



## Dokumentation

---

### zapp!

*Eine Applikation zur Vorbereitung auf Prüfungen und Zertifizierungen*

Autor(en) des Dokuments	René BrodKorb	Ersterstellung:	21.02.2019
Dateiname	zapp! - Dokumentation BrodKorb		
Seitenanzahl	38	© 2019 René BrodKorb	Version 1.7

# Inhaltsverzeichnis

1	Einleitung.....	4
2	Anforderungen und Informationen.....	4
2.1	Das Angular-Framework.....	4
2.2	Anforderungen an die Applikation (Stage 1) .....	5
2.3	Zielgruppe .....	5
2.4	Vergleich mit bestehenden Lösungen.....	5
2.5	Zertifizierungen .....	6
2.6	Ressourcen .....	6
2.6.1	Anwender .....	6
2.6.2	Programmierung .....	6
2.7.1	Stage 2 .....	6
2.7.2	Stage 3 (Optional).....	6
3	Umsetzung der Applikation .....	6
3.1	Stage 1 – Fragenliste, Einzelansicht, Lernmodi und Navigation .....	6
3.1.1	Aufbau der Applikation .....	7
3.1.2	Ablaufplan .....	9
3.1.3	EPK (Ausgehend vom Menü, nur Funktionalität der Fragebeantwortung) .....	10
3.1.4	Angular-Klassen .....	11
3.1.5	Angular-Komponente.....	11
3.1.6	Anzeige einer Liste von Fragen (abb. 4).....	11
3.1.7	Anzeige einer Einzelfrage nach Auswahl in der Liste (Abb. 6) .....	11
3.1.8	Auswählen von Antworten .....	12
3.1.9	Lernmodus und Prüfungsmodus .....	12
3.1.10	Navigieren zur nächsten/vorherigen Frage .....	14
3.1.11	Auswertung der Antworten .....	15
3.2	Stage 2.....	16
3.3	Erweiterungen.....	18
3.3.1	Statistiken .....	18
3.3.2	Fortschrittsbalken.....	19
3.3.3	Responsive Design .....	19
3.3.4	Nicht umgesetzte Erweiterungen.....	20
3.3.5	Native Android Applikation.....	20
4	Fazit .....	21
5	Anhang .....	22
5.1	Datenstrukturbaum .....	22
5.2	Quellcode.....	23
5.2.1	Index.html.....	23
5.2.2	app-routing.module.ts .....	23
5.2.3	app.component.css .....	24
5.2.4	app.component.html .....	24

5.2.5	app.component.ts .....	25
5.2.6	app.module.ts .....	25
5.2.7	about.component.html .....	26
5.2.8	about.component.ts .....	26
5.2.9	frage-details.component.css.....	26
5.2.10	frage-datails.component.html .....	28
5.2.11	frage-datails.component.ts .....	32
5.2.12	frage-liste-item.component.html .....	34
5.2.13	frage-liste-item.component.ts .....	34
5.2.14	fragenliste.component.html .....	34
5.2.15	fragenliste.component.ts .....	34
5.2.16	home.component.html.....	35
5.2.17	home.component.ts .....	35
5.2.18	shared/class-antworten.ts.....	35
5.2.19	shared/frage.ts .....	35
5.2.20	shared/frage-service.service.ts.....	36
5.2.21	fragen_von_tom.json (AUSZUG).....	37
5.4	Reviewvermerke.....	38
5.4.1	Ersterstellung.....	38
5.4.2	Revision 1.....	38
5.4.3	Revision 2.....	38
5.4.4	Revision 3.....	38
5.4.5	Revision 4.....	38
5.4.6	Revision 5.....	38
5.4.7	Revision 6.....	38
5.4.8	Revision 7.....	38
5.5	Historie der Dokumentversionen.....	38

# 1 Einleitung

Dieses Dokument beschreibt das Projekt „zapp!“, einen Prüfungs- und Zertifizierungssimulator. Dieses Projekt wurde im Rahmen der Fachinformatiker-Umschulung der BITLC Business IT Learning Center GmbH im Modul Enterprise-Development mit Angular unter Linux durchgeführt und kommt daher in verschiedenen Versionen vor.

Dieses Dokument beschreibt die Anforderungen, die Umsetzung und den Aufbau der Applikation zapp!

Die Fachinformatiker-Umschulung beim BITLC ist modular aufgebaut. Im Zuge des Angularmoduls entschließen wir uns einen Prüfungssimulator zur Vorbereitung auf Prüfungen und externe Zertifizierungen mit dem Angular-Framework zu realisieren. Basis ist der „lpisim“ Prüfungssimulator zur Vorbereitung auf LPIC-1 Zertifizierungen. (Siehe 2.4)

## 2 Anforderungen und Informationen

### 2.1 Das Angular-Framework

Angular ist ein Web-Front-End-Applikationsframework das TypeScript als Superset zu JavaScript, welches ausschließlich in Webbrowsern verwendet wird, verwendet. Es beinhaltet zudem CSS3 und HTML5 Komponenten. Angular ist ein Open-Source Projekt von Google und wird von einer Community aus Einzelpersonen und Unternehmen stetig weiterentwickelt. Die, beim Erstellen der App, aktuellste Version ist Angular 7.

Angular priorisiert durch den Mobile-First-Ansatz mobile Plattformen, ist aber auch auf Desktop-PCs problemlos nutzbar. Zudem besteht die Möglichkeit aus Angularanwendungen native Smartphone Apps zu erzeugen.

Der Vorteil von One-Page-Webapplikationen ist, dass Teilbereiche nach Bedarf nachgeladen oder generiert werden. Am Beispiel dieses Projektes zapp! (Siehe Abb. 1) lässt sich gut erkennen, dass das Menü, sowie der Fußbereich statisch sind. Diese werden nur einmal geladen. Der Bereich dazwischen ist dynamisch. Dieser Teil wird aus den Einzelkomponenten generiert, sobald diese benötigt werden.



Abbildung 1: Statische und Dynamische Teile

## 2.2 Anforderungen an die Applikation (Stage 1)

Die Applikation (App) soll in allen aktuellen Webbrowsern lauffähig sein und soll in der ersten Ausbaustufe (Stage 1) folgende Grundfunktionen besitzen:

- ⚡ Anzeige einer Liste von Fragen
- ⚡ Anzeige einer Einzelfrage nach Auswahl in der Liste
- ⚡ Anzahl der Fragen: 10
- ⚡ Auswählen von Antworten
- ⚡ Abfrage von Multiple-Choice-Fragen
- ⚡ Lernmodus - zeigt richtige Antworten an
- ⚡ Prüfungsmodus - wertet Eingabe des Benutzers aus
- ⚡ Navigieren zur nächsten/vorherigen Frage

## 2.3 Zielgruppe

Zielgruppe sind sowohl Teilnehmer von Zertifizierungs-Schulungen als auch Personen, die sich privat auf eine Prüfung vorbereiten wollen.

## 2.4 Vergleich mit bestehenden Lösungen

Aktuell gibt es einige Lösungen, die aber nicht das Lernen des Stoffes in den Vordergrund stellen. Vielmehr liegt bei den vorhandenen Lösungen der Fokus auf der Simulation einer Prüfung. Zudem sind einige der vorhandenen Lösungen nur als klassisches Programm (z.B. LPISIM) verfügbar und können daher nur auf Geräten genutzt werden, auf denen sie installiert sind. Also nicht per Webbrowser oder Smartphone-App nutzbar.

The screenshot shows a web-based application window titled "Probeprüfung: Quickie/LPI101a". At the top left, there is a checkbox labeled "Wiedervorlage". At the top right, it says "Verbleibende Zeit: 0:04:51". The main content area displays "QNr: 1 --" followed by the question: "Which of the following information is stored within the BIOS? (Choose TWO correct answers.)". Below the question are five numbered options, each with an unchecked checkbox:

1. ☐ Hardware configuration
2. ☐ Boot device order
3. ☐ The system's hostname
4. ☐ Timezone
5. ☐ Linux kernel version

At the bottom of the window, there is a footer bar. On the left, it says "Wählen Sie alle zutreffenden Antworten". On the right, it says "Frage 8 von 10". Below this bar are three buttons: "Vorige Frage", "Nächste Frage", and "Prüfung beenden".

Abbildung 2: Beispiel lpisim

## 2.5 Zertifizierungen

Institute und Unternehmen wie Microsoft, das LPI Linux Professional Institute, Cisco und andere bieten Zertifizierungen für seine/ihre Produkte und Dienste an (z. B. LPIC-1). Man kann sich weltweit bei zugelassenen Zertifizierungsstellen prüfen und somit zertifizieren lassen.

## 2.6 Ressourcen

### 2.6.1 Anwender

Der Nutzer benötigt lediglich einen Computer mit aktuellem Webbrowser, Eingabegeräte (z.B. Maus) und einen Internetzugang.

### 2.6.2 Programmierung

- ⚡ Computer und Eingabegeräte
- ⚡ Angular-Framework
- ⚡ CSS-Framework Semantic-UI
- ⚡ Aktueller Webbrowser, z.B. Mozilla Firefox, Google Chrome oder ähnlich.
- ⚡ IDE z.B. Microsoft Visual Studio Code
- ⚡ Literatur, z.B. Bücher, Internetquellen
- ⚡ CentOS 7 mit Angular-Framework Version 7 und npm (node package manager)

## 2.7 Aufbauende Entwicklungsstufen (Stage 2+)

### 2.7.1 Stage 2

Aufbauend auf Stage 1 sollten, bei ausreichender Zeit, auch folgende Fragentypen integriert werden.

- Single-Choice-Fragen und
- FillIn-Fragen

Diese beiden Funktionen wurden in die Applikation integriert. (Siehe Punkt 3.2)

### 2.7.2 Stage 3 (Optional)

Aufbauend auf Stage 1 und 2 sollten auch Bilderfragen gestellt werden können. (Z.B. das richtige Zuordnen oder Anordnen von Bildern.) Dies wurde im Rahmen des Projektes jedoch nicht realisiert.

## 3 Umsetzung der Applikation

### 3.1 Stage 1 – Fragenliste, Einzelansicht, Lernmodi und Navigation

Die erste Ausbaustufe soll, wie unter 2.2. beschrieben, eine Liste von zehn Multiple-Choice-Fragen, sowie eine Einzelansicht der Frage enthalten. Es soll zur nächsten oder vorherigen Frage navigiert werden können. Zudem soll eine Abfrage der Eingaben (Prüfung) bzw. die Anzeige richtiger Antworten realisiert werden.

### 3.1.1 Aufbau der Applikation

Beim Starten der Webapplikation erscheint zuerst die s. g. Home-Komponente. Dies wird durch das Routing-Module von Angular realisiert.

