29. Dezember 2021

Manuel Ebneter

UIFZ-2026-005-A

Control Freak Parent - Projektdokumentation

Modul 151

Inhaltsverzeichnis

[Projektdokumentation 2](#_Toc91697245)

[Management Summary 2](#_Toc91697246)

[Informieren 2](#_Toc91697247)

[Planen 3](#_Toc91697248)

[Use-Case Diagramm 3](#_Toc91697249)

[Klassendiagramm 3](#_Toc91697250)

[ERD 3](#_Toc91697251)

[Aussehen 4](#_Toc91697252)

[Realisieren 6](#_Toc91697253)

[View-Components 6](#_Toc91697254)

[Control-Components 6](#_Toc91697255)

[Model-Components 6](#_Toc91697256)

[Hilfestellungen 7](#_Toc91697257)

[PHP 7](#_Toc91697258)

[JS 7](#_Toc91697259)

[Ehrenwort 7](#_Toc91697260)

[Github 7](#_Toc91697261)

# Projektdokumentation

## Management Summary

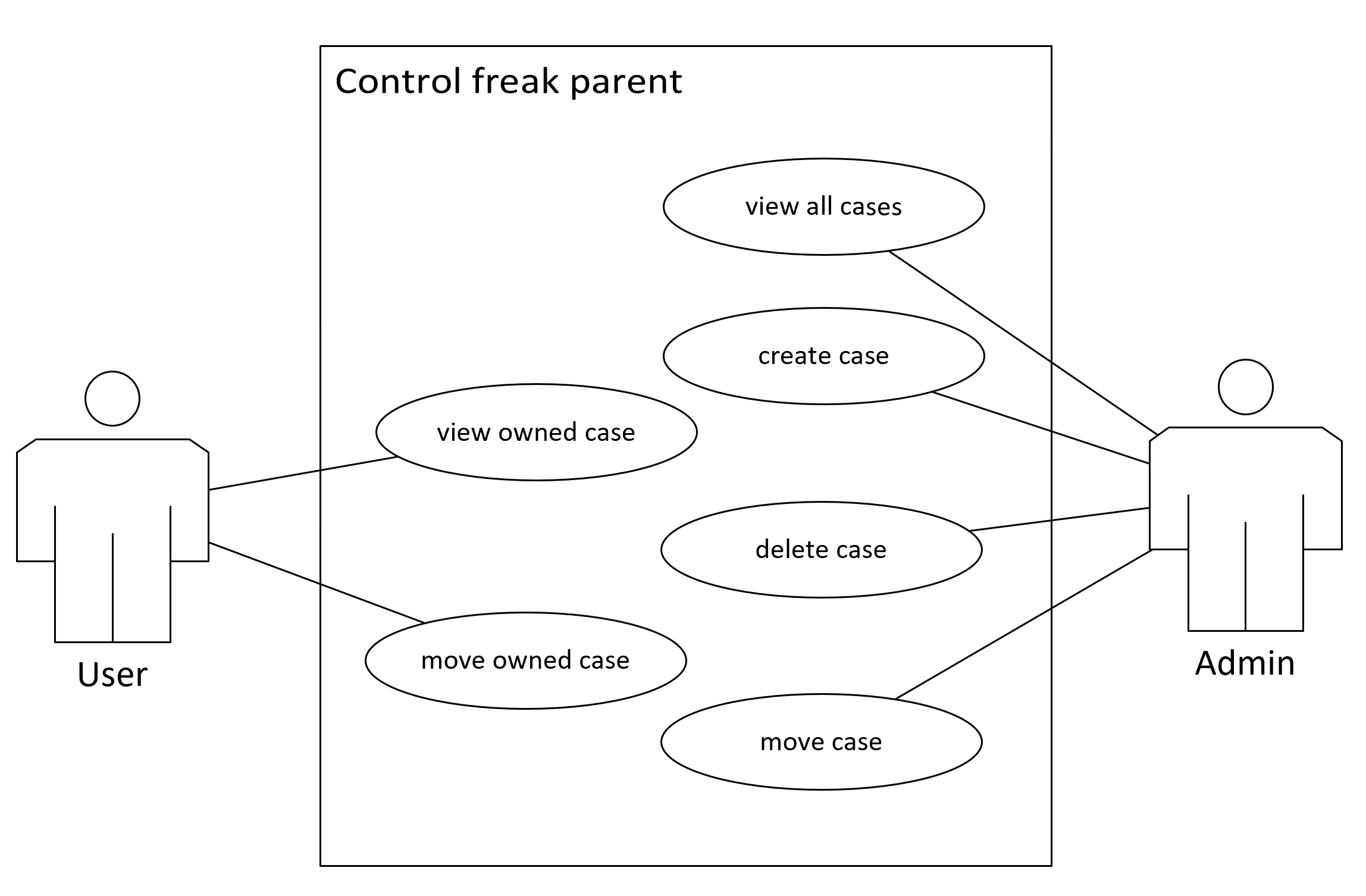
Eine Web-CRUD-App zum managen von Aufgaben für Familienmitglieder. Das Familienoberhaupt kann Aufträge erstellen und sie den jeweiligen Familienmitgliedern zuordnen. Familienmitglieder können ihre eigenen Aufträge betrachten und Fortschritt bestätigen. Die Applikation besteht aus einem Quasar(vue.js) Frontend und PHP im Backend welche die entsprechende Seite via Session-Management anzeigt. Auch im Backend befindet sich eine Rest-API welche Anfragen bearbeitet und die Qery’s zur Datenbank(MariaDB) managen.

## Informieren

* Aufträge anzeigen
  + Für den Admin werden alle Aufträge angezeigt
  + Für User werden die ihnen zugewiesenen Aufträge angezeigt
* Auftrag erstellen
  + Aufträge sollen vom Admin erstellt werden können
  + Auftrag hat einen Text sowie einen verantwortlichen User
* Auftrags Status ändern
  + Statusänderung von ‘zu Erledigen’ --> ‘in Bearbeitung’ --> ‘erledigt’
  + Admin kann alle Aufträge bewegen
  + User kann nur seine Aufträge bewegen
* Auftrag löschen
  + Admin kann Auftrag löschen
* Login
  + Hier können die Benutzter einloggen und werden ensprechend ihrer rechte auf die jeweilige Seite weitergeleitet.

## Planen

### Use-Case Diagramm



### Klassendiagramm

Werden von vom REST-Service genutzt.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

### ERD

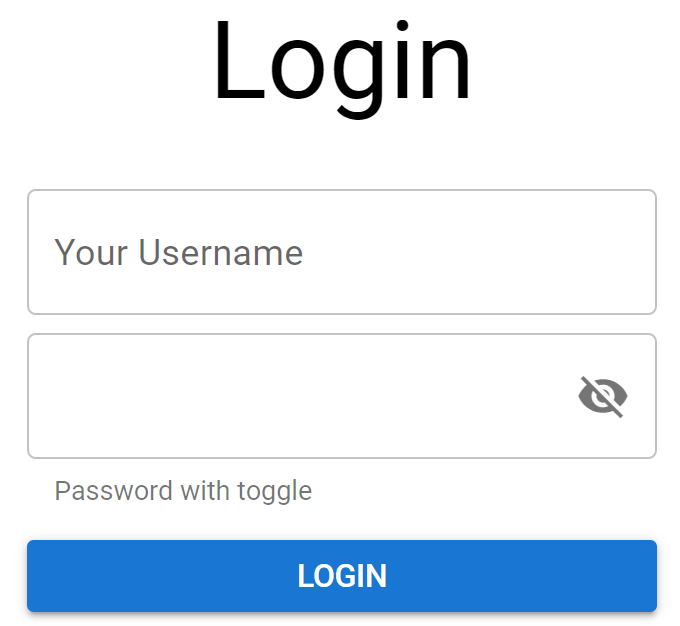
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

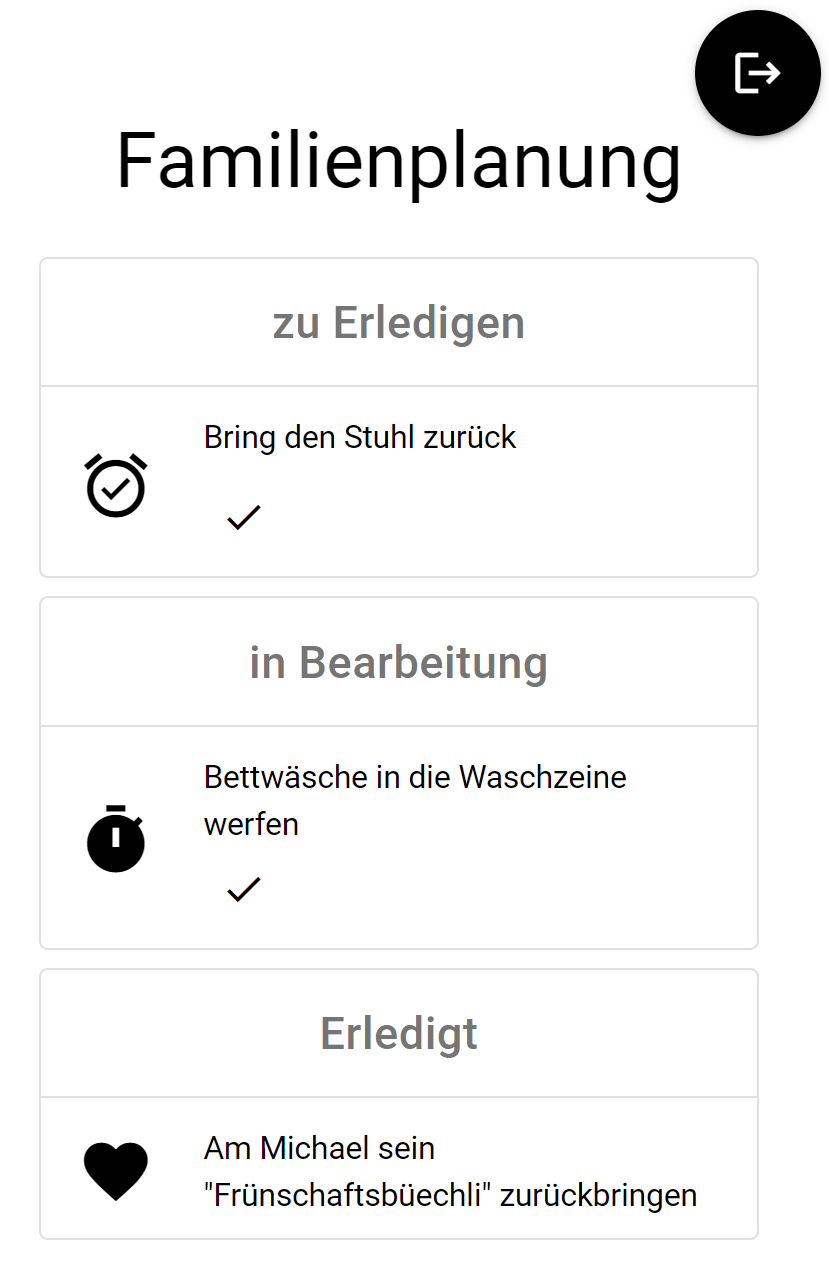
### Aussehen

#### Login-View

Einfache Login-Seite, nach erfolgreichem Login wird User-View oder Admin-View retuned.



#### User-View

Enthält Anzeige der eigenen Aufträge, welche via ‘done’-Button zum nächsten Status gebracht werden können und einen Button zum Logout.

#### Admin-View

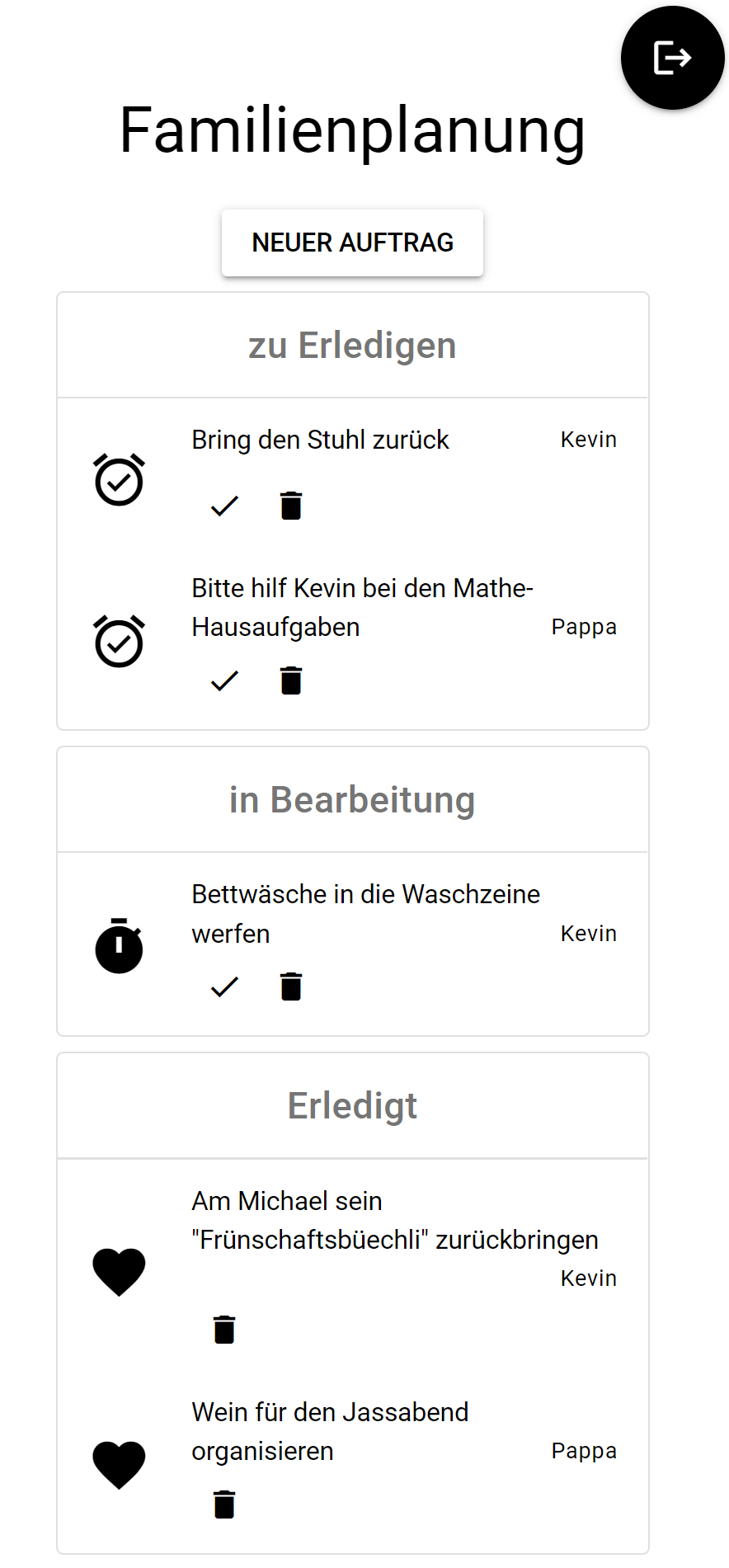
Button:   
Auftrag State-Fortschritt

Button:   
Auftrag Löschen

Auftrag mit Verantwortlicher Person

Container für Arbeitsauftrag (State-spezific)

Neuer Auftrag hinzufügen



Logout

Auftrag hinzufügen mit Textfeld für die Auftragsbeschreibung und Select-Input für die Bestimmung des Verantwortlichen.

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungEin Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung - Verantwortung-Select

Abbildung - Auftrag hinzufügen

## Realisieren

### View-Components

#### login.html

Die erste Seite die beim Aufrufen der URL angezeigt wird. Entsprechend des gewählten Logins wird der User auf die entsprechende View weitergeleitet.

#### adminView.html

Beinhaltet Logik zum Anzeigen, Hinzufügen, Löschen und Bearbeiten von Aufträgen. Daten werden parallel zu den Updates via Call zum REST-Service auch Client seitig angepasst, um Anfragen an das Backend zu reduzieren.

#### userView.html

Beinhaltet Logik zum Anzeigen und Bearbeiten von Aufträgen. Daten werden parallel zu den Updates via Call zum REST-Service auch Client seitig angepasst, um Anfragen an das Backend zu reduzieren

### Control-Components

#### CaseHandler.php

Klasse die eine Datenbankverbindung und query’s für die Case-Tabelle über verschiedene Methoden zur Verfügung stellt.

#### UserHandler.php

Klasse die eine Datenbankverbindung und query’s für die User-Tabelle über verschiedene Methoden zur Verfügung stellt.

#### case.api.php

Stellt ein Interface für HTTP-Calls zur Verfügung und beantwortet diese entsprechend.

#### user.api.php

Stellt ein Interface für HTTP-Calls zur Verfügung und beantwortet diese entsprechend. Auch werden hier die Session-Variablen vergeben, die für Useridentifikation genutzt werden.

### Model-Components

#### cfp.sql

Script welches die SQL-Datenbank darstellt. Bestehend aus einer Tabelle für User und einer für Cases die in Beziehung miteinander stehen.

## Hilfestellungen

### PHP

* <https://www.php-einfach.de/mysql-tutorial/crashkurs-pdo/>
* [https://www.php.net/manual/de/book.pdo.php](https://www.php.net/manual/de/book.pdo.phpphp)
* <https://phptherightway.com/>

### JS

* <https://quasar.dev/>
* <https://developer.mozilla.org/en-US/>

## Ehrenwort

Hiermit bestätige ich, alle Teile meiner Projektarbeit ohne fremde Hilfe gemacht zu haben. Ich habe keinen fertigen Code aus einem Buch oder dem Internet kopiert, alle Quellen zur Hilfestellung habe ich oben angegeben.

## Github

[GIT-Link](https://github.com/MEbneter/M151_CFP)