Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften, Bereich EuI

Projektarbeit

Webapp für Studiende des Studienganges AIMMT

Gedruckt am: 7. Juni 2020

von: Nicole Goldmann
geboren am 20. August 1997
in Herbolzheim

von: Stefanie Weidemann
geboren am 25. August 1995
in Anklam

Betreuer: Prof. Dr.-Ing. Herbert Litschke

Inhaltsverzeichnis

1 Einleitung

Möchten Studierende wissen welche Module sie nächstes Semester haben oder in welchem Haus sie einen bestimmten Professor finden können, müssen sie sich durch endlos viele Seiten der Hochschul-Website klicken, um an das Ziel zu gelangen. Die Website ist sowohl für zukünftige Studierende und Interessierte, als auch für eingeschriebene Studierende konzipiert. Daher gibt es eine Fülle von Informationen und Querverweise, sodass es schwer ist die wichtigen Aspekte herauszufiltern.

Um dies zu vereinfachen wird im folgenden eine Applikation entwickelt die alle wichtigen Informationen zum Studiengang Angewandte Informatik - Multimendiatechnik zusammenfasst und gebündelt darstellt. Diese kann kann von allen Studierenden des Studienganges bequem auf dem Smartphone ausgeführt werden.

Nach einer Analyse der Hochschulseite und der Beschreibung des IST-Zustandes folgt die Festlegung der Systemanforderungen an die Applikation. In den Grundlagen wird erläutern was Webcomponents, im speziellen die LitElements, sind und wie sie aufgebaut sind. Hier wird zudem auf den Shadow-Dom, die Templates und die Properties eingegangen. Außerdem wird die interne Navigation mittels eines Routers beschrieben. Im Abschnitt Konzept wird der Aufbau und die Funktionsweise der Applikation näher erläutert. Im Kapitel 5 Implementierungen werden wichtige Ausschnitte aus dem Programmcode gezeigt und näher erklärt.

2 Untersuchung der Hochschul-Website

Studenten die Informationen zu ihrem Studiengang suchen brauchen teilweise sechs Klicks um von der Startseite der HS Wismar zur Semsterübersicht (AIMMT) zu gelangen. Innerhalb dieser Seiten gibt es einige Unstimmigkeiten und fehlerhaftes Verhalten, welches im folgendem näher erläutert wird.

2.1 Analyse des Ist-Zustandes

Die Hochschulwebsite hat einen großen allgemeinen Teil der Informationen über die Hochschule enthält und hat Verlinkungen zu den drei Fakultätsseiten. Da die entwickelte Webanwendung für Studierende des Studiengangs Angewandte Informatik und Multimendiatechnik gedacht ist wird an dieser Stelle nur die Fakultätsseite der Ingenieurstechnik bzw. des Bereiches Elektrotechnik und Information.

In der oberen Navigationsleiste gibt es nicht eindeutig erkennbare Icons für den Schnelleinstig und Informationen. Werden diese angeklickt, klappt sich ein Panel nach oben aus. Die obere Navigationsleiste ist nicht fixiert. Der Footer ist sehr groß und enthält teilweise die selben Verlinkungen wie in der Seitennavigation oben. Auf der Seite der Semsterübersicht gib es eine Akkordeonmenü welches nicht an allen Punkten anklickbar ist. Dieses Menü enthält ein Pfeil-Icon nach unten zeigend, welches dem User sugerriert das sich hinter dem Icon mehr Informationen enthalten. Dieses Icon ist aber nicht anklickbar. Die Pfeil-Icons gibt es auf der gesamten Website als wiederkehrendes Symbol für Verlinkungen. Auf den Informationsseiten werden diese Icons allerdings auch als Auflistungszeichen verwendet. Weiterhin gibt es viel zu viele Querverweise. Beispielsweise auf einer Modulübersichtsseite gibt es Verlinkungen zu weiteren Studiengängen die dieses Modul besuchen, weitere Module die der/die Professor |in unterrichtet, die Forschungsthemen, Thesenthemen, und Jobs die der/die Professor|in anbietet. Diese Informationen gibt als sowohl auf der Modulübersicht als auch auf der persönlichen Seite der/des Professor|in. Der User wird überfordert, da zu viele Informationen gegeben werden. Desweiteren fehlt die

Auflistung der Wahlpflichtmodule. Diese müssen umständlich im Modulhandbuch gesucht werden.

