|  |  |
| --- | --- |
|  |  |
| **M223 Doku** |  |
|  |  |
|  | *Abgabe: 24.3.2023*  *M223* |
|  | *M223*  *Experte: Remo Steinmann* |

Table of Contents

[Teil 1: Konzeptioneller Teil 3](#_Toc130450302)

[1.1 Aufgabenstellung 3](#_Toc130450303)

[1.1.1 Ausgangslage 3](#_Toc130450304)

[1.1.2 Aufgabenstellung 3](#_Toc130450305)

[1.1.3 Technologien 3](#_Toc130450306)

[1.2 Projektaufbau 4](#_Toc130450307)

[1.3 Zeitplan 4](#_Toc130450308)

[1.3.1 Termine 4](#_Toc130450309)

[1.3.2 Geplante Arbeiten 4](#_Toc130450310)

[1.4 Arbeitsjournal 4](#_Toc130450311)

[Teil 2: Praktischer Teil 7](#_Toc130450312)

[2.1 I – Informieren 7](#_Toc130450313)

[2.1.1 Ausgangslage 7](#_Toc130450314)

[2.1.2 Umsetzung 8](#_Toc130450315)

[2.1.3 Kriterien 8](#_Toc130450316)

[2.1.4 Use-Cases 9](#_Toc130450317)

[2.2 P – Planen 10](#_Toc130450318)

[2.2.1 Geplante Seiten 10](#_Toc130450319)

[2.2.2 Zeitplan 10](#_Toc130450320)

[2.3 E – Entscheiden 10](#_Toc130450321)

[2.3.1 Lösungswege 11](#_Toc130450322)

[2.4 R – Realisieren 11](#_Toc130450323)

[2.4.1 – Frontend 11](#_Toc130450324)

[2.4.2 – Backend 11](#_Toc130450325)

[2.4.3 - Datenbank 11](#_Toc130450326)

[2.5 K – Kontrollieren 11](#_Toc130450327)

[2.6 A - Auswerten 11](#_Toc130450328)

[3 Quellenverzeichnis 11](#_Toc130450329)

[4 Glossar 12](#_Toc130450330)

[5 Anhang 12](#_Toc130450331)

[5.1 Code 12](#_Toc130450332)

# Teil 1: Konzeptioneller Teil

## Aufgabenstellung

### 1.1.1 Ausgangslage

Aufgabenstellung im Modul 223 ist es eine Projektarbeit im Sinne einer IPA-Simulation durchzuführen. Der Arbeitsaufwand soll dabei ca. 33 Stunden betragen. Ziel dabei ist es sich optimal auf die richtige IPA vorzubereiten, mit dem Unterschied, dass der Arbeitsaufwand, sowie die individuellen Ziele, sich von dieser Arbeit unterscheiden.

### 1.1.2 Aufgabenstellung

Das Projekt soll folgende Punkt erfüllen:

#### 1.1.2.1 Funktionale Anforderungen:

Die objektorientierte Applikation erfüllt folgende Kriterien:

Front- und Backend

Zentrale Datenbank

Mehrere User greifen gleichzeitig auf den gleichen Datenbestand zu

Zentrale Benutzer- und Rechteverwaltung

#### 1.1.2.2 Nicht-Funktionale Anforderungen

Das Projekt muss komplett neu sein und darf keine Erweiterung eines bestehenden Projekts sein.

Ein Projekt von der Abteilung und das Arbeiten auf Firmen-Infrastruktur (Notebook, Entwicklungsumgebung, Server) ist möglich, aber nicht obligatorisch (bei der Nutzung von Firmeninfrastruktur, die Erlaubnis von der Abteilung einholen).

Die Wahl der Technologie ist euch überlassen (Rahmenbedingungen oben müssen aber immer eingehalten werden, => Empfehlung: Technologie von richtiger IPA verwenden).

Beispiel für eine solche Aufgabenstellung findet ihr auf der Webseite der PK19 in Zürich [www.pk19.ch](http://www.pk19.ch) (<https://pk19.ch/wp-content/uploads/2020/12/Aufgabenstellung-API-Beispiel1.pdf>)

Das gesamte Projektresultat (Programmcode, Dokumentation und Präsentation) wird am Schluss abgegeben, bei uns archiviert sowie steht der Berufsbildung als Muster für zukünftige Durchführungen komplett zur Verfügung. Bitte geeignete Projekte wählen.

