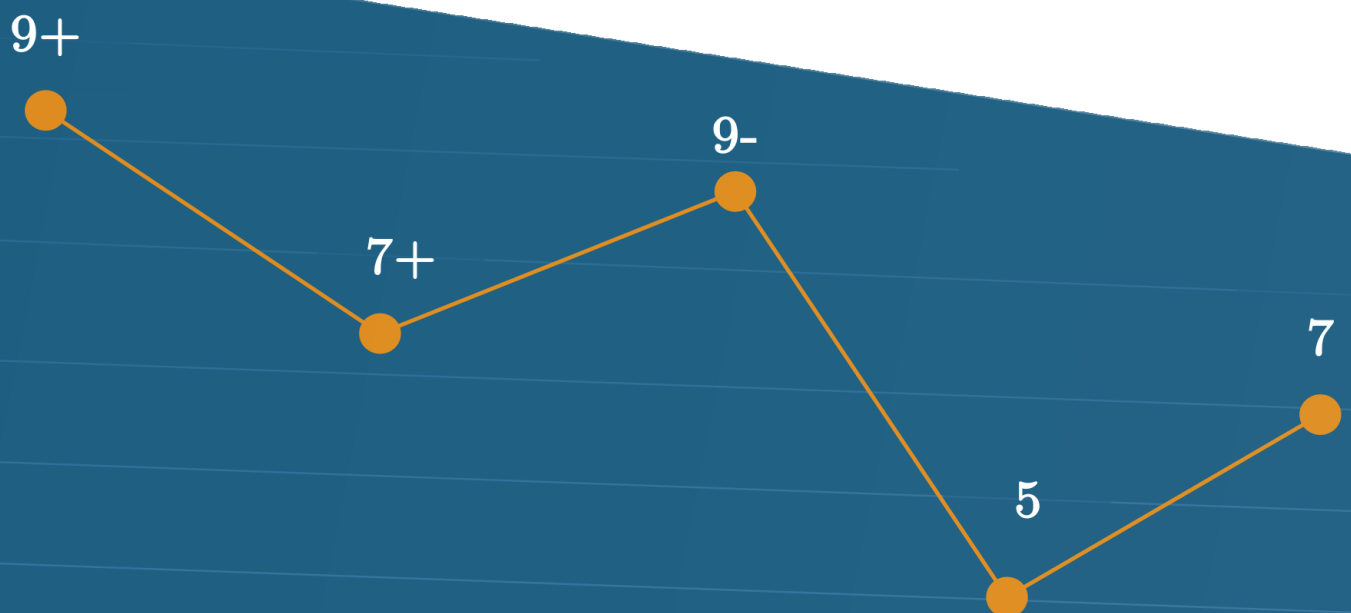


# Notendarstellung

## Angular-Applikation



Landesberufsschule  
„Christian J. Tschuggmall, Brixen“  
5. HIM - Informatik



## Inhaltsverzeichnis

1	Einleitung.....	4
2	Planung.....	5
2.1	Zielsetzung.....	7
2.1.1	Hauptziel.....	8
2.1.2	Teilziele.....	8
2.1.3	Nicht-Ziele.....	8
2.2	Stakeholder.....	9
2.3	Projektstrukturplan.....	12
2.4	Risikoanalyse.....	14
2.5	Projektzeitplan.....	19
3	Programmierung.....	21
3.1	TypeScript.....	21
3.2	Angular.....	22
3.3	Angular Vertiefung.....	22
3.3.1	Components.....	23
3.3.2	Templates.....	24
3.4	ClassMate.....	25
3.4.1	API.....	26
3.4.1.a	Services.....	28
3.4.1.b	Authentifizierung.....	29
3.4.2	Login.....	31
3.4.2.a	Routing.....	31
3.4.2.b	Guard.....	32
3.4.3	Noten.....	33
3.4.3.a	Tables.....	33
3.4.3.b	Expansion Panel.....	34
3.4.3.c	Cards.....	35
3.4.4	Details der Noten.....	37
3.4.4.a	NGX-Chart.....	37
3.5	Design.....	40
3.5.1	Material Design.....	40
3.5.2	Angular und Material.....	41
4	Deployment.....	42
4.1	Hosting.....	42
4.1.1	Reverse-Proxy.....	43
4.2	PWA.....	43
5	Endergebnis.....	44
5.1	Zugang und Login.....	44

5.2 Notenübersicht.....	45
5.3 Details.....	46
5.4 Installation.....	47
6 Fazit.....	48
6.1 Problematiken.....	48
6.1.1 Kommunikation mit der API.....	48
6.1.2 SSO und WebUntis.....	49
6.2 Was verbirgt die Zukunft?.....	49
7 Literaturverzeichnis.....	50
8 Abbildungsverzeichnis.....	51
9 Erklärung.....	52
10 Anhang.....	53

# 1 Einleitung

Im Schulalltag ist es unverzichtbar, sich über die aktuelle Situation zu informieren und auf dem neuesten Stand zu sein, um das Schuljahr erfolgreich abzuschließen. Dazu gehört auch die aktuelle Notensituation. Viele Schüler, aber auch Elternteile interessieren sich brennend dafür, welche Note man bei der letzten Schularbeit erhalten hat. Leider ist die Oberfläche von WebUntis unübersichtlich und unpraktisch für einen schnellen Überblick. So kam die Idee für eine eigenentwickelte Applikation, die die Noten übersichtlich darstellt.

WebUntis ist eine umfangreiche Lösung für Schulen aller Art. Der Aufgabenbereich erstreckt sich von der Erstellung und Verwaltung der Stundenpläne und Raumbuchungen bis hin zur Abwesenheitsprotokollierung und Notenverwaltung. Viele Schulen haben sich entschlossen, ihr Vertrauen in diese Softwarelösung zu legen. Da das Unternehmen Untis jedoch in ganz Europa tätig ist, sind Anpassungen für einzelne Schulen schwer umzusetzen. Aus diesem Grund habe ich in Zusammenarbeit mit dem Zuständigen unserer Schule für WebUntis, Herrn Helmut Faller, ein Projekt ins Leben gerufen, das es ermöglicht, die Noten von WebUntis abzurufen und in einer gewünschten Form anzuzeigen.

Ohne dieses Projekt wäre es umständlich, die Noten der einzelnen Fächer herauszusuchen, um sich einen Überblick zu verschaffen. Zusätzlich ermöglicht die WebUntis-App für das Smartphone keine Überprüfung der Noten, man muss daher die Website aufrufen und dann aufwendig nach den Noten suchen, die in einer für das Smartphone nicht optimierten Darstellung angezeigt werden.

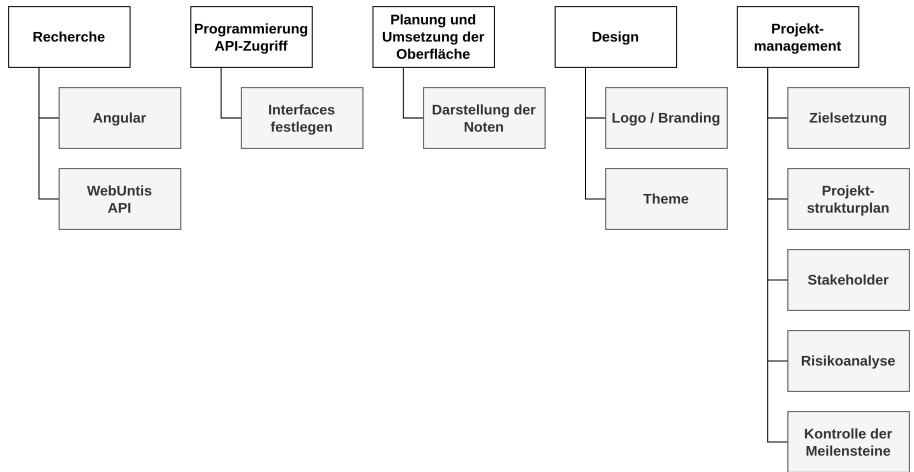
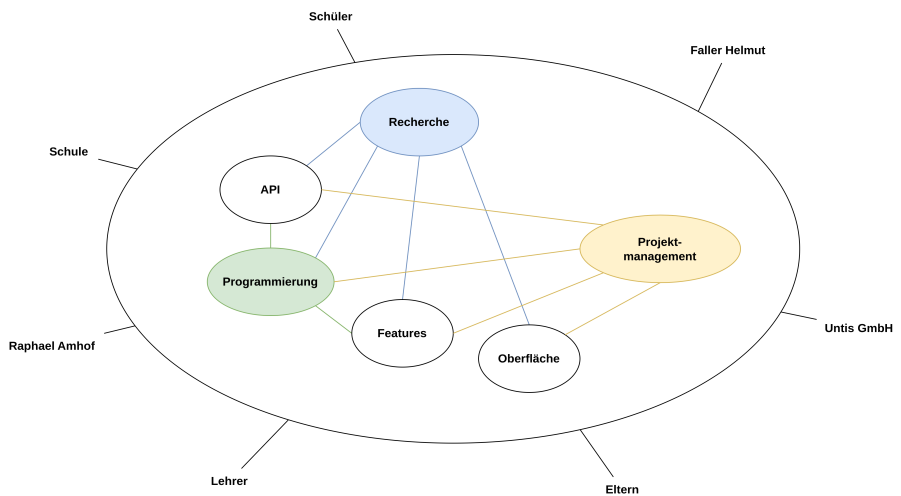
Bevor ich jedoch auf den technischen Teil des Projektes eingehe, ist eine sorgfältige Planung unerlässlich. Um eine effiziente Umsetzung und erfolgreiche Realisierung zu gewährleisten, habe ich einen detaillierten Projektplan entwickelt. Im folgenden Abschnitt werde ich auf die Planung des Projektes eingehen, bevor ich später im Dokument den technischen Teil fortführe.