2.2 Festlegung Systemanforderung

Die Hochschulwebsite ist sowohl für Studieninterssierte, als auch für eingeschriebene Studierende konzipiert. Um schneller an die wichtigsten Informationen zu gelangen, soll die Anwendung nur für eingeschriebene Studierende des Studiengangs Angewandte Informatik und Multimediatechnik dienen. Weiterhin sollen nur die wichtigsten Information kurz und knapp dargestellt werden. Daher hat jede Seite ein klares Ziel. So ergeben sich weniger Verlinkungen, sodass der User immer genau weiß wo er sich befindet. Die heutigen Studierenden gehören zur Generation Smartphone. Daher soll der Ansatz "mobile first"verfolgt werden.

3 Grundlagen

Um die folgenden Kapitel besser einzuordnen, werden nun einige technische Grundlagen betrachten. Zuerst wird auf die Webcomponents und im speziellen auf die LitElemente und deren Prinzipien eingegangen. Dann wird auf die Navigation mittels eines Routers eingegangen.

3.1 Webcomponents

Web Components stellen eine Reihe von Webplattform-APIs dar. Mit Ihnen kann wiederverwendbares gekapseltes HTML erstellt und erweitert werden. Das HTML wird in Komponenten gegliedert die in jedem modernen Browser und jeder JavaScript-Bibliothek und jedem Framework verwendet werden kann. Web Components basieren auf vier Anforderungen:

- *Custom Elements:* Grundlage für das Erstellen neuer DOM-Elemente
- *Shadow Dom:* Definiert wie gekapseltes HTML in den Components verwendet wird
- *ES Modules:* Definiert die Einbindung und Wiederverwendung der JavaScript-Dateien
- *HTML Templates:* Definiert HTML-Fragmente die erst zur Laufzeit istanziiert werden

Es existieren viel Bibliotheken mit denen die Erstellung von Web Components erleichtert wird, wie zum Beispiel Hybrids, LitElement, Polymer usw. Im folgenden wird die Bibliothek LitElement näher erläutert. [1]

3.1.1 LitElement

LitElement ist eine Basisklasse zum Erstellen von Web Components. Sie verwendet lit-html um Templates zu definieren und zu rendern und fügt eine API zum Verwalten

von Eigenschaften und Attributen hinzu. Die Eigenschaften werden beobachtet und die Elemente werden asynchron aktualisiert, wenn sich ihre Eigenschaft ändert.

3.1.2 Shadow-Dom

Der Shadow-Dom wird verwendet um den Template-Dom zu Kapseln. Er bietet drei wesentliche Vorteile:

- *Dom-Scoping:* DOM-APIs finden keine Element im Schatten-Dom sodass globale Scripte keinen Zugriff haben
- *Style-Scoping:* gekapselte Styles haben keine Auswirkung auf den Rest des DOM-Baumes
- *Composition:* der Schatten-Dom der Komponente ist von untergeordneten Elementen getrennt, so kann gesteuert werden wie untergeordnete Elemente in das Template gerendert werden sollen [1]

3.1.3 Templates

In einer Render-Funktion der Elementklasse wird das Template für die Component definiert. In dieser Funktion wird das rohe HTML in einem JavaScript template literal innerhalb von back-ticks geschrieben (siehe Abb.). Die Render Methode kann alles zurückgeben was lit-html rendern kann. [2]

```
render() {
  return html`
    <p id="message">Loading</p>
  `;
}
```

Abbildung 3.1: Render-Funktion
[2]

