Yago Romero

Projekt modul 307

Inhalt

[1 Formular des Projekts 3](#_Toc10755326)

[2 Einleitung 4](#_Toc10755327)

[3 Anforderungen 4](#_Toc10755328)

[4 Schnittstellen zu anderen Systemen 4](#_Toc10755329)

[5 Architektur 5](#_Toc10755330)

[5.1 Komponenten 5](#_Toc10755331)

[6 Das Projekt 7](#_Toc10755332)

[6.1 Die Vorbereitung der Entwicklungsumgebung 7](#_Toc10755333)

[6.2 Gestaltung des Designs 10](#_Toc10755334)

[6.3 Mockup 10](#_Toc10755335)

[6.4 UML Diagramm 19](#_Toc10755336)

[7 Test 19](#_Toc10755337)

[8 Benutzerhandbuch 24](#_Toc10755338)

[8.1 Sich registrieren 24](#_Toc10755339)

[8.2 Sich einloggen 25](#_Toc10755340)

[8.3 Chatten 27](#_Toc10755341)

[8.4 Navigieren 28](#_Toc10755342)

[8.5 Chats löschen 29](#_Toc10755343)

[8.6 Profil editieren 29](#_Toc10755344)

[9 Arbeitsjournal 30](#_Toc10755345)

[10 Versionen Kontrolle 31](#_Toc10755346)

[Abbildung 1 Auftrag Formular 3](#_Toc10797484)

[Abbildung 2 Node herunterladen 8](#_Toc10797485)

[Abbildung 3 Angular installieren 8](#_Toc10797486)

[Abbildung 4 Angular Version 8](#_Toc10797487)

[Abbildung 5 Firebase Projekt erstellen 9](#_Toc10797488)

[Abbildung 6 Firebase Projekt vorbereiten 9](#_Toc10797489)

[Abbildung 7 Control Panel Firebase Projekt 10](#_Toc10797490)

[Abbildung 8 Typescript Installation 10](#_Toc10797491)

[Abbildung 9 Firebase Installation 10](#_Toc10797492)

[Abbildung 10 Login Mockup 11](#_Toc10797493)

[Abbildung 11 Registrierung Mockup 12](#_Toc10797494)

[Abbildung 12 User List Mockup 13](#_Toc10797495)

[Abbildung 13 User Profile Mockup 14](#_Toc10797496)

[Abbildung 14 Chat Mockup 15](#_Toc10797497)

[Abbildung 15 Profile Mockup 16](#_Toc10797498)

[Abbildung 16 Edit Profile Mockup 17](#_Toc10797499)

[Abbildung 17 Chat liste Mockup 18](#_Toc10797500)

[Abbildung 18 UML Diagramm ChatApp 19](#_Toc10797501)

[Abbildung 19 Profile edditing 24](#_Toc10797502)

[Abbildung 20 Registrieren 25](#_Toc10797503)

[Abbildung 21 Einloggen 26](#_Toc10797504)

[Abbildung 22 Google Login 26](#_Toc10797505)

[Abbildung 23 Password zurücksetzen 27](#_Toc10797506)

[Abbildung 24 Benutzer Liste 27](#_Toc10797507)

[Abbildung 25 Chat Knopf 27](#_Toc10797508)

[Abbildung 26 Chat 28](#_Toc10797509)

[Abbildung 27 Menu 28](#_Toc10797510)

[Abbildung 28 Chat löschen 29](#_Toc10797511)

[Abbildung 29 Menu Profil 29](#_Toc10797512)

[Abbildung 30 Profil 29](#_Toc10797513)

[Abbildung 31 Profil editieren 30](#_Toc10797514)

[Abbildung 32 Authentifikation durch Firebase 31](#_Toc10797515)

[Abbildung 33 Chat Datenbank 32](#_Toc10797516)

[Abbildung 34 Benutzern Datenbank 32](#_Toc10797517)

# 1 Formular des Projekts

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 1 Auftrag Formular

# 2 Einleitung

Da ich sehr in Angular interessiert bin, habe ich entschlossen eine Angular[[1]](#footnote-1) basierte App zu entwickeln. Da ich noch nicht viel Erfragung mit Mobile Apps habe, wird es eine einfache Web App sein. Um es ein wenig interessanter zu machen werde ich eine Chat App entwickeln, dafür werde ich verschiedene Funktionalitäten der Plattform Firebase benutzen und das GUI werde ich mithilfe CSS und SASS gestalten.