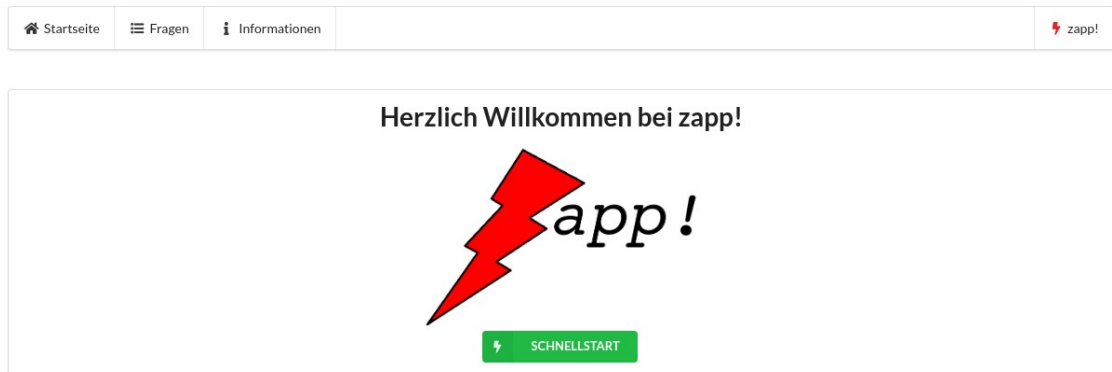


Abbildung 3: App Startseite

Von dort kann der Nutzer über das Menü zur Fragenliste oder zur Infoseite wechseln. (Über Router-Links)

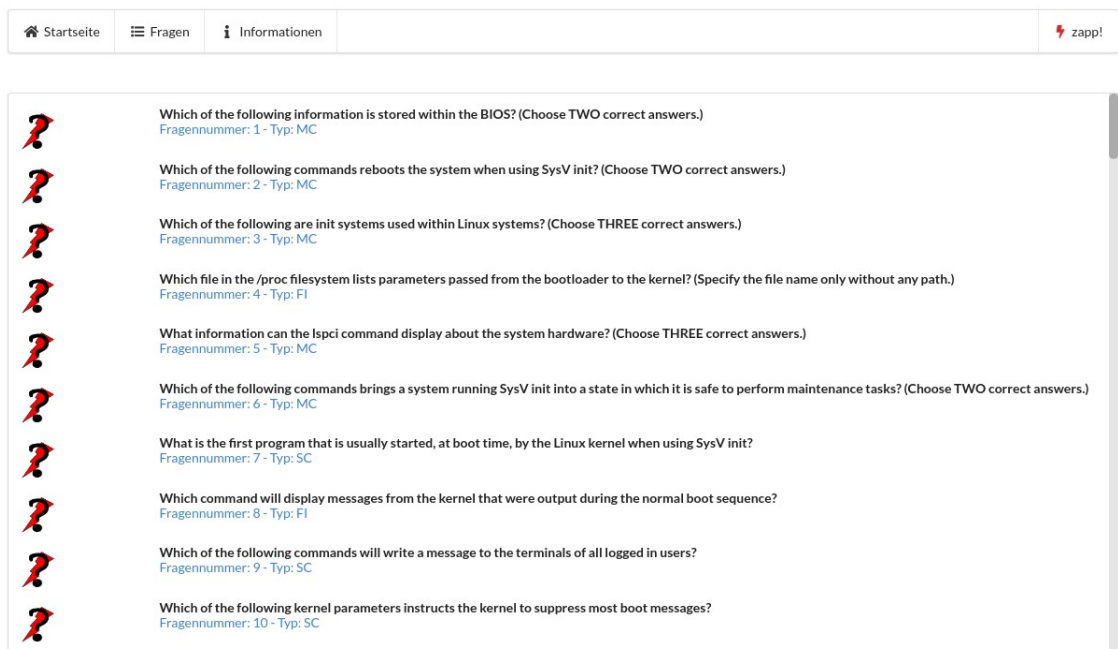


Abbildung 4: Fragenliste

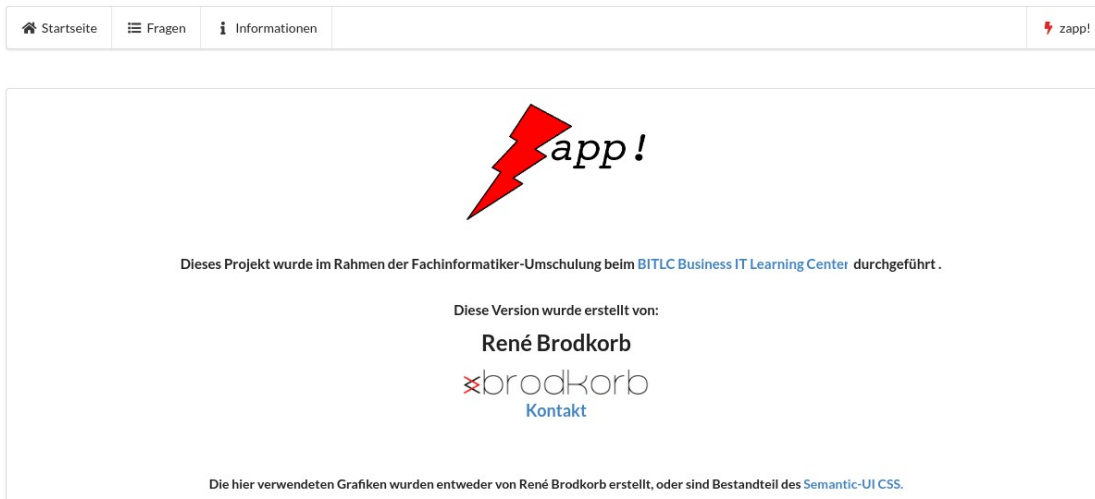


Abbildung 5: Info-Seite

Durch das klicken auf eine Frage in der Fragenliste wird die Einzelansicht der angeklickten Frage angezeigt.

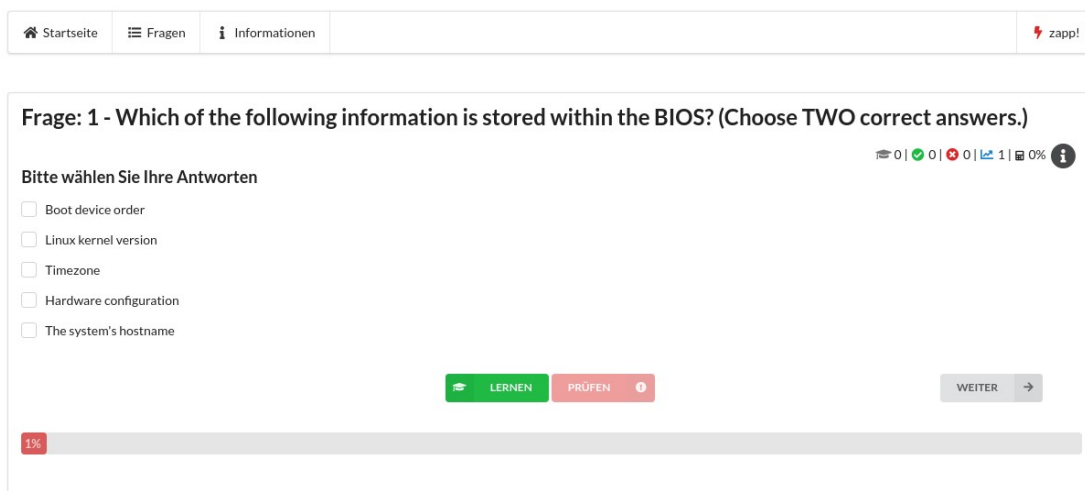
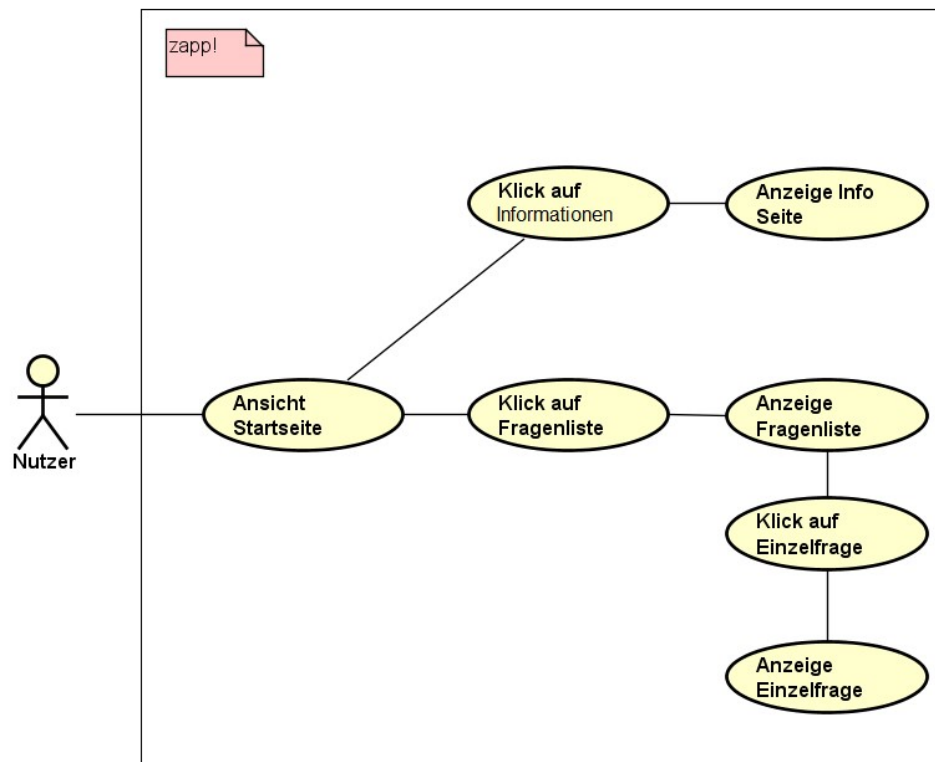