### 1.1.3 Technologien

* Angular
  + Front-End Framework
  + Basiert auf Typescript, CSS/SCSS/SASS und HTML
  + Entwickelt von Google
* Express
  + Server Framework
  + Benutzt Typescript
  + CORS kompatibel
* JSON
  + Datenbank
  + Daten werden nicht relational in ein File gespeichert
  + Benötigt ein Typescript Teil, der Sachen speichert und lädt
* Node.js
  + NPM ist ein CLI tool, um node pakages zu installieren
  + Node ist ein JavaScript Interpreter
  + Node Kompiliert TS zu JS

## Projektaufbau

Remo Steinmann wird in diesem Projekt die Rolle des Experten übernehmen. In diesem Fall gibt es keine Fachvorgesetzte Person.

Als Projektmethode habe ich mich für IPERKA entschieden. Auch wenn ich es nicht gern habe, ist es die Methode am sinnvollsten, da man damit einen Auftrag Punkt für Punkt analysieren und ausführen kann. Zudem lassen sich die Projektphasen sauber trennen.

## Zeitplan

### 1.3.1 Termine

1. Projekttag(14.3.23): Ein Expertenbesuch findet statt, um erstes Feedback zu geben und um die Grundlage der Arbeit zu beachten.

5. Projekttag(22.3.23): Ein 2. Expertenbesuch soll schon ein ziemlich finales Produkt aufzeigen, bei dem auch Feedback zur Dokumentation gegeben werden soll.

6. Projekttag(24.3.23): Abgabe des Projektes.

11.4.23: Präsentation und Fachgespräch der Projektarbeit.

### 1.3.2 Geplante Arbeiten

Folgende Arbeiten sind geplant:

* Aufbau der Dokumentation
* Grobe Version des Zeitplanes erstellen
* Expertenbesuche
* Kriterienkatalog studieren
* Zeitplan fertigstellen
* Konzept für die Realisierung erstellen
* Testkonzept erstellen
* Lösungsvariante festlegen
* Einrichten der Projektumgebung
* Implementieren der Datenbank
* Implementieren der REST API
* Implementieren Der REST Routes
* Login und Signup Pages Implementieren
* To-Do Page Implementieren
* Admin Panels Implementieren
* Umgebung fürs Testen einrichten
* Testfälle Testen
* Reflexion & Fazit
* Finalisierung und Abgabe

## Arbeitsjournal

Gelb: Teilweise Erledigte Arbeiten

Rot: Nicht erledigte Arbeiten

|  |  |
| --- | --- |
| Datum: | 14.3.2023 (1. Tag, Dienstag) |
| Geplante Arbeiten: | Aufbau der Dokumentation(1h)  Grobe Version des Zeitplans erstellen(1h)  Expertenbesuch(1h)  Kriterienkatalog analysieren(0.5h)  Konzept für die Realisierung erstellen(1h)  Datenbank Modell erstellen(0.5h)  Testkonzept erstellen(1h)  Zeitplan festlegen(2h)  Lösungsvariante festlegen(0.5h)  Projektumgebung einrichten(1h) |
| Erledigte Arbeiten: | Aufbau der Dokumentation(Partiell) (1h)  Grobe Version des Zeitplans(1h)  Expertenbesuch(0.5h)  Kriterienkatalog studiert(0.5h)  Konzept für die Realisierung erstellt(0.5h)  Datenbankmodell erstellt(1h)  Projektumgebung einrichten(1h)  Lösungsvariante festlegen(0.5h) |
| Detailbeschrieb: | Der Erste Tag war ein Planungs- und Einrichtungstag.  Wir haben zuerst eine Einführung von Lara erhalten und dann haben wir festgestellt, dass Jürg uns den falschen Projektantrag mit den falschen Anforderungen gegeben hat.  Die wichtigste Änderung ist die Anforderung der Datenbank(sie muss nur noch zentral anstatt relational sein), was es mir ermöglicht MongoDB oder einfach ein JSON zu verwenden. Ansonsten hatten die Änderungen keinen nennenswerten Effekt.  Da die Datenbank nun nicht-relational ist, musste ich kein ERD mehr machen.  Der wichtigste Punkt des heutigen Tages war das Expertengespräch. Zur Vorbereitung habe ich den Zeitplan, sowie den Grundaufbau der Dokumentation erstellt. Im Expertengespräch habe ich dann bereits erstes wichtiges Feedback zum Zeitplan und der Doku erhalten. Die wichtigsten Punkte sind folgende:  - Tage im Zeitplan in sinnvolle Blöcke aufteilen  - Tasks im Zeitplan sollen dokumentieren bereits beinhalten.  - Der Zeitplan soll nicht einfach eine Treppe sein.  - Die Dokumentation muss in den Konzeptionellen und den Praktischen Teil aufgeteilt sein.  Der Zeitplan wurde schnell erledigt, allerdings habe ich ihn, wie oben erwähnt, noch einmal angepasst.  Ich hatte hier Mühe einzuschätzen, wie viel Zeit ich für den jeweiligen Task brauche.  Zum Schluss des Tages habe ich die Projekte generiert und das Github Repo erstellt. Auch hier hatte ich keine grossen Probleme. |
| Hilfestellungen: | Zeitplan festlegen: Vorlage Zeitplan Alpay Ildrim [1]  Aufbau der Dokumentation: Dokumentation M326 |
| Erfolge und Missgeschicke: | Missgeschicke:   * Projektantrag Überarbeitung   Erfolge:   * Datenbank kann mit JSON gemacht werden |