## 2 Planung

Alles fängt mit einer Idee im Kopf an, aber um ein Ziel zu erreichen, muss man es auf Papier bringen. Für ein Projekt ist dies einer der wichtigsten Schritte und sollte daher nicht vernachlässigt werden. Zur Planung gehören mehrere grundlegende Schritte. Als erstes muss man die Idee definieren und eine erste Projektskizze anfertigen. Diese gibt einen kurzen Überblick über die Ziele und Aufgaben, die bewältigt werden müssen.

Die Projektskizze umfasst eine detaillierte Beschreibung der Ziele, Aufgaben und des geplanten Ablaufs. Darüber hinaus werden auch Rahmenbedingungen identifiziert und mögliche Umfeldler skizziert, wie zum Beispiel das sachliche, zeitliche und soziale Umfeld. Die Projektskizze dient als Leitfaden für das gesamte Projekt und ermöglicht es, einen klaren Überblick über den geplanten Projektablauf zu behalten. Sie wird als Grundlage für die weitere Planung und Umsetzung des Projekts dienen.

Die Projektskizze für mein Projekt lautet wie folgt:

<b>Projekttitlel</b>	Notendarstellung – Web-App für WebUntis
<b>Projektmitglieder</b>	Amhof Raphael
<b>Zeitraahmen</b>	01.10.2022 – 12.05.2023
<b>Ziel</b>	Bereitstellung einer übersichtlichen Notendarstellung
<b>Ausgangssituation</b>	WebUntis als Datenquelle für Noten und Zugriff mittels API
<b>Grobziele mit Teilaufgaben</b>	 <p><b>Recherche</b></p> <ul style="list-style-type: none"> <li>Angular</li> <li>WebUntis API</li> </ul> <p><b>Programmierung API-Zugriff</b></p> <ul style="list-style-type: none"> <li>Interfaces festlegen</li> </ul> <p><b>Planung und Umsetzung der Oberfläche</b></p> <ul style="list-style-type: none"> <li>Darstellung der Noten</li> </ul> <p><b>Design</b></p> <ul style="list-style-type: none"> <li>Logo / Branding</li> <li>Theme</li> </ul> <p><b>Projektmanagement</b></p> <ul style="list-style-type: none"> <li>Zielsetzung</li> <li>Projektstrukturplan</li> <li>Stakeholder</li> <li>Risikoanalyse</li> <li>Kontrolle der Meilensteine</li> </ul> <p><i>Bild 2.1: Grobziele</i></p>
<b>Rahmenbedingungen</b>	<p>Analyse des sachlichen Umfeldes:</p> <p>Auftraggeber: Hat starkes Interesse daran, dass das Projekt gelingt</p> <p>Schüler: Erhalten einen Vorteil durch die Applikation und sind somit für das Projekt</p> <p>Analyse des zeitlichen Umfeldes:</p> <p>Vorheriges Projekt einer ähnlichen Windows-Applikation ist zu berücksichtigen</p> <p>Folgeprojekt könnte die Instandhaltung und Erweiterung der Applikation sein</p> <p>Analyse des sozialen Umfeldes:</p>  <p><i>Bild 2.2: Soziales Umfeld</i></p>

## 2.1 Zielsetzung

Nachdem im Allgemeinen die Ziele definiert wurden und man sich einen kleinen Überblick über das Projekt verschafft hat, gilt es, die Ziele genau zu definieren. Ziele helfen dabei, motiviert und fokussiert das Projekt zu erledigen und ermöglichen es, Prozesse zu messen und zu überprüfen. Für eine präzise Zieldefinition eignet sich die SMART-Methode, die erstmals von Peter Drucker vorgeschlagen wurde.<sup>1</sup> Dieses Konzept besagt, dass ein Ziel folgende fünf Kriterien erfüllen sollte:

- **S**pezifisch
- **M**essbar
- **A**kzeptiert
- **R**ealistisch
- **T**erminiert

Um ein klar definiertes Ziel zu formulieren, ist es wichtig, es spezifisch und klar zu definieren, um Unklarheiten zu vermeiden. Zudem sollte das Ziel messbar sein, damit man weiß, wann es erreicht wurde und um ein klares Ziel vor Augen zu haben. Ebenso wichtig ist, dass das Ziel von den Teammitgliedern und Stakeholdern akzeptiert wird, da sonst die Akzeptanz und Unterstützung für das Projekt möglicherweise fehlen. Ein Ziel sollte immer realistisch sein und nicht unerreichbar wirken. Hierbei ist es wichtig zu beachten, dass das Sprichwort "Nichts ist unmöglich" zwar stimmt, jedoch die Motivation der Mitarbeiter leiden kann, wenn ein Ziel als unmöglich erscheint. Es ist daher oft sinnvoll, ein großes Ziel in kleinere Teilziele aufzuteilen, die einfacher erreichbar erscheinen. Oft wirken die Teilziele einfacher, motivierender und erreichbarer, als ein klobiges Ziel. Zu guter Letzt sollte ein Ziel zeitlich terminiert sein, um zu verhindern, dass es auf die lange Bank geschoben wird und um sicherzustellen, dass es bis zu einem bestimmten Zeitpunkt abgeschlossen wird. Diese Herangehensweise hilft enorm bei der Planung von Projekten, da sie hilft, einst schwammige Gedanken in ein glasklares Ziel zu formen. Sie zwingt uns auch dazu, darüber nachzudenken, ob das Ziel realistisch und akzeptiert ist.

Des Weiteren können Ziele in Haupt-, Teil-, Neben- und Nicht-Ziele eingeteilt

---

1 [https://de.wikipedia.org/wiki/SMART\\_\(Projektmanagement\)#cite\\_note-1](https://de.wikipedia.org/wiki/SMART_(Projektmanagement)#cite_note-1) (Stand: 09.03.23)

werden. Diese Einteilung hilft dabei, wichtige und unwichtige Prozesse zu differenzieren und festzulegen, was nicht erreicht werden soll.

Nach einer kleinen Einführung in die richtige Zielsetzung möchte ich nun meine Zieldefinition aufzeigen:

### **2.1.1 Hauptziel**

Entwicklung einer Web-Anwendung zur klaren Darstellung von Noten eines Schuljahres in Verbindung mit der bestehenden App WebUntis. Die Noten sollen übersichtlich und klar dargestellt werden, um einen Anreiz zur Nutzung zu schaffen. Es soll eine Verlinkung von der Web-App zur App WebUntis mit Single Sign On (SSO) ermöglicht werden, um eine erneute Anmeldung zu vermeiden. Die Applikation soll online von einem Anbieter oder idealerweise von der Schule bereitgestellt werden. Die Fertigstellung ist bis zum 26.04.2023 geplant.

### **2.1.2 Teilziele**

- Programmierung des API-Zugriffes mit einem Proxy dazwischen
- Notenanzeige
  - Klare Skizze der Anzeige
  - Abrufen der Daten
  - Umsetzung in HTML und SCSS
- Single-Sign-On (SSO)
- Hosting

### **2.1.3 Nicht-Ziele**

- Ersetzen von WebUntis
- Vollständiges Interface für alle Funktionen bereits von WebUntis bereitgestellt
- Autonomes Schulmanagement-Programm



## 2.2 Stakeholder

An einem Projekt nehmen normalerweise mehrere Personen und Gruppen Einfluss. Jeder Einzelne hat eigene Ziele und Interessen. Damit das Projekt ein Erfolg wird, muss man die einzelnen Gruppen identifizieren und sich in ihre Position versetzen. Andernfalls könnte es sein, dass das Projekt nicht den Vorstellungen wichtiger Akteure entspricht und somit die Unterstützung entfällt. Dies kann im schlimmsten Fall zum Scheitern des Projektes führen. Um das zu verhindern, fängt man an, die sogenannten Stakeholder zu identifizieren und zu kategorisieren. Dabei werden der Einfluss auf das Projekt, ihre Einstellungen, Interessen sowie die Risiken und das Konfliktpotenzial in einer Tabelle aufgelistet. In Fachkreisen nennt man diesen Vorgang Stakeholderanalyse. Weiterhin kann man diese Stakeholder in verschiedene Gruppen einteilen, um die Art ihres Einflusses auf das Projekt zu unterteilen. Die Stakeholder meines Projektes habe ich in vier Gruppen unterteilt. Diese unterscheiden sich in der Art des Einflusses auf das Projekt, der Konfrontation mit dem Projekt und den Interessensgruppen.

Die erste und wichtigste Gruppe sind die **Benutzer** meiner Applikation. Sie werden am häufigsten mit dem Projekt interagieren und sind somit ausschlaggebend für den Erfolg des Projektes. Sie haben Einfluss auf die Funktionen des Projektes und geben direktes Feedback zum Projekt.

Als nächstes kommt die Gruppe **Bildungsdirektion/Administration**. Sie sind dafür zuständig, das Projekt an die Schüler weiterzugeben und es bereitzustellen. Obwohl sie nicht direkt mit dem Projekt interagieren, sind sie dennoch wichtig, insbesondere für die Aufrechterhaltung nach Abschluss des Projekts.

Eine Gruppe mit mäßigem Einfluss sind die **externen Unternehmen**. Der wichtigste Stakeholder in dieser Gruppe ist Untis, das Unternehmen hinter WebUntis. Die Besonderheit dieser Gruppe ist, dass sie wenig Interesse am Projekt haben, aber dennoch einen wichtigen Einfluss ausüben können.