Abbildung 6: Einzelansicht Multiple-Choice-Frage

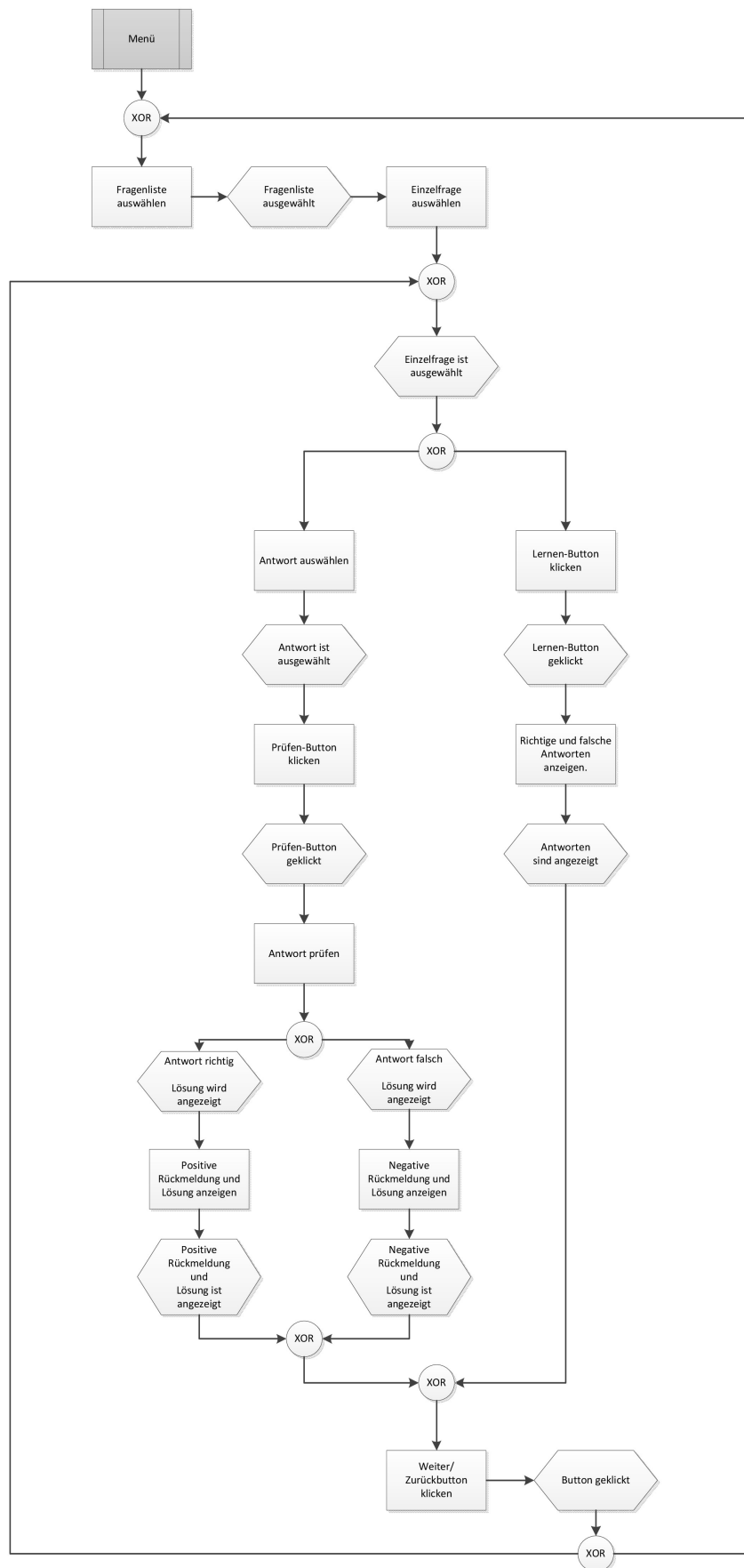
Eine Navigation zur Fragenliste, der Startseite oder der Info-Seite ist jederzeit durch das statische Menü gegeben.



### 3.1.2 Ablaufplan



### 3.1.3 EPK (Ausgehend vom Menü, nur Funktionalität der Fragebeantwortung)



### 3.1.4 Angular-Klassen

Um Fragen, Antworten und die Fragenlisten anzeigen zu können erstellen wir Daten-Klassen. Eine Frage-Klasse, die Attribute, sowie das Antworten-Array enthält. Dieses Antworten-Array ist ebenfalls eine Klasse, die Antworten-Klasse. Mit diesen Werten lassen sich Funktionen und Ausgaben realisieren. Einzelne Werte lassen sich später abfragen und auswerten.

Auszug Klasse Frage

```
export class Frage {
  constructor(
    public id: number,
    public fragentext: string,
    public hinweis: string,
    public antwort: ClassAntworten[]
    public typ: string,
    public first?: boolean,
    public last?: boolean
  ) {}
}
```

Auszug Klasse ClassAntworten

```
export class ClassAntworten {
  constructor(
    public id: number,
    public antworttext: string,
    public pruefung: boolean
  ) {}
}
```

### 3.1.5 Angular-Komponente

Durch den modularen Aufbau des Angular-Frameworks lassen sich Applikationen in Komponente (wie Module), s.g. components, unterteilen. Wir erstellen dazu notwendige Komponenten mit dem Node package manager **npm** nach. Dazu nutzen wir das Kommando **ng g component component-name**. Ähnlich lassen sich auch Klassen erzeugen. Dazu nutzt man folgendes npm Kommando: **ng g class classname**

Komponente bestehen i. d. R. aus einer .css-, einer .html- und einer .ts-Datei.

Die .ts-Datei beinhaltet TypeScript und somit die Logik/Funktionen. Die .html-Datei beinhaltet den sichtbaren Teil der Komponente. Die .css-Datei kann zusätzliche Styles für die .html-Datei enthalten.

Zwingend notwendig sind weder die .css-, noch die .html-Datei. Die Inhalte der beiden Dateien können auch direkt in der .ts-Datei integriert werden. Dies bietet sich z. B. bei sehr kurzen Codes an. Wird der Code hier zu lang, so wird er schnell, insbesondere bei der Fehlersuche, unübersichtlich.

### 3.1.6 Anzeige einer Liste von Fragen (abb. 4)

Um die Fragenliste anzeigen zu lassen, erstellen wir die fragenliste.component. Beim Klick im Menü auf „Fragenliste“ wird somit die fragenliste.component.html angezeigt. Der Inhalt wird dabei dynamisch erzeugt und besteht aus weiteren Komponenten. Es wird eine Schleife aus Elementen der frage-liste-item.component angezeigt.

Die Fragen der Fragenliste werden über einen Service, den frage-service.service, zur Verfügung gestellt. Wird eine Frage in diesen Service hinzugefügt, so wird diese automatisch in der Fragenliste angezeigt.

### 3.1.7 Anzeige einer Einzelfrage nach Auswahl in der Liste (Abb. 6)

Um die Fragenliste anzeigen zu lassen, erstellen wir die frage-details.component. Beim Klick auf eine Frage in der Fragenliste wird somit die frage-details.component.html angezeigt. Diese Datei ist die größte .html-Datei des Projektes.

Sie beinhaltet die Anzeige der Fragen, sowie die Buttons zur Navigation zwischen den Fragen und die Buttons zur Nutzung der Funktionen „Lernen“ und „Prüfen“. Darüber hinaus gibt es ein ⓘ „Info“-Icon. Dieses blendet ein fragespezifisches Themengebiet und aktuelle Statistiken ein.

Die wichtigsten Buttons in der Übersicht.

1. Weiter: Überspringt die aktuelle Frage und wechselt direkt zur nächsten Frage. Bei der letzten Frage wird der Weiter-Button ausgeblendet. Dafür erscheint ein „Zurück zur Frageliste“-Button.
 

→ WEITER

ZURÜCK ZUR LISTE →
2. Lernen: Deaktiviert die Möglichkeit der Antwortauswahl und zeigt die richtige(n) Antwort(n) an. Auswählen der Antworten / Prüfen ist dann nicht mehr möglich.
 

📖 LERNEN
3. Prüfen: Prüft die Antwort auf Richtigkeit und gibt bei einer falschen Antwort eine Popup-Fehlermeldung, mit einem Hinweis auf die falsche Antwort, aus.
 

❗ PRÜFEN
4. Zurück: Ähnlich wie Weiter-Button. Bei der ersten Frage ist der Button ausgeblendet.
 

← ZURÜCK

### 3.1.8 Auswählen von Antworten

Das Auswählen der Antworten geschieht durch Anwahl der Checkboxes. *Zur Auswertung siehe Punkt 3.1.11.*

🏠 Startseite ☰ Fragen ℹ Informationen ⚡ zapp!

**Frage: 1 - Which of the following information is stored within the BIOS? (Choose TWO correct answers.)**

🗣 0 | 🟢 0 | 🚫 0 | 📄 1 | 📊 0%

Bitte wählen Sie Ihre Antworten

☒ Boot device order  
☐ Linux kernel version  
☐ Timezone  
☒ Hardware configuration  
☐ The system's hostname

📖 LERNEN

❗ PRÜFEN

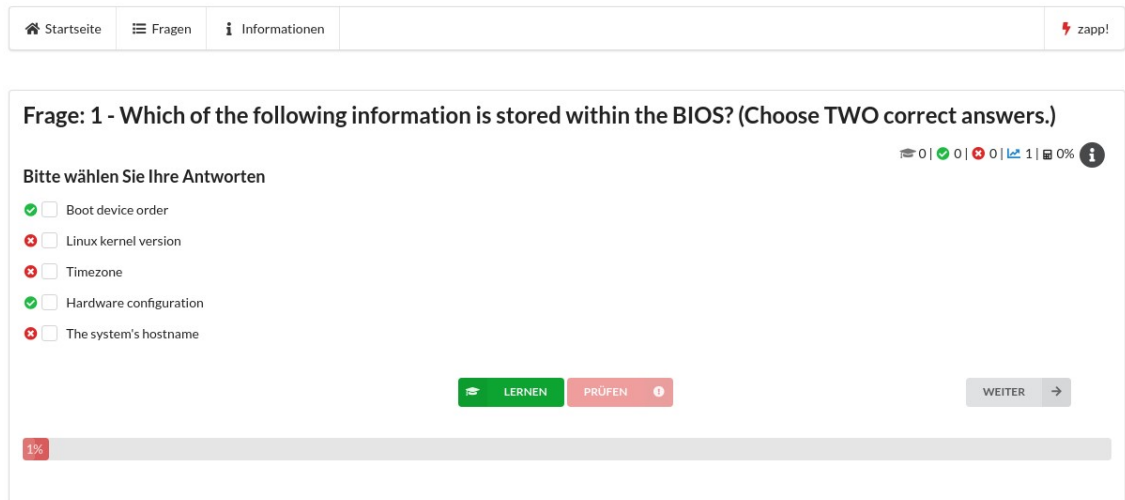
WEITER →

Abbildung 7: Checkboxes angewählt

### 3.1.9 Lernmodus und Prüfungsmodus

Kommt ein Nutzer auf eine Einzelfragenansicht, so ist automatisch der Lernmodus aktiviert. Der Nutzer kann sich mit dem „Lernen“-Button die richtigen Lösungen anzeigen lassen. Der „Prüfen“-Button ist in diesem Modus inaktiv. (Siehe Abb. 10))

Bei Fill-In Fragen können keine Antworten markiert werden. Daher wird bei diesen Fragen die korrekte Antwort als HTML-Schnipsel eingeblendet. (Ähnlich dem Hinweisbutton)



Startseite Fragen Informationen zapp!

Frage: 1 - Which of the following information is stored within the BIOS? (Choose TWO correct answers.)