3.1.4 Properties

LitElement verwaltet deklarierte Eigenschaften in einem statischen properties Getter, die entsprechenden Attribute werden in einem Elementkonstruktor initialisiert. So kann sicher gestellt werden das das bei einer geplanten Elementaktualiserung sich die deklarierte Eigenschaft ändert. [2]

3.2 Router

Der Router übernimmt das Navigieren auf der gesamten Seite. Er legt die angezeigte URL fest und verwaltet welche Ansicht gezeigt werden soll. Die verschiedenen Routen werden mit Namen und einem URL Pattern angelegt und können so auseinander gehalten werden. Über Parameter und Querys können dann Informationen übertragen werden, die für die neu aufgerufene Seite von belangen sind.

Wird ein Button mit einem Redirect auf eine neue Ansicht belegt, registriert auch der Router den Klick und wechselt die Path Variable auf den entsprechenden Namen, welcher zu der neuen URL zugehörig ist. Anhand dieser Variable können alle nicht benötigten Ansichten ausgeblendet und die neuen angezeigt werden.

4 Konzept

Anhand der in Kapitel 2 festgelegten Anforderungen wird im folgenden ein Konzept in Form von einem Zustandsdiagramm und die Aufteilung in Komponenten entworfen.

4.1 Systementwurf

Die Abbildung 4.1 zeigt die Zustände, in denen sich der User während der Benutzung der Anwendung befinden kann. Im Zustand der Startseite kann durch ein klick auf die gewünschte Übersicht zum einen in den Zustand der Semester Übersicht und zum andern in die Übersicht der Professoren und Mitarbeiter gewechselt werden. Diesen Zustandswechsel der beiden Übersichten kann jederzeit durch die untere Hauptnavigation getätigt werden. Im Zustand der Semester Übersicht kann durch ein klick auf ein Semester in den Zustand der Modul Übersicht gewechselt werden. Wählt der User hier durch ein klick ein Modul aus, wird die entsprechende Detailseite aufgerufen. Wird im sechsten Semester das Wahlpflichtmodul gewählt, wird wieder ein Übersichtszustand aufgerufen, von dem ebenfalls in die Detailansicht gewechselt werden kann. Im Zustand der Professoren und Mitarbeiter Übersicht kann durch einen klick auf eine Person die entsprechende Detailseite der Person aufgerufen werden.

4.2 Ansichten und Komponenten

Die Webapplikation wurde in sechs Ansichten aufgeteilt: eine Startseite, die Semesterübersicht, die Modulübersicht, die Detailseite der Module, die Professorenübersicht und die Detailseite der Professoren. Aufgrund der verschiedenen Wahlpflichtmodule gibt es noch eine siebte Ansicht dieser Module.

Innerhalb dieser Ansichten gibt es drei Komponenten die in allen wieder verwendet werden: die Überschrifts-Komponente, die Return-Button-Komponente und die Hauptnavigations-Komponente. Der Hauptteil der Ansichten ist in weitere Komponenten gegliedert:

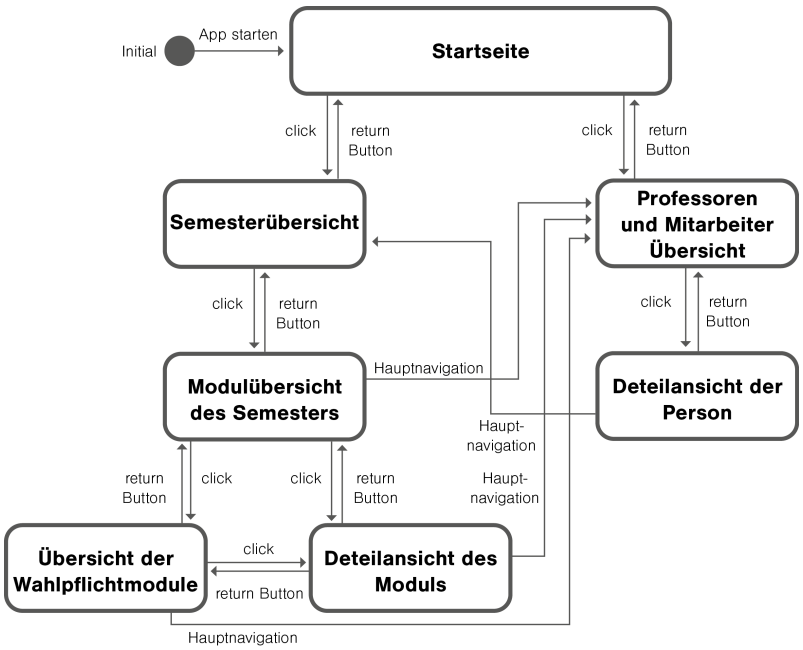

Abbildung 4.1: Zustandsdiagramm (eigene Darstellung)

- Auflistung der Semester
- Auflistung der Professoren und Mitarbeiter
- Auflistung der Module
- Semester-Wochen-Stundenübersicht
- Prüfungsinformationen
- Inhalt der Module
- Bild des Professors bzw. des Mitarbeiters
- Kontaktdaten des Professors bzw. des Mitarbeiters
- Lehre des Professors bzw. des Mitarbeiters

Diese wiederverwendbaren Komponenten können mit dem speziellen Content des jeweilig ausgewählten Semesters, Moduls oder einer bestimmten Person gefüllt werden. Dieser Content wird in einer erweiterbaren JSON-Datei gebündelt (siehe Kapitel 5.2 Datenhandling).

5 Implementierungen

Im folgendem werden auf die für die Anwendung spezifischen Implementieren eingegangen, die auf Basis des im vorherigem Kapitels entworfenen Konzepte beruhen.

5.1 Components

Alle Ansichten legen die Struktur der Webseite fest, indem sie die jeweiligen Komponenten durch dessen Custom Element (definiertes HTML-Tag mit eigenem Verhalten) einbinden (siehe Abb. 5.1 Ansicht mit den jeweiligen Custom Elements).

```
27      render() {
28
29          return html`
30
31          <!--<studyguide-breadcrumb></studyguide-breadcrumb>-->
32          <studyguide-headline></studyguide-headline>
33          <studyguide-professor semesterId="${this.semesterId}" courseId="${this.courseId}"></studyguide-professor>
34          <studyguide-exam semesterId="${this.semesterId}" courseId="${this.courseId}"></studyguide-exam>
35          <studyguide-modulContent semesterId="${this.semesterId}" courseId="${this.courseId}"></studyguide-modulContent>
36          <studyguide-returnButton semesterId="${this.semesterId}"></studyguide-returnButton>
37          <studyguide-navigation></studyguide-navigation>
38          `;
39      }
40  }
```

Abbildung 5.1: Ansicht mit den jeweiligen Custom Elements

Das Verhalten des jeweiligen Custom Elements wird über ein HTML-Template innerhalb der Render-Funktion implementiert. Dieses besteht aus herkömmlichen HTML-Tags (siehe Abb. 5.2 HTML-Template eines Custom Elements).

5.2 Datenhandling

Wie in Abbildung 5.3 zu sehen ist, besteht die Struktur der Daten aus JavaScript-Objekten. Das Objekt *dataSem* enthält ein Array das wiederum Objekte der einzelnen Semester beinhaltet. Dieses besteht wieder aus einem Array mit Objekten der einzelnen Module. Das Objekt *dataProf* enthält ein Objekt Professoren und eine Objekt Mitarbeiter. Diese enthalten wieder ein Array aus Objekten mit den Daten der einzelnen Personen.

```
render() {

    const teach = this.getTeaching();

    return html`
    <link rel="stylesheet" href="../src/styles/font-style.css">
    <link rel="stylesheet" href="../src/styles/details.css">

    <div class="teach detail detail-style detail-style-prof font-fam">
        <h4 class="font-weight-600 font-size-md">Lehre</h4>
        </br>
        ${teach.map((i) => html`<p class="font-weight-300 font-size-s">${i}</p>`)}
    </div>

    `;
}
```

