Dokumetation

Entwicklung des Backends

Stand: 16/05/22

Autoren :

Jan Händl

Dustin Heyer

Inhaltsverzeichnis

1 Zielbestimmung

Entwickelt des Backend. Das inkludiert das festlegen der Werkzeuge, das Erstellen von Speicherstrucktur und das nachvollziehbare Dokumentieren der Arbeit.

2 Toolaufstellung

In Folge zeigen und Begründen wir unsere Toolauswahl für die einzelnen Teilbereiche des Projektes.

2.1 Server/ Online-Speicher

Die beschränkte Auswahl ist auf die folgenden drei Optionen gefallen. Zum einen aus Erfahrung und zum anderen damit von jeder Speicher Möglichkeit ein Vertreter aufgeführt ist.

	Minio	Express	MongoDB
Vorrangiger Einsatz	Cloud Storage	Rest API	DB Speicherung
Vorteil	Umgang mit großen Datenmengen Docker	Schnelles Aufsetzten und einsetzten Node.js	JSON Format speicherbar Docker
Erfahrung	Dustin vorhanden	Jan vorhanden	Keine

Die Wahl wird hier nur zum Teil von unserem Einsatz-Specktrum bestimmt. Damit entscheiden wir uns für Mongo DB, da dies die Speicherung von XApi Statements direkt unterstützt. Eine solche Datenbank aufzusetzen ist extra Aufwand aber die Funktionalität von MongoDB passt dafür perfekt zu unserem Zwecke. Alle anderen Daten werden in Minio Ablegen und Speichern. Express nutzen wir in diesem Kontext nicht, da es nicht weiter nötig ist und alle Funktionalitäten die wir benötigen bereits vorhanden sind.

2.2 Auswerten von Daten

R vs. SPSS vs. Python vs. Excel

	R	SPSS	Python	Exel
Vorteile	Statistiksoftware Viele Pakete Gut für komplexe Statistik	Alt Hat die gängigsten Methoden (ausgereift) meist aber im Bereich der Sozial-wissenschaft genutzt	Allzweckwaffe Viele Tools Pandas Datenanalyse	nur einfache statistische Aufgaben möglich Add in Datenanalyse Macros nutzbar

Hier schließen wir die ersten zwei Optionen aus. Diese wären erneut unverhältnismaßig Komplex für unser Projekt. Damit bleiben uns noch Python und Exel. Wobei wir erstmal bei beiden bleiben können. Python können wir nutzen um einen neuen Server-Stack aufzusetzen welcher aktuelle Auswertungen durchführt oder passiv, auf Bedarf, Datenanalysen durchführt. Exel würde nur passiv genutzt werden ist aber unsere Backup Lösung.

2.3 Anwendungsmöglichkeiten

Hier zählen wir die vier gängigsten Applikationmöglichkeiten vor und wägen ab welche Option für uns am passensten ist.

	Webapplikation	normale Applikation	Handy App	Eigenes Tool
Allgemein	leicht zu Entwickelnbekanntester Bereich Html-Auszug möglich	Auf dem Computer nutzbar Näher an xApi	Direkt fürs Handy Am nächsten am Nutzer und der xApi	Am schwersten zu machen Entwicklung + Elektronik und BS
unsere Möglichkeiten	React, Angular,VoeJS	Elektron, Java (AWT)	Android Studio	/

Erstmal werden wir alles als Webapplikation entwickeln. Dort ist die meiste Erfahrung vorhanden und somit der bester Fortschritt zu erwarten. Außerdem existieren Applikationen zum Umwandeln von Wepapp-Code zur den Anderen Applikationstypen, womit mehreres auf einmal erreicht werden kann.

2.4 Webapplikationen

	React	Angular	VueJS
Basis	Open-Source JavaScript-Bibliothek	Entwicklungsplattform, die auf dem Typeskript basiert	JavaScript-Framework
standart Einsatz	React wird verwendet, um UI-Komponenten in jeder App mit häufig variablen Daten zu erstellen	Angular wird hauptsächlich verwendet, um komplexe Apps für Unternehmen wie Single-Page-Apps und progressive Web-Apps zu erstellen	Progressivität und inkrementelle Einsetzbarkeit Setzt mehr auf HTML
Lernkurve	moderat aufgrund kleiner Package-Größe	steil aufgrund vieler eingebauten Funktionalitäten	flache Lernkurve vorsicht vor Spagetti-Code
Hauptargument	Flexibilität Großes Ökosystem Kleine Teams Viel Wahlfreiheit bei den benutzbaren Packages	TypeScript Große Comunity Für große Apps Objekt orientiert	Großes Ökosystem Keine langwierigen Build-Prozesse keine Syntaxerweiterungen wie TypeScript oder JSX
Erfahrung	vorhanden	keine	keine

Diese Entscheidung ist die komplizierteste. Da mit jeder der drei Optionen unser Ziel erreicht werden kann. Anhand der Fakten entscheiden wir uns gegen Angular. Es müste neu gelernt werden und ist eher für größere Projekte als unseres gedacht. Somit behindert es eher den Prozess anstatt ihn zu unterstützen. Weiterhin entscheiden wir uns auch gegen VueJS da es durch den Fokus auf HTML nicht in unsere Hauptintension passt. Damit entscheiden wir uns für React, da vorallem die oben gelisteten Hauptargumente zu unserer Aufgabe und Projektgruppe passt. Zusätlich ist keine weitere Einarbeitungszeit nötig und Erfahrung vorhanden.

3 UI-Frameworks

Die UI (UserInterface) einer Applikation ist die direkte Schnittstelle zwischen dem Nutzer und der Anwendung. Dementsprechen ist eine gute, intuitive UI (oder auch GUI) ein muss. Aber eine komplett UI selbst zu Gestalten und Umzusetzen ist schwer und kostet Zeit. Deswegen wird in Folge das Thema UI-Frameworks näher beleuchtet.

3.1 Was und Wofür sind UI-Frameworks ?

Solch ein Framework ist im Allgemeinen eine Software-Suite, die dem Benutzer bereits vorgefertigte Schnittstellen präsentiert. Diese dienen dazu eigenen UI-Komponenten zu erstellen oder bereits fertige Elemente selbst zu modifizieren. Framework bezieht sich auf APIs und Dokumentationen, durch diese ist beschrieben wie die genannten Vorgänge ausführbar sind. Dabei werden sie meist in der Webentwicklung genutz und bassieren so vollständig auf CSS, Javascript und HTML. Durch sie ist es möglich schnell und flexibel eigenen UIs mit wenig Vorwissen zu erstellen.

3.2 Warum werden sie genutzt ?

Die Nutzung eines UI-Frameworks bringt einige Vorteile mit sich. Zum einen kann man einfach und schnell Oberflächen erstellen. Zum anderen ist der Aspekt der Wiederverwendbarkeit und Einheitlichkeit gegeben . Frameworks bieten meist ein einheitliches Set an UI-Elementen an welche harmonisch miteinander genuzt werden können. dazu kommt das der Stil somit auf mehrern Seiten (Produkten, Aplikationen,...) ohne Probleme wiederverwendet werden kann. Was einem eine eigenen Erstellung ersparrt.

3.3 Wíe werden sie installiert

Die meisten Frameworks wie Angular Material, Bootstrap, NGX Bootstrap, DHTMLX, Webix, Material UI, Evergreen oder Rebass konnen simple per NPM oder YARN installiert werden. Was eine simple und schnelle Anwendung bedeutet. Ein paar von den gennanten (zb. Bootstrap) können auch als Programm installiert werden.

3.4 Unserer Entscheidung

Wir haben uns für Material UI entschieden. Es gibt uns genügen Freiheit um unsere Ideen umzusetzen. Dazu kommt, dass bereits jemand Erfahrung mit dem Umgang hat und es eine Figma-Integrations gibt. Die vielen bereitgestellten Komponenten ermöglichen einen schnelles und flexibles Arbeiten.

Wir nutzen folgende Komponenten: accordion, app-bar, card, drawer, list, list-item, paper, step, stepper.

3.4.1 Beispiel der Erzeugung (Anhand App-Bar)

```
import * as React from 'react';
import AppBar from '@mui/material/AppBar';

export default function ButtonAppBar() {
     return (
   <Box sx={{ flexGrow: 1 }}>
      <AppBar position="static">
        <p>NEWS</p>
      <button color="inherit">Login</button>
      </AppBar>
   </Box>    );
}
```

4 Datenstruktur

Die in der Applikation verwendeten Daten werden in Strukturen gehalt. Dazu zählen die Nutzerdaten Moduldaten und Anwendungsdaten. Diese werden Im JSON-Format ("JavaScript Object Notation"gehalten und auf Minio im Raw gespeichert.

4.1 Nutzerdaten

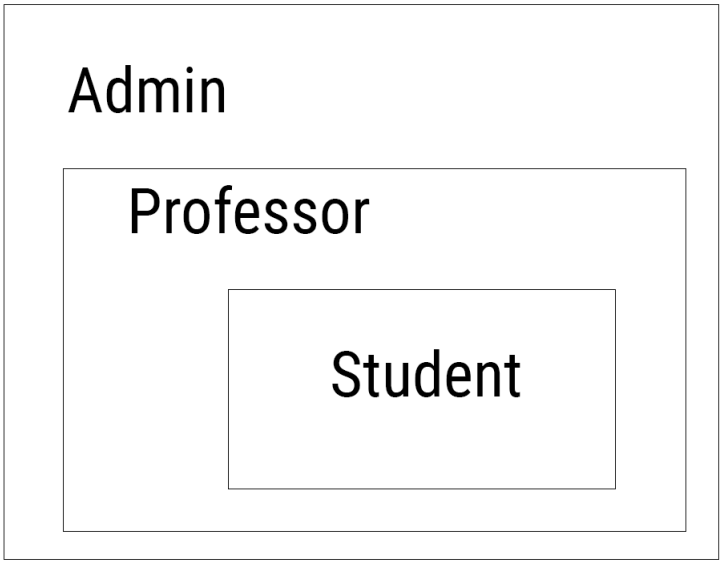
Die Nutzerdaten werden nach folgenden Prinzip gehalten

```
{
    "username": "m",
     "email": "m",
     "age": "11",
     "Nutzergruppe": "Studen_User",
     "passwort":"423a4ca5d3447996de10c2e5f7e3fc38f0c4f8790f84...bd468c3e",
```

```
7       "id": 6,
8       "module":["100000","100001"]
9   }
```

Hierbei ist der Username und die Id eindeutig. Die anderen Parameter werden für die Anmeldung und die Zuweisung der Module genutzt.

Username, Email und Age, sowie Id werden genutzt um den Nutzer Anzumelden und ihn als eigene Identität zu kennzeichnen. Das Passswort ist zum anmelden gedacht. Dabei ist zu beachten das aus Datensicherheitsgründen das Passwort gehasht wird. Hier mit dem sha512 Algortihmus. Für die weiter Sicherheit wird das Passwort bevor es gehast wird mit einen zusätlichen String konkateniert. Das Passwort wird in dem Gebiet des Hashings dann als Pfeffer und der zusätliche String als Salz bezeichnent. Die Nutzergruppe gibt an um welche Art von Nutzer es sich handelt. Dabei gibt es die Auswahl zwischen Student_User, Professor_User und Admin_User. Je nach dem welche Art Nutzer angemeldet ist stehen zusätliche Funktionen zur Verfügung. Die Nutzergruppen sind hierarchisch geordnet was den Zugriff auf Funktionen angeht.



Der letzte Parameter Modul wird genutz um festzuhalten in welchen Modulen der jeweilige Nutzer sich angemeldet hat. Dabei wird die Id des Moduls genuzt da diese eindeutig ist.

4.2 Studiengang-Daten

Die Studiengang-Daten werden anch folgendem Prinzip gehalten:

```
1   {
2       "sgNummer": "I43",
3       "sgName": "Medieninformatik",
4       "sgRegelstudienzeit": "6",
5       "sgCredits": 180,
6       "module": [
7                   ...
8               ]
9   }
```

Die Ersten Parameter dienen der Identifikation des Studiengangs. Dazu gehören die Studiengang-Nummer (sgNummer), der Name des Studiengangs (sgName), die Regelstudienzeit (sgRegelstudienzeit) und die zu erreichenden Credits (sgCredits). Danach folgen die Auflistung der Module wie folgt:

```
1   {
2       "modul":"Elektronik für Medieninformatiker",
3       "modulnummer":"E802",
4       "credits":3,
5       "semester":1
6   },
```

Für jedes Modul werden vier Parameter gespeichert.

- modul: Name des Moduls
- modulnummer: Identifikationsnummer des Moduls

- credits: Anzahl an zu erhaltenden Credits bei Abschluss
- semester: angedachte Stelle nach Studienablaufplan

4.3 Moduldaten

Die Moduldaten werden nach folgenden Prinzip gehalten:

```
{
    "modulId": "100000",
    "modulName": "Test",
    "leader": "Prof.Prof",
    "leaderMail": "prof.p@gmail.com",
    "usedModules": [
                    ...
                      ],
     "deadlines": [
                    ...
                  ],
    "news": [
           ...
         ],
    "schedule": [
              ...
             ],
    "grades": [
              ...
         ]
}
```

Die Parameter ModulId und ModulName dienen der Identifikation des Moduls. Leader und leaderMail sind die Daten des Erstellers. Der Ersteller ist ein Nutzer vom Nutzertyp Professor_User oder Admin_User. Die restlichen Daten sind zur Spezifikation des Moduls.

Dort werden verschiedene Sachverhalte festgehalten.

```
"usedModules": [
                {
                   "id": "0",
                         "name":"Programm1",
                         "spezificationNumber": "1111111"
                     },
                 {
                       "id": "1",
                       "name": "Programm2",
                        "spezificationNumber": "1111110"
                   }
                ]
```

Der Parameter "usedModules" wird verwendet um verschiedene Anwendungen im Modul zu speichern. Diese können auch leer sein. Die module werden wieder als neue Objekte gespeichert bestehend aus id, name, und spezificationNumber. Die Id und der Name sind zum identfizieren der Anwendung und die Spezifikationsnummer um die Ausprägung dazustellen. Die Ausprägung kann gesehen werden wie ein Speicherstand welcher geladen und gespeichert werden kann. Dies Erfolgt dann über ein extra Menü.

```
"deadlines": [
                {
                  "name": "abgabe1",
                  "doDatum": "2012-07-05T18:30:32.360Z"
                },
                   {
                     "name": "abgabe2",
                     "doDatum": "2012-08-05T18:30:32.360Z"
                  }
               ]
```

Der Parameter "deadlines" wird genutz um Abgaben im Modul dazustellen. Diese können dann dem Nutzer direk angezeigt werden sowie im Modul selbst. Es werden hier wieder einzelne Objekte gespeichert. Diese verfügen über zwei weiter Parameter:

- name: zum Kennzeichnen um welche Abgabe es sich handelt
- doDatum: zum Kennzeichnen des Abgabedatums

```
"news": [
          {
              "name": "Zwischneprüfung",
                  "expiryDate": "2012-07-05T18:30:32.360Z"
              }
      ]
```

Der Parameter "news" wird genutzt um kurzfristige Änderungen/Informationsen vom Kursleiter darzustellen. Diese können dann dem Nutzer direk angezeigt werden sowie im Modul selbst. Dieser verfügt ebenfalls über zwei weiter Parameter:

- name: Name/Beschreibung der Information
- expiryDate: zum Einstellen wie lange Die Information angezeigt wird.

```
"schedule": [
            {
              "room": "Z100",
               "time": "9:20-10:50",
               "dof" : "Montag",
               "week": "1",
               "type": "VL"
            }
        ]
```

Der Parameter "schedule" wird genutzt um den Wochenplan darszutellen. Hier wird als Objekt jede Unterrichteinheit festgelegt. Der Parameter verfügt über folgende Objekt-Parameter:

- room: Raum wo die Einheit stattfindet
- time: Zeitraum der Einheit
- dof: Wochentag
- week: Gerade und ungerade Woche bzw. AB-Woche
- type: Unterscheidung in VL-Vorlesung, ÜB-Übung und Prak-Praktikum

```
"grades": [
            {
              "name": "abgabe2",
               "datum": "2012-07-05T18:30:32.360Z",
               "grade": "1",
               "studentID": "1"
            }
        ]
```

Der Parameter "Grades" wird genutzt um die Noten in einem Modul zu speichern. Diese können dann gefiltert dem Nutzer direk angezeigt werden sowie ebenfalls gefiltert im Modul selbst. GEfiletert steht in diesem Zusammenhang für einem Nutzer werden nur seine eigenen Noten angezeigt. Der Parameter verfügt über folgende Objekt-Parameter:

- name: Zur Angabe der Prüfungsnummer und Prüfungsname
- datum: Zur Angabe des Prüfungsdatums
- grade: Note des Nutzers
- id: Nutzer-ID zur Identifikations des geprüften Nutzers.