

Dokumentation

Entwicklung des Backends

Stand: 16/05/22

Autoren :

Jan Händl

Dustin Heyer

Inhaltsverzeichnis

| | |
|---|----------|
| 1 Zielbestimmung | 1 |
| 2 Toolaufstellung | 1 |
| 2.1 Server/ Online-Speicher | 1 |
| 2.2 Auswerten von Daten | 1 |
| 2.3 Anwendungsmöglichkeiten | 1 |
| 2.4 Webapplikationen | 2 |
| 3 UI-Frameworks | 2 |
| 3.1 Was und Wofür sind UI-Frameworks ? | 3 |
| 3.2 Warum werden sie genutzt ? | 3 |
| 3.3 Wie werden sie installiert ? | 3 |
| 3.4 Unserer Entscheidung | 3 |
| 3.4.1 Beispiel der Erzeugung (Anhand App-Bar) | 3 |
| 4 Datenstruktur | 3 |
| 4.1 Nutzerdaten | 3 |
| 4.2 Studiengang-Daten | 4 |
| 4.3 Moduldaten | 5 |
| 5 xApi | 6 |
| 5.1 xApi-Anweisungen | 7 |
| 5.2 Learning Record Store | 8 |
| 5.3 Learning Management System | 10 |

1 Zielbestimmung

Entwickelt des Backend. Das inkludiert das festlegen der Werkzeuge, das Erstellen von Speicherstrukturen 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 Aufsetzen und einsetzen Node.js | JSON Format speicherbar Docker |
| Erfahrung | Dustin vorhanden | Jan vorhanden | Keine |

Die Wahl wird hier nur zum Teil von unserem Einsatz-Spektrum 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 | Excel |
|----------|---|---|--|--|
| Vorteile | Statistiksoftware Viele Pakete Gut für komplexe Statistik | Alt Hat die gängigsten Methoden (ausgereift) meist aber im Bereich der Sozialwissenschaft 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 Excel. 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. Excel 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 besten 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. Zusätzlich 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 genutzt und basieren 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 genutzt werden können. dazu kommt das der Stil somit auf mehreren Seiten (Produkten, Applikationen,...) ohne Probleme wiederverwendet werden kann. Was einem eine eigenen Erstellung erspart.

3.3 Wie werden sie installiert ?

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

3.4 Unserer Entscheidung

Wir haben uns für Material UI entschieden. Es gibt uns genügend Freiheit um unsere Ideen umzusetzen. Dazu kommt, dass bereits Erfahrung mit dem Umgang vorhanden ist und es eine Figma-Integration gibt. Die vielen bereitgestellten Komponenten ermöglichen einen schnellen und flexiblen 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)

```
1 import * as React from 'react';
2 import AppBar from '@mui/material/AppBar';
3
4 export default function ButtonAppBar() {
5   return (
6     <Box sx={{ flexGrow: 1 }}>
7       <AppBar position="static">
8         <p>NEWS</p>
9         <button color="inherit">Login</button>
10       </AppBar>
11     </Box>
12   );
13 }
```

4 Datenstruktur

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

4.1 Nutzerdaten

Die Nutzerdaten werden nach folgenden Prinzip gehalten

```
1 {
2   "username": "m",
3   "email": "m",
4   "age": "11",
5   "Nutzergruppe": "Studen_User",
6   "password": "423a4ca5d3447996de10c2e5f7e3fc38f0c4f8790f84...bd468c3e",
7 }
```

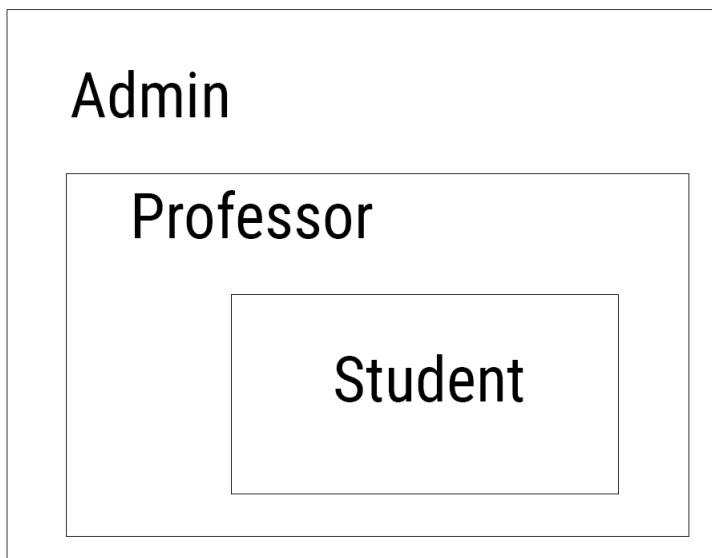
```

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 Passwort ist zum anmelden gedacht. Dabei ist zu beachten das aus Datensicherheitsgründen das Passwort gehasht wird. Hier mit dem sha512 Algorithmus. Für die weitere Sicherheit wird das Passwort bevor es gehasht wird mit einem zusätzlichen String konkateniert. Das Passwort wird in dem Gebiet des Hashings dann als Pfeffer und der zusätzliche String als Salz bezeichnet. 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 zusätzliche Funktionen zur Verfügung. Die Nutzergruppen sind hierarchisch geordnet was den Zugriff auf Funktionen angeht.



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

4.2 Studiengang-Daten

Die Studiengang-Daten werden nach folgendem Prinzip gehalten:

```

1 {
2   "sgNummer": "143",
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:

```

1 {
2   "moduleId": "100000",
3   "moduleName": "Test",
4   "leader": "Prof. Prof",
5   "leaderMail": "prof.p@gmail.com",
6   "usedModules": [
7     ...
8   ],
9   "deadlines": [
10    ...
11   ],
12   "news": [
13     ...
14   ],
15   "schedule": [
16     ...
17   ],
18   "grades": [
19     ...
20   ]
21 }
```

Die Parameter moduleId und moduleName 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.

```

1 "usedModules": [
2   {
3     "id": "0",
4     "name": "Programm1",
5     "spezifikationNumber": "1111111"
6   },
7   {
8     "id": "1",
9     "name": "Programm2",
10    "spezifikationNumber": "1111110"
11  }
12 ]
```

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 spezifikationNumber. Die Id und der Name sind zum identifizieren 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.

```

1 "deadlines": [
2   {
3     "name": "abgabe1",
4     "doDatum": "2012-07-05T18:30:32.360Z"
5   },
6   {
7     "name": "abgabe2",
8     "doDatum": "2012-08-05T18:30:32.360Z"
9   }
10 ]
```

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

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

```

1 "news": [
2   {
3     "name": "Zwischneprüfung",
4     "expiryDate": "2012-07-05T18:30:32.360Z"
5   }
6 ]
```

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

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

```
1 "schedule": [  
2   {  
3     "room": "Z100",  
4     "time": "9:20-10:50",  
5     "dof": "Montag",  
6     "week": "1",  
7     "type": "VL"  
8   }  
9 ]
```

Der Parameter "schedule" wird genutzt um den Wochenplan darzustellen. Hier wird als Objekt jede Unterrichtseinheit 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

```
1 "grades": [  
2   {  
3     "name": "abgabe2",  
4     "datum": "2012-07-05T18:30:32.360Z",  
5     "grade": "1",  
6     "studentID": "1"  
7   }  
8 ]
```

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

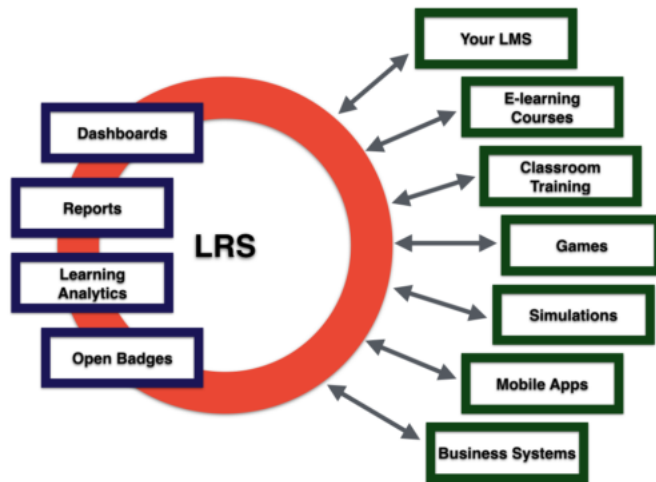
- name: Zur Angabe der Prüfungsnummer und Prüfungsname
- datum: Zur Angabe des Prüfungsdatums
- grade: Note des Nutzers
- id: Nutzer-ID zur Identifikation des geprüften Nutzers.

5 xApi

Die xApi (Experience API) ist eine Spezifikation um das Sammeln von Daten, welche die Erfahrungen und das Verhalten von Nutzern von Lernplattformen, vereinheitlichen soll. Die xApi wurde 2013 von der ADL-Initiative vorgestellt, um neue Herausforderungen im Feld des E-Learnings zu bewältigen. Dazu gehören in erster Linie das breitere Aufkommen von mobilen Lernen (via Smartphones oder Tablets), Implementieren von AR, VR und Offline-Lernen.

Das Ecosystem der xApi besteht aus drei wesentlichen Teilen:

- Client
- LRS (Learnign Record Store)
- LMS (Learning Management System)



Der Client kommuniziert via xApi-Anweisungen mit dem LRS. Der LRS stellt demnach die Daten für die Auswertung bereit, welche von dem LMS übernommen wird.

5.1 xApi-Anweisungen

Die xApi Anweisungen sind im JSON Format mit den Properties „Actor“, „Object“ und „Verb“. Weitere Properties können nach belieben hinzugefügt werden.

Der Actor beschreibt wer die Action ausführt. Dabei kann dies eine einzelne Person, aber auch eine Gruppe von Personen sein. Der Actor muss eindeutig indentifizierbar über eine ID oder eine email Adresse sein.

```

1 {
2   "actor": {
3     "mbox": "test@test.com",
4     "id": "12312312341"
5   }
6 }

```

Das Verb beschreibt die Interaktion des Actors mit dem System. Hierbei ist die Anzahl unterschiedlicher Verben nicht limitiert. Um ein Verb zu beschreiben wird das Property „id“ verwendet, welche auf eine Definition verweist, und optional das Property „display“ welches eine Definition der Aktion in einer lebaren Form enthält, bestenfalls in mehreren Sprachen.

```

1 {
2   "id": "http://example.com/xapi/verbs#defenestrated",
3   "display": {
4     "en-US": "defenestrated",
5     "es": "defenestrado"
6   }
7 }

```

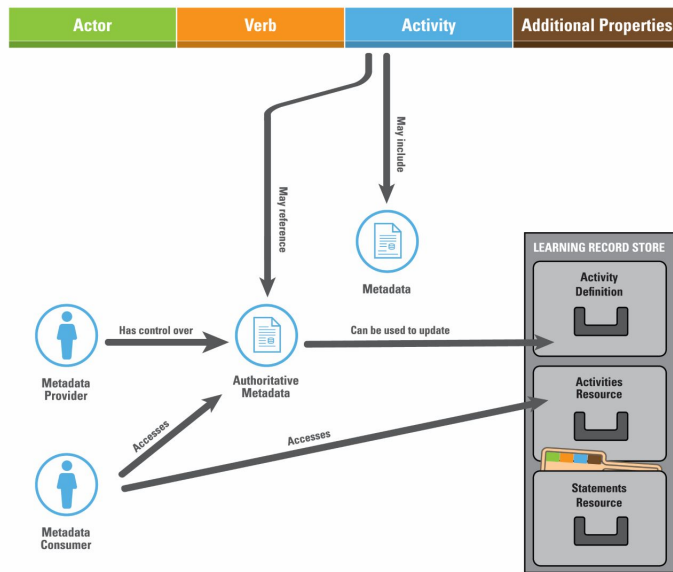
Das Object beschreibt das Object auf welchem die Aktion ausgeführt wurde. Dabei kann das Object ein Actor, eine Activity, ein Statement oder ein Substatement sein. Hierbei muss eine „id“ angegeben werden. Optional ist die Property „objectType“. Die „id“ ist ein identifier für eine eindeutige Activität, der objectType ist eine Activität, falls vorhanden.

```

1 {
2   "objectType": "StatementRef",
3   "id": "8f87ccde-bb56-4c2e-ab83-44982ef22df0"
4 }

```

xAPI Statement



5.2 Learning Record Store

Die Aufgabe eines LRS ist das Speichern von xApi Statements. Die Übermittlung dieser Statements erfolgt über REST Apis.

| Endpunkt | Methode | Parameter | Typ | Standard | Beschreibung | Notwendig | Rückgabewert |
|-------------|-----------------------------|--------------|------------------------|----------|--|-----------|----------------|
| /statements | PUT | statementId | String | - | ID des Statements welches gespeichert wird | + | 204 |
| /statements | Post | StatementID | Array[String] | - | IDs der Statements welche gespeichert werden | + | 200 |
| /statements | GET | statementID | String | - | Gibts statements gefiltert nach Werten zurück: Filtert nach ID des Statements | - | 200 |
| | | verb | verbID | - | Filtert nach ID des Verbs | - | |
| | | activity | activityID | - | Filtert nach ID der Activity | - | |
| | | registration | UUID | - | Filtert nach UUID | - | |
| | | since | Timestamp | - | Filtert nach Timestamp seit | - | |
| | | until | Timestamp | - | Filtert nach Timestamp until | - | |
| /state | (PUT POST GET DELETE) | activityID | String | | Speichert. löscht oder downloaded states welche mit den gegebenen Werten in Kontext stehen | + | 204 (GET: 200) |
| | | agent | JSON (not implemented) | | Agent Parameter nicht implementiert | (+) | |
| | | registration | UUID | | Registration ID welche mit dem state in Kontext steht | - | |
| | | stateID | String | | Id des states | + | |
| /activities | GET | activityId | String | | downloaded documente welche mit den gegebenen Werten in Kontext stehen | + | 200 |
| /activities | (PUT POST DELETE) | activityId | String | | Speichert. oder löscht documente welche mit den gegebenen Werten in Kontext stehen | + | 204 |
| | | profileId | String | | (nicht implementiert) | (+) | |

5.3 Learning Management System

Das Learning Management System ist eine Software zur Verwaltung und Auswertung der Daten und der einzelnen Nutzer. Wichtige Funktionalitäten in der Verwaltung ist die Registrierung, das Löschen, die Authentification und die Rechteverwaltung der Nutzer. Bei der Auswertung sollte Wert auf eine aussagekräftige Visualisierung der Daten gelegt werden, sowie die Zugangsbeschränkung der Daten basierend auf einem vorgegebenen Rollen bzw. Rechtesystem der Nutzer.

Weitere Features eines Learning Managent Systems können sein:

- Management von Inhalten und Bereitstellung von Materialien
- Soziale Features (Chats,Foren etc.. .)
- Lernaktivitäten (Quiz,Test, oder andere dynamische Aufgaben)
- Zeitmanagement, wie Terminplanung via Kalender
- Anerkennung von Lernzielen und Prüfungsleistungen