Abbildung 5.2: HTML-Template eines Custom Elements

```
let dataSem =
    [{
        id: 1,
        modules:
            [
                {
                    id: 1,
                    "name": "Mathematik für Ingenieure 1",
                    "modulID": "MA1",
                    "profname": "Ekaterina Auer",
                    "hour": {
                        "vorlesung": 4,
                        "uebung": 2,
                        "praktikum": 0
                    },
                    "exam": "120 Minuten schriftlich",
                    "credits": 8,
                    "content": [
                        "Lineare Algebra",
                        "Komplexe Zahlen",
                        "Vektoren und Matrizen",
                        "Lineare Gleichungssysteme",
                        "Analysis",
                        "Funktionen",
                        "Grenzwerte",
                        "Differential- und Integralrechnung"
                    ]
                },
                {
                    id: 2,
                    "name": "Grundlagen der Technischen Informatik",
```

```
let dataProf =
    {
        professoren:
            [{
                id: 1,
                infos: {
                    "profID": 1,
                    "name": "Ekaterina Auer",
                    "title": "Prof. Dr. rer. nat. habil",
                    "telNumber": "03841753-7322",
                    "email": "ekaterina.auer@hs-wismar.de",
                    "office": {
                        "house": 17,
                        "room": 305
                    },
                    "teaching": [
                        "Mathematik f0r Ingenieure 1",
                        "Mathematik f0r Ingenieure 2"
                    ]
                }
            },
            {
                id: 2,
                infos: {
                    "profID": 2,
                    "name": "Matthias Kreuseler",
```

Abbildung 5.3: Objektstruktur der Semester und Module und der Professoren und Mitarbeiter

Durch spezielle Getter-Methode werden die jeweiligen Daten aus den Objekten in die Komponenten geschrieben. In der Abbildung 5.4 Methode für das Datenhandling wird am Beispiel *getTeaching* gezeigt wie die Daten für die Komponente textitLehre aus der textitdataProf gezogen werden. Hierzu wird die Professoren ID der angeklickten Person aus der Route mit der ID in der *dataProf* verglichen und die benötigten Daten in ein Array geschrieben. Dieses Array wird dann, wie in Abbildung 5.2 HTML-Template eines Custom Elements gezeigt, mithilfe der Map-Funktion in das Template geschrieben.

```
13      getTeaching() {
14          let teach = [];
15          if (this.profId) {
16              const professoren = prof.professoren;
17              const professor = professoren.find((prof) => prof.id === this.profId);
18              if (!professor) {
19                  const dozenten = prof.dozenten;
20                  const dozent = dozenten.find((doz) => doz.id === this.profId);
21                  teach = dozent.infos.teaching;
22              } else {
23                  teach = professor.infos.teaching;
24              }
25          }
26          return teach
27      }
```

Abbildung 5.4: Methode für das Datenhandling

5.3 Navigation mithilfe des Routers

Jedes anklickbare Element enthält einen Clickhandler, welcher die Navigation zur nächsten Seite ermöglicht und über den Router gesteuert wird. Welche Daten benötigt sind kann über die URL herausgefunden werden und wird in der speziellen Getter-Methode abgefragt.

Wie in Abbildung 5.5 zu sehen, werden in der *routes( )*-Methode alle benötigten Routen festgelegt. Der Name gibt an welches Ziel die die Route hat. Während das Pattern festlegt wie die zugehörige URL aussieht. Diese wird mit Variablen versehen (*semesterID*), um beispielsweise die verschiedenen Semester auseinander zu halten. Parameter werden nicht in der URL angezeigt, sondern direkt an die einzelnen Ansichten übergeben, um die Information weiter zu verarbeiten.