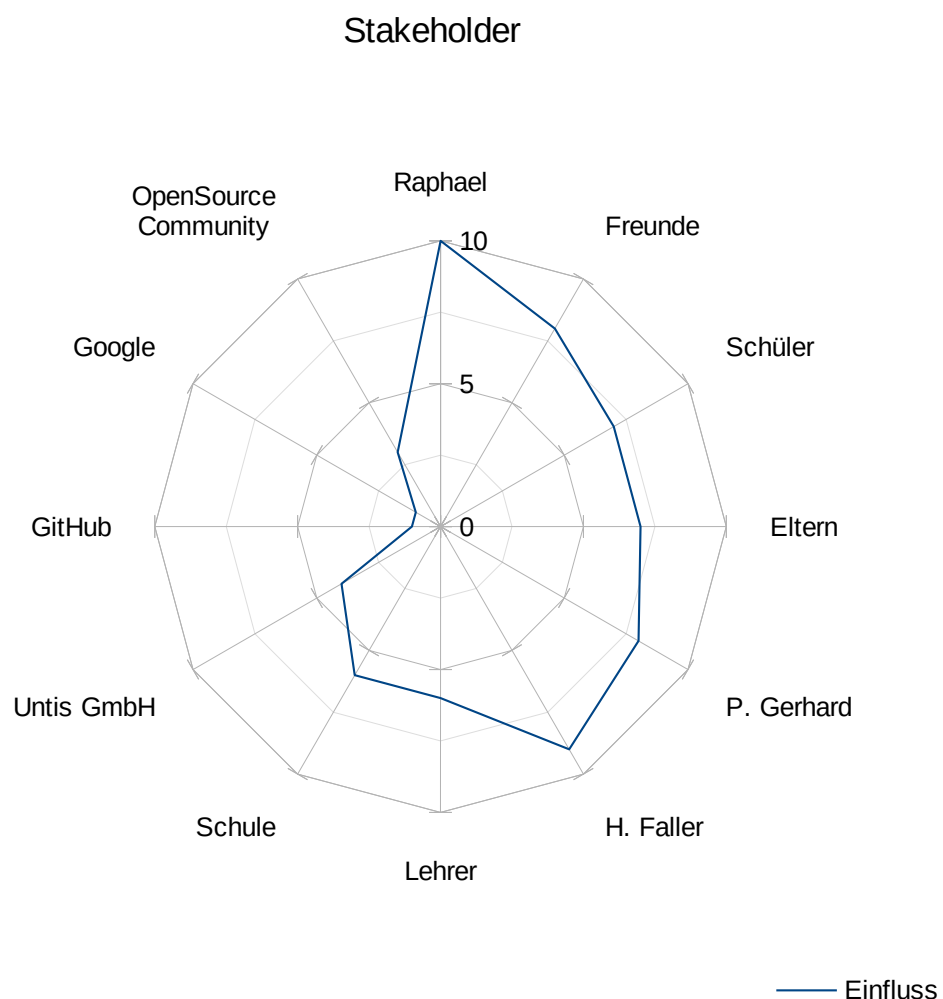
Eine vernachlässigbare Gruppe sind die **externen Stakeholder**, zu denen die Open Source Community gehört.

Name	Einfluss	Auftrag, Ziele	Chancen, Interessen	Risiken, Konfliktpotenzial	Maßnahmen
Raphael	10	Erstellung Projekt	Erfolgreicher Abschluss des Projektes		
Freunde	8	Tipps und Tricks	Nutzen der Applikation		
Schüler	7	Schuljahr mit positive Noten abschließen	Überprüfung Noten	Noten mit der App zu überprüfen ist umständlicher, unübersichtlicher oder bietet keinen Vorteil	Oberfläche gut planen und designen; Funktionen nach Wunsch umsetzen
Eltern	7	Sohn/Tochter überprüfen und motivieren	Leistungsüberprüfung Schüler		
P. Gerhard	8	Kontrolle der Projektarbeit, Ideen	Interessiert an der Leistung der Schüler		
H. Faller	9	Einbindung der Applikation in WebUntis; Ideen für Features; Bereitstellung aller nötigen WebUntis Daten	Einwandfreie Bereitstellung der App	Keine Einbindung möglich in das aktuelle System; Sterben des Projektes	Kommunikation der Wünsche und Vorschläge; Kommunikation mit Untis
Lehrer	6	Eintragen der Noten in WebUntis	Mitteilen der Noten an die Schüler		
Schule	6	Kommunikation mit den Schülern über WebUntis (Noten, Abwesenheiten, ...)	Interesse daran, dass die Schüler ihren aktuellen Status bewerten und darauf aufbauend für den zukünftigen Schulalltag planen können		
Untis GmbH	4	Verkauf der eigenen Software		WebUntis könnte etwas dagegen haben, wenn ich einfach so auf ihre API zugreife. Ein alternatives Programm könnte gegen ihre Nutzungsbedingungen verstoßen	Bei einer Konfliktsituation einen Kompromiss eingehen
GitHub	1	Bereitstellung des Quellcodes			
Google	1	Weiterentwicklung des Angular Frameworks			
OpenSource Community	3	Nutzung und Weiterentwicklung; Codefeedback	Verbesserung des Codes; Nutzen in eigener Verantwortung		

	Benutzer
	Bildungsinstitution/Administration
	Externe Unternehmen
	Externe Stakeholder
1-10	Einfluss

Vorhin in der Tabelle sind alle Stakeholder meines Projekts aufgelistet. Daraus lässt sich ableiten, welche Stakeholder einen großen Einfluss haben und welche Interessen die dazugehörige Gruppe aufweist. Falls die Gruppe ein Risiko oder Konfliktpotenzial darstellt, werden auch mögliche Maßnahmen genannt, um das Risiko zu minimieren.

In der unten stehenden Grafik sieht man nochmals den Einfluss der einzelnen Gruppen. Es lässt sich herauslesen, dass Raphael für die Umsetzung des Projektes den größten Einfluss auf die Prozesse des Projektes hat. Zusätzlich hat Helmut Faller einen Einfluss von neun. Dies bedeutet in einfachen Worten, dass die Interessen dieser Stakeholder wichtiger sind als andere. Daraus können die Aufgaben des Projektes auf diese Interessen ausgerichtet werden, um die Chance für ein erfolgreiches Projekt zu erhöhen.



## **2.3 Projektstrukturplan**

Sobald man sich einen guten Überblick über die Situation verschafft hat, kann man beginnen, über die einzelnen Arbeitsschritte des Projektes nachzudenken. Dabei hilft es, die Schritte in sogenannte Arbeitspakete einzuteilen. Diese Pakete spiegeln einen bestimmten Arbeitsschritt wider. Diese Arbeitspakete können dann weiter in Sammelaufgaben unterteilt werden. Wenn man diese Aufgaben hierarchisch darstellt, erhält man einen Projektstrukturplan. Dieser hilft dabei, die wichtigsten Schritte eines Projektes zu erfassen und dient zudem als Grundlage für weitere Planungsschritte.

Auf der nächsten Seite ist der Projektstrukturplan meines Projektes dargestellt. Dort kann man auf einen Blick sehen, welche Schritte erledigt werden müssen. Zudem habe ich die Schritte chronologisch von links nach rechts sortiert, das heißt, je weiter man nach rechts geht, desto später müssen sie erledigt werden.