Bitte wählen Sie Ihre Antworten

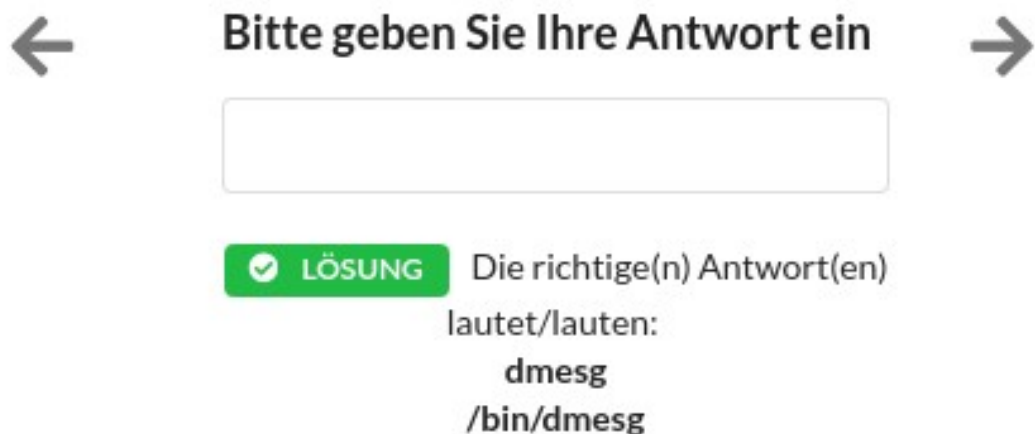
0 | 0 | 0 | 0 | 1 | 0%

- ☒ Boot device order
- ☐ Linux kernel version
- ☐ Timezone
- ☒ Hardware configuration
- ☐ The system's hostname

LERNEN PRÜFEN WEITER

1%

Abbildung 8: Lernmodus, Ausgabe nach Klick auf „Lernen“



← Bitte geben Sie Ihre Antwort ein →

LÖSUNG Die richtige(n) Antwort(en) lautet/lauten:  
dmesg  
/bin/dmesg

Abbildung 9: Lösung Fill-In Fragen

Wird das Auswahlfeld, bei Multiple-Choice und Single-Choice-Fragen, mit den generierten Antworten aus der Antworten-Klasse, oder das Fill-In-Feld bei Fill-In-Fragen, durch klicken manipuliert, so deaktiviert sich der „Lernen“-Button. Der „Prüfen“-Button aktiviert sich gleichzeitig. Damit ist der Prüfungsmodus aktiviert. Eine Anzeige der richtigen Fragen, über den „Lernen“-Button ist nun nicht mehr möglich.



vor Klick im Formular:  nach Klick: 

Abbildung 10: Inaktiver und aktiver Button

Auszug Code des „PRÜFEN“-Buttons:

```
[...] „[class.disabled]=\"!(multiForm.dirty || singleForm.dirty || fillinForm.dirty)\"" [...]
```

Wählt man die richtigen Antworten aus, so erfolgt eine positive Bestätigung durch eine Ausgabe. Diese erfolgt als HTML\_Einblendung. Im Anschluss werden die Fragen, wie im Lernmodus, farblich markiert.

Erfolgt eine falsche Eingabe, wird eine Meldung per Alert ausgegeben. Diese weist auf die Falscheingabe hin.

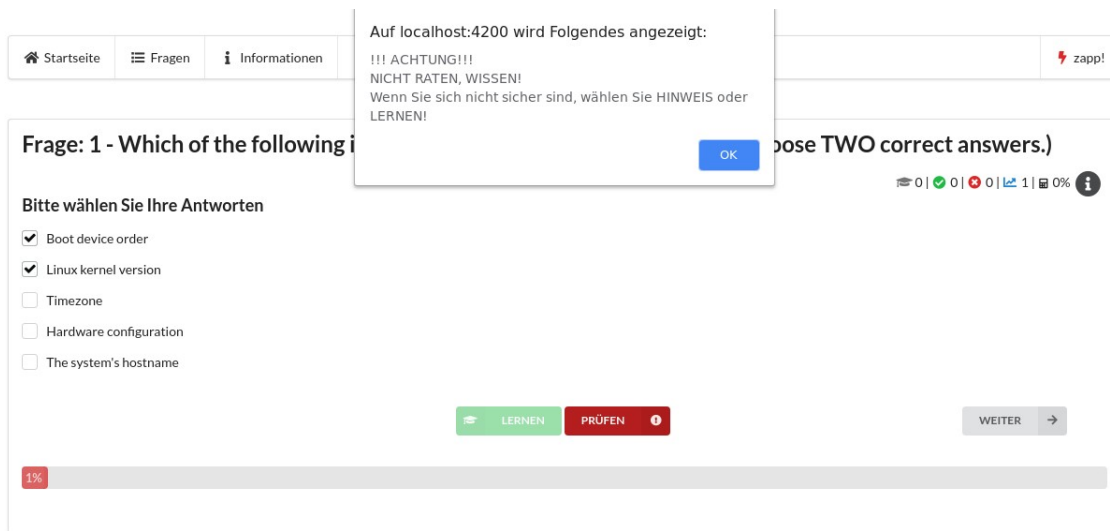


Abbildung 11: Alert nach Falscheingabe

### 3.1.10 Navigieren zur nächsten/vorherigen Frage

Die Navigation von Frage zu Frage erfolgt über die Buttons „ZURÜCK“ und „WEITER“

Am Beispiel des „WEITER“-Buttons sehen wir, dass zum einen die Funktion `weiter()`, durch das Angular-Event `(click)="weiter()"`, ausgeführt, und zum anderen der Routerlink manipuliert wird.

Durch `{{frage.id+1}}`, wird die ID der Frage, die in der Frage-Klasse hinterlegt ist, und beim Initialisieren der Frage ausgelesen wurde, um eins erhöht.

Dadurch wird jedoch nur die angezeigte URL manipuliert. Durch die `weiter()`-Funktion wird dann die ID erneut aus der Frage-Klasse ausgelesen und um eins erhöht. Dies geschieht in der Zuweisung: `this.frage = this.fs.getSingle(this.frage.id + 1)`.

Anschließend wird die Frage, durch die `this.initFrage()`-Funktion, neu initialisiert, d. h. Variablen werden zurückgesetzt.

Nun korrespondiert die erhöhte URL mit der ID der Frage, sodass die richtige ID in der URL, und die passende Frage ausgegeben wird.

Auch der Fortschrittsbalken greift auf die, in der Fragen-Klasse hinterlegte, ID zurück.

HTML-Code des „WEITER“-Buttons aus frage-details.component.html:

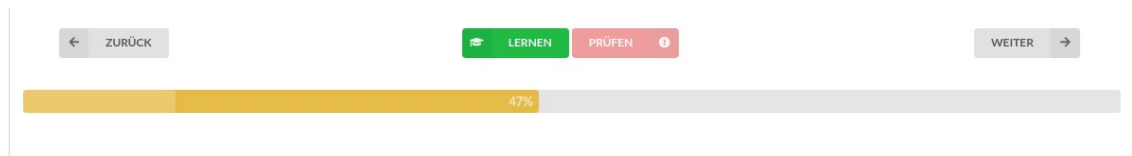
```
<div class="three wide column" (click)="weiter()"
routerLink="/fragen/{{frage.id+1}}">

  <i class="right grey arrow icon" *ngIf="frage.last"></i>
</div>
```

weiter() –Funktion die beim Klick auf den Button ausgeführt wird. Aus frage-details.component.ts:

```
weiter() {

  this.frage = this.fs.getSingle(this.frage.id + 1)
  this.initFrage();
}
```



*Abbildung 12: Navigation mit Fortschrittsbalken*

### 3.1.11 Auswertung der Antworten

Sobald der „PRÜFEN“-Button, der erst durch die Manipulation des Formulars aktiviert wird, geklickt wird, wird, durch das Angular-Event `(click)="pruefen()"`, die `pruefen()`-Funktion ausgeführt. Diese vergleicht die gegebenen Antworten mit den Werten in der Frage-Klasse, bzw. in der Antworten-Klasse.

#### Die Funktion `pruefen()` im Detail:

Neben der Ausführung der `aufdecken()`-Funktion, die im Rahmen der `pruefen()`-Funktion ausgeführt und unten erläutert wird, wird die `correct`-Variable auf `true` gesetzt.

Wir gehen zuerst davon aus, dass die Frage richtig beantwortet wurde. Im Anschluss prüfen wir, mittels einer `for`-Schleife für jede Antwortmöglichkeit einzeln, ob der eingegebene Wert ungleich des Wertes aus der Antworten-Klasse ist. Ist dies der Fall, so wird die Variable `correct` auf `false` gesetzt.

Ist die Variable `correct = true`, so ist die Frage richtig beantwortet und es wird eine positive Meldung als HTML-Element ausgegeben. (Siehe Abbildung 8)

Ist die Variable `correct = false`, so ist mindestens eine Antwort nicht korrekt gewesen und es wird eine negative Meldung als `alert` ausgegeben. (Siehe Abbildung 9)

#### Die Funktion `auswerten()` im Detail:

Diese setzt mit `this.aufgedeckt = true;` den Wert von `aufgedeckt`, der standardmäßig auf `false` eingestellt ist, auf `true`. Dies wird mit einer `*ngIf`-Abfrage im HTML-Code abgefragt. Stellt die `aufdecken()`-Funktion diesen Wert, durch klicken auf „LERNEN“ oder „PRÜFEN“, auf `true`, so wird das Komponenteigene-CSS aktiviert. Dieses markiert die Antworten mit prägnanten Icons. (🟢, 🔴)

Auszug `pruefen()` -Funktion des Multiple-Choice-Formulars in `frage-details.component.ts`

```

pruefen() {
  this.aufdecken();
  let correct = true;
  if (this.frage.typ === 'mc') {
    for (let i = 0; i < this.frage.antwort.length; i++) {
      if (
        this.frage.antwort[i].pruefung !==
        this.antwortArray.value[i]
      ) {
        correct = false;
      }
    }
  }
  [...]
  if (!correct) {
    alert("!!! ACHTUNG!!! \nNICHT RATEN, WISSEN! \nWenn Sie
    sich nicht sicher sind, wählen Sie HINWEIS oder LERNEN!")

    } else {
      this.richtig = true;
    }
  }
}

```

Den vollständigen Quellcode finden Sie unter 5. Quellcode.

## 3.2 Stage 2

Aufbauend auf Stage 1 wurden zwei weitere Formulare integriert, die weitere Fragetypen, wie Single-Choice und Fill-In-Fragen integrieren soll.

Die Angularfunktion `*ngIf` hilft uns dabei. Diese prüft auf `true` oder `false`. Hier wird der Wert aus der Frage-Klasse ausgelesen und verglichen.