In der Render-Funktion (siehe Abb. 5.6) werden alle Ansichten über ihre Tags eingebunden und anhand der 'route' Variable, welche den Namen der aktuell aufgerufenen Seite enthält, ein und ausgeblendet.

```
24      static get routes() {
25          return [{
26              name: 'home',
27              pattern: ''
28          }, {
29              name: 'semesterview',
30              pattern: 'semesteruebersicht'
31          }, {
32              name: 'moduls',
33              pattern: 'semesteruebersicht/semester/:semesterId/modul'
34          }, {
35              name: 'moduldetail',
36              pattern: 'semesteruebersicht/semester/:semesterId/modul/:courseId'
37          }, {
38              name: 'obligation',
39              pattern: 'semesteruebersicht/semester/:semesterId/modul/:courseId/obligation_modules'
40          }, {
41              name: 'obligationdetail',
42              pattern: 'semesteruebersicht/semester/:semesterId/modul/:courseId/obligation_modules/:obligationId'
43          }, {
44              name: 'profview',
45              pattern: 'professor_und_mitarbeiter'
46          }, {
47              name: 'profdetail',
48              pattern: 'professor_und_mitarbeiter/:profId'
49          }];
50      }
```

Abbildung 5.5: routes( )-Methode

```
66      render() {
67          return html`
68            <app-main active-route=${this.route}>
69              <studyguide-home route="home" ?hidden="${this.route !== 'home'}"></studyguide-home>
70              <studyguide-semester route="semesterview" ?hidden="${this.route !== 'semesterview'}"></studyguide-semester>
71              <studyguide-moduls route="moduls" semesterId="${this.params.semesterId}" ?hidden="${this.route !== 'moduls'}"></studyguide-moduls>
72              <studyguide-obligation-moduls route="obligation" semesterId="${this.params.semesterId}" ?hidden="${this.route !== 'obligation'}"></studyguide-obligation-moduls>
73              <studyguide-moduldetails route="moduldetail" semesterId="${this.params.semesterId}" courseId="${this.params.courseId}" ?hidden="${this.route !== 'moduldetail'}"></studyguide-moduldetails>
74              <studyguide-obligation-moduldetails route="obligationdetail" semesterId="${this.params.semesterId}" courseId="${this.params.courseId}" obligationId="${this.params.obligationId}" ?hidden="${this.route !== 'obligationdetail'}">
75              <studyguide-prof route="profview" ?hidden="${this.route !== 'profview'}"></studyguide-prof>
76              <studyguide-profdetails route="profdetail" profId="${this.params.profId}" ?hidden="${this.route !== 'profdetail'}"></studyguide-profdetails>
77            </app-main>
78          `;
79      }
```

Abbildung 5.6: Render-Methode des Routers

6 Zusammenfassung und Ausblick

Innerhalb dieses Projektes wurde eine Anwendung entwickelt die es Studenten erleichtert an wichtige Informationen, während ihres Studiums der Angewandten Informatik an der Hochschule Wismar, zu gelangen. Dazu wurde als Erstes eine Analyse der Hochschul-Website vorgenommen und anhand der Ergebnisse Anforderungen an die Webapplikation festgelegt. Weiterhin wurden Webcomponents allgemein und im speziellen die LitElemente erläutert. Im Kapitel 4 Konzepte wurde der Systementwurf und die Aufteilung der Anwendung in die einzelnen Komponenten und Ansichten vorgestellt.

Ergebnis

Ausblick

Literaturverzeichnis

[1] *Introduction - What are web componts?*
https://www.webcomponents.org/introduction [17.05.2020]

[2] *LitElement* 2018 Polymer Project
https://lit-element.polymer-project.org/ [17.05.2020]

[3] *LitElement Router*
https://www.npmjs.com/package/lit-element-router [28.05.2020]

Abbildungsverzeichnis

Selbstständigkeitserklärung

Hiermit erklären wir, dass wir die hier vorliegende Arbeit selbstständig, ohne unerlaubte fremde Hilfe und nur unter Verwendung der aufgeführten Hilfsmittel angefertigt haben.

Ort, Datum Unterschriften