|  |  |
| --- | --- |
| Datum: | 17.3.2023 (2. Tag, Freitag) |
| Geplante Arbeiten: | Implementieren der Datenbank(1h)  Implementieren der REST API(2h)  Implementieren der Routes(3h) |
| Erledigte Arbeiten: | Implementieren der Datenbank(2h)  Implementieren der REST API(2h)  Implementieren der Routes(2h) |
| Detailbeschrieb: | Als erstes habe ich heute die Datenbank implementiert.  Für mich keine Challenge, da ich gerade eben(bei der Arbeit) eine andere gemacht habe, welche noch ein wenig komplexer war.  Trotzdem habe ich mehr Zeit benötigt, als erwartet, vor allem weil ich noch nicht so viel dokumentiert habe.  Die REST API ist das Grundgerüst des Backends, dazu gehören Error handling, Interfaces, Klassen und Not Found responses. Diese habe ich in der eingeplanten mehr oder weniger fertig gemacht, allerdings habe ich viel zu wenig Zeit für die Dokumentation aufgewendet. Hier war auch das YouTube Tutorial noch eine gute Hilfestellung, welches mir mit Errorhandling noch stark geholfen hat.  Die Routes haben definitiv länger als erwartet gebraucht, ich hatte heute Abend bis zu einem gewissen Grad auch ein wenig Konzentrationsschwierigkeiten. Ansonsten habe ich mit diesem Task das Backend eigentlich fertig gemacht. |
| Hilfestellungen: | Codinggarden YouTube Tutorial[2] |
| Erfolge und Misserfolge: | Erfolg:   * Ganzes Backend gemacht   Misserfolge:   * Zu viel Zeit für gewisse Routes aufgewendet * Kaum Zeit für Dokumentation aufgewendet |

|  |  |
| --- | --- |
| Datum: | 20.3.2023 (3. Tag, Montag)  Anmerkung: Dieser Tag ist der Ersatz für den Mittwoch, an dem ich krank war. |
| Geplante Arbeiten: | Login und Signup Pages(2h)  ToDo Page implementieren(2h)  Admin-Panels implementieren(partiell)(2h) |
| Erledigte Arbeiten: | Fertigstellung Zeitplan(1h)  Login und Signup(3h)  ToDo Page implementieren(2h) |
| Detailbeschrieb: | Heute habe ich noch den Zeitplan fertiggestellt. Auch das Testkonzept muss ich noch machen und sollte dies nicht zu weit herausschieben.  Das Login und Signup hat mir keine grossen Probleme bereitet, allerdings habe ich auch keine Material Komponenten verwendet, weshalb das Design noch etwas zu wünschen übrig lässt.  Die ToDo Seite hat mir eher Probleme bereitet, da ich mit dem Table Mühe hatte. |
| Hilfestellungen | Offizielle Angular Dokumentation [3]  Angular Material Docs [4] |
| Erfolge, Missgeschicke: | Erfolge:   * Der Service ist heute schon fertig geworden und ich muss ihn nicht kontinuierlich implementieren.   Missgeschicke:   * Ich bin mit der ToDo Page in Verzögerung gekommen, das muss ich nun aufarbeiten * Ich habe noch fast nichts dokumentiert, das muss ich alles noch aufholen |