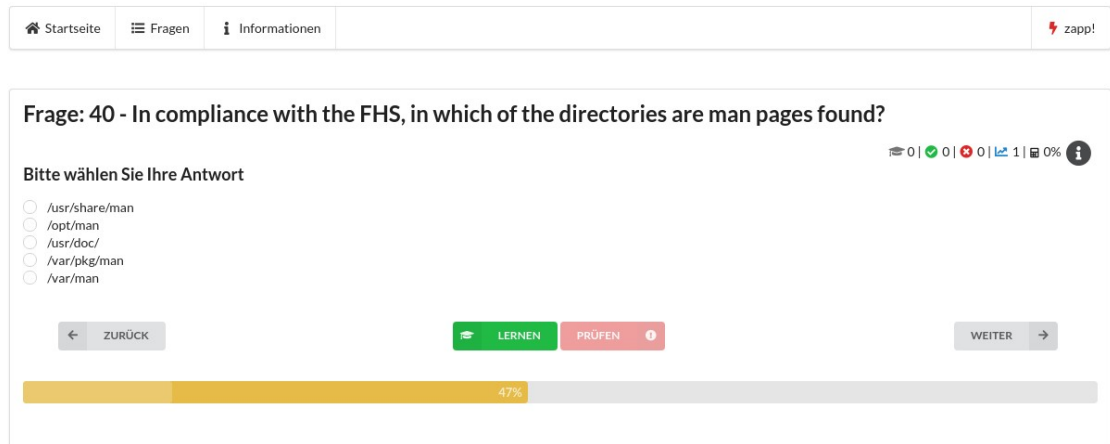
Ist der Wert = `'mc'`, dann wird das Multiple-Choice Formular verwendet. Es werden die gleichen Funktionen zur Auswertung genutzt wie in Stage 1.

Durch folgende Abfragen wird das jeweils richtige Formular für jede einzelne Frage ausgewählt:

<code>&lt;div *ngIf="frage.typ==='mc'"&gt; ... &lt;/div&gt;</code>	→ wählt das Multiple-Choice Formular, bereits ab Beginn implementiert, (Siehe Abb. 7)
<code>&lt;div *ngIf="frage.typ==='sc'"&gt; ... &lt;/div&gt;</code>	→ wählt das Single-Choice Formular (Siehe Abb. 13)
<code>&lt;div *ngIf="frage.typ==='fi'"&gt; ... &lt;/div&gt;</code>	→ wählt das Fill-In Formular (Siehe Abb. 14)

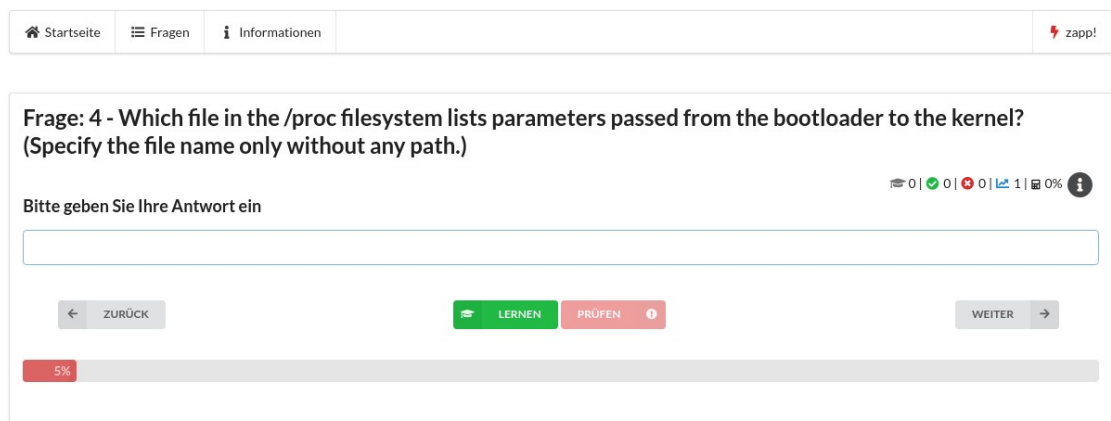
Den vollständigen Quellcode der Formulare finden Sie im Anhang.





The screenshot shows a web interface for a quiz. At the top, there is a navigation bar with links for 'Startseite', 'Fragen', and 'Informationen', and a 'zapp!' logo on the right. The main content area displays 'Frage: 40 - In compliance with the FHS, in which of the directories are man pages found?'. Below the question, it says 'Bitte wählen Sie Ihre Antwort'. There are five radio button options:   
☐ /usr/share/man  
☐ /opt/man  
☐ /usr/doc/  
☐ /var/pkg/man  
☐ /var/man  
At the bottom of the question area, there are three buttons: 'ZURÜCK' (with a left arrow), 'LERNEN' (green), and 'PRÜFEN' (red with a question mark icon). To the right of these is a 'WEITER' button (grey with a right arrow). A progress bar at the bottom shows 47% completion.

Abbildung 13: Single-Choice Fragenformular








The screenshot shows a web interface for a quiz. At the top, there is a navigation bar with links for 'Startseite', 'Fragen', and 'Informationen', and a 'zapp!' logo on the right. The main content area displays 'Frage: 4 - Which file in the /proc filesystem lists parameters passed from the bootloader to the kernel? (Specify the file name only without any path.)'. Below the question, it says 'Bitte geben Sie Ihre Antwort ein'. There is a text input field. At the bottom of the question area, there are three buttons: 'ZURÜCK' (grey with a left arrow), 'LERNEN' (green), and 'PRÜFEN' (red with a question mark icon). To the right of these is a 'WEITER' button (grey with a right arrow). A progress bar at the bottom shows 5% completion.


Abbildung 14: Fill-In Fragenformular

### 3.3 Erweiterungen

#### 3.3.1 Statistiken

Statistiken sind für Nutzer immer interessant. Sie helfen dabei einen Überblick zu behalten. Deshalb werden folgende Werte erfasst:

- ⚡  Fragen beantwortet
- ⚡  Fragen richtig beantwortet
- ⚡  Fragen falsch beantwortet
- ⚡  Fragen gesehen (auch wiederholte)
- ⚡  Prozentual richtig beantwortete Fragen. (In Bezug auf Anzahl angezeigter Fragen)

Um dem Nutzer diese Informationen stets zur Verfügung zu stellen, werden die Icons stetig oberhalb der Antworten angezeigt. Eine Erklärung öffnet sich, sobald der Nutzer ein Modal/Popup öffnet. Hierzu muss der Nutzer lediglich auf das  Info-Icon klicken.

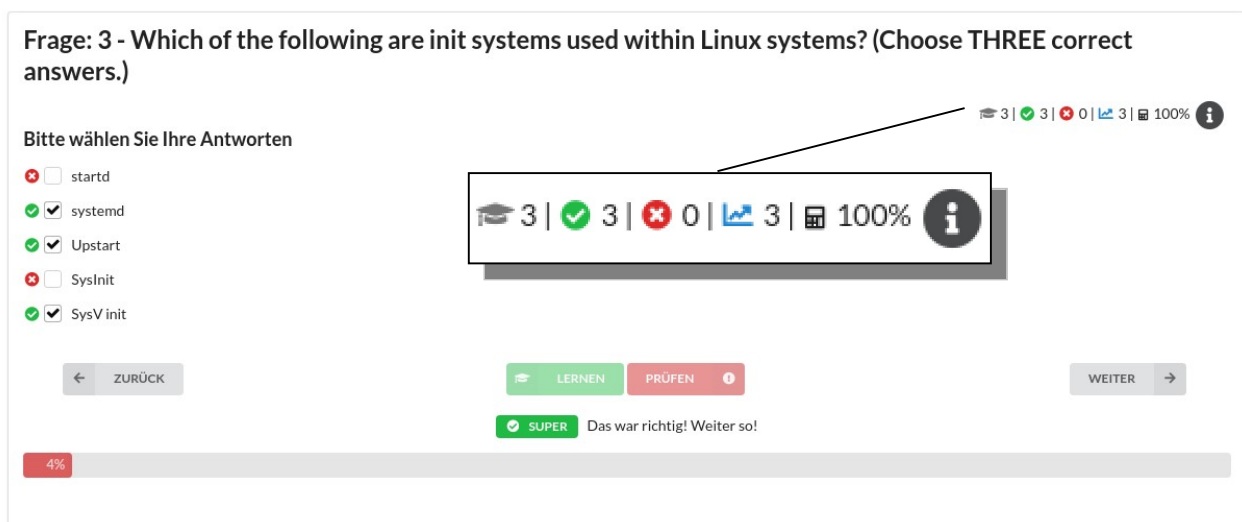


Abbildung 15: Statistik Schnellübersicht

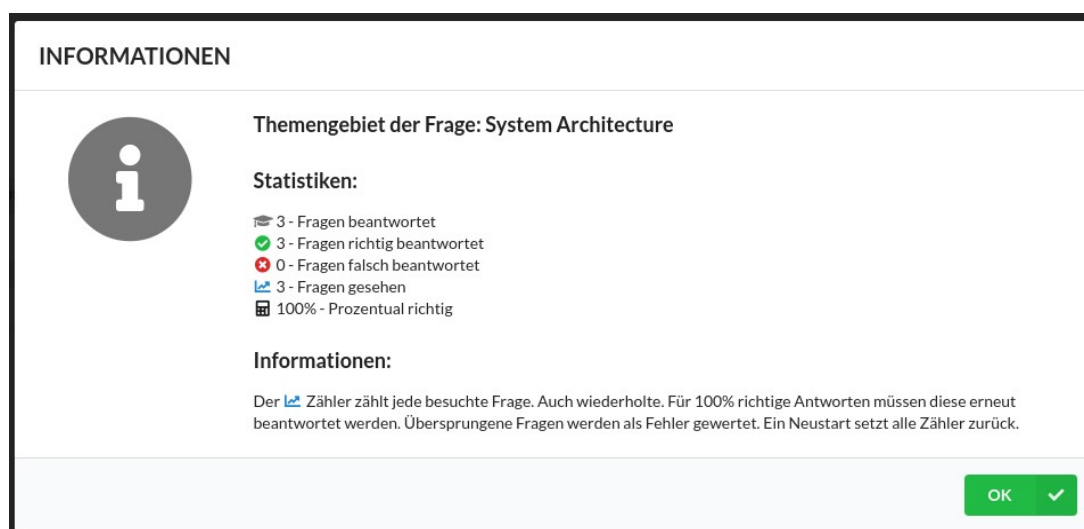


Abbildung 16: Modal mit Infoangaben

### 3.3.2 Fortschrittsbalken

Eine weitere Hilfe für Nutzer ist der Fortschrittsbalken. Er soll dem Nutzer zeigen, an welcher Stelle er sich in der Fragenliste befindet. Steigt der Nutzer über die Fragenliste z. B. bei Frage 40 ein, so startet die der Fortschrittsbalken natürlich nicht bei 0%. Er ermittelt die aktuelle Position aus der Frage-ID und die Gesamtanzahl der Fragen aus der Länge des Arrays. Mit diesen Werten gibt der Fortschrittsbalken nun die aktuelle Position in Prozent aus. Je nach Wert färbt sich der Balken z. B. rot, gelb oder grün.

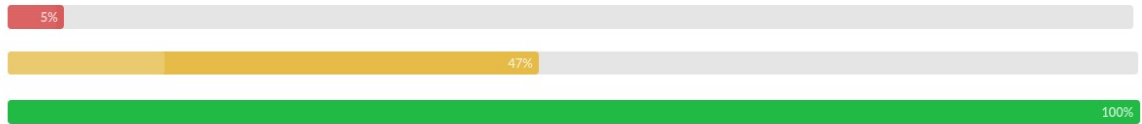


Abbildung 17: Fortschrittsbalken in verschiedenen Stadien

### 3.3.3 Responsive Design

Das Angularframework verfolgt den Mobile-First-Ansatz. Daher sollen Applikationen primär für mobile Endgeräte entwickelt werden. Da unterschiedliche Displayausrichtungen und Auflösungen haben können, kann man mit Angular und dem, hier verwendeten, CSS3 Framework Semantic-UI verhältnismäßig leicht Applikationen entwickeln, die sich diesen Gegebenheiten anpassen.