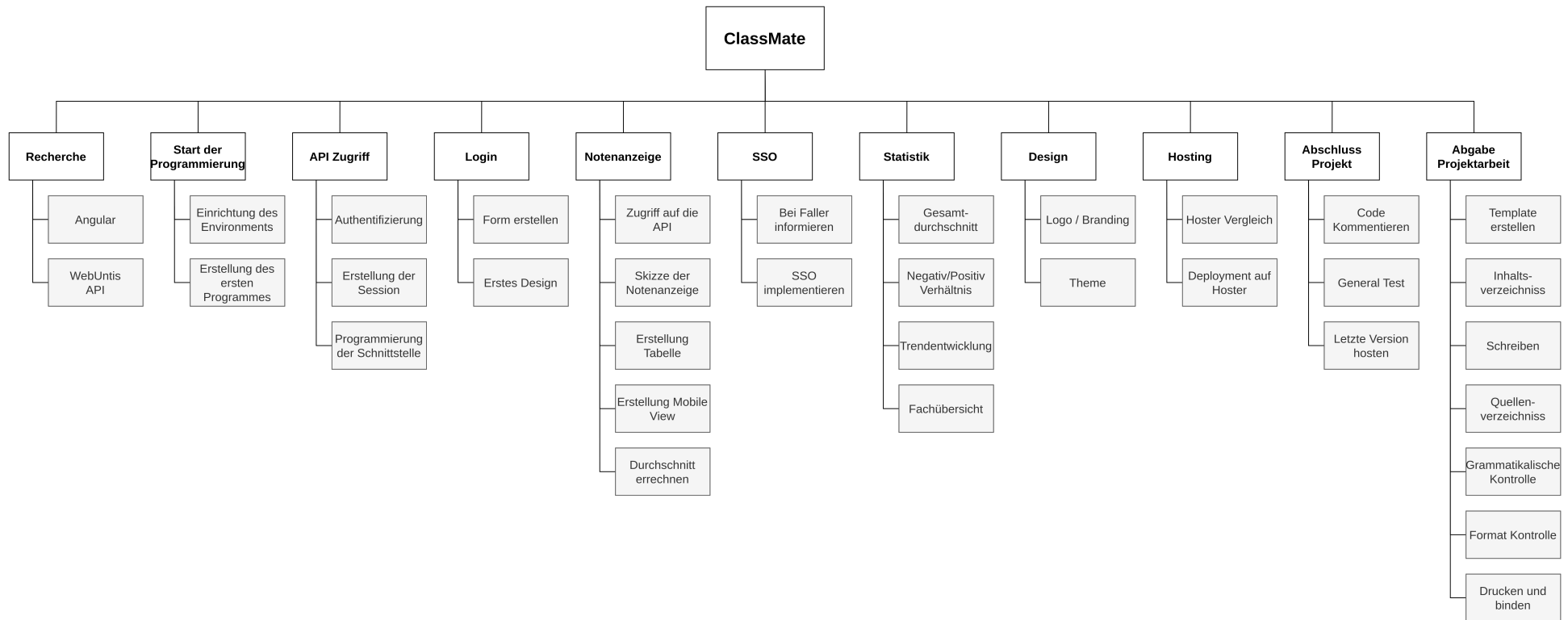


Bild 2.3: Projektstrukturplan

## 2.4 Risikoanalyse

Im Leben gibt es tatsächlich eine Vielzahl von Risiken und Unklarheiten, und das ist auch gut so. Denn sonst wäre es ja langweilig. Diese etwas radikale Sichtweise begleitet mich schon seit längerem. Sie hat mir geholfen, meine Komfortzone zu verlassen und Neues auszuprobieren. Natürlich muss man nicht alles einfach so hinnehmen, wie es kommt. Man kann durchaus im Voraus Risiken einplanen, um vorbereitet zu sein, wenn es darauf ankommt. Mit einem solchen Ansatz gerät man nicht in Panik, wenn es darauf ankommt, sondern ist besser darauf vorbereitet.

Auch in der Kunst des Projektmanagements gibt es diesen Ansatz. Um Risiken zu identifizieren und Lösungen auszumalen, erstellt man eine Risikoanalyse. Diese beinhaltet eine Liste aller Risiken. Um nicht von der schierenden Menge an Risiken überwältigt zu werden, werden die Risiken nach ihrer Eintrittswahrscheinlichkeit und Auswirkung klassifiziert. In der Praxis multipliziert man diese beiden Kennzahlen, um den R-Wert zu erhalten. Anhand dieses Wertes können die Risiken sortiert werden, um die schwerwiegendsten Risiken zu identifizieren.

Auf den folgenden Seiten habe ich die Risikoanalyse beigefügt.

**Auflistung der Risiken**

<b>AP-Nr.</b>	<b>Thema</b>	<b>Risiko</b>
<b>1</b>	<b>API</b>	
1.1		Änderungen der API Spezifikationen
1.2		Keine Erreichbarkeit
1.3		Zu viele Zugriffe eines einzelnen Clients sperren
1.4		Authentifizierung ändert sich
<b>2</b>	<b>SSO</b>	
2.1		Keine Möglichkeit für SSO
2.2		Session Key nicht übertragbar
<b>3</b>	<b>Facharbeit schreiben</b>	
3.1		Keine Themen über was man schreiben kann
3.2		Zeit wird knapp
3.3		Drucker funktioniert nicht
3.4		Verlust der Datei
<b>4</b>	<b>Hosting</b>	
4.1		Keinen Hosting Anbieter finden
4.2		Die Schule hosted es nicht
4.3		Zu viele Zugriffe Überlastung
4.4		Zu hohe Kosten
4.5		Keine Möglichkeit neue Technologien wie Angular zu hosten
<b>5</b>	<b>Notenübersicht</b>	
5.1		Interface ist nicht verwendbar
5.2		Interface ist unübersichtlich
5.3		Daten können nicht abgerufen werden
5.4		Verschiedene Geräte zeigen es falsch an
<b>6</b>	<b>Generell</b>	
6.1		Verlust der Daten
6.2		Korruption der Daten
6.3		Laptop geht kaputt
6.4		Sicherheit der Applikation niedrig
6.5		Übertragung unsicher
6.6		Session Hijacking
<b>7</b>	<b>Frameworks</b>	
7.1		Frameworks sind veraltet
7.2		Sicherheitslücken

## Risikobewertung

AP-Nr.	Risiko	Eintrittswahrscheinlichkeit	Schwere der Auswirkung	R-Wert
1.1	Änderungen der API Spezifikationen	3	8	24
1.2	Keine Erreichbarkeit	6	6	36
1.3	Zu viele Zugriffe eines einzelnen Clients sperren	5	3	15
1.4	Authentifizierung ändert sich	4	4	16
2.1	Keine Möglichkeit für SSO	5	2	10
2.2	Session Key nicht übertragbar	7	1	7
3.1	Keine Themen über was man schreiben kann	3	7	21
3.2	Zeit wird knapp	7	8	56
3.3	Drucker funktioniert nicht	7	3	21
3.4	Verlust der Datei	5	10	50
4.1	Keinen Hosting Anbieter finden	3	8	24
4.2	Die Schule hosted es nicht	8	3	24
4.3	Zu viele Zugriffe Überlastung	6	4	24
4.4	Zu hohe Kosten	9	7	63
4.5	Keine Möglichkeit neue Technologien wie Angular zu hosten	2	5	10
5.1	Interface ist nicht verwendbar	1	10	10
5.2	Interface ist unübersichtlich	8	7	56
5.3	Daten können nicht abgerufen werden	2	9	18
5.4	Verschiedene Geräte zeigen es falsch an	8	7	56
6.1	Verlust der Daten	4	10	40
6.2	Korruption der Daten	3	9	27
6.3	Laptop geht kaputt	1	8	8
6.4	Sicherheit der Applikation niedrig	8	5	40
6.5	Übertragung unsicher	5	5	25
6.6	Session Hijacking	8	5	40
7.1	Frameworks sind veraltet	3	3	9
7.2	Sicherheitslücken	4	4	16



### Auflistung nach Dringlichkeit

AP-Nr.	Risiko	R-Wert	Kumuliert	In %
4.4	Zu hohe Kosten	63	63	8,4 %
3.2	Zeit wird knapp	56	119	16,0 %
5.2	Interface ist unübersichtlich	56	175	23,5 %
5.4	Verschiedene Geräte zeigen es falsch an	56	231	31,0 %
3.4	Verlust der Datei	50	281	37,7 %
6.1	Verlust der Daten	40	321	43,0 %
6.4	Sicherheit der Applikation niedrig	40	361	48,4 %
6.6	Session Hijacking	40	401	53,8 %
1.2	Keine Erreichbarkeit	36	437	58,6 %
6.2	Korruption der Daten	27	464	62,2 %
6.5	Übertragung unsicher	25	489	65,5 %
1.1	Änderungen der API Spezifikationen	24	513	68,8 %
4.1	Keinen Hosting Anbieter finden	24	537	72,0 %
4.2	Die Schule hosted es nicht	24	561	75,2 %
4.3	Zu viele Zugriffe Überlastung	24	585	78,4 %
3.1	Keine Themen über was man schreiben kann	21	606	81,2 %
3.3	Drucker funktioniert nicht	21	627	84,0 %
5.3	Daten können nicht abgerufen werden	18	645	86,5 %
1.4	Authentifizierung ändert sich	16	661	88,6 %
7.2	Sicherheitslücken	16	677	90,8 %
1.3	Zu viele Zugriffe eines einzelnen Clients sperren	15	692	92,8 %
2.1	Keine Möglichkeit für SSO	10	702	94,1 %
4.5	Keine Möglichkeit neue Technologien wie Angular zu hosten	10	712	95,4 %
5.1	Interface ist nicht verwendbar	10	722	96,8 %
7.1	Frameworks sind veraltet	9	731	98,0 %
6.3	Laptop geht kaputt	8	739	99,1 %
2.2	Session Key nicht übertragbar	7	746	100,0 %