|  |  |
| --- | --- |
| Datum: | 21.3.2023 (4. Tag, Dienstag) |
| Geplante Arbeiten: | Admin Panels implementieren(2h)  Testumgebung einrichten(2h) |
| Erledigte Arbeiten: | Admin Panels implementieren(partiell)(2h) |
| Detailbeschrieb: | An diesem Punkt habe ich ein Zeitproblem. Ich bin mehr oder weniger noch nirgends und hinke stark im Zeitplan hinterher. Aus diesem Grund habe ich entschieden, die Admin Panels sein zu lassen und mich auf die Dokumentation zu konzentrieren, da ich dort noch nicht genug gemacht habe. |
| Hilfestellungen | Kriterienkatalog IPA  Andere Dokumentationen von vorherigen Projekten |
| Erfolge und Misserfolge | Erfolge:   * Dokumentation teilweise aufgeholt   Misserfolge:   * Kaum Zeit für Admin Panels gehabt * Gar nicht bis zum Testen gekommen |

|  |  |
| --- | --- |
| Datum: | 14.3.2023 (5. Tag, Mittwoch) |
| Geplante Arbeiten: | Expertenbesuch(1h)  Testfälle testen(2h) |
| Erledigte Arbeiten: | Expertenbesuch(1h)  Testfälle testen(2h) |
| Detailbeschrieb: | Heute war der 2. Expertenbesuch, was auch heute wieder der wichtigste Tagespunkt war. Da die Dokumentation schon um einiges vollständiger ist, konnte ich bereits weiteres Feedback einholen, wie ich eine optimale Dokumentation schreibe. Der für mich wichtigste Punkt ist, dass die Dokumentation auch viel reflektieren und meine persönlichen Erfahrungen beschreiben soll. Bis jetzt habe ich nur fachliche Teile gemacht.  Ich habe auch erfasst, dass der 2. Teil der Dokumentation eher der fachliche ist, wo die App beschrieben wird und der 1. Teil vor allem auch persönliche Erfahrungen und Take-Home Messages aufzeigen soll.  Um die Testfälle durchzuführen, musste ich natürlich zuerst das Testkonzept erstellen. Das habe ich zwar früh eingeplant, aber da ich am 1. Tag keine Zeit mehr hatte, habe ich das herausgeschoben. Hier hat auch der Experte Steinmann noch wichtiges Feedback gegeben: Wenn ich etwas nicht machen kann, nicht damit warten bis es ein freies Fenster gibt, sondern alle anderen Arbeiten herausschieben und diese Arbeit erledigen. Vor allem wenn es so eine wichtige Arbeit, wie das Testkonzept ist |
| Hilfestellungen: | Experte Remo Steinmann |
| Erfolge und Misserfolge: | Erfolge:   * Testkonzept erledigt. * Getestet * Wichtiges Feedback vom Gespräch umgesetzt   Misserfolge:   * Keine |

|  |  |
| --- | --- |
| Datum: | 14.3.2023 (6. Tag, Freitag) |
| Geplante Arbeiten: | Reflexion und Fazit(2h)  Finalisierung und Abgabe(2h) |
| Erledigte Arbeiten: |  |
| Detailbeschrieb: |  |
| Hilfestellungen: |  |
| Erfolge und Misserfolge: |  |

# Teil 2: Praktischer Teil

## 2.1 I – Informieren

In diesem Teil werden Informationen gesammelt und allfällige Fragen geklärt, sowie die Aufgabenstellung definiert. Die wichtigsten Punkte sind in der Dokumentation dokumentiert.

### 2.1.1 Ausgangslage

Im Rahmen des M223 soll eine Projektarbeit mit dem Arbeitsaufwand von 33h durchgeführt werden, wovon 15h in die Dokumentation, 15h in die Realisierung und 3h ins testen investiert sollen. Das Projekt soll eine IPA simulieren, aber in einem etwas kleinerem Umfang. Die allgemeinen Kriterien sind zwar gleich, jedoch mussten nur 3 individuelle Kriterien ausgewählt werden, anstelle der 7, welche an der richtigen IPA verlangt werden.