# 3 Anforderungen

Kundenanforderungen:

Entwicklung einer Chat App mit folgenden Anforderungen:

* Der Benutzer kann sich mit email und Password, sowie mit Google Account einloggen.
* Der Benutzer kann mit anderen Nutzern per Chat schriftlich kommunizieren.
* Jeder Benutzer hat ein eigenes Profil.

Die funktionalen Anforderungen für dieses Projekts lauten:

* Benutzer kann sich mit Benutzername und Password Einloggen.
* Benutzer kann sich mittels Google Account Einloggen.
* Benutzer kann sich ausloggen, indem er auf einen „Sign. Out“ Knopf drückt.
* Nach dem Registrieren wird ein Einloggen erforderlich sein.
* Nach dem Registrieren wird am Nutzer ein verifikationsemail geschickt.
* Nach dem Einloggen wird der Benutzer zur eine Chat Seite weitergeleitet.
* In der Chat Seite kann der Benutzer zu allen Nutzern zugreifen, um mit denen zu chatten.
* Benutzer muss folgende Daten angeben, um ein Konto zu erstellen:
  + Email
  + Password: Nicht kürzer als 6 Zeichen und nicht länger als 25 Zeichen
  + Gender: Muss zwischen männliches und weibliches, mittels zwei Radio Buttons, wählen (nicht pflichtig)
* Wenn der Benutzer einer dieser Felder nicht korrekt ausfüllt, kann er kein Konto erstellen.
* Falls der Benutzer ein falsches Password und/oder Email, während den Login eingibt, bekommt er eine Fehler Meldung über.

Nicht funktionale Anforderungen:

* Grafische Oberfläche ist attraktiv und ansprechend.

# 4 Schnittstellen zu anderen Systemen

Diese App wird die Authentifikation, sowie die Datenbank Funktion von Google Firebase benutzen. Sie werden sich um das Back-End kümmern.

Die ganze Logik wird mit Hilfe von Angular/Fire möglich.

# 5 Architektur

Ich verwende für dieses Projekt den MVC sowohl, wie das Client/Server style:

Ein Bild, das drinnen, Text, Himmel, Wand enthält.

Automatisch generierte Beschreibung

## 5.1 Komponenten

Die Komponenten sind folgende:

* Auth
  + Login: Anmeldungskomponenten:
    - login.component.ts
    - login.component.html
    - login.component.css
  + Register: Kontoherstellungskomponenten:
    - register.component.ts
    - register.component.html
    - register.component.css
  + Reset-password: Password Wiederherstellungskomponenten:
    - reset-password.component.ts
    - reset-password.component.html
    - reset-password.component.css
  + auth.module: Beinhaltet die Logik für die funktionsweise der anderen Komponenten.
* Chat
  + chat-detail:
    - chat-detail.component
    - chat-detail.component.html
    - chat-detail.component.css
  + chat-feed
    - chat-feed.component
    - chat-feed.component.html
    - chat-feed.component.css
  + chat-input:
    - chat-input.component
    - chat-input.component.html
    - chat-input.component.css
  + chat-list:
    - chat-list.component
    - chat-list.component.html
    - chat-list.component.css
  + chat-message:
    - chat-message.component
    - chat-message.component.html
    - chat-message.component.css
  + chat-messages:
    - chat-messages.component
    - chat-messages.component.html
    - chat-messages.component.css
  + chat-thread:
    - chat-thread.component
    - chat-thread.component.html
    - chat-thread.component.css
  + chat-threads:
    - chat-threads.component
    - chat-threads.component.html
    - chat-threads.component.css
  + chat.module
  + message.model: Classe für message
  + message.service
  + thread.model: Klasse für thread
  + thread.service
* core:
  + auth.service: Authentifikationsfunktionen und Logik
  + core.module
* shared: Alle Komponenten die von anderen Komponenten gebraucht werden:
  + navbar: Navigation Komponenten der Applikation
    - navbar.component
    - navbar.component.html
    - navbar.component.css
  + upload: Hochladen von Bildern
    - upload.component
    - upload.component.html
    - upload.component.css
    - upload.service.ts
  + from-now.pipe
  + shared.module
* user
  + user-dashboard
    - user-dashboard.component
    - user-dashboard.component.html
    - user-dashboard.component.css
  + user-detail:
    - user-detail.component
    - user-detail.component.html
    - user-detail.component.css
  + user-list:
    - user-list.component
    - user-list.component.html
    - user-list.component.css
  + user-list-item:
    - user-list-item.component
    - user-list-item.component.html
    - user-item.component.css
  + user.model
  + user.module
  + user.service
* app.component.ts
* app.component.html
* app.component.ts
* app.module.ts
* material.module.ts
* routing.guard.ts
* routing.module.ts

# 6 Das Projekt

## 6.1 Die Vorbereitung der Entwicklungsumgebung

Da ich mit Angular arbeiten möchte, muss ich erst die Entwicklungsumgebung vorbereiten. Dafür muss ich folgendes installieren: Node.js, Angular/cli, Typesript und das Firebase SDK. Ausserdem muss ich eine neue Projektmappe bei Firebase.com herstellen, damit sich die Applikation mit den Firebase Servern kommuniziert.

1. Auf Node.js Homepage gehen und Node.js herunterladen:

Ein Bild, das Screenshot, Monitor enthält.

Automatisch generierte Beschreibung

Abbildung 2 Node herunterladen

Um Inkompatibilitätsprobleme zu vermeiden, werde ich die Version 10.15.3 LTS benutzen, diese ist die letzte stabile Version.

1. Nach der Installation von Node.js werde ich mittels npm Befehl Angular herunterladen und installieren:



Abbildung 3 Angular installieren

Nachdem ich auf Enter drücke, wird Angular auf dem Rechner Installiert werde.

Ein Bild, das Screenshot, Text enthält.

Automatisch generierte Beschreibung

Abbildung 4 Angular Version

Wie man sehen kann, handelt es sich um die Version 7.3.9 von Angular.[[2]](#footnote-2)

1. Als nächstes werde ich eine neue Projekt Mappe in Firebase erstellen damit meiner App, all die Firebase Funktionalitäten, die ich brauche, benutzen kann.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 5 Firebase Projekt erstellen

Mit dem Klicken auf dem Kreuz, kann ich ein neues Projekt erstellen.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 6 Firebase Projekt vorbereiten

Jetzt muss ich nur einem passenden Namen für meine App wählen und die Geschäftsbedingungen akzeptieren und auf Projekt Erstellen drücken.

1. Hier habe ich den Control Panel meiner App:

Ein Bild, das Screenshot, Monitor, Mobiltelefon, Telefon enthält.

Automatisch generierte Beschreibung

Abbildung 7 Control Panel Firebase Projekt

1. Installation von Typescript

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 8 Typescript Installation

1. Installation von Firebase



Abbildung 9 Firebase Installation

## 6.2 Gestaltung des Designs

Ich wollte ein modernes und attraktives Design gestalten, deshalb habe ich die angular-material Komponenten für meine App benutzt. Da das Design dieser Komponenten sich nach die Google-Material Standards orientieren, denn Angular wird von Google betreut, kann ich davon ausgehen, dass diese Komponenten eine hohe Qualität haben, sowie Kompatibel. Ausserdem vermeide ich so mögliche Fehler, die durch falsch programmierte Formulare entstehen können.

## 6.3 Mockup

Damit es mir leichter fällt meine App zu programmieren, habe ich einige Mockups gemacht, die den Fertigen Design entsprechen sollen.