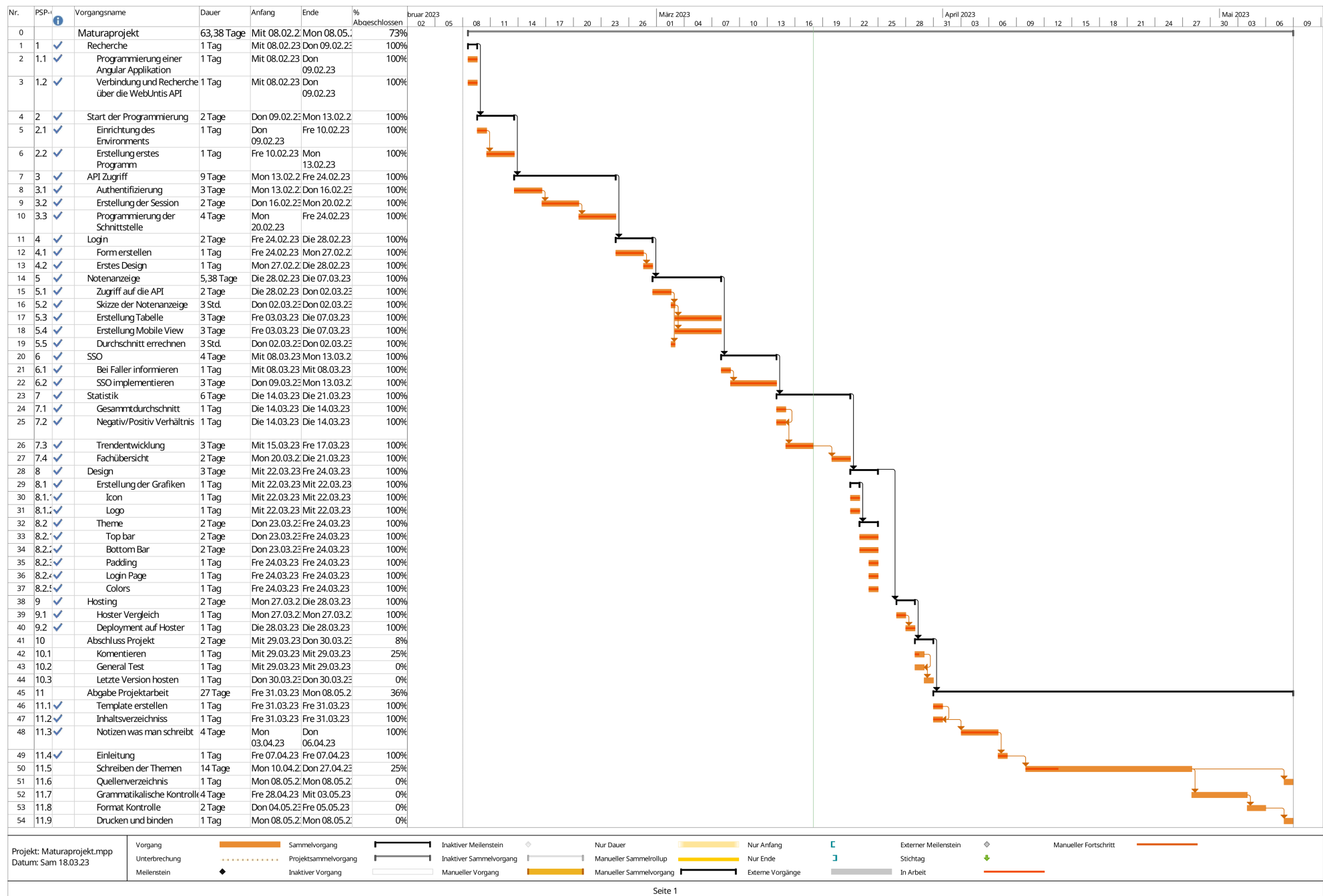
## Maßnahmen

AP-Nr.	Risiko	Maßnahme
4.4	Zu hohe Kosten	Frage Schule für Hosting
3.2	Zeit wird knapp	Genug einplanen plus Dokument vorbereiten
5.2	Interface ist unübersichtlich	Genaues Skizzieren des Interface und Befragung
5.4	Verscheidende Geräte zeigen es falsch an	Alle möglichen Größen einkalkulieren
3.4	Verlust der Datei	Backup auf mehreren Clouds nach Änderung
6.1	Verlust der Daten	Github Code speichern
6.4	Sicherheit der Applikation niedrig	Recherche für eine sichere Programmierung und Verwendung neuester Protokolle
6.6	Session Hijacking	Sichere Speicherung von Keys
1.2	Keine Erreichbarkeit	Keine Lösung, Fehler abfangen
6.2	Korruption der Daten	Backup
6.5	Übertragung unsicher	HTTPS verwenden
1.1	Änderungen der API Spezifikationen	Keine mögliche Maßnahme
4.1	Keinen Hosting Anbieter finden	Schule fragen
4.2	Die Schule hosted es nicht	Anbieter Heroku kostenloses Hosting
4.3	Zu viele Zugriffe Überlastung	Besserer Anbieter wie Schule
3.1	Keine Themen über was man schreiben kann	Während Prozess mitschreiben was man gemacht hat
3.3	Drucker funktioniert nicht	In der Schule drucken
5.3	Daten können nicht abgerufen werden	Programmierung eines resistenten API Zugriff
1.4	Authentifizierung ändert sich	Keine Lösung
7.2	Sicherheitslücken	Code prüfen
1.3	Zu viele Zugriffe eines einzelnen Clients sperren	Nachfragen für Freigabe bei WebUntis
2.1	Keine Möglichkeit für SSO	Möglichkeiten suchen
4.5	Keine Möglichkeit neue Technologien wie Angular zu hosten	Anderen Hoster
5.1	Interface ist nicht verwendbar	Planung des Interfaces
7.1	Frameworks sind veraltet	Updaten
6.3	Laptop geht kaputt	Backup
2.2	Session Key nicht übertragbar	Andere Möglichkeit suchen

Ich gebe zu, die vorherigen Tabellen sind ein bisschen einschüchternd. Nehmen wir aber die Tabellen Stück für Stück auseinander. In der ersten Tabelle sind die Risiken aufgelistet und in Arbeitspakete eingeteilt. Soweit noch verständlich. In der nächsten Tabelle habe ich die Risiken nach ihrer Schwere und Eintrittswahrscheinlichkeit eingeteilt. Das heißt, sie werden von 1 (unwahrscheinlich/leicht) bis 10 (sehr wahrscheinlich/schwer) bewertet. Durch diese beiden Zahlen ergibt sich der R-Wert. Nach diesem Wert kann man dann sortieren. Das habe ich dann auch in der nächsten Tabelle gemacht. Zusätzlich habe ich die kumulierten Werte berechnet, um klarzustellen, dass man sich auf die oberen Probleme konzentrieren soll. Dabei kann man sehen, dass sich die letzten Punkte nur wenige Prozent auf das Projekt auswirken. Zum Schluss habe ich mögliche Maßnahmen oder Vorbeugungen aufgelistet, die bei einem Eintritt des Risikos getroffen werden können.

## **2.5 Projektzeitplan**

Sobald man die Phasen definiert, die Schritte identifiziert und die Stakeholder gegliedert hat, und die Risiken eingeschätzt wurden, ist man so weit, das Projekt zu planen und in ein Zeitfenster einzutragen. Dabei eignet sich ein Programm, das die Vorgänge automatisch plant und entsprechende Grafiken generiert. Im Unterricht haben wir MS-Project kennengelernt, das sich perfekt für eine präzise Projektplanung eignet. Neben dem reinen Projektzeitplan bietet MS-Project die Möglichkeit, Ressourcen, Kosten und Meilensteine einzubinden.



## 3 Programmierung

Das World Wide Web ist eine der wichtigsten Erfindungen unserer Zeit. Zum ersten Mal konnten wir auf unzählige Informationen zugreifen, ohne eine Bibliothek besuchen zu müssen. Das WWW ist auch die einzige Plattform, die es ermöglicht, für alle Betriebssysteme zu programmieren, ohne sich um Kompatibilitätsprobleme kümmern zu müssen, sei es für iPhone, Android, Windows oder Linux - das Web wird von allen unterstützt. Daher liegt es nahe, Programme direkt im Web freizugeben. Dies ist kein neues Konzept und es gibt bereits eine Vielzahl an Frameworks, um dies ohne großen Aufwand zu erreichen. Eines der nennenswerten Frameworks ist sicherlich Angular, das von Google entwickelt wurde und alle Funktionen bietet, um Web-Apps schnell und einfach zu programmieren, zu gestalten und zu verwalten.<sup>2</sup> Mehr dazu später.

### 3.1 TypeScript

Die Programmiersprache TypeScript ist das grundlegende Werkzeug, um eine Web-App in Angular zu programmieren. Es ist eine Voraussetzung, bevor man überhaupt mit Angular starten kann. TypeScript ist eine von Microsoft entwickelte Programmiersprache für das Web.<sup>3</sup> Diejenigen die schon mal eine Website erstellt haben, werden sich jetzt denken: "Aber im Web nutzt man doch nur JavaScript." Das ist korrekt, aber lassen Sie mich es erklären. TypeScript läuft auf JavaScript. Einfach gesagt wird der TypeScript-Programmcodem in JavaScript umgewandelt, um ihn dann auszuführen. Man könnte natürlich auch direkt in JavaScript schreiben, mag sich der eine oder andere denken. Natürlich bietet TypeScript einige Vorteile gegenüber JavaScript. Zum einen bringt TypeScript viele Konzepte aus der objektorientierten Programmierung mit und implementiert sie in JavaScript. Zum Beispiel gibt es Klassen und Interfaces, die für eine einfache und strukturierte Programmierung verwendet werden können.<sup>4</sup> Zusätzlich werden Variablen und Funktionen eindeutige Typen zugewiesen. Das bedeutet, dass nur ein bestimmter Datentyp verwendet werden kann. Dies wird

---

2 <https://angular.io/guide/what-is-angular> (Stand: 19.03.23)

3 <https://github.com/microsoft/TypeScript> (Stand: 20.03.23)

4 <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes-func.html> (Stand: 20.03.23)

bei der Übersetzung zu JavaScript überprüft. JavaScript selbst unterstützt diese Funktionalität nicht, da es unerheblich ist, welcher Datentyp in einer Variable gespeichert wird.<sup>5</sup> Dies führt oft zu Fehlern und unerwünschten Effekten im Code. Deshalb hat sich Google wahrscheinlich für TypeScript entschieden.

## 3.2 Angular

Wenn bereits eine Programmiersprache existiert, warum kann man dann nicht einfach damit die Web-App entwickeln?