Abgabepunkt: 24.3.2023 11:30 Uhr

### 2.1.2 Umsetzung

Die Umsetzung wird, wie man Am Punkt 2.1 bereits erkennen kann, mit IPERKA durchgeführt.

Die Phasen haben alle ihre eigenen Punkte und haben eine Beschreibung zum jeweiligen Punkt bereits geschrieben.

### 2.1.3 Kriterien

#### 2.1.3.1 Akzeptanzkriterien

* Realisierung der Applikation
* Saubere Dokumentation
* Pünktliche Abgabe gemäss definiertem Datum

#### 2.1.3.2 Ausgewählte individuelle Kriterien

1. Individuelles Bewertungskriterium

|  |  |
| --- | --- |
| *Nummer Katalog-Kriterium - Bezeichnung* | |
| 194 - Plausibilisierung der Benutzer-Eingaben | |
| *Definition (Leitfrage)* | |
| Eingaben des Users werden validiert, sowie wird der User auf Pflichtfelder hingewiesen. | |
| *Gütestufe 3* | *Gütestufe 2* |
| Alle Eingabefelder werden überprüft. Es ist eindeutig gekennzeichnet, welche Felder Pflichtfelder sind. Für den Benutzer ist ersichtlich, welche Wertebereiche zulässig sind. Findet die Plausibilisierung eine Fehleingabe, so wird der Benutzer mit konkreten Hinweisen geführt, das entsprechende Feld wird aktiviert. | Plausibilisierung findet statt, Feedback an Benutzer ist mangelhaft/nicht eindeutig/unvollständig. Nur korrekte Daten werden übermittelt |
| *Gütestufe 1* | *Gütestufe 0* |
| Eingaben werden plausibilisiert, aber bei Fehlern oder fehlenden Eingaben sind die bisher gemachten Eingaben verloren oder die fehlerhaften Eingaben werden trotzdem übermittelt. Oder: es werden nicht alle Eingaben ueberprueft, welche ueberprueft werden sollten. | Es findet keine Plausibilisierung statt. |

1. Individuelles Bewertungskriterium

|  |  |
| --- | --- |
| *Bezeichnung* | |
| 250 - Schichtentrennung (Applikation) | |
| *Definition (Leitfrage)* | |
| Gibt es Presentation Logic, Application Logic und Service Layer. Sind sie sinnvoll unterteilt. | |
| *Gütestufe 3* | *Gütestufe 2* |
| 1. Gibt es eine Persistenz-, eine Service- und eine Präsentationsschicht mit klarer Schichtentrennung 2. Die Schichten sind stimmig aufgebaut und sinnvoll auf Module aufgeteilt 3. Trennung der Packagestruktur ersichtlich 4. Sprechende Namensgebung 5. Firmenvorgaben eingehalten Alle 5 Aspekte erfüllt | 4 Aspekte erfüllt |
| *Gütestufe 1* | *Gütestufe 0* |
| 3 Aspekte erfüllt | Weniger als 3 Aspekte erfüllt |

1. Individuelles Bewertungskriterium

|  |  |
| --- | --- |
| *Bezeichnung* | |
| 123 - Kommentare im Quellcode | |
| *Definition (Leitfrage)* | |
| Ist der Code sinnvoll Dokumentiert | |
| *Gütestufe 3* | *Gütestufe 2* |
| Der Sourcecode der Applikation ist vollumfänglich kommentiert: 1. Funktionen, Parameter, Rückgabewerte, 2. Wichtige Stellen im Sourcecode, 3. weitere zusätzliche/nützliche Kommentare. | Der Sourcecode der Applikation ist im Grossen und Ganzen kommentiert. Einer der genannten Punkte könnte präziser sein. |
| *Gütestufe 1* | *Gütestufe 0* |
| Der Sourcecode der Applikation ist nur teilweise kommentiert. | Der Sourcecode der Applikation ist unzureichend kommentiert. |

### 2.1.4 Use-Cases

Für was kann die App verwendet werden?

- ToDos anywhere: Man soll die ToDo App wo immer man ist öffnen können, ohne eine richtige App zu installieren(öffnen im browser).