Die Applikation wurde so konzipiert, dass sie in drei verschiedenen Modi verfügbar ist. Dem Desktopmodus, dem Tabletmodus und dem Mobilemodus.

Im Tablet-, sowie im Mobilemodus werden die grauen „ZURÜCK“ und „WEITER“-Buttons ausgeblendet. Gleichzeitig werden größere Schaltflächen mit Richtungspfeilen eingeblendet. Diese sind über die Touchscreens der Tablet-PCs und Smartphones besser zu bedienen. Sofern die letzte Frage angezeigt wird, wird der Pfeil für Weiter → durch einen Wiederholpfeil ↺ ersetzt. Dieser führt den Nutzer zurück zur Fragenliste. (Siehe Abb. 18 und Abb. 19)

The screenshot shows the desktop view of the application. At the top, there is a navigation bar with links for 'Startseite', 'Fragen', and 'Informationen', and a 'zapp!' logo. Below this, a question is displayed: 'Frage: 85 - Which umask value will result in the default access permissions of 600 (rw-----) for files and 700 (rwx-----) for directories? (Specify only the numerical umask value.)'. Below the question is a text input field with the placeholder 'Bitte geben Sie Ihre Antwort ein'. To the right of the input field are several icons: a speech bubble, a green checkmark, a red circle with a white 'X', a blue magnifying glass, and a green circle with a white 'X'. Below the input field are three buttons: 'ZURÜCK' (grey), 'LERNEN' (green), and 'ZURÜCK ZUR LISTE' (red). At the bottom, there is a green progress bar labeled '100%'.

Abbildung 18: Desktopansicht (mind. Breite: 1000px)

Abbildung 19: Mobilgerätansicht (hier: Tablet mit Richtungspfeilen bei 1. Frage)

### 3.3.4 Nicht umgesetzte Erweiterungen

Natürlich kann man das Projekt weiter ausbauen. Weitere Ideen sind z. B. eine Auswahl von verschiedenen Fragenkatalogen. Diese könnte z.B. durch ein Dropdown-Menü realisiert werden. Aufbauend darauf könnte man die Integration von externen Fragenkatalogen z. B. von einem Server umsetzen. Auch eine Internationalisierung ist möglich und sicherlich sinnvoll.

### 3.3.5 Native Android Applikation

Als Erweiterung kommen auch native Smartphone und/oder Tablet Apps im Betracht. Eine Portierung der Applikation für Android-Smartphones wurde mittels Apache Cordova durchgeführt. Dies wird in einer separaten Dokumentation beschrieben.

Mittels Cordova wären auch weitere Portierungen, z. B. für das Apple Betriebssystem iOS, möglich.



Abbildung 20: Android-zapp!

## 4 Fazit

Die Entwicklung der Webapplikation zapp! stellte mich von Beginn an vor einige Herausforderungen. Der Start wurde durch das Angular-Buch (*Angular; Woiwode, Malcher, Koppenhagen, Hoppe; ISBN: 978-3-86490-357-1; dpunkt-Verlag*) jedoch deutlich erleichtert. Dort konnte ich einige Schritte, wie Anlegen des Projektes und Routing, übernehmen und für zapp! umschreiben.

Die Idee zu zapp! finde ich persönlich sehr gut. Eine Browserbasierte Webapplikation läuft nahezu auf jedem Endgerät und ist schnell aufgerufen. Dies ermöglicht Nutzern eine schnelle Nutzung ohne unnötige Software herunterladen und installieren zu müssen.

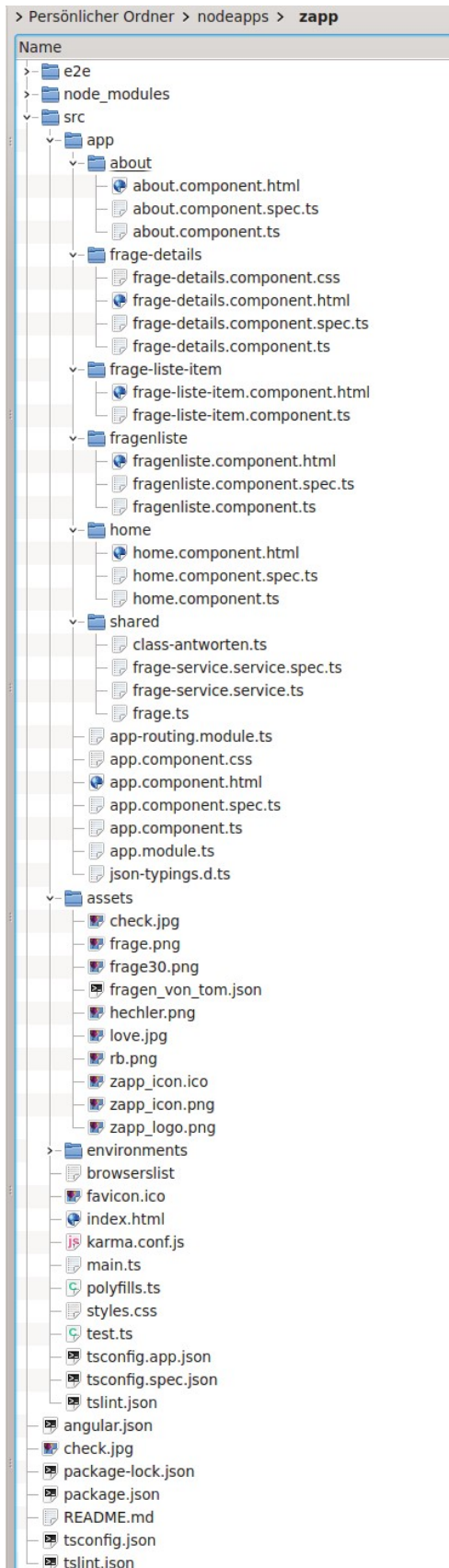
Im Laufe des Projektes konnte ich immer wieder kleinere und mittlere Erfolge erzielen. Genau diese Erfolge haben mir immer wieder gezeigt, dass mein Verständnis für Angular wächst. Die Zusammenarbeit mit anderen Teilnehmern war oftmals am effektivsten. Gemeinsam wurden Ideen kreiert und umgesetzt. Einige Lösungsansätze wurden auch durch Internetrecherche ermittelt. Die große Community rund um Angular bietet viele Informationen.

Parallel zum Angularframework konnte ich auch meine Kenntnisse in HTML5 und CSS3 weiter ausbauen. Angular ist durch das Zusammenspiel von HTML, TypeScript, CSS und auch JavaScript ein mächtiges Werkzeug um anspruchsvolle und komplexe Webapplikationen zu realisieren.

Abschließend kann ich sagen, dass ich grundlegende Kenntnisse in Angular sammeln und anwenden konnte. Ich fühle mich nun in der Lage, um erste Projekte mit Angular eigenständig zu entwickeln.

## 5 Anhang

### 5.1 Datenstrukturbaum



## 5.2 Quellcode

### 5.2.1 Index.html

```
<!doctype html>
<html lang="de">
<head>
  <meta charset="utf-8">
  <title>zapp! ...und wie lernst du?</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="./assets/zapp_icon.ico">
</head>

<body>
  <zp-root></zp-root>
</body>
</html>
```

### 5.2.2 app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { FragenlisteComponent } from '../fragenliste/fragenliste.component';
import { HomeComponent } from '../home/home.component';
import { AboutComponent } from '../about/about.component';
import { FrageDetailsComponent } from '../frage-details/frage-
details.component';

export const routes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'},
  {
    path: 'fragen',
    component: FragenlisteComponent
  },
  {
    path: 'fragen/:id',
    component: FrageDetailsComponent
  },
  {
    path: 'about',
    component: AboutComponent
  },
  {
    path: 'home',
    component: HomeComponent
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

### 5.2.3 app.component.css

```
#paddingbody {
  padding-bottom: 3em;
}

#footerfest {
  position: fixed;
  padding: 4px;
  bottom: 0;
  left: 0;
  right: 0;
  width: 100%;
  background-color: red;
  text-align: center;
  color: white;
}
```

### 5.2.4 app.component.html

```
<!--COMPUTER ONLY RAHMEN OBEN-->

<div class="ui grid">
  <div class="three wide computer three wide tablet column"></div>
  <div class="computer only ten wide column">
  </div>
  <div class="computer only three wide column"></div>
</div>

<!--COMPUTER ONLY RAHMEN OBEN ENDE-->

<!--MENUE-->
<div class="ui grid">
  <div class="three wide computer two wide tablet one wide mobile
column"></div>
  <div class="ten wide computer twelve wide tablet fourteen wide mobile
column">
    <div class="ui pointing menu">
      <a class="item" routerLink="home">
        <i class="home icon"></i>
        <p class="computer only">Startseite</p>
      </a>
      <a class="item" routerLink="fragen">
        <i class="list icon"></i> Fragen
      </a>
      <a class="item" routerLink="about">
        <i class="info icon"></i> Informationen
      </a>
      <div class="right menu">
        <div class="item">
          <div class="list icon">
            <i class="bolt red icon"></i>zapp!
          </div>
        </div>
      </div>
    </div>
  <div class="three wide computer two wide tablet one wide mobile
column"></div>
</div>
<!--MENUE-ENDE-->
<div class="ui grid">

  <div class="three wide computer two wide tablet one wide mobile
column"></div>
```



```

    <div class="ten wide computer twelve wide tablet fourteen wide mobile
column">
      <div id="paddingbody">
        <router-outlet></router-outlet>
      </div>
    </div>
    <div class="three wide computer two wide tablet one wide mobile column">
    </div>
  </div>
<!--FOOTER-FIX-->

<div id="footerfest">
  <h5><i class=" bolt red icon"></i>zapp! v1.4 (c)2019</h5>
</div>

```

### 5.2.5 app.component.ts

```

import { Component } from '@angular/core';

@Component({
  selector: 'zp-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'zapp';
}

```

### 5.2.6 app.module.ts

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FragenlisteComponent } from './fragenliste/fragenliste.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { FrageListItemComponent } from './frage-liste-item/frage-liste-
item.component';
import { FrageDetailsComponent } from './frage-details/frage-
details.component';
import { FrageServiceService } from './shared/frage-service.service';
import { ReactiveFormsModule } from "@angular/forms";

@NgModule({
  declarations: [
    AppComponent,
    FragenlisteComponent,
    HomeComponent,
    AboutComponent,
    FrageListItemComponent,
    FrageDetailsComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule
  ],
  providers: [
    FrageServiceService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