Web-Applikationen folgen immer ein bestimmten Grundschemata. Beispielsweise besteht fast immer die Notwendigkeit eines Login-Bildschirms, Datenverarbeitung und -verifikation, Datenanzeige und das Laden von Daten im Hintergrund. Diese Funktionen sind universell und werden häufig benötigt. Hier kommt Angular ins Spiel. Angular implementiert diese Funktionen im Framework und ermöglicht es, sie problemlos und einfach zu nutzen. Dadurch muss man nicht bei jeder neuen Applikation das Rad neu erfinden, sondern kann auf optimierte und ausführlich getestete Funktionen zurückgreifen.<sup>6</sup>

## 3.3 Angular Vertiefung

Angular basiert auf Komponenten, die es ermöglichen, eine Single-Page-Applikation zu skalieren und zu erweitern. Komponenten sind unabhängige Teile einer Webseite, die mehrfach eingebunden werden können, wie zum Beispiel ein Anmeldeformular.<sup>7</sup>

---

5 <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html> (Stand: 20.03.23)

6 <https://angular.io/docs> (Stand: 21.03.23)

7 <https://angular.io/guide/what-is-angular#what-is-angular> (Stand: 21.03.23)

### 3.3.1 Components

Eine Angular-App besteht aus mehreren Komponenten (Components), die jeweils für die Steuerung der einzelnen Inhalte in der App zuständig sind. Eine Komponente besteht aus drei Teilen. Erstens eine TypeScript-Klasse, in der hauptsächlich die Logik des Programms definiert wird. Zweitens ein Template bzw. eine HTML-Struktur, mit der die Klasse interagieren kann. Drittens wird noch eine CSS-Datei benötigt, um das kosmetische Aussehen der Komponente zu gestalten.<sup>8</sup>

Um eine Komponente zu definieren, müssen wir in einer TypeScript-Datei die Komponentenkasse definieren:

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

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

*Code 1: Minimale Komponente*

Der erste Schritt besteht darin, die Komponente aus der Angular-Bibliothek mit `import` einzubinden, damit die Direktive `@Component` verwendet werden kann. Diese Direktive gibt dem Code an, dass die folgende Klasse eine Komponente ist und auch als solche behandelt wird. Zusätzlich wird ein Objekt übergeben mit zwei Elementen. Erstens `selector` für die eindeutige Identifizierung in Angular, um die Komponente beispielsweise in anderen Komponenten einzufügen. Das ermöglicht das beliebige Verschachteln von Komponenten. Das Nächste ist `template`, mit diesem Element kann die Struktur der Seite mithilfe von HTML erstellt werden. Alternativ zur direkten Übergabe des HTML-Codes kann auch eine Datei verlinkt werden.

---

<sup>8</sup> <https://angular.io/guide/what-is-angular#components> (Stand: 21.03.23)

### 3.3.2 Templates

Davor habe ich bereits kurz die Templates erwähnt, aber nicht genau erklärt, wie sie funktionieren. Templates sind HTML-Fragmente, die die Struktur der Ansicht angeben und einfach gesagt vorgeben, was angezeigt werden soll. Dieser Code kann reines HTML sein aber es kann auch mit Angular Erweiterungen ausgeschmückt werden. Diese helfen einen dabei die Struktur dynamisch zu generieren und basierend auf Daten anzuzeigen. Dadurch ist es möglich, eine generische Struktur vorzugeben und zur Laufzeit mit Daten zu füllen. Eine weitere Funktion ist die automatische Aktualisierung des DOMs, sobald ein Wert geändert wird. Für die Programmierung bedeutet dies volle Flexibilität und einfache Aktualisierung der Daten in der App, ohne immer die komplette Ansicht manuell aktualisieren zu müssen. Folgendes Beispiel wird Ihnen dabei helfen, das Konzept zu verstehen:

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

@Component ({
  selector: 'hello-world-interpolation',
  templateUrl: './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
  message = 'Hello, World!';
}
```

Code 2: Klasse für das Template

Das Template ist gespeichert in `hello-world-interpolation.component.html`:

```
<p>{{ message }}</p>
```

Code 3: Einfaches Template

Das Resultat ist folgendes:

```
<p>Hello, World!</p>
```

Code 4: Ergebnis Template

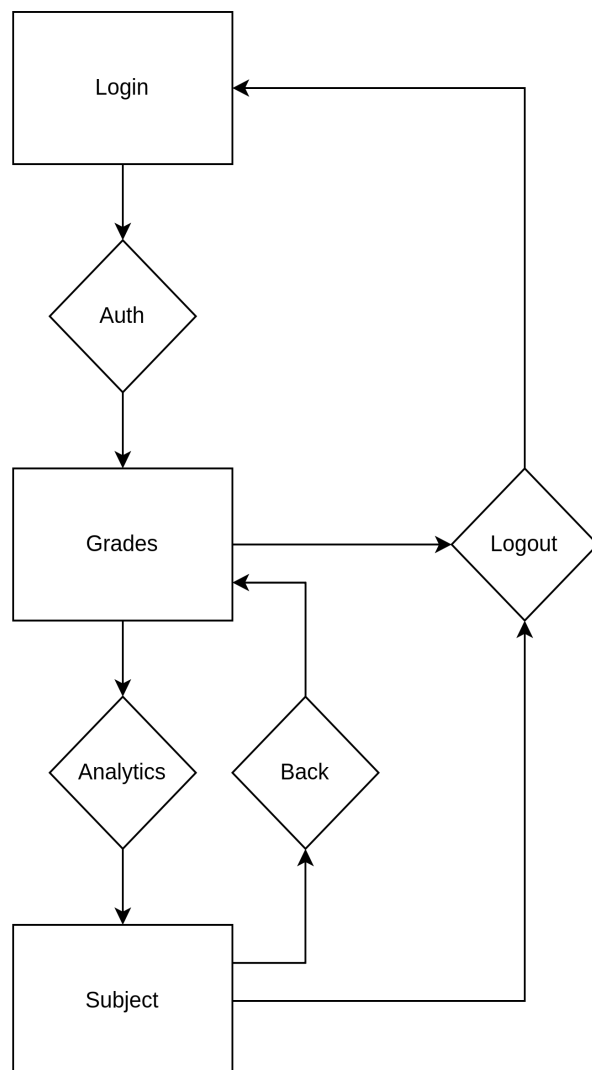
Wir stellen fest, dass `{{ message }}` ausgetauscht wird. Angular wird automatisch diese Teile mit den im Code definierten Daten austauschen.



### 3.4 ClassMate

Mit den grundlegenden Werkzeugen ausgerüstet, können wir mit der Programmierung einer Web-Anwendung beginnen. Natürlich war das hier nur eine sehr kurze und vereinfachte Einleitung in die Welt von Angular, und es gibt noch zahlreiche Themen und Konzepte zu erforschen. Eine gute Ressource bietet die offizielle Angular-Dokumentation, die nicht nur sehr detailliert in die Themen eintaucht, sondern auch Konzepte einfach erklärt.

Die App plant man in mehreren Schritten. Der erste Schritt ist, sich zu überlegen, was der Benutzer sehen soll und wie er dorthin gelangt. Ein solches Navigationsdiagramm ermöglicht es uns, in kurzer Zeit zu sehen, wie der Benutzer mit der App interagieren soll:



#### Beschreibung:

Der Benutzer soll bei seinem ersten Besuch auf eine Login-Seite gelangen.

Sobald er sich erfolgreich eingeloggt hat, wird er zur Hauptübersicht (Grades) weitergeleitet, wo der Durchschnitt der einzelnen Fächer angezeigt werden soll.

Falls er auf ein Fach klickt, wird er zur Detailseite weitergeleitet, auf der die genauen Daten des Fachs angezeigt werden sollen, wie z.B. Trenddaten und Durchschnitt. Hier kann er auch mittels eines Knopfes zur Übersicht zurückgelangen.

Der Benutzer kann sich in jeder Situation abmelden und gelangt somit wieder auf die Startseite.

Bild 3.1: Navigationsdiagramm

Jetzt stellt sich die Frage, woher die Daten kommen. Damit die Benutzer keine Daten eigenhändig eingeben müssen, lade ich die Daten direkt von WebUntis. Um den Datenfluss besser darzustellen, habe ich ein Data Flow Diagramm erstellt. Dieses verdeutlicht die Richtung des Datenflusses und vernachlässigt bewusst die Implementierung, um übersichtlich zu bleiben.

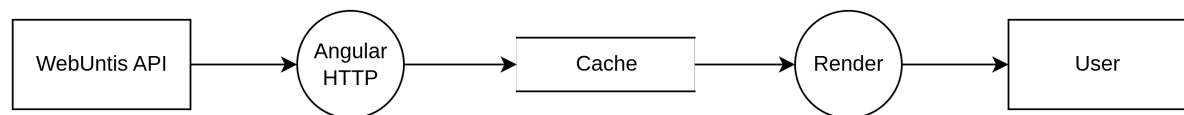


Bild 3.2: Data Flow Diagramm

Folglich kommen die Daten von der WebUntis API, diese werden mittels HTTP abgerufen. Danach werden die Daten in einen Cache gegeben, um die Anfragen zu minimieren. Schließlich werden die Daten dem Renderer übergeben und dem Benutzer präsentiert.

Hoffentlich haben diese Grafiken den logischen Aufbau der Applikation verdeutlicht. Sobald ich den logischen und abstrakten Teil der Applikation ausgedacht habe kann ich mit der Implementierung starten.

### 3.4.1 API

Der erste Schritt besteht darin, die Daten mittels einer API von WebUntis abzurufen. Ein Application Programming Interface (API), auf deutsch Anwendungsschnittstelle, ermöglicht es Programmierern auf bereitgestellte Funktionen eines Dienstes zuzugreifen. Glücklicherweise nutzt auch WebUntis diese Technologie. Somit war der Plan, in den Quellcode der offiziellen Website zu schauen und die benötigten Zugriffe zu identifizieren und später im eigenen Code zu verwenden. Um den Datenverkehr einer Website zu analysieren, kann man die sogenannten Entwicklertools im Browser öffnen. In Chrome genügt es, die gewünschte Seite zu öffnen und F12 zu drücken. Es öffnet sich ein neues Fenster, welches alle möglichen Daten anzeigt. Für die Applikation sind besonders die Kommunikation mit der API wichtig. Dazu wechseln wir zum "Network"-Reiter. Dort sehen wir dann alle Anfragen, die der Browser an verschiedene Endpunkte im Internet gemacht hat. Dieses Fenster wird in Bild 3.3 dargestellt. Dort sieht man auch die Details der einzelnen Anfragen, wie die

Header oder die Antwort des Servers. Mit dieser Methode habe ich schrittweise die einzelnen Schritte herausgefiltert, die für einen erfolgreichen Datenzugriff notwendig sind.

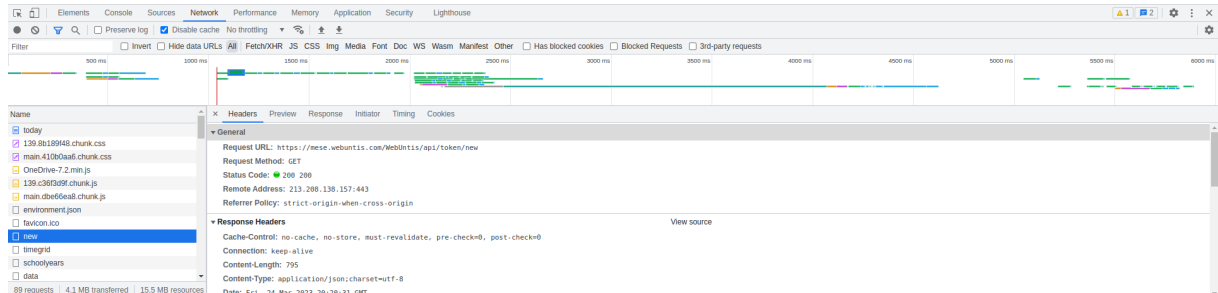


Bild 3.3: Netzwerkanalyse von WebUntis

Zusammenfassend braucht meine Applikation folgende Daten:

- Login Daten
- Schüler Daten
- Fächer
- Noten der Fächer

Mit der bereits erklärten Methode habe ich diese Netzwerkkommunikation herausgefiltert und analysiert. Ich bin zu folgendem Schluss gekommen:

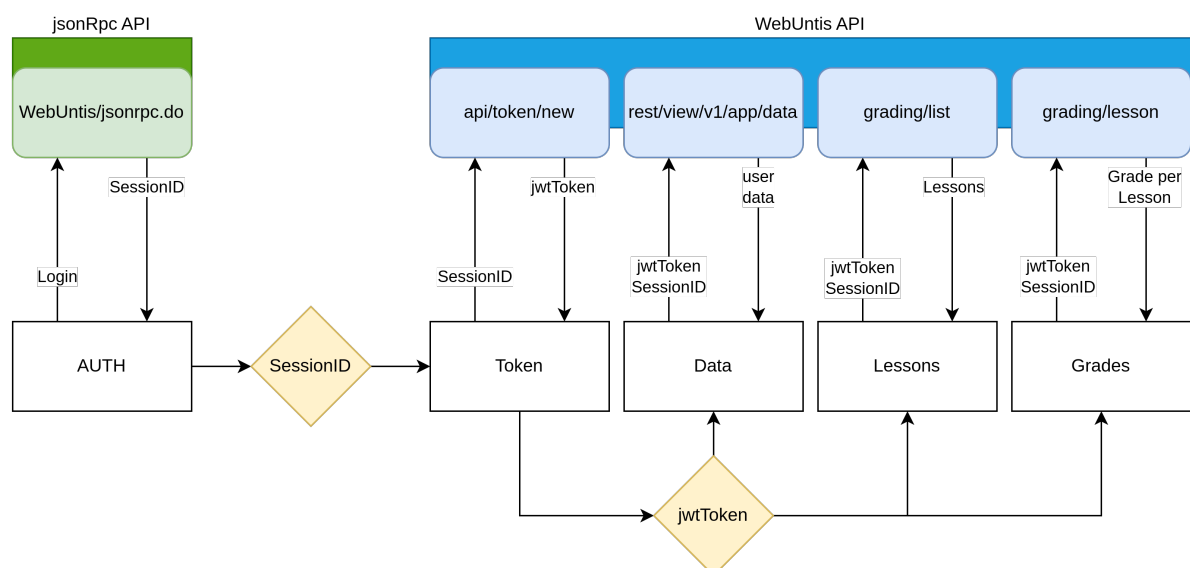


Bild 3.4: API Zugriffsschema

Wie man aus der Grafik erkennen kann, gibt es zwei Schnittstellen, um mit WebUntis zu kommunizieren: einmal die json-RPC API und die eigentliche API. Als erstes muss die json-RPC API mit Benutzername und Passwort aufgerufen werden, um einen sogenannten Session Key zu erhalten. Diesen Schlüssel verwendet man dann anschließend, um sich beim Server zu authentifizieren. Mehr zur Implementation später im Kapitel 3.4.1.b Authentifizierung. Als Nächstes kann man mit der anderen API kommunizieren, die uns mehr Daten als die erste API zur Verfügung stellt. Diese API wird genutzt, um verschiedene Daten abzurufen, wie Noten, Fächer, Abwesenheiten und vieles mehr. Bevor wir jedoch diese Daten erhalten können, benötigen wir einen Token. Diesen erhalten wir, wenn wir mit dem vorherigen Session Key eine Anfrage an die API stellen. Bei korrektem Schlüssel gibt uns die API einen Token, den wir bei den nachfolgenden Anfragen mitführen müssen.

### **3.4.1.a Services**

In Angular gibt es glücklicherweise bereits eine Möglichkeit, eine API-Verbindung einfach und schnell einzubinden. Dazu werden Angular-Services verwendet. Diese sind Module, die ausschließlich für die Datenbeschaffung verantwortlich sind, das heißt, sie implementieren nicht die Ansicht. Ihre reine Aufgabe besteht darin, Daten abzurufen, zu senden oder zu verarbeiten. Die Implementierung soll für den Rest des Programms unsichtbar sein. Das einzige, was für andere Klassen freigegeben wird, sind Funktionen, die ein definiertes Interface haben, damit klar ist, in welcher Form die Daten zurückkommen.<sup>9</sup> Die Services werden anschließend mittels Dependency Injection (DI) in die Klasse eingefügt, bei der man diese Daten benötigt. Einfach gesagt fügt Angular automatisch die benötigten Abhängigkeiten in seine Komponenten ein, ohne dass sich der Programmierer darum kümmern muss. Ich werde hier jedoch nicht weiter darauf eingehen, wie DI funktioniert, weil das eindeutig den Rahmen dieser Projektarbeit sprengen würde. Nichtsdestotrotz werde ich jetzt das Grundgerüst eines HTTP-Services erklären. Es sieht wie folgt aus:

---

9 <https://angular.io/guide/creating-injectable-service#creating-an-injectable-service> (Stand: 25.03.23)

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable()
export class ConfigService {
  constructor(private http: HttpClient) {}

  getApi<T>(path: string) {
    return this.http.get<T>(path)
  }
}
```

Code 5: Grundgerüst HTTP Service

Zuerst werden wie immer die benötigten Bibliotheken importiert. Danach wird eine Direktive `@Injectable()` angegeben. Wie schon besprochen, gibt diese Direktive dem Compiler an, dass die folgende Klasse eine bestimmte Funktion hat. In diesem Fall kann sie anschließend für DI verwendet werden. Der nächste wichtige Punkt ist der `constructor`, der folgendes Attribut hat: `private http: HttpClient`. Hier sehen wir die Dependency Injection in Aktion. Angular stellt uns den `HttpClient` zu Verfügung. Schließlich nutzt die Beispielfunktion diese `http`-Klasse, um eine GET-Anfrage an eine Adresse zu senden.<sup>10</sup>

### 3.4.1.b Authentifizierung

Jetzt wissen wir, wie man in Angular Anfragen an einen Webserver stellt. Der nächste Schritt ist, dieses Konzept auf die spezifischen Anfragen für WebUntis anzuwenden. Dazu möchte ich nochmals auf das vorherige Bild 3.4 hinweisen. Dort sieht man, dass man sich, bevor man eine Anfrage stellen kann, authentifizieren muss. Um dies zu erreichen, sind mehrere aufeinanderfolgende Anfragen erforderlich. Das Ziel ist es, `/app/data` aufzurufen, um die Daten des angemeldeten Benutzers zu erhalten. Die Implementierung erfolgt mit einem bisher nicht behandelten Konzept, nämlich der Funktionalen Programmierung. Kurz gesagt ist dies ein Konzept, bei dem nicht mehr gewartet wird, bis eine Aktion ausgeführt wird und das gesamte Programm stoppt, sondern es wird weitergegangen und im Hintergrund ausgeführt. Sobald die Daten eintreffen, können sie bearbeitet werden. Dies wird mit der `.pipe()` Funktion erreicht.