- Whiteboard ersetzen. Man kann das Whiteboard mit

## 2.2 P – Planen

In dieser Phase des Projektes geht es um den Ablauf und Aufbau des Projektes dazu gehören die geplanten Arbeiten, sowie der Zeitplan.

Testweise sind nur e2e Tests geplant, welche von Hand nach Testkonzept ausgeführt werden (siehe Testkonzept unter K – Kontrollieren)

### 2.2.1 Geplante Seiten

Für das Frontend sind folgende Seiten geplant:

* Login & Signup
* ToDo List
* Create ToDo Page
* Admin Panel mit User List
* Admin Panel – Edit User Page
* Admin Panel – Edit ToDos Page

### 2.2.2 Zeitplan

Chart

Description automatically generated

## 2.3 E – Entscheiden

Da es für alles verschiedene Lösungswege gibt, ist hier die Entscheidung für den jeweilige Projektteil dokumentiert.

### 2.3.1 Lösungswege

#### 2.3.1.1 Frontend

Was gibt es für Optionen(und warum man sie nicht nehmen sollte):

* Nuxt(Vue)
* Gradle
* Dart
* Angular

Die oben genannten Optionen sind die, mit denen ich schon einmal zu tun hatte und ich darum evaluierte.

Ich habe mich schlussendlich für Angular entschieden, da ich damit am meisten Erfahrung habe und es auch am einfachsten zu verwenden ist. Ebenso ist Performance, bezüglich den vielen Dependencies, kein Problem.

#### 2.3.1.2 Backend

Diese Optionen gibt es für das Backend:

* C#
* Java
* Rust
* Golang
* ANSI-C
* JavaScript
* TypeScript

Auch hier habe ich alle Optionen, mit denen ich irgendwo Erfahrung habe evaluiert.

Schlussendlich habe ich mich auch hier für die Option entschieden, mit der ich am meisten Erfahrung habe: TypeScript.

TypeScript habe ich JavaScript bevorzugt, da ich eine Type annotierung noch gerne habe, damit ich weiss, was sich in einem Objekt befinden kann.

#### 2.3.1.3 Datenbank

Auch für die Datenbank gibt es folgende Optionen:

* MongoDB
* MariaDB
* JSON

Hier Gibt es vor allem die Unterscheidung zwischen Datenbanken, die auf SQL aufgebaut sind und welche, die einfach JSON verwenden. Ich habe mich für JSON entschieden, da ich bereits selber einen JSON DataStorage gemacht habe und ich da erfahrung habe. MongoDB war auch eine sehr gute Option, da man, mit dem Mongoose Modul, MongoDB einfach und ohne grossen Aufwand verwenden kann. MariaDB habe ich ever weniger gut evaluiert, da ich nicht so viel Erfahrung mit SQL habe und es auch etwas komplexer ist.

## 2.4 R – Realisieren

In dieser Phase habe ich die 3 Layers des Projektes realisiert. Hier wurde dann immer klarer, wie knapp die Zeit ist und ich dann langsam

### 2.4.1 – Frontend

Das Frontend umfängt den Teil, der mit Angular gemacht ist.

Hier habe ich als Hilfestellung die offizielle Angular Dokumentation[3], welche auch ein gutes Tutorial enthält.

Die Tables sind von Angular Material[4], welches bereits Styling dafür bereitstellt.

#### 2.4.1.1 Struktur

Graphical user interface, text, application

Description automatically generated with medium confidenceDie Struktur des Frontends ist bereits von Angular generiert und leicht verständlich. Nichtsdestotrotz möchte ich die Struktur nochmals beschreiben.

Im Projekt Folder befinden sich die ganzen node\_modules und configs. Im src Folder(siehe Bild rechts) befindet sich der ganze Source code, darunter der Index(entrypoint), sowie die global styles. Die Environments bestimmen, ob der Production mode enabled ist und in meinem Fall, ob die Backend Daten gemockt werden oder http Requests Ausgefhrt werden.

Text

Description automatically generated

Im App Folder befinden sich die einzelnen Components und Services, sowie das app.module.ts(alle deklarationen und imports) und das app-routing.module.ts(Das File, welches die Routes bestimmt). Die Components sind in meinem Fall eigentlich immer die Pages(siehe 2.2.1), bis auf die Menu-Bar, welche ein Navigationsmenu darstellt.

Graphical user interface, text, email, website

Description automatically generated

Ein Component besteht immer aus HTML, CSS und TS files. Ein .spec.ts ist ein optionales Testfile, welches für Unit Testing verwendet wird

A blue screen with white text

Description automatically generated with medium confidence

Ein Service besteht nur aus einem TS file mit einem optionalen .spec.ts file, welches ich aber nicht habe.

#### 2.4.1.2 Routes

In meinem Projekt habe ich die Routes folgendermassen definiert:

|  |
| --- |
| const routes: Routes = [    { path: "login", component: LoginComponent },    { path: "signup", component: SignupComponent },    { path: "view-todo", component: ViewTodosComponent },    { path: "edit-todo", component: EditTodoComponent },    { path: "new-todo", component: CreateTodoComponent },    { path: "edit-user/:id", component: EditUserComponent},    { path: "edit-todo/:id", component: EditTodoComponent },    { path: "admin-panel", component: AdminPanelComponent},    { path: "", component: LoginComponent },    { path: "\*", component: LoginComponent },  ]; |

Wie man sehen kann, ist die LoginPage die LandingPage.

Ebenso sind, für die beiden Edit Pages, Route Params notwendig, um das dementsprechnde ToDo zu editieren.

#### 2.4.1.3 Login/Signup

Das Login Page ist die LandingPage, jedoch ist sie ziemlich ähnlich, weshalb ich diese hier zusammenfasse. Der Unterschied ist vor allem, dass die Login Page einen Token request macht, während die Signup Page mehr Inputs hat und einen Request auf Signup macht.

Beide enthalten jedoch ein simples form, welches mehrere Inputs, einen Submit Button und einen Error output hat.

|  |
| --- |
| <h1>signup</h1>      <form #signupForm (ngSubmit)="onSubmit()" >          <input type="email" name="username" id="username" placeholder="\*e-mail"v [(ngModel)]="email"><br>          <input type="number" name="username" id="username" placeholder="\*age" value="0" [(ngModel)]="age"><br>          <input type="text" name="username" id="username" placeholder="\*name" [(ngModel)]="name"><br>          <input type="password" name="password" id="password" placeholder="\*password" [(ngModel)]="password"><br>          <label>Passwort muss 8 Zeichen lang sein<br>\*Pflichtfelder</label>          <label [(ngModel)]="error"></label><br>          <button type="submit">submit</button>      </form> |

Danach wird mit der onSubmit() Methode der Request verarbeitet und an den Server gesendet.

Bei Success wird man bei beiden Forms automatisch eingeloggt, während ein Error zu einem Output führt.

Dieses Beispiel lässt sich in dem Sinn auch verallgemeinern, da in jedem Component(mit einem Server Request), eine Methode aus dem ServerAccesService aufgerufen wird. Je nach Environment werden dann die jeweiligen Daten(Mock Daten oder Server Daten) provided.

|  |
| --- |
| this.userService.createUser(this.email, this.age, this.name, this.password).subscribe({          next: response => {            console.log('success: post token response', response.id);            this.router.navigateByUrl("/view-todo");            this.state.UpdateUser(response.id);          },          error: err => console.log(err),        }); |

#### 2.4.1.4 View-ToDo

#### 2.4.1.x Service

Der Server Besteht aus Methoden, welche alle nach dem folgenden Prinzip aufgebaut sind:

|  |
| --- |
| updateToDo( title: string, description: string, id: string): Observable<ToDo> {        let params: URLSearchParams = new URLSearchParams();        params.set("surveyid", id);        const data = this.http.patch<ToDo>('http://localhost:5000/todos/' + id, { title, description });        return data;      } |

Der RETURN Value ist bei allen Methoden ein Observable, welches einen gewissen Content enthält.

Jeder Server Request wird Intercepted, damit die Authorization headers gesetzt werden können.

Ich habe den oben genannten Request gewählt, da in diesem auch demonstriert wird, wie URL params gesetzt werden. Der 2. Teil des Request gibt dann natürlich den Body mit.

#### 2.4.1.x Interceptor

Der Interceptor hat die simple Aufgabe, jeden HTTP request mit einem Authorizationheader zu versehen. Man könnte noch verschiedene Interceptoren für verschiedene Requests machen, allerdings ist in meinem Fall einer völlig ausreichend.

|  |
| --- |
| export class HttpInterceptorMain implements HttpInterceptor {    constructor(private userService: IServerAccessService) {    }    intercept(httpRequest: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {        const modifiedRequest = httpRequest.clone({        headers: httpRequest.headers.set('Authorization', this.userService.currentUser.email),      });      console.log('interceptor: ', this.userService.currentUser.email);      return next.handle(modifiedRequest);    }  } |

### 2.4.2 – Backend

### 2.4.3 - Datenbank

## 2.5 K – Kontrollieren

## 2.5.1 Testkonzept

|  |  |
| --- | --- |
| Testfall #1 | Login |
| Testbeschreibung | Der User kann sich (mit bestehendem Login) einloggen |
| Testablauf | * Bewegen auf URL der Webseite * Eingeben eines Bestehenden Logins (E-Mail und Passwort) * Drücken des Login Buttons |
| Erwartetes Resultat | Der User befindet sich auf der Landing Page und kann seine ToDos einsehen. |

|  |  |
| --- | --- |
| Testfall #2 | Login mit falschen Daten |
| Testbeschreibung | Der User versucht sich mit falschen Daten auf die Webseite einzuloggen und erhält dann eine Fehlermeldung |
| Testablauf | * Bewegen auf URL der Webseite * Eingeben eines falschen Logins (E-Mail und Passwort) * Drücken des Login Buttons |
| Erwartetes Resultat | Der User hat eine Fehlermeldung erhalten und kann seine Daten neu eingeben. |

|  |  |
| --- | --- |
| Testfall #3 | Signup |
| Testbeschreibung | Der User kann ein neues Login erstellen |
| Testablauf | * Bewegen auf URL der Webseite * Klicken des Signup Buttons * Eingeben, der gewünschten User Daten * Drücken des Signup Buttons |
| Erwartetes Resultat | Der User befindet sich auf der Landing Page und könnte seine ToDos einsehen, wenn er welche hätte. |

|  |  |
| --- | --- |
| Testfall #4 | Signup mit falschen Daten |
| Testbeschreibung | Der User versucht ein neues Login zu erstellen, verwendet aber invalide Daten.  Getestet werden sollen zu kurzes Passwort, invalides Alter und invalide E-Mail Addresse |
| Testablauf | * Bewegen auf URL der Webseite * Klicken des Signup Buttons * Eingeben, der gewünschten Userdaten * Drücken des Signup Buttons |
| Erwartetes Resultat | Der User erhält die korrekte Fehlermeldung. |

|  |  |
| --- | --- |
| Testfall #5 | ToDo erstellen |
| Testbeschreibung | Der User kann ein neues ToDo erstellen |
| Testablauf | * Bewegen auf URL der Webseite * Eingabe der Nutzerdaten * Drücken des Login Buttons * Drücken des “ToDo erstellen” Buttons * ToDo Felder ausfüllen * Auf bestätigen klicken |
| Erwartetes Resultat | Der User ist wieder auf der Landing page und sieht das neu Erstellte ToDo mit Status Offen |

|  |  |
| --- | --- |
| Testfall #6 | ToDo Editieren |
| Testbeschreibung | Der User kann ein neues Login erstellen |
| Testablauf | * Bewegen auf URL der Webseite * Klicken des Signup Buttons * Drücken des Login Buttons |
| Erwartetes Resultat | Der User befindet sich auf der Landing Page und kann seine ToDos einsehen. |

## 2.5.2 Testprotokoll

## 2.6 A - Auswerten

# 3 Quellenverzeichnis

[1] <https://github.com/AYIDouble/IPA-2018-Informatiker-EFZ-Applikationsentwicklung-Alpay-Yildirim>

IPA 2018 Alpay Ildrim

[2] [https://www.youtube.com/watch?v=vDLE8hqzA8I](%20https:/www.youtube.com/watch?v=vDLE8hqzA8I)

CodingGarden – Build a CRUD API with Express, TypeScript, MongoDB, Zod and Jest

[3] <https://angular.io/docs>

Angular official docs

[4] <https://material.angular.io/components/categories>

Angular Material Docs

# 4 Glossar

# 5 Anhang

## 5.1 Code