### 5.2.7 about.component.html

```
<div style="text-align: center">



<h4 class="ui header">
  Dieses Projekt wurde im Rahmen der Fachinformatiker-Umschulung beim <a
href="http://www.bitlc.de"
  target="_blank">BITLC Business IT Learning Center</a> im Modul Web-
Development durchgeführt und kommt daher in
  verschiedenen Versionen vor.<br><br>
  Diese Version wurde erstellt von:
</h4>

<div class="column">
  <h2 class="ui header">René Brodkorb</h2>
  
  <div>
    <h3 class="ui header"><a href="mailto:service@pcrb.de">Kontakt</a></h3>
    <br>
  </div>
</div>
<h5 class="ui header">Die hier verwendeten Grafiken wurden entweder von René@
Brodkorb erstellt, oder sind Bestandteil
  des <a href="https://semantic-ui.com/" target="_blank">Semantic-UI
CSS.</a></h5>
</div>
```

### 5.2.8 about.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'zp-about',
  templateUrl: './about.component.html',
  styles: []
})
export class AboutComponent implements OnInit {
  constructor() { }
  ngOnInit() {
  }
}
```

### 5.2.9 frage-details.component.css

```
#mittigid {
  display: flex;
  justify-content: center;
}
#progressbar {
  background: rgb(0, 0, 0);
  border-radius: 13px;
  height: 20px;
  width: 250px;
  padding: 3px;
}
#progressbar:after {
  content: '';
  display: block;
  background: #DB2828;
  width: 60%;
  height: 100%;
  border-radius: 9px;
}
#progressbarcenter {
  display: flex;
```

```
        align-items: center;
        justify-content: center;
    }
    #transparent {
        color: #FF000000
    }
    #footerfest {
        position: fixed;
        padding: 4px;
        bottom: 20px;
        left: 0;
        right: 0;
        width: 100%;
        text-align: center;
        /* background-color: red; */
        /* color: white; */
        background-color: white;
        color: black;
    }
}
```

### 5.2.10 frage-details.component.html

```

<h2 class="ui header">Frage: {{ frage.id }} - {{ frage.fragentext }}</h2>
<div class="ui grid">
  <div class="mobile tablet only three wide column" (click)="zurueck()"
routerLink="/fragen/{{frage.id-1}}">
    <i class="big left grey arrow icon" *ngIf="frage.first"></i>
  </div>

  <!--FORMULARE-->
  <div class="ten wide mobile sixteen wide computer ten wide tablet column">
    <div class="six wide column" style="text-align:right">
      <i *ngIf="frage.antwort[0]" class="graduation cap icon grey"></i>{{
zfalsch + zrichtig }} |
      <i *ngIf="frage.antwort[0]" class="check circle icon green"></i>{{
zrichtig }} |
      <i *ngIf="frage.antwort[0]" class="times circle icon red"></i>{{ zfalsch
}} |
      <i *ngIf="frage.antwort[0]" class="chart line icon blue"></i>{{ zview }}
    </div>
    <i *ngIf="frage.antwort[0]" class="small calculator icon black"></i>{{
prozent | number:'1.0' }}%
    <i *ngIf="frage.antwort[0]" (click)="statsfkt()" class="big info circle
icon black link"></i>
  </div>
  <!-- MULTI-->
  <div *ngIf="frage.typ==='mc'">
    <form [formGroup]="multiForm">
      <h3 class="ui header">Bitte wählen Sie Ihre Antworten</h3>
      <div formArrayName="antwortArray">
        <div class="ui form" *ngFor="let a of frage.antwort; index as i">
          <div class="inline field"></div>
          <span *ngIf="aufgedeckt && a.pruefung">
            <i *ngIf="frage.antwort[0]" class="check circle icon green"></i>
          </span>
          <span *ngIf="!a.pruefung && aufgedeckt">
            <i *ngIf="frage.antwort[0]" class="times circle icon red"></i>
          </span>
          <div class="ui checkbox">
            <input [formControlName]="i" type="checkbox" tabIndex="0"
[id]="i">
            <label [for]="i">{{a.antworttext}}</label>
          </div>
        </div>
      </div>
    </form>
  </div>
  <!-- ENDE-MULTI -->

  <!-- SINGLE-->
  <div *ngIf="frage.typ==='sc'">
    <form [formGroup]="singleForm">
      <h3 class="ui header">Bitte wählen Sie Ihre Antwort</h3>
      <div class="grouped fields">
        <div class="inline field">
          <div class="ui form" *ngFor="let a of frage.antwort; index as i">
            <div class="field">
              <span *ngIf="aufgedeckt && a.pruefung">
                <i *ngIf="frage.antwort[0]" class="check circle icon
green"></i>
              </span>
              <span *ngIf="!a.pruefung && aufgedeckt">
                <i *ngIf="frage.antwort[0]" class="times circle icon
red"></i>
              </span>
              <div class="ui radio checkbox">

```

```

        <input type="radio" name="antname" formControlName="antname"
tabindex="0" class="hidden" [id]="i"
        [value]="i">
        <label [for]="i">{{a.antworttext}}</label>
    </div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<!--ENDE-SINGLE-->

<!--FILL-IN-->
<div *ngIf="frage.typ==='fi'">
    <form [formGroup]="fillinForm">
        <h3 class="ui header">Bitte geben Sie Ihre Antwort ein</h3>
        <div class="ui form">
            <div class="field">
                <input formControlName="textfeld" name="tf12" type="text"
(keyup.enter)="pruefen()" autofocus>
            </div>
        </div>
    </form>
    <!--EINBLENDUNG LÃ-SUNG-->
    <div class="item" style="text-align:center" *ngIf="aufgedecktfi">

        <br>
        <div class="ui green horizontal icon label">
            <i class="middle check circle icon"></i>LÃ-SUNG</div>
        Die richtige(n) Antwort(en) lautet/lauten: <br>
        <div *ngFor="let a of frage.antwort; index as
i"><b>{{a.antworttext}}</b></div>
        </div>
    <!--ENDE EINBLENDUNG LÃ-SUNG-->
    </div>
    <!--ENDE-FILL-IN-->

    <!--ENDE-FORMULARE-->
</div>

<div class="mobile tablet only three wide column" *ngIf="frage.last"
style="text-align: right" (click)="weiter()"
    routerLink="/fragen/{{frage.id+1}}">
    <i class="big right grey arrow icon"></i>
</div>
<div class="mobile tablet only three wide column" *ngIf="!frage.last"
style="text-align: right"
    routerLink="/fragen/">
    <i class="big undo grey arrow icon"></i>
</div>

</div>

<!-------BUTTONS START----->
<br><br>
<div class="ui grid" style="text-align:center">

    <div class="computer only three wide column" id="mittigid">
        <div *ngIf="frage.first">
            <button class="ui tiny labeled icon button" (click)="zurueck()"
routerLink="/fragen/{{frage.id-1}}">
                <i class="left arrow icon"></i>
                ZURÃœCK
            </button>
        </div>
    </div>
</div>

```

```

    <div class="sixteen wide mobile sixteen wide tablet ten wide computer wide
column">
    <button class="ui tiny green labeled icon button" (click)="aufdecken()"
    [class.disabled]="multiForm.dirty || singleForm.dirty ||
fillinForm.dirty">
    <i class="graduation cap icon"></i>
    LERNEN
    </button>

    <!-- <button class="ui tiny orange labeled icon button"
(click)="hinweisbutton()">
    <i class="question circle icon"></i>
    HINWEIS
    </button> -->

    <button class="ui tiny red right labeled icon button" (click)="pruefen()"
    [class.disabled]="aufgedeckt || aufgedecktfi || !(multiForm.dirty ||
singleForm.dirty || fillinForm.dirty)">
    <i class="attention icon"></i>
    PRÜFEN
    </button>
</div>

<div class="computer only three wide column" id="mittigid">
    <div *ngIf="frage.last">
    <button class="ui tiny right labeled icon button" (click)="weiter()"
routerLink="/fragen/{{frage.id+1}}">
    <i class="right arrow icon"></i>
    WEITER
    </button>
    </div>

    <div *ngIf="!frage.last">
    <button class="ui red tiny right labeled icon button"
routerLink="/fragen/">
    <i class="right arrow icon"></i>
    ZURÜCK ZUR LISTE
    </button>
    </div>

</div>
</div>
<!-------BUTTONS ENDE----->

<!--Flexible-AUSGABE-->
<br>
<div class="item" style="text-align:center" *ngIf="hinweis">
    <div class="ui orange horizontal icon label">
    <i class="question circle icon"></i>HINWEIS</div>
    {{ frage.hinweis }} <br><br>
</div>

<div class="item" style="text-align:center" *ngIf="richtig">
    <div class="ui green horizontal icon label">
    <i class="middle check circle icon"></i>SUPER</div>
    Das war richtig! Weiter so!
</div>

<!-- FORTSCHRITTSBALKEN -->
<div class="ui indicating progress" data-value="1" id="example4">
    <div class="bar">
    <div class="progress"></div>
    </div>
</div>
<!-- FORTSCHRITTSBALKEN ENDE-->

```

```

<!-- MODAL -->
<div class="ui modal">
  <i class="close icon"></i>
  <div class="header">
    INFORMATIONEN
  </div>
  <div class="image content">
    <div class="ui medium image" style="text-align:center">
      <!--  -->
      <i class="massiv info circle icon grey"></i>
    </div>
    <div class="description">
      <h3 class="ui header">Themengebiet der Frage: {{ frage.hinweis }}</h3>
      <h3 class="ui header">Statistiken:</h3>
      <i *ngIf="frage.antwort[0]" class="graduation cap icon grey"></i>{{
zfalsch + zrichtig }} - Fragen beantwortet<br>
      <i *ngIf="frage.antwort[0]" class="check circle icon green"></i>{{
zrichtig }} - Fragen richtig beantwortet <br>
      <i *ngIf="frage.antwort[0]" class="times circle icon red"></i>{{ zfalsch
}} - Fragen falsch beantwortet<br>
      <i *ngIf="frage.antwort[0]" class="chart line icon blue"></i>{{ zview }}
- Fragen gesehen <br>
      <i *ngIf="frage.antwort[0]" class="calculator icon black"></i>{{ prozent
| number:'1.0' }}% - Prozentual richtig
      <h3 class="ui header">Informationen:</h3>
      <p>Der <i *ngIf="frage.antwort[0]" class="chart line icon
blue"></i>Zähler zählt jede besuchte Frage. Auch
wiederholte. Für 100% richtige Antworten müssen diese erneut
beantwortet werden. Übersprungene Fragen werden als
Fehler gewertet. Ein Neustart setzt alle Zähler
zurück.</p>
    </div>
  </div>
  <div class="actions">
    <div class="ui positive right labeled icon button">
      OK
      <i class="checkmark icon"></i>
    </div>
  </div>
</div>
<!-- MODAL ENDE -->

```

### 5.2.11 frage-details.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Frage } from '../shared/frage';
import { FrageServiceService } from '../shared/frage-service.service';
import { ActivatedRoute } from '@angular/router';
import { FormArray, FormGroup, FormBuilder, FormControl } from
 '@angular/forms';

declare var jQuery: any;

@Component({
  selector: 'zp-frage-details',
  templateUrl: './frage-details.component.html',
  styleUrls: ['./frage-details.component.css']
})
export class FrageDetailsComponent implements OnInit {
  frage: Frage;
  hinweis: boolean;
  aufgedeckt: boolean;
  aufgedecktfi: boolean;
  richtig: boolean;
  antwortArray: FormArray;
  multiForm: FormGroup;
  singleForm: FormGroup;
  fillinForm: FormGroup;
  antname: FormControl;
  zfalsch: number = 0;
  zrichtig: number = 0;
  zview: number = 0;
  prozent: number = 0;
  anzFragen: number;

  location = { first: false, last: false };

  constructor(
    private fs: FrageServiceService,
    private route: ActivatedRoute,
    private fb: FormBuilder
  ) { }

  ngOnInit() {
    const id: number = parseInt(this.route.snapshot.params['id'], 10);
    this.frage = this.fs.getSingle(id);
    this.hinweis = false;
    this.aufgedeckt = false;
    this.aufgedecktfi = false;
    this.initFrage();
    this.zfalsch = 0;
    this.zrichtig = 0;
    this.prozent = 0;
  }

  initFrage() {
    this.antwortArray = this.fb.array(this.frage.antwort.map(e => false));
    this.multiForm = this.fb.group({
      antwortArray: this.antwortArray
    })
    this.singleForm = this.fb.group({
      antname: ''
    })
    this.fillinForm = new FormGroup({
      textfeld: new FormControl()
    })
    this.hinweis = false;
  }
}
```



```
this.aufgedeckt = false;
this.aufgedecktfi = false;
this.richtig = false;
this.zview++;
this.anzFragen = this.fs.getlaenge();
this.scrippte()

}

weiter() {
  this.frage = this.fs.getSingle(this.frage.id + 1)
  this.initFrage();
}

zurueck() {
  this.frage = this.fs.getSingle(this.frage.id - 1)
  this.initFrage();
}

hinweisbutton() {
  if (this.hinweis) {
  } else {
    this.hinweis = true;
  }
}

aufdecken() {
  if (this.frage.typ === 'mc') {
    this.aufgedeckt = true;
  } else if (this.frage.typ === 'sc') {
    this.aufgedeckt = true;
  } else if (this.frage.typ === 'fi') {
    this.aufgedecktfi = true;
  }
}

pruefen() {
  this.aufdecken();
  let correct = true;

  if (this.frage.typ === 'mc') {
    for (let i = 0; i < this.frage.antwort.length; i++) {
      if (
        this.frage.antwort[i].pruefung !== this.antwortArray.value[i]
      ) {
        correct = false;
      }
    }
  } else if (this.frage.typ === 'sc') {
    if (this.frage.antwort[this.singleForm.controls.ansname.value].pruefung
    === false) {
      correct = false;
    }
  } else if (this.frage.typ === 'fi') {
    if (this.frage.antwort[0].antworttext !==
    this.fillinForm.controls.textfeld.value) {
      correct = false;
    }
  }

  if (!correct) {
    alert("!!! ACHTUNG!!! \nNICHT RATEN, WISSEN! \nWenn Sie sich nicht
    sicher sind, wÄhlen Sie HINWEIS oder LERNEN!")
    this.zfalsch++;
  } else {
    this.richtig = true;
    this.aufgedecktfi = false;
  }
}
```

```

        this.zrichtig++;
    }
    this.prozent = (this.zrichtig / this.zview) * 100
}

statsfkt() {
    jQuery('.ui.modal')
        .modal('show')
;
}

scripte() {
    jQuery('#example4')
        .progress('set progress', this.frage.id)
        .progress('set total', this.anzFragen);
}

onKeydown(event) {
    if (event.key === "ArrowRight") {
        this.weiter();
    } else if (event.key === "ArrowLeft") {
        this.zurueck();
    }
}
}

```

### 5.2.12 frage-liste-item.component.html

```

<div class="content">
    <div class="ui grid">
        <div class="two wide column">
             </div>
        <div class="fourteen wide column">
            <div class="header" style="text-align: left">{{ frage.fragentext
        }}</div>
            <div class="extra" style="text-align: left">Fragennummer: {{
frage.id }} - Typ: {{ frage.typ | uppercase }}</div>      <br>
            </div>
        </div>
    </div>
</div>

```

### 5.2.13 frage-liste-item.component.ts

```

import { Component, Input } from '@angular/core';
import { Frage } from '../shared/frage';

@Component({
    selector: 'a.zp-frage-liste-item',
    templateUrl: './frage-liste-item.component.html',
    styles: []
})
export class FrageListItemComponent {
    @Input() frage: Frage;
}

```

### 5.2.14 fragenliste.component.html

```

<div class="ui middle aligned animated list">
    <a class="zp-frage-liste-item" *ngFor="let f of fragen" [frage]="f"
[routerLink]="f.id"></a></div>

```

### 5.2.15 fragenliste.component.ts

```

import { Component, OnInit, Output } from '@angular/core';
import { Frage, ClassAntworten } from '../shared/frage';

```

```
import { FrageServiceService } from '../shared/frage-service.service';
import { EventEmitter } from 'events';

@Component({
  selector: 'zp-fragenliste',
  templateUrl: './fragenliste.component.html',
  styles: []
})
export class FragenlisteComponent implements OnInit {
  fragen: Frage[]
  constructor(private fs: FrageServiceService) { }
  ngOnInit() {
    this.fragen = this.fs.getAll();
  }
}
```

### 5.2.16 home.component.html

```
<div style="text-align:center">
<h1 class="ui header">Herzlich Willkommen bei zapp!</h1>

<br>
<button class="ui green small labeled icon button" routerLink="/fragen/1">
  <i class="bolt white icon"></i>
  SCHNELLESTART
</button>
</div>
```

### 5.2.17 home.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'zp-home',
  templateUrl: './home.component.html',
  styles: []
})
export class HomeComponent implements OnInit {

  constructor() { }
  ngOnInit() {
  }
}
```

### 5.2.18 shared/class-antworten.ts

```
export class ClassAntworten {
  constructor(
    public id: number,
    public antworttext: string,
    public pruefung: boolean
  ) {}
}
```

### 5.2.19 shared/frage.ts

```
import { ClassAntworten } from './class-antworten';
export { ClassAntworten } from './class-antworten';

export class Frage {
  constructor(
    public id: number,
    public fragentext: string,
    public hinweis: string,
    public antwort: ClassAntworten[],
    public typ: string,
    public first?: boolean,
    public last?: boolean
  ) {}
}
```

### 5.2.20 shared/frage-service.service.ts

```
import { Injectable } from '@angular/core';
import { Frage, ClassAntworten } from '../shared/frage';
import * as TomsFragen from '../../assets/fragen_von_tom.json';

@Injectable()

export class FrageServiceService {
  fragen: Frage[]

  constructor() {
    this.fragen = TomsFragen.default
  }
  getAll() {
    return this.fragen;
  }

  getSingle(id: number) {
    let position = this.fragen.map(e => e.id).indexOf(id);
    const frage = this.fragen[position];
    frage.first = position == 0 ? false : true;
    frage.last = position == this.fragen.length - 1 ? false : true;
    return frage;
  }
}
```

### 5.2.21 fragen\_von\_tom.json (AUSZUG)

```
[
  {
    "id": 1,
    "fragentext": "Which of the following information is stored within the BIOS? (Choose TWO correct answers.)",
    "hinweis": "System Architecture",
    "antwort": [
      { "id": 1, "antworttext": "Boot device order", "pruefung": true},
      { "id": 2, "antworttext": "Linux kernel version", "pruefung": false},
      { "id": 3, "antworttext": "Timezone", "pruefung": false},
      { "id": 4, "antworttext": "Hardware configuration", "pruefung": true},
      { "id": 5, "antworttext": "The system's hostname", "pruefung": false}
    ],
    "typ": "mc"
  },
  {...},

  {
    "id": 4,
    "fragentext": "Which file in the /proc filesystem lists parameters passed from the bootloader to the kernel? (Specify the file name only without any path.)",
    "hinweis": "System Architecture",
    "antwort": [
      { "id": 1, "antworttext": "cmdline", "pruefung": true },
      { "id": 2, "antworttext": "/proc/cmdline", "pruefung": true }
    ],
    "typ": "fi"
  },
  {...},

  {
    "id": 9,
    "fragentext": "Which of the following commands will write a message to the terminals of all logged in users?",
    "hinweis": "System Architecture",
    "antwort": [
      { "id": 1, "antworttext": "bcast", "pruefung": false},
      { "id": 2, "antworttext": "mesg", "pruefung": false},
      { "id": 3, "antworttext": "print", "pruefung": false},
      { "id": 4, "antworttext": "wall", "pruefung": true},
      { "id": 5, "antworttext": "yell", "pruefung": false}
    ],
    "typ": "sc"
  },
  {...}, //INSGESAMT 85 FRAGEN
```

## 5.3 Ablage, Gültigkeit und Bezüge zu anderen Dokumenten

Die jeweils aktuellste, und damit einzig gültige, Version dieses Dokumentes wird auf dem BITLC-Erk3-Server im folgenden Unterordner gespeichert und gepflegt: \\bitlcrw\RB\zapp

## 5.4 Reviewvermerke

### 5.4.1 Ersterstellung

Diese Dokumentation wurde, in Ihrer ersten Version, am 21.02.2019 erstellt.

### 5.4.2 Revision 1

EPK hinzugefügt. Code aktualisiert.

### 5.4.3 Revision 2

Code aktualisiert. Schwerpunkt: Fehlerbehebungen

### 5.4.4 Revision 3

Code aktualisiert. Schwerpunkt: Statistik hinzugefügt.

### 5.4.5 Revision 4

Code aktualisiert. Schwerpunkt: Design überarbeitet.

### 5.4.6 Revision 5

Dokumentation aktualisiert. Schwerpunkt: EPK, Inhalt, Fazit.

### 5.4.7 Revision 6

Dokumentation aktualisiert. Aufbau, Layout, kleinere Korrekturen

### 5.4.8 Revision 7

Dokumentation, nach Durchsicht durch Herrn Antonius Ehlen, aktualisiert.

## 5.5 Historie der Dokumentversionen

Version	Datum	Autor	Bemerkungen
1.0	22.02.2019	R. Brodkorb	Ersterstellung
1.1	25.02.2019	R. Brodkorb	Revision 1
1.2	25.02.2019	R. Brodkorb	Revision 2
1.3	26.02.2019	R. Brodkorb	Revision 3
1.4	27.02.2019	R. Brodkorb	Revision 4
1.5	28.02.2019	R. Brodkorb	Revision 5
1.6	05.03.2019	R. Brodkorb	Revision 6
1.7	06.03.2019	R. Brodkorb	Revision 7

---

Datum

---

René Brodkorb