---

<sup>10</sup> <https://angular.io/guide/http#setup-for-server-communication> (Stand: 25.03.23)

Mehr muss man jetzt nicht wissen, da es schlichtweg zu viel wäre, genau auf dieses Konzept einzugehen. Wenn man jedoch mehr darüber erfahren möchte, gibt es sehr gute Online-Quellen zu diesem Thema, sowohl allgemein zur Funktionalen Programmierung als auch zur Umsetzung in JavaScript mit RxJS.<sup>11</sup>

```
login(school: string, username: string, password: string)
{
  const data$ = this.getData().pipe(
    tap((loginData) => {})
  );

  const token$ = this.http
    .get('/api/WebUntis/api/token/new', { responseType: 'text' })
    .pipe(
      switchMap(() => data$)
    );

  const session$ = this.postJsonRpcApi<LoginDtoResponse>(Method.AUTH, {
    user: username,
    password: password,
    client: this.apiDefinition.client,
  }).pipe(
    shareReplay()
  );

  return session$.pipe(switchMap(() => token$));
}
```

Code 6: Login Funktion

Der oben gezeigte Code ist eine stark verkürzte Form des Originals. Die Funktion führt den Login aus, indem sie Benutzernamen und Passwort vom User Input entgegennimmt. Danach werden mehrere sogenannte Observables definiert. Diese sind einfach asynchrone funktionale Definitionen, die erst später ausgeführt werden. Diese Observables kommen entweder von Zwischenfunktionen oder direkt von `this.http`, welche die Klasse ist, die uns durch Dependency Injection zur Verfügung gestellt wurde, um HTTP-Anfragen zu erstellen. Somit sind das einfach gesagt nur Anfragen an die API. Zum Schluss werden noch die Observables verkettet und zurückgegeben, damit der Hauptprogrammcode das Observable ausführen kann. Alles verstanden? Falls nicht, nicht verzweifeln, es ist ein bisschen schwierig Funktionales Programmieren auf Anhieb zu verstehen, auch ich habe sehr lange gebraucht, bis ich es überhaupt ansatzweise verstanden habe.

<sup>11</sup> <https://www.learnrxjs.io/> (Stand: 25.03.23)

### 3.4.2 Login

Mit dem vorherigen Code kann man nun den ersten Versuch starten, einen Login durchzuführen. Dazu muss lediglich eine Komponente erstellt und ein Anmeldeformular erstellt werden. Ich werde hier nicht ins Detail gehen und erklären, wie ich dies gemacht habe, da es ziemlich einfach ist und es viele Anleitungen online gibt. Stattdessen werde ich etwas über den Mechanismus erzählen, der dafür sorgt, dass ein nicht angemeldeter Benutzer auf eine Seite geleitet wird, die eine Anmeldung verlangt.

#### 3.4.2.a Routing

Davor benötigt es jedoch eine funktionierende Navigation. In Angular gibt es die Möglichkeit, die Komponenten basierend auf der aktuellen Position auszutauschen. Hierfür verwendet man einen `Router`. Dieser ermöglicht das Definieren von Routen, die mit einer ganz bestimmten URL übereinstimmen.<sup>12</sup>

```
import { NgModule } from '@angular/core';
import { RouterModule } from '@angular/router';
import { AppComponent } from './app.component';
import { GradesComponent } from './grades/grades.component';
import { AuthGuard } from './guard/auth.guard';
import { LoginComponent } from './login/login.component';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    GradesComponent,
  ],
  imports: [
    RouterModule.forRoot([
      { path: 'grades', component: GradesComponent },
      { path: 'subject', component: SubjectComponent },
      { path: 'login', component: LoginComponent },
      { path: '**', component: LoginComponent },
    ]),
  ],
  providers: [],
  bootstrap: [AppComponent],
})
```

Code 7: Router Modul

<sup>12</sup> <https://angular.io/guide/routing-overview> (Stand: 28.03.23)

Zu beachten ist das `RouterModule.forRoot()`. Dort wird ein Array angegeben, welches die Routen für die Applikation angibt. Als Erstes gibt man den Pfad an, auf den die Applikation die Komponente setzen soll. Danach wird die Komponente angegeben. Den Rest erledigt Angular. Angular sorgt dafür, dass die richtige Komponente angezeigt wird, wenn man `example.com/grades` aufruft.<sup>13</sup>

### 3.4.2.b Guard

Jetzt stellt sich die Frage, wie wir die Komponenten vor unerlaubten Zugriffen schützen. Dazu gibt es einen Guard, der im Prinzip, bevor die Komponente ausgeführt wird, überprüft, ob der Benutzer berechtigt ist. Um einen solchen Guard zu erstellen, führt man in der Angular CLI folgendes Kommando aus:

`ng generate guard NAME` => Generiert ein Injectable für das Router Modul<sup>14</sup>

Mit diesem Kommando wird eine neue Datei generiert, in der unser Guard enthalten ist. Jetzt bearbeiten wir die Funktion `canActivate()` und geben eine Umleitung zurück, wenn der Benutzer nicht authentifiziert ist.

```
import { Injectable } from '@angular/core';
import { CanActivate, Router, UrlTree } from '@angular/router';
import { WebuntisApiService } from '../webuntis/webuntisApi.service';

@Injectable({
  providedIn: 'root',
})

export class AuthGuard implements CanActivate {
  constructor(
    private router: Router,
    private webuntis: WebuntisApiService
  ) {}

  canActivate(): true | UrlTree {
    if (!this.webuntis.isLoggedIn()) {
      return this.router.parseUrl('/login');
    }
    return true;
  }
}
```

Code 8: Guard

<sup>13</sup> <https://angular.io/guide/router#defining-a-basic-route> (Stand: 28.03.23)

<sup>14</sup> <https://angular.io/cli/generate#guard> (Stand: 28.03.23)



Der nächste Schritt besteht darin, den Guard dem zuvor erstellten Router hinzuzufügen. Es sieht dann wie folgt aus:

```
RouterModule.forRoot([
  { path: 'grades', component: GradesComponent,
    canActivate: [AuthGuard] },
  { path: 'subject', component: SubjectComponent,
    canActivate: [AuthGuard] },
  { path: 'login', component: LoginComponent },
  { path: '**', component: LoginComponent },
])
```

*Code 9: RouterModule mit einen Guard*

Wenn ein Benutzer beispielsweise `grades` aufruft, wird zuerst überprüft, ob er authentifiziert ist. Falls das nicht der Fall ist, wird er zur Login-Seite umgeleitet.

### 3.4.3 Noten

Sobald der Login einsatzbereit ist, ist es endlich so weit, sich mit dem eigentlichen Teil des Programms zu beschäftigen. Dazu benötigen wir zuerst die Daten, welche wir wie zuvor erläutert von der API erhalten. Der eigentliche Schritt besteht jedoch darin, die Noten übersichtlich darzustellen. Hierfür habe ich ein Konzept entwickelt. Ich habe mich für die Desktop-Darstellung für eine einfache Tabelle und für den mobilen Sektor für eine Art Akkordeon-Darstellung entschieden. Im folgenden Abschnitt wird genauer erläutert, was genau damit gemeint ist.

#### 3.4.3.a Tables

Man könnte jetzt eine einfache HTML Tabelle zusammenschustern, dies ist jedoch in Angular nicht nötig. Falls man Angular Material verwendet, gibt es eine Sammlung von vorgefertigten Komponenten, die einfach verwendet werden können. Jetzt fragen sich einige, was ist denn schon wieder Angular Material? Dies ist eine Erweiterungsbibliothek für Angular, die vor allem für das Frontend entscheidend ist.<sup>15</sup> Dazu aber später im Kapitel 3.5.1 mehr. Wichtig zu wissen ist nur, dass es verschiedene Komponenten gibt, die man benutzen kann, beispielsweise Tabellen.

---

<sup>15</sup> <https://material.angular.io/> (Stand: 29.03.23)

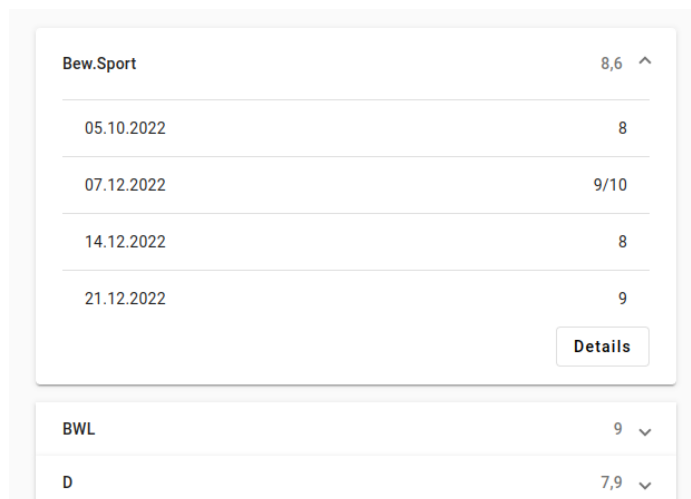
Mit Material-Tables können Daten einfach an die Tabelle übergeben werden. Dies erfolgt mit der `dataSource`. Diese ist dafür zuständig, die Daten als Observable zur Verfügung zu stellen. Sobald der `mat-table` die Daten erhält, erstellt er in diesem Fall eine Reihe mit dem Titel `Subject` und mit den einzelnen Namen in `element.lesson.subjects`.<sup>16</sup>

```
<mat-table #table [dataSource]="dataSource">
  <ng-container matColumnDef="subject">
    <mat-header-cell *matHeaderCellDef>
      Subject
    </mat-header-cell>
    <mat-cell *matCellDef="let element">
      {{ element.lesson.subjects }}
    </mat-cell>
  </ng-container>
</mat-table>
```

Code 10: Einfacher Table

### 3.4.3.b Expansion Panel

Diese Tabelle lässt sich jedoch nicht auf dem Smartphone anzeigen, daher habe ich dafür eine separate Ansicht erstellt. Eine sogenannte "Expansion Panel" