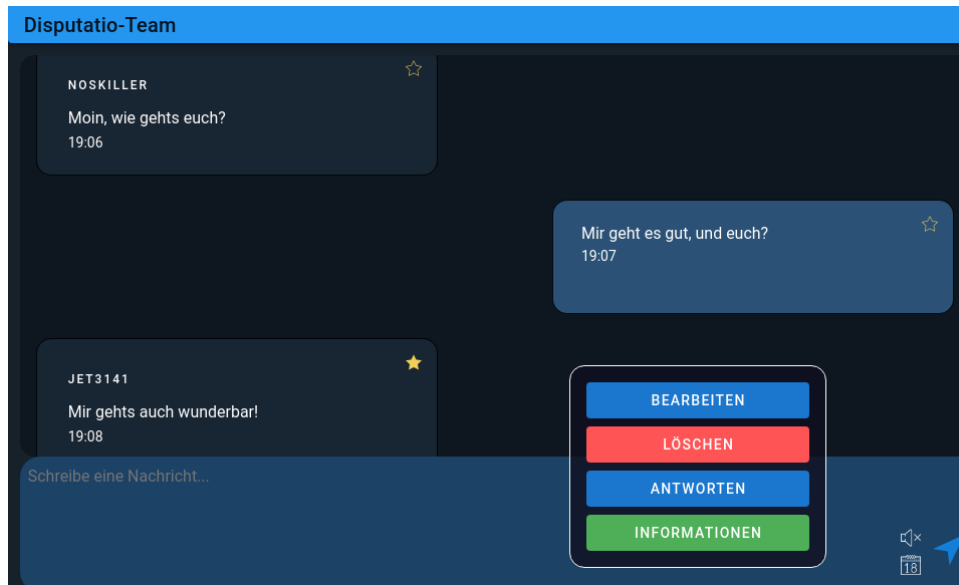


Abbildung 32: Die Chatoptionen

Mit einer Nachricht kann auf verschiedene Arten interagiert werden. Das Menü zum Auswählen aus diesen Möglichkeiten öffnet sich mit einem Rechtsklick auf eine eigene Nachricht. Die Nachricht kann entweder gelöscht oder bearbeitet werden, oder es kann eine Antwort auf sie gesendet werden. In den letzten beiden Fällen wird an das Textfeld oben ein weiterer Block angehängt, wie in der folgenden Abbildung zu erkennen ist. Beim Bearbeiten der Nachricht wird der vorherige Nachrichtentext ins Textfeld geladen, um das Bearbeiten zu vereinfachen.



Abbildung 32: Die "Bearbeiten"-Funktion

Nachrichteninfos

Im Nachrichtenmenü aus 2.2.2 befindet sich auch die Möglichkeit, sich Informationen über eine Nachricht anzeigen zu lassen. Das beinhaltet, wer die Nachricht gelesen hat und wer nicht. Wenn ein Nutzer eine Nachricht gelesen hat, wird das durch ein kleines Auge gekennzeichnet.

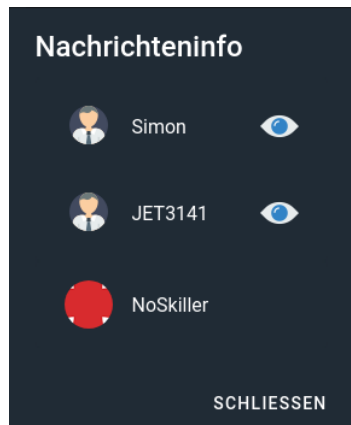


Abbildung 33

2.2.3. Nachrichten markieren

Wenn einem Nutzer eine Nachricht besonders gut gefällt oder der Nutzer sich die Nachricht für später speichern möchte, kann das mit einem Klick auf den kleinen Stern oben rechts in jeder Nachricht geschehen. Ist die Nachricht von einem selbst markiert worden, wird das durch einen ausgefüllten Stern deutlich. Die markierten Nachrichten können in den Einstellungen eingesehen werden.

2.3. neue Gruppen erstellen

Abbildung 34: Ansicht der "eine neue Gruppe erstellen"-Karte (diese Ansicht ist in Mobil- und Desktop-Version genau gleich)

Generell findet sich auf der Website in der unteren rechten Ecke immer ein lilafarbener Button, auf dem ein Benutzer-Icon zu sehen ist. Beim Klick auf diesen Button gelangt man in der Desktop- sowie in der Mobil-Version auf die Karte, die in Abbildung 15 dargestellt ist. Auf dieser Karte ist es möglich, eine neue Gruppe zu erstellen.

Für eine Gruppe muss ein Gruppenname festgelegt werden, eine Gruppenbeschreibung ist nicht erforderlich.

Aus der Freundesliste können nun Teilnehmer der Gruppe mit einem Linksklick entsprechend ausgewählt werden. Eine Gruppe muss mindestens einen anderen Nutzer enthalten, kann aber

unbegrenzt groß sein. Wird die Mindestanforderung nicht erfüllt, bleibt der Button „Absenden“ grau und kann nicht angeklickt werden.

2.4. Archivierte Chats



Abbildung 35: Icon-Ansicht der Nicht-archivierten Chats

Archivierte Chats werden gesondert angezeigt. Wie der Button zum Erstellen von neuen Gruppen, so findet sich am unteren linken Ende der Seite ein Button mit abgebildetem Icon. Beim Klick auf diesen Button wird die Ansicht die jeweils auf andere Chatliste gewechselt. Das Icon ändert sich zur Kennzeichnung (vgl. Abbildung 16 bzw. 17). Dieser Button bleibt in der Mobilversion genau wie in der Desktop-Version erhalten.



Abbildung 36: Icon Ansicht der archivierten Chats

3. Channel

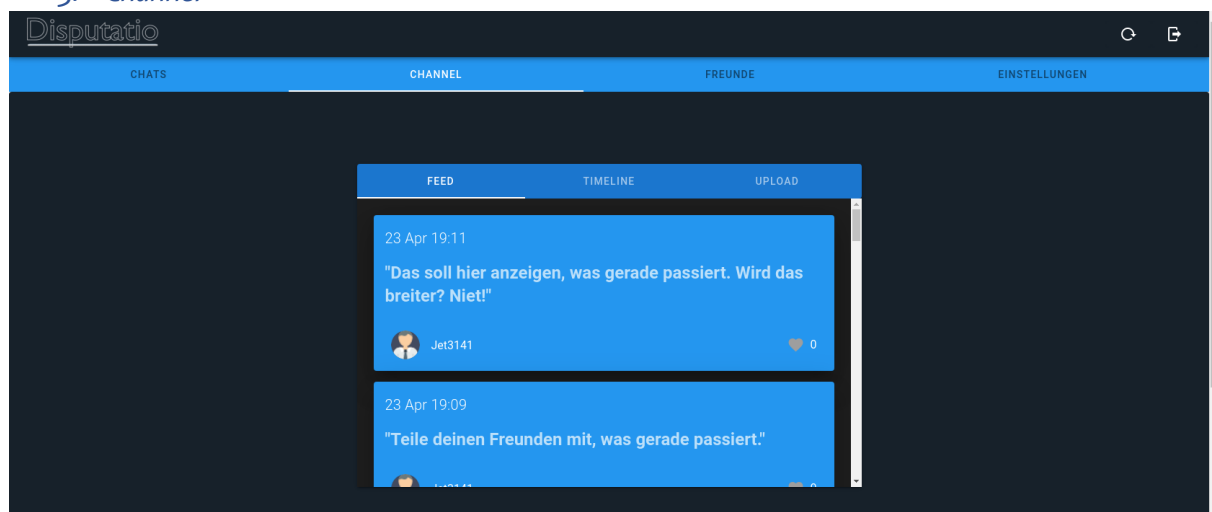


Abbildung 37: Die Desktop-Ansicht des kompletten Channel-Reiters

Der zweite Reiter in der Tab-Leiste am oberen Rand der Website führt den Nutzer zur Funktion „Channel“.

3.1. Grundsätzliche Erklärung

Ein Channel ist die Möglichkeit jedes Nutzers, zu gewissen Sachverhalten Stellung zu beziehen. Diese Stellungnahmen können so lang sein, wie der Nutzer möchte, sind allerdings eher als Kurzmitteilung gedacht.

3.2.Feed

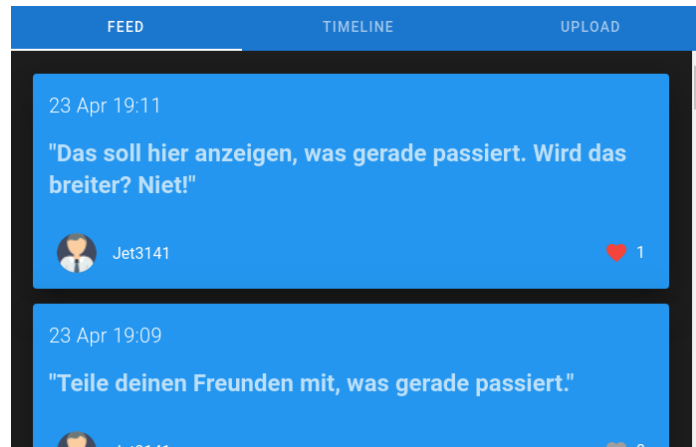


Abbildung 38: Die Desktop-Ansicht des Feeds

Der Channel stellt wieder eine Karte mit drei Reitern dar. Der erste dieser Reiter ist der „Feed“, der die neusten Channel-Nachrichten der befreundeten Nutzer anzeigt. Dieser Bereich kann gescrollt werden, wodurch man absteigend zu den ältesten Channel-Mitteilungen der anderen Nutzer kommt.

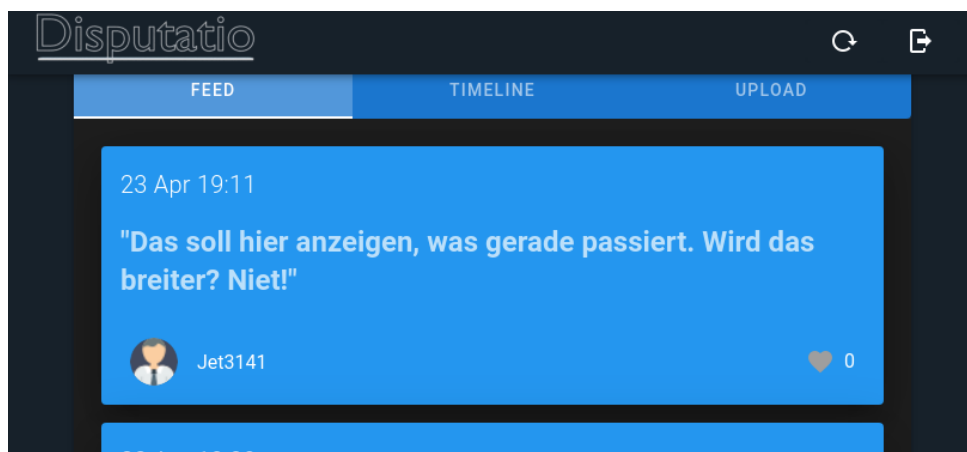


Abbildung 39: Die mobile Ansicht des Feeds

3.3.2 Nachrichten liken

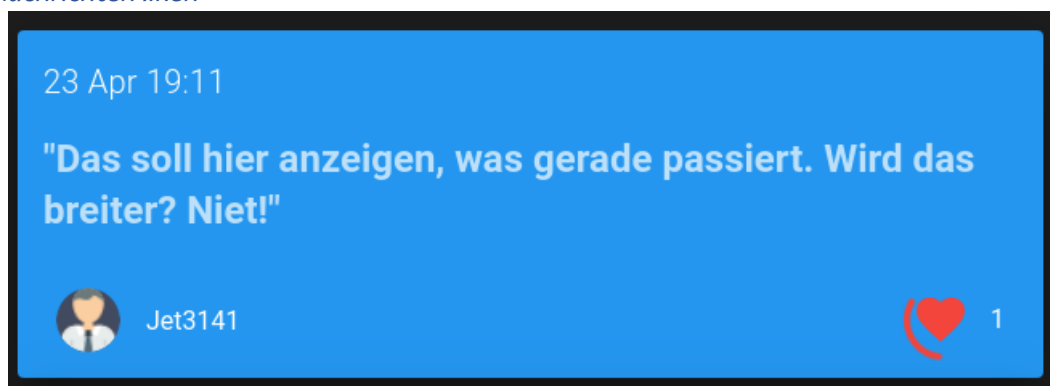


Abbildung 40: Darstellung der Like-Animation

Channel-Nachrichten können von allen Freunden des Nutzers/ der Nutzerin eingesehen, sowie durch das Herz in der unteren rechten Ecke mit einem „gefällt mir“ kommentiert werden.

Wenn man auf das Herz klickt, wird die Nachricht geliket. Die Übermittlungsdauer vom Frontend zum Server wird durch eine Animation (siehe Abbildung 20) überbrückt. Das Herz wird für den Nutzer, der die Channel-Nachricht ansieht, nach dem liken rot.

Rechts neben dem Herz wird die Anzahl der bereits erteilten Likes anderer Nutzer angezeigt.

3.3. Timeline

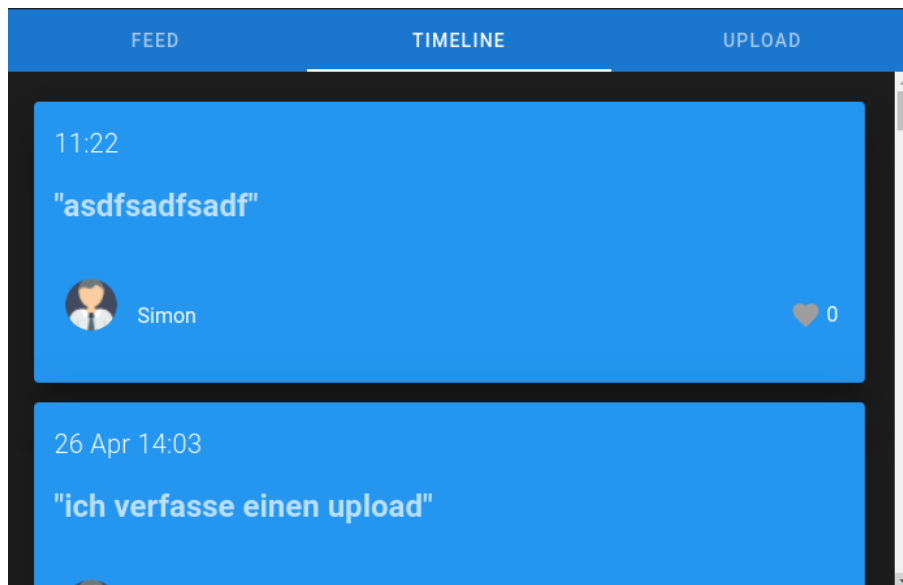


Abbildung 41: Darstellung der Desktop-Ansicht des Timeline-Tabs

Unter dem Reiter „Timeline“ finden sich die eigenen Channel-Nachrichten in gleicher Reihenfolge wie die Nachrichten unter dem Reiter „Feed“. Die Anzahl der Likes kann durch die Zahl entnommen werden, eigene Nachrichten können allerdings nicht geliket werden.

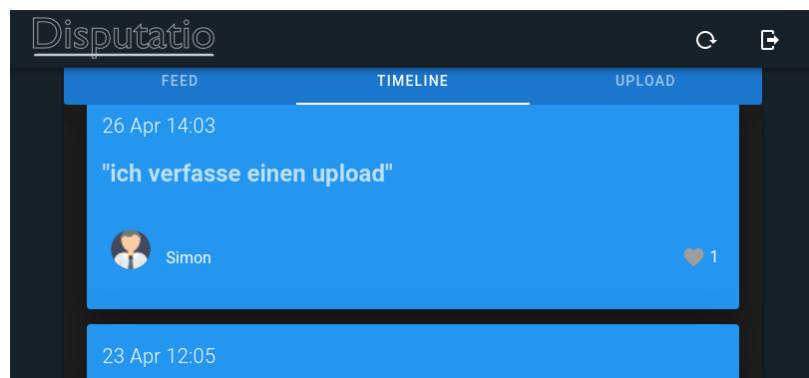


Abbildung 42: Darstellung der Mobil-Ansicht des Timeline-Tabs

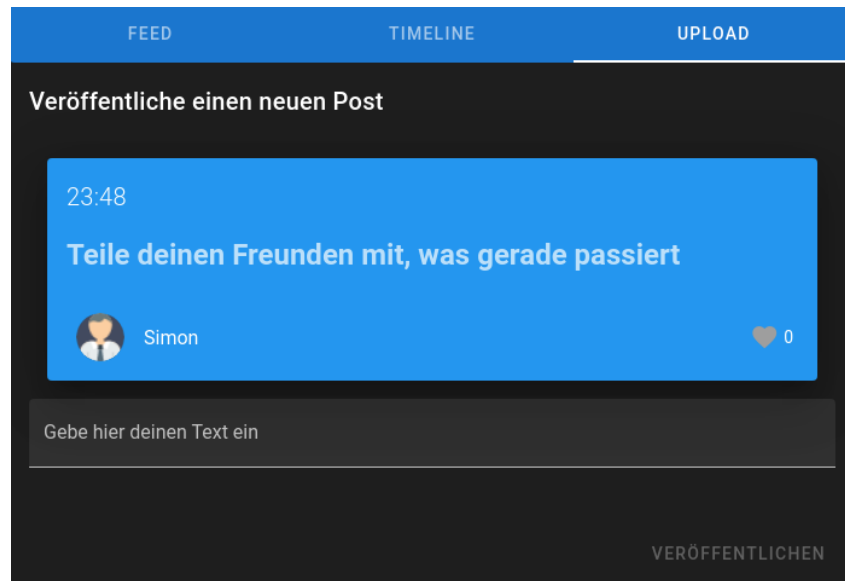


Abbildung 43: Darstellung der Desktop-Ansicht des Upload-Reiters

Unter dem Reiter „Upload“ kann eine neue Channel-Nachricht erstellt werden. Hierzu muss unter „Gebe hier deinen Text ein“ der Inhalt der Nachricht eingetragen werden. Darüber werden die Änderungen direkt synchronisiert und der Nutzer erhält eine Vorschau des späteren Posts. Sobald etwas eingetragen wurde, ist es möglich, über den Button „Veröffentlichen“ seine Channel-Nachricht zu senden.

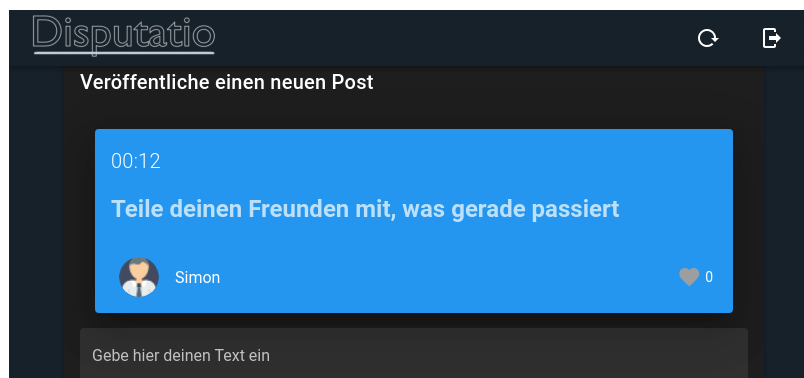


Abbildung 44: Darstellung der Mobil-Ansicht des Upload-Reiters

4. Freunde

Damit die Nutzer ihre Kontakte verwalten und mit Ihnen in Kontakt bleiben können, gibt es den Reiter „Freunde“. Dieser ist wieder in verschiedene Teile untergliedert.

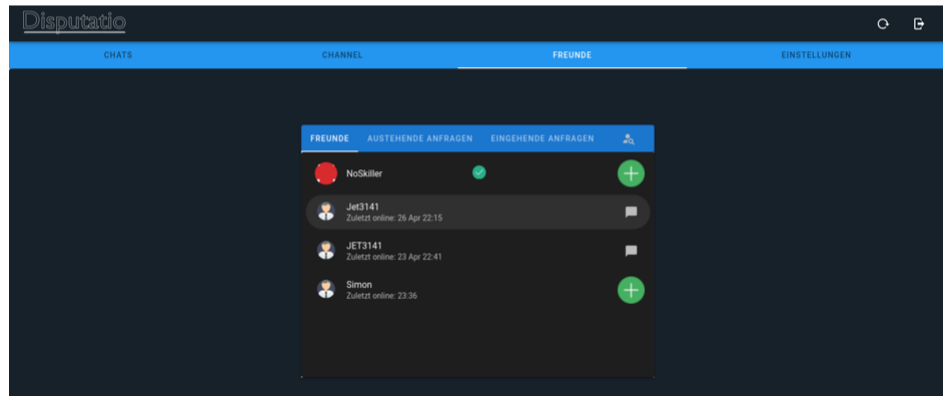


Abbildung 45: Desktopansicht der Freunde

4.1 Freundesliste

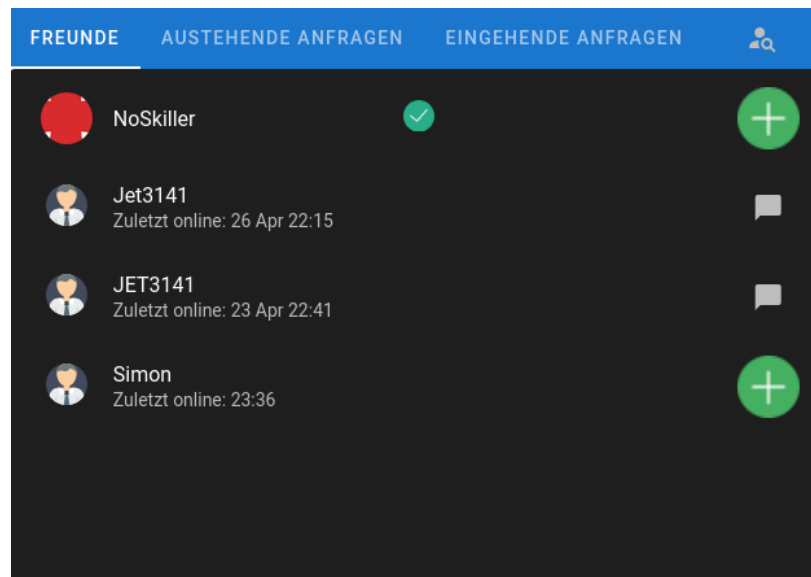


Abbildung 46: Die Freundesliste bereits befreundeter Personen (diese Ansicht ist bei der mobilen Version genau gleich)

In der Freundesliste werden alle Freunde des Nutzers aufgelistet. Wenn diese online sind, ist das durch das kleine grüne Häkchen in der Mitte gekennzeichnet, ist der Nutzer offline, wird das Datum, zu dem der Nutzer zuletzt online war, angegeben. Jeweils rechts eines Freundes ist ein Knopf, mit dem eine Aktion durchgeführt werden kann. Wenn man selbst und der Freund schon einen Chat haben, kann durch einen Klick auf das graue Icon der Chat geöffnet werden, existiert dieser noch nicht, kann der Chat hier erstellt werden.

4.2 Ausstehende Anfragen

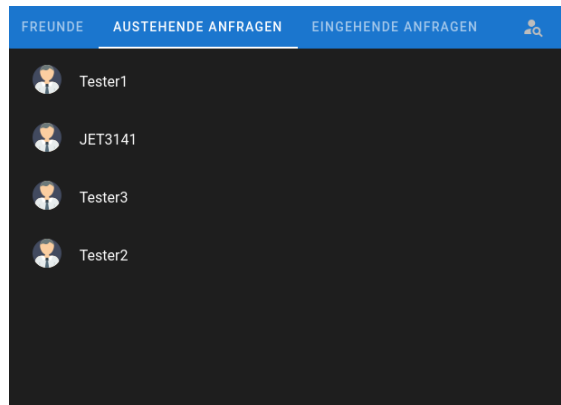


Abbildung 47: Die Karte der ausstehende Freundschaftsanfragen (diese Ansicht ist bei der mobilen Version genau gleich)

Hier werden die Freundschaftsanfragen aufgelistet, auf deren Antwort gewartet wird.

4.3 Eingehende Anfragen

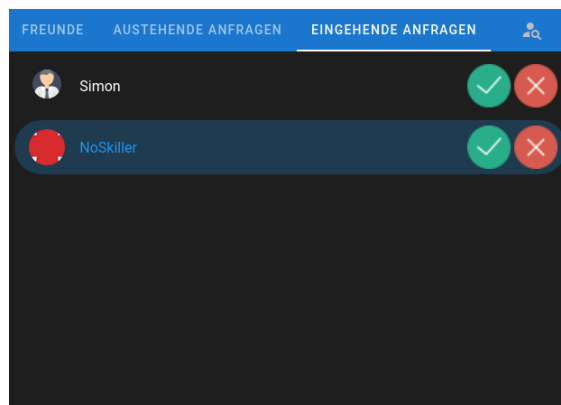


Abbildung 48: Die Ansicht der eingehenden Freundschaftsanfragen (diese Ansicht ist bei der mobilen Version genau gleich)

Hier werden die Freundschaftsanfragen aufgelistet, die an den Nutzer selbst gerichtet sind. Sie können angenommen oder abgelehnt werden.

4.4 Freunde hinzufügen

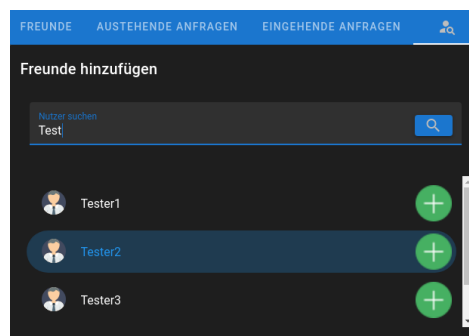


Abbildung 49: Die Möglichkeit, Freunde hinzuzufügen (diese Ansicht ist bei der mobilen Version genau gleich)

Hier können Freunde hinzugefügt und nach Nutzern gesucht werden. Dafür ist lediglich die Eintragung des Namens sowie ein Klick auf die Lupe nötig. Mit dem grünen Plusknopf kann eine Freundschaftsanfrage verschickt werden.

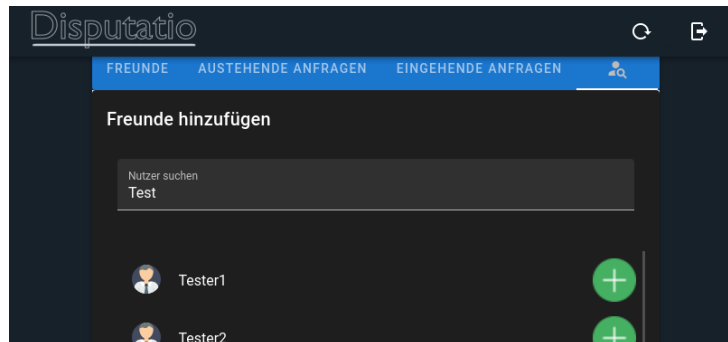


Abbildung 50: Die mobile Ansicht zum Hinzufügen von Freunden

5. Einstellungen

Hier finden sich Einstellungen, die der Nutzer vornehmen kann und Übersichten über seine Daten.

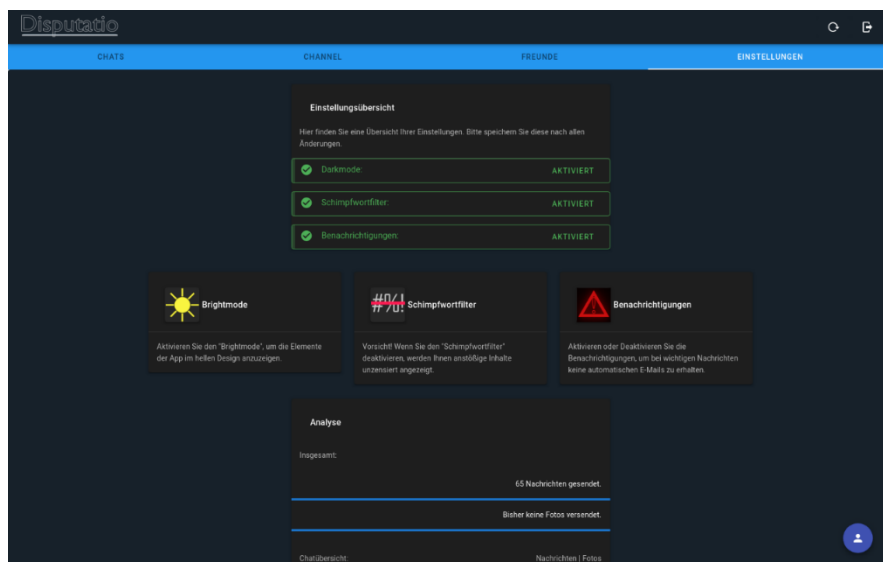


Abbildung 51: Der Einstellungs-Reiter in der Desktop-Ansicht (Teil 1)

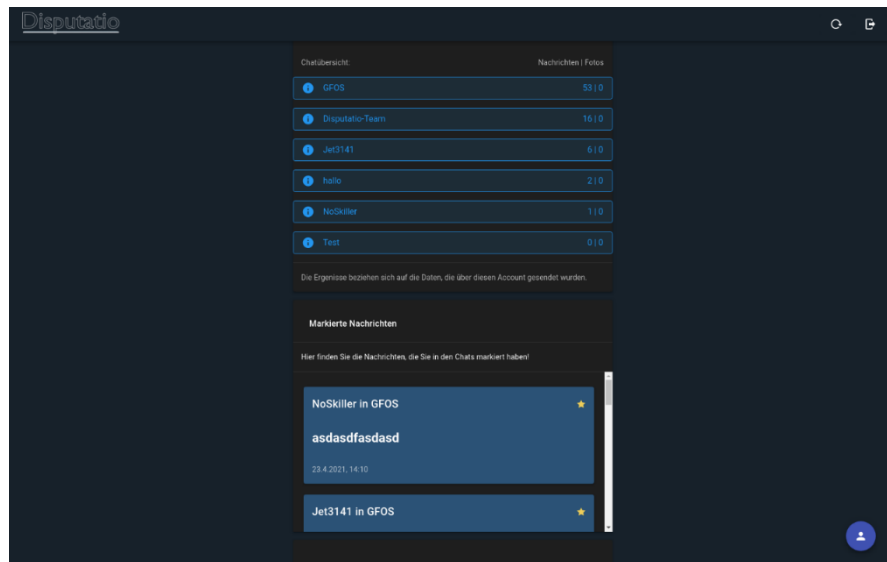


Abbildung 52: Der Einstellungs-Reiter in der Desktop-Ansicht (Teil 2)

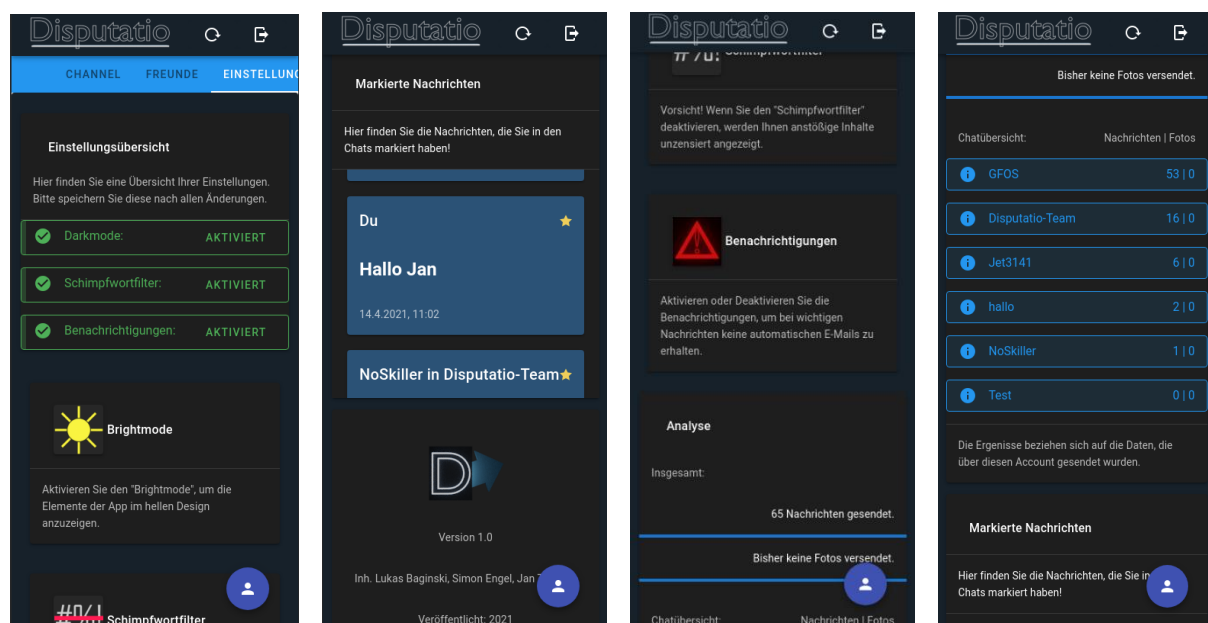


Abbildung 53: Die Einstellungsseite in der mobilen Version (Scrollaufnahme)

5.1 Einstellungsmöglichkeiten

Es gibt drei Haupteinstellungsmöglichkeiten, die ein Nutzer vornehmen kann. Zuerst kann er sich für einen Dark- oder Brightmode entscheiden, der die komplette Anwendung in ihrem Farbdesign ändert.

Außerdem kann ein Schimpfwortfilter aktiviert werden, der unangemessene Wörter herausfiltert, um das Nutzererlebnis zu verbessern. Abschließend gibt es die Möglichkeit, den Erhalt von E-Mails bei wichtigen Nachrichten zu deaktivieren.

5.2 Analyse

Nutzer haben die Möglichkeit, eine Analyse über die Daten, die von ihrem Account aus versendet wurden, durchführen zu lassen. Dafür muss auf „Analyse Starten“ geklickt werden. Daraufhin klappt sich ein Fenster auf, in dem die Daten eingesehen werden können. Am oberen Ende wird die Summe der selbst gesendeten Nachrichten und Fotos angezeigt, darunter werden diese Informationen pro Chat bzw. Gruppe dargestellt.

5.3 Markierte Nachrichten

Hier werden die Nachrichten, die ein Nutzer mit einem Stern markiert hat, angezeigt. Sollte diese Markierung nicht mehr gewünscht sein, kann sie mit einem Klick auf den kleinen Stern rechts oben beendet werden.

5.4 Account ändern

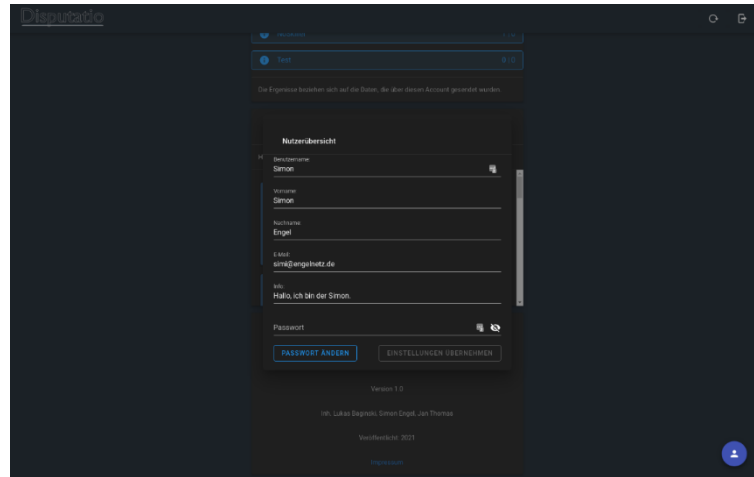


Abbildung 54: Die Möglichkeit, die Nutzerdaten zu ändern in der Desktop-Version

Die Accountinformationen eines Nutzers können hier angezeigt bzw. verändert werden. Um zu dieser Karte zu gelangen, genügt ein Klick auf den Knopf rechts unten in der Einstellungsansicht.

Um die Informationen zu verändern, können diese einfach neu in eines der Felder eingetragen werden. Zum Bestätigen muss noch das Passwort eingegeben werden, woraufhin der Knopf zum Speichern grün wird.

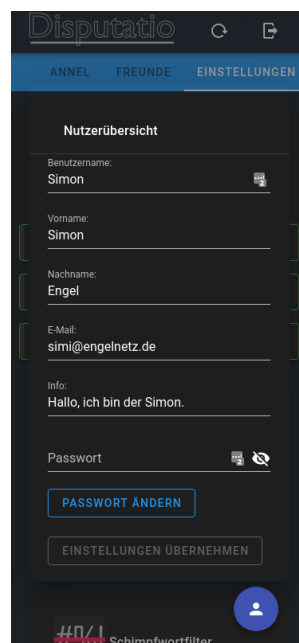


Abbildung 55: Die Möglichkeit, die Nutzereinstellungen einzusehen, in der mobilen Ansicht

Installationsanleitung

Für eine lauffähige Anwendung ist das Programm „Netbeans“ in der Version 12.0 LTS erforderlich, wobei es sich um eine integrierte Entwicklungsumgebung handelt. Die verwendete Java-Version ist

Java 8 OpenJDK. Bei der Datenbank haben wir uns für die in Netbeans eingebaute Java-Datenbank entschieden. Die dazugehörigen Dateien sind im Ordner „Datenbank“ zu finden.

Nach der Installation von Netbeans muss die Installation des GlassFish-Servers durchgeführt werden. Diese kann unter dem Reiter Services/Servers mit einem Rechtsklick und „Add Server“ durchgeführt werden. Wir haben die Version 5.1.0 verwendet, bei der Installation können alle Voreinstellungen beibehalten werden.

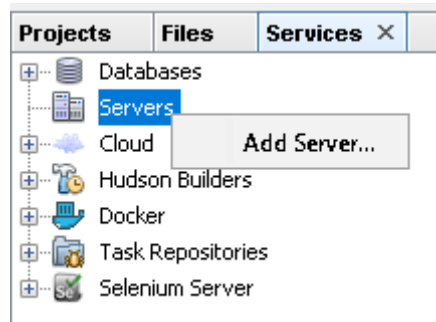


Abbildung 56

Nun muss die Datenbank importiert werden. Dafür Windows + R drücken und %AppData%\Netbeans\Derby“ eingeben.

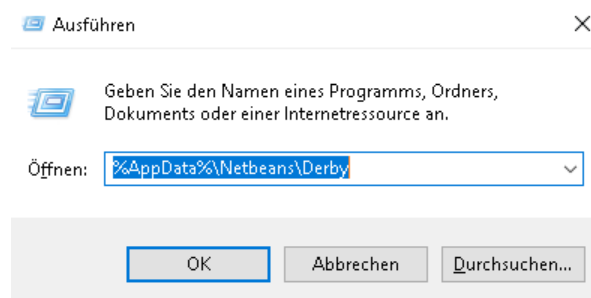


Abbildung 57

In diesen Ordner muss der Ordner db aus dem Ordner Datenbank verschoben werden, woraufhin NetBeans einen Neustart erfordert.

Unter dem Reiter Services/JavaDB/ muss dann nur noch einmal die Verbindung zur Datenbank „db“ hergestellt werden. Dafür muss ein Rechtsklick auf „db“ getätigt werden und die Option „Connect“ ausgewählt werden. Im dann erscheinenden Fenster müssen die Daten eingegeben werden: Der Nutzernamen und das Passwort sind beide „root“. Hier muss noch die Funktion „Remember password“ ausgewählt werden.

Wenn Netbeans richtig installiert und die Datenbank verbunden ist, kann das Projekt unter Datei/Projekt öffnen geöffnet werden. Das zu öffnende Projekt befindet sich im Ordner „Backend“.

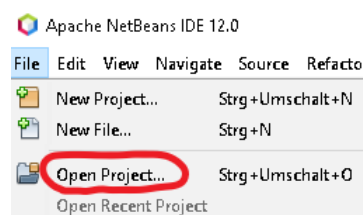


Abbildung 58

Mit einem Klick auf den grünen Play-Button kann das ganze Projekt jetzt gestartet werden. Die komplette Webanwendung ist dann unter der Adresse localhost:8080/GFOS zu erreichen.



Abbildung 59

Zusammenfassung/Fazit

In diesem Kapitel wollen wir auf unsere Bearbeitungen zurückblicken und diese mit den anfänglichen Planungen und Erwartungen an das Projekt reflektieren.

Datenbank

Die Datenbank konnten wir so aufsetzen, dass sie alle Nutzerdaten sinnvoll und möglichst effizient speichert. In sofern sind wir auch mit der Datenbank zufrieden, weil sie die Anwendung gut unterstützt.

Backend

Mit Blick auf den Server lässt sich sagen, dass wir mit diesem sehr zufrieden sind. Während der Bearbeitungszeit traten an dieser Stelle zwar die meisten Probleme auf, allerdings ist das auch natürlich- der Server ist schließlich das Herzstück der Anwendung.

Da der Server nun allerdings alle Funktionen im Frontend entsprechend hinterlegt und die Brücke zwischen Frontend und Datenbank baut, können wir uns mit diesem zufrieden geben.

Frontend

Von den Seiten des Frontends haben wir unsere Planungen ebenfalls erfüllen können.

Die Basis-Elemente des Frontends sollten dazu dienen, dem Nutzer eine Oberfläche zur Kommunikation mit dem Backend bereitzustellen.

Um das gewährleisten zu können, müssen die Funktionen des Frontends mit denen des Backends Deckungsgleich sein.

Darüber hinaus war es uns wichtig, einen Messenger zu schaffen, der ein „Gleichgewicht zwischen einem hohen Funktionsumfang und gleichzeitig einer einfachen und intuitiven Bedienung der Anwendung“ (Kapitel: „Planungen des Projekts“; Idee) darstellt. Mit Blick auf diese von uns selbst gestellte Anforderung können wir sagen, dass wir ihr gerecht geworden sind. Durch die „UI-components Designgewohnheiten modernen den App unsere wir konnten vuetify.js von ¹ anpassen. Darüber hinaus halfen uns Icons, Farbe und Charakter in die App zu bringen und den Nutzern eine Alternative zum Text zu geben.

Bemerkungen

In unserer Dokumentation werden einige Schaubilder dargestellt. Diese haben wir selber kreiert, weshalb sich keine Angabe zu Quellen o.Ä. findet. Ähnlich verhält es sich mit den Informationen zu Frontend und Design. Diese stammen von den Seiten vue.js (<https://vuejs.org/> 2021-04-29 19:42) und vuetify.js (<https://vuetifyjs.com/en/> 2021-04-29 19:42), wo die Programmiersprachen vorgestellt werden.

¹UI-Components oder UI-Elements: User Interface Elements, Elemente der App, die die Eingabemöglichkeit für den Nutzer darstellen.

Anhang

Der Anhang umfasst Dokumente, die zum Kontext des Projekts gehören oder die Dokumentation ergänzen.

Aufgabenstellung

Technisch:

- Java Enterprise Edition
- Application Server (Wildfly/Glassfish)
- JavaScript|HTML|CSS
- JavaScript-Frameworks zur Visualisierung (Bootstrap, Angular, React, vue)
- MySQL oder Java DB

Leistungsumfang:

- Umsetzung der Anforderungen mit den genannten Entwicklungsumgebungen in Java
- Zum Verständnis des Quellcodes sollte dieser gängige Stilrichtlinien berücksichtigen und – wo sinnvoll- kommentiert sein.
- Erstellung einer Dokumentation zur Realisierung und Installation der Anwendung: Dazu gehört mindestens eine Beschreibung der Anwendungsarchitektur inklusive einer kurzen Beschreibung der Klassen und Skripte sowie eine Installationsanleitung zur Inbetriebnahme der Anwendungen. Der geforderte Umfang der Dokumentation beträgt 25-35 Seiten.
- Screenshots aller Masken, jeweils in Desktop- und Smartphone-Ansicht
- Präsentation der Ergebnisse in Form einer Demonstration der lauffähigen Anwendung in Form eines Screencasts.

Design und Zusatzfunktionen:

Das Oberflächendesign der Anwendung ist frei wählbar, sollte sich jedoch an der Gestaltung aktueller Software orientieren.

Hinweis:

Die oben beschriebenen Funktionen stellen die Minimalanforderung dar. Lasse Deiner Fantasie bei der Realisierung des Projektes freien Lauf. Gut durchdachte und umgesetzte Zusatzfunktionen können bei der Bewertung Deines Projekts relevante Unterschiede machen.

Nutzungsbedingungen der App

Um unserer App den nötigen professionellen „Touch“ zu geben, fertigten wir ebenfalls eine eigene Sammlung an Nutzungsbedingungen und eine Hilfe an. Verhaltensregeln werden festgelegt und Bedingungen zur Nutzung der App werden festgelegt und den Nutzern wird eine Möglichkeit gegeben, wie sie sich bei Problemen oder Fragen erkundigen können.

Beginnen möchten wir an dieser Stelle mit den Nutzungsbedingungen, die wir selber erdacht haben. Uns war wichtig, den Nutzern der App sowohl Zusicherung hinsichtlich nicht anfallender Kosten und Datenschutz zu geben, wie auch einen Grundsatz an Verhaltensregeln im Programm festzulegen. Diese sollen nämlich das „Miteinander“ der

Nutzer gewährleisten und- sofern der Messenger je veröffentlicht werden sollte- schlechte Nachrichten hinlänglich Cybermobbing oder Ähnlichem vermeiden.

Die folgenden Bedingungen sind bei der Benutzung des Messenger-Dienstes “Disputatio” einzuhalten. Sollten diese Regelungen missachtet werden, kann es zum Ausschluss der Benutzerin oder des Benutzers von der Plattform kommen. Die Bedingungen sind:

§1:

Allgemeine Bestimmungen

1 Anmeldebedingungen

- a Jeder Account, welcher in dem Messenger-Dienst “Disputatio” angelegt wird, hat die angegebenen Pflichtfelder zu erfüllen. Diese Pflichtfelder sind: “Benutzername, Passwort und E-Mail-Adresse”. Andere Daten, die im Zuge der Registrierung angegeben werden, werden gemäß der Datenschutzerklärung (siehe Datenschutzerklärung) zur Verbesserung der Dienste verwendet, sind aber nicht angabepflichtig.
- b Sofern über die Pflichtfelder hinaus Angaben getätigt werden, müssen diese der Wahrheit entsprechen und nach bestem Wissen und Gewissen ausgefüllt werden. Verstöße gegen diese Bedingung werden nach §2 a geahndet und können- bei weit reichenden Folgen- weitere Konsequenzen haben.
- c Jeder Mensch hat das Recht, über den Messenger-Dienst “Disputatio” so viele Accounts anzulegen, wie er möchte. Dabei wird allerdings darauf hingewiesen, dass bei einer sehr hohen Anzahl von Accounts pro Person die persönliche Organisation verkompliziert werden könnte.
- d Der Messenger-Dienst “Disputatio” ist kostenlos. Durch die Nutzung von Privatpersonen oder Unternehmen sollen keine Kosten entstehen. Darüber hinaus ist auf der gesamten Plattform des Messenger-Dienstes der gewerbliche Handel mit allen Waren, Dienstleistungen oder Ressourcen verboten.

2 Ausschluss

- a Sollten Nutzer auf dem Messenger-Dienst sich nicht nach Ordnungen der allgemeinen Nutzungsbedingungen verhalten oder durch sonstige Verhaltensweisen oder Begebenheiten (strafrechtliche Verfolgung etc.) auffällig werden, behält sich der Betreiber vor, die jeweiligen Personen von der Nutzung auszuschließen. In diesem Fall entfallen §1 a-c sowie alle nachfolgenden Paragraphen.
- b Der Betreiber behält sich überdies vor, den Messenger-Dienst zu jeder Zeit einzustellen. Somit besteht kein Recht auf die Aushändigung gewisser Inhalte oder die Weiterführung der Dienstleistung. Zur Erklärung wird besonders wird auf §1 c verwiesen.

§2:

Umgang mit Chatverläufen

1 Inhalt von Chatverläufen

- 1.a Der Inhalt der Chatverläufe im Messenger-Dienst “Disputatio” sind privat. Sie sind nur von den Personen, die an der jeweiligen Chatsituation beteiligt sind, zu lesen. Diese Funktion wird zum einen durch den Betreiber sichergestellt, muss aber von den Benutzern ebenso beachtet werden. Ausnahmen zu dieser Regelung finden sich in Absatz
- 1.b Jeder Nutzer hat sich im Messenger-Dienst “Disputatio” gemäß gesellschaftlicher Konventionen zu verhalten. Dabei ist darauf zu achten, dass beim Verschicken

von Textnachrichten, Bildern, Memes oder Stickern die Freiheit und Würde anderer Chatteilnehmer nicht begrenzt wird. Dieser Umstand wird aufgrund von Absatz 1 vom Betreiber nicht kontrolliert.

- 1.c Beide vorangegangenen Absätze entfallen, sofern durch oder gegen die Inhalte der Chats oder dessen Teilnehmer strafrechtlich ermittelt wird. In diesem Falle behält sich der Betreiber vor, die Inhalte sowie die eingetragenen Nutzerdaten den örtlichen Behörden zu übergeben. Dies wird allerdings nur durch gerichtliche Anordnung wirksam.
- 1.d Im Allgemeinen wird für die Benutzung von “Disputatio” darum gebeten, einen friedvollen und respektvollen Umgang mit anderen Nutzern zu pflegen.
- 2 Versendung von relevanten Inhalten
 - a Der Betreiber des Messenger-Dienstes “Disputatio” übernimmt für den Inhalt von Textnachrichten sowie für den Inhalt von Bildnachrichten (siehe §2, 3) keine Haftung.
 - b Sollte es zum begründeten Verdacht kommen, dass die Inhalte der Chats einer Person strafrechtlich relevant sein könnten, behält sich der Betreiber vor §2, 1a außer Kraft zu setzen. In diesem Falle können die Inhalte der Chatverläufe an Behörden der jeweiligen Länder weitergeben werden.
 - c Darüber hinaus wird für die Sicherheit der Inhalte der Textnachrichten keine Haftung übernommen. “Disputatio” schützt die Chats zwar, ist allerdings im Falle von Datenklau o.Ä. nicht für eine Entschädigung heranzuziehen. Bemühungen, dem Datenklau entgegenzuwirken werden vom Betreiber unterstützt.
- 3 Versendung von Bildinhalten
 - a Die Versendung von Bildinhalten ist über die Plattform “Disputatio” möglich. Das wird durch den Betreiber gewährleistet.
 - b Für den Inhalt der Bildinhalte übernimmt der Messenger keine Haftung. Sofern strafrechtlich relevante Inhalte auf Bildern zu sehen sind oder ein begründeter Verdacht zu dieser Annahme besteht, ist der Messenger bereit, Nutzungsdaten von Chatverläufen und deren Inhalte weiterzugeben.
 - c Darüber hinaus wird nicht für die Sicherheit und die Anonymität der Bildinhalte gehaftet. Deshalb wird darum gebeten, weniger relevante Bilder über den Messenger zu versenden.