* Log In:

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 10 Login Mockup

Nachdem der Benutzer sich eingeloggt hat, wird er zu dem Benutzer liste weitergeleitet und wenn er auf Register drückt, kommt er zur Registrierung Seite.

* Registrierung:

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 11 Registrierung Mockup

Wenn der Benutzer auf Register drückt, bekommt er ein verifikationsemail und wird zur Login Seite weitergeleitet.

* Benutzer liste

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 12 User List Mockup

Hier kann der Nutzer auf einen beliebigen User der Liste klicken, um an der Profil Seite dieses Users zu gelangen. Von dort aus, kann er auf den Chat Knopf drücken, um ein Chat zu öffnen.

* User-profile

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 13 User Profile Mockup

Hier hat der Benutzer auf einen der User geklickt und wenn er nun auf das Chat Knopf drückt, beim User unten rechts, kann er einen Chat mit diesem User Beginen.

* Chat

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 14 Chat Mockup

Das ist die Chat Seite, hier kann der Benutzer Nachrichten senden und Empfängen.

* Profile

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 15 Profile Mockup

Dies ist die Profile Seite des Eingeloggten Benutzers, er kann hier ein neues Profilbild zufügen oder die Profile Information ändern, indem er auf Edit Profile klickt.

* Edit-profile

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 16 Edit Profile Mockup

* Chat liste

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 17 Chat liste Mockup

Hier kann der Benutzer alle aktiven Chats ansehen und sie löschen.

## 6.4 UML Diagramm

Ein Bild, das Text, Karte enthält.

Automatisch generierte Beschreibung

Abbildung 18 UML Diagramm ChatApp

# 7 Test

**Test Nr.: 1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Email Input wird angeklickt und danach wird auf der Seite geklickt.**

**Resultat: Console.log() email is badly formatted, Input wird rot angezeigt.**

**Erwartetes Resultat: Console.log() email is badly formatted und Input soll rot angezeigt werden.**

**Test Nr: 1.0**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Password Input wird angeklickt und danach wird auf der Seite geklickt.**

**Resultat: Console.log() email is badly formatted, Input wird rot angezeigt.**

**Erwartetes Resultat: Console.log() email is badly formatted und Input soll rot angezeigt warden.**

**Test Nr: 1.0.1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Register Knopf wird angeklickt während weide Inputs lehr sind.**

**Resultat: Console.log() email is badly formatted, Inputs werden rot angezeigt.**

**Erwartetes Resultat: Console.log() email is badly formatted und beide Inputs sollen rot angezeigt werden.**

**Test Nr: 1.1.0**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Asdrdssarr wird im Email Input geschrieben, während Password Input lehr ist. Danach wird auf Registrieren geklickt.**

**Resultat: Console.log() email is badly formatted, Input wird rot angezeigt.**

**Erwartetes Resultat: Console.log() email is badly formatted und Input soll rot angezeigt werden und Registrierung soll nicht möglich sein.**

**Test Nr: 1.1.1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Asdrdssarr wird im Password Input geschrieben, während Email Input lehr ist. Danach wird auf Registrieren geklickt.**

**Resultat: Console.log() email is badly formatted, Input wird rot angezeigt.**

**Erwartetes Resultat: Console.log() email is badly formatted und Input soll blau angezeigt werden und Registrierung soll nicht möglich sein.**

**Test Nr: 2**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht:** [yagoromero50@gmail.com](mailto:yagoromero50@gmail.com) **wird in Email Input geschrieben, während Password Input lehr ist. Danach wird auf Registrieren geklickt.**

**Resultat: Console.log() The password must be 6 characters long or more, Input wird blau angezeigt.**

**Erwartetes Resultat: Console.log() The Password must be 6 characters long or more und Input soll blau angezeigt werden. Registrierung soll nicht möglich sein.**

**Test Nr.: 2.0**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: 1 wird in Password Input geschrieben, während Email Input lehr ist. Danach wird auf Registrieren geklickt.**

**Resultat: Console.log() email is badly formatted, Input wird rot angezeigt.**

**Erwartetes Resultat: Console.log() email is badly formatted und Input soll rot angezeigt werden und Registrierung soll nicht möglich sein.**

**Test Nr: 2.0.1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: password1 wird in Password Input geschrieben, während Email Input** [yagoromero50@gmail.com](mailto:yagoromero50@gmail.com) **beinhaltet. Danach wird auf Registrieren geklickt.**

**Resultat: Beide Inputs werden blau angezeigt. Console.log() Welcome, your account has been created! Und We sent you an email verification. E-Mail mit verifikationslink wird geschickt. Danach wird man zur Login Seite weitergeleitet.**

**Erwartetes Resultat: Inputs sollen blau angezeigt werden und ein einen verifikationsemail soll an die eingegebene Email Adresse geschickt werden und man soll zur Login Seite weitergeleitet werden.**

**Test Nr: 2.1.1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Im Login Formular Registrieren wird geklickt.**

**Resultat: Man wird zur Registrierung Seite weitergeleitet.**

**Erwartetes Resultat: Man soll zur Registrierung Seite weitergeleitet werden.**

**Test Nr: 3**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Im Login Formular gibt man die Anmeldungsdaten ein und klickt auf Sign In.**

**Resultat: Man wird zu dem Benutzerliste Seite weitergeleitet. unsafe:nullhttps://www.gravatar.com/avatar/cb60af7770d84063b66bb39edbd81fcd?d=identicon:1 GET unsafe:nullhttps://www.gravatar.com/avatar/cb60af7770d84063b66bb39edbd81fcd?d=identicon net::ERR\_UNKNOWN\_URL\_SCHEME. Profilbilder werden nicht angezeigt**

**Erwartetes Resultat: Man soll zu dem Benutzerliste Seite weitergeleitet werden. Profilbilder sollen angezeigt werden.**

**Test Nr: 3**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: In der Benutzerlist Seite wird ein Nutzer angeklickt.**

**Resultat: Man wird zum Profil dieses Nutzers weitergeleitet. unsafe:nullhttps://www.gravatar.com/avatar/cb60af7770d84063b66bb39edbd81fcd?d=identicon:1 GET unsafe:nullhttps://www.gravatar.com/avatar/cb60af7770d84063b66bb39edbd81fcd?d=identicon net::ERR\_UNKNOWN\_URL\_SCHEME. Profilbilder werden nicht angezeigt**

**Erwartetes Resultat: Man soll zum Profil dieses Nutzers Seite weitergeleitet werden. Profilbilder sollen angezeigt werden.**

**Test Nr: 3.0**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Beim Nutzer Profil klickt man auf Chat Knopf.**

**Resultat: Man wird zur Chat Seite weitergeleitet. Console.log() Thread created!**

**Erwartetes Resultat: Man soll zur Chat Seite weitergeleitet werden. Console.log() Thread created!**

**Test Nr: 3.0.1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Man schickt eine Nachricht am Nutzer.**

**Resultat: ChatInputComponent.html:2 ERROR TypeError: Cannot read property 'doc' of undefined**

**at ChatInputComponent.saveLast (chat-input.component.ts:42)**

**at ChatInputComponent.send (chat-input.component.ts:35)**

**at ChatInputComponent.handleSubmit (chat-input.component.ts:47)**

**at Object.eval [as handleEvent] (ChatInputComponent.html:2)**

**at handleEvent (core.js:34777)**

**at callWithDebugContext (core.js:36395)**

**at Object.debugHandleEvent [as handleEvent] (core.js:36031)**

**at dispatchEvent (core.js:22519)**

**at core.js:33709**

**at HTMLInputElement.<anonymous> (platform-browser.js:1789)**

**View\_ChatInputComponent\_0 @ ChatInputComponent.html:2**

**proxyClass @ compiler.js:19199**

**logError @ core.js:36330**

**handleError @ core.js:7239**

**dispatchEvent @ core.js:22523**

**(anonymous) @ core.js:33709**

**(anonymous) @ platform-browser.js:1789**

**invokeTask @ zone-evergreen.js:391**

**onInvokeTask @ core.js:30873**

**invokeTask @ zone-evergreen.js:390**

**runTask @ zone-evergreen.js:168**

**invokeTask @ zone-evergreen.js:465**

**invokeTask @ zone-evergreen.js:1603**

**globalZoneAwareCallback @ zone-evergreen.js:1629**

**ChatInputComponent.html:2 ERROR CONTEXT DebugContext\_ {view: {…}, nodeIndex: 11, nodeDef: {…}, elDef: {…}, elView: {…}}**

**View\_ChatInputComponent\_0 @ ChatInputComponent.html:2**

**proxyClass @ compiler.js:19199**

**logError @ core.js:36330**

**handleError @ core.js:7244**

**dispatchEvent @ core.js:22523**

**(anonymous) @ core.js:33709**

**(anonymous) @ platform-browser.js:1789**

**invokeTask @ zone-evergreen.js:391**

**onInvokeTask @ core.js:30873**

**invokeTask @ zone-evergreen.js:390**

**runTask @ zone-evergreen.js:168**

**invokeTask @ zone-evergreen.js:465**

**invokeTask @ zone-evergreen.js:1603**

**globalZoneAwareCallback @ zone-evergreen.js:1629**

**message.service.ts:43 Message sent**

**Nachricht wird gesendet**

**Erwartetes Resultat: Nachricht soll ohne Fehler gesendet werden. Console.log() Message sent**

**Test Nr: 3.1.1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Man klickt im Menu auf Profil.**

**Resultat: Man wird zur Profil Seite von** [yagoromero50@gmail.com](mailto:yagoromero50@gmail.com) **weitergeleitet. Profilbild wird nicht angezeigt.**

**GET nullhttps://www.gravatar.com/avatar/cb04ff574e27bcd6f557c0ce90d29ecb?d=identicon net::ERR\_UNKNOWN\_URL\_SCHEME**

**Erwartetes Resultat: Man soll zur Profil Seite des Eingeloggtes Nutzers weitergeleitet werden. Profilbild soll angezeigt werden.**

**Test Nr: 4**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Bei der Profil Seite von** [yagoromero50@gmail.com](mailto:yagoromero50@gmail.com)**, klickt man auf dem Profilbild, um es zu ändern.**

**Resultat: core.js:7187 ERROR FirebaseStorageError {code\_: "storage/object-not-found", message\_: "Firebase Storage: Object 'users/MXARggehIEgFiVOmM8cQFmNAob33/photos/Momon.jpg' does not exist.", serverResponse\_**

**Erwartetes Resultat: Man soll das Profilbild anderen können und danach soll es als Profilbild bleiben.**

**Test Nr: 4.0**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Bei der Profil Seite von** [yagoromero50@gmail.com](mailto:yagoromero50@gmail.com)**, klickt man auf Edit Profil.**

**Resultat: Man wird zur Edit Profil Seite weitergeleitet.**

**Erwartetes Resultat: Man soll zur Edit Profil Seite weitergeleitet werden.**

**Test Nr: 4.1**

**Typ: White box**

**Art: Unit Test**

**Was wird gemacht: Bei der Profil Seite von** [yagoromero50@gmail.com](mailto:yagoromero50@gmail.com)**, wird folgendes eingetragen:**

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 19 Profile edditing

**Bei update your name und Update your email, wird auf dem Save Knopf geklickt.**

**Danach wird auf Save geklickt.**

**Resultat: Console.log() Your profile has been updated!**

**user.service.ts:43 Your email has been updated to: yagoromero50@gmail.com**

**user.service.ts:47 We sent you an email verification**

**Erwartetes Resultat: Console.log() Your profile has been updated!**

**user.service.ts:43 Your email has been updated to: yagoromero50@gmail.com**

**user.service.ts:47 We sent you an email verification**

**Ein Verifikation Email soll gesendet werden.**

# 8 Benutzerhandbuch

## 8.1 Sich registrieren

Um sich erfolgreich zu registrieren, müssen Sie ein gültiges Email eingeben wie zum Beispiel: [yagoromero@gmail.com](mailto:yagoromero@gmail.com).

Danach müssen Sie ein Password eingeben, der minimal 6 Zeilen lang ist und maximal 25 Zeilen lang ist. Ausserdem muss es mindestens eine Zahl beinhalten, um das gewählte Password zu überprüfen können Sie auf den Augen Knopf drücken, so wird Ihrem Password angezeigt und Sie können es korrigieren.

Das Gendern ist freiwillig. Sie müssen es nicht angeben, wenn Sie nicht möchten.

Wenn Sie fertig sind, drücken Sie auf Register. Sie werden ein verifikationsemail bekommen.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 20 Registrieren

Nachdem Sie auf dem Link, der in der verifikationsemail zu finden ist, geklickt haben, können sie sich einloggen.

Falls Sie schon ein Account habe, können Sie oben links, neben den Namen der App, auf Log in klicken und sich einloggen.

## 8.2 Sich einloggen

Um sich einloggen zu können, müssen Sie ihr Anmeldungsdaten eingeben oder Sie können sich mit ihr Gmail Account einloggen.

Wenn Sie email und Password eingegeben haben, drücken sie auf Sign In. Falls sie ihr Password oder Email vergessen haben, können sie auf Forgot Password? Klicken, um es zurückzustellen.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 21 Einloggen

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 22 Google Login

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 23 Password zurücksetzen

Nach dem einloggen, werden Sie automatisch zur Benutzern Liste weitergeleitet.

## 8.3 Chatten

Damit Sie anderen Benutzern eine Nachricht senden können, müssen sie folgende Schritte folgen:

* Erstens wählen Sie ein Nutzer aus der Benutzer Liste aus, indem sie auf den Namen dieses Nutzers klicken.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 24 Benutzer Liste

* Sie werden zur Profil Seite dieses Nutzers weitergeleitet, dort können sie auf dem Nachricht Knopf, unten rechtes beim Nutzer Profil, um einen neuen Chat anzufangen.

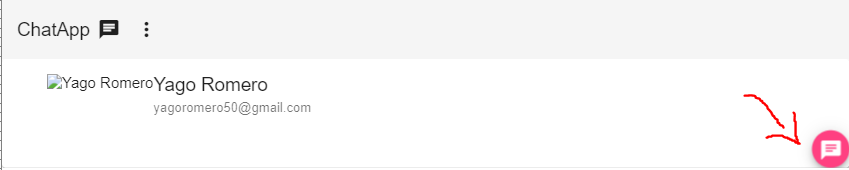


Abbildung 25 Chat Knopf

Sie werden durch den klicken auf diesen Knopf zur Chat Seite gelangen.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 26 Chat

Ganz unten können sie einen Text eingeben, der versendet werden soll.

## 8.4 Navigieren

Um durch die App zu navigieren, können sie oben links neben den Namen der App, auf den Menu Knopf drücken.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 27 Menu

So können sie zum Profil, Benutzer List oder Chats gelangen, sowie sich auch Ausloggen.

## 8.5 Chats löschen

Falls Sie einen Chatverlauf löschen möchten, können sie auf oben links auf den Chat Icon drücken, so gelangen sie zu Ihrem aktiven Chat. Danach müssen Sie nur noch beim ausgewählten Chatverlauf, auf dem Papierkorb drücken.

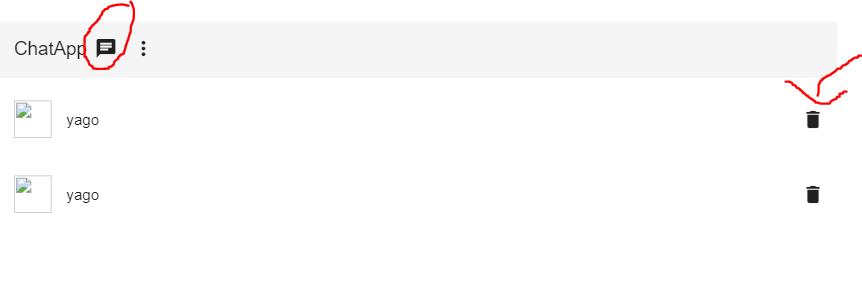


Abbildung 28 Chat löschen

## 8.6 Profil editieren

Klicken sie beim Menu auf Profil, um an ihre Profil Seite zu gelangen. Dort können sie auf Edit Profile drücken, um die Profil daten zu verändern.

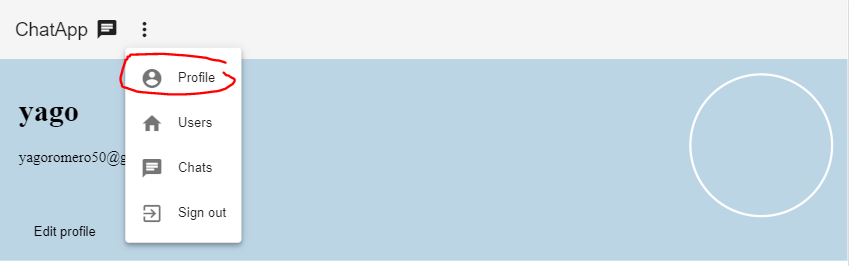


Abbildung 29 Menu Profil

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 30 Profil

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 31 Profil editieren

Wenn Sie fertig sind, drücken Sie auf Save.

# 9 Arbeitsjournal

|  |  |
| --- | --- |
| Was wurde gemacht? | Wenn |
| Thema gesucht | 19/04/2019 |
| Ich habe mich für ein Thema entschieden | 26/04/2019 |
| Umsetzung recherchieren | 29/04/2019 |
| Ich habe mich für Angular um das Projekt um zu setzen entschieden | 01/05/2019 |
| Entwicklungsumgebung wird vorbereitet | 10/05/2019 |
| Dokumentation wird geschrieben | 10/05/2019 |
| Mockups gemacht | 11/05/2019 |
| Erste Angular Komponenten und Services werden vorbereitet | 14/05/2019 |
| Authentifikationsprobleme mit Gitlab | 17/05/2019 |
| Routing der verschiedenen Komponenten | 17/05/2019 |
| Hauptfunktionen programmiert | 18/05/2019 |
| Login und Registrierung | 19/05/2019 |
| Formular für Login und Registrierung | 21/05/2019 |
| Login und Registrierung Funktionen/Methoden | 21/05/2019 |
| Password-reset programmiert | 22/05/2019 |
| Users Komponenten programmiert | 23/05/2019 |
| Chat Komponenten programmiert | 26/05/2019 |
| Gesamtes Design überprüft | 02/06/2019 |
| Dokumentation geschrieben | 05/06/2019 |
| Tests durchgeführt | 05/06/2019 |
| Bugs korrigiert | 05/06/2019 |
| Dokumentation geschrieben | 06/06/2019 |
| Navbar und letzte Änderungen durchgeführt | 07/06/2019 |
| Test durchgeführt | 07/06/2019 |

# 10 Versionen Kontrolle

|  |  |
| --- | --- |
| Chatapp | Version |
|  | 1.0.0 |

# 11 Anhang

Ein Bild, das Screenshot, drinnen enthält.

Automatisch generierte Beschreibung

Abbildung 32 Authentifikation durch Firebase

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 33 Chat Datenbank

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 34 Benutzern Datenbank

1. Ich werde Angular und nicht Angular.js verwenden. Obwohl sie sehr ähnlich sind, finde ich, dass Angular einfacher ist. [↑](#footnote-ref-1)
2. Später habe ich das angular/cli auf die Version 8.0.1 aktualisiert, um einige Bugs zu korrigieren. [↑](#footnote-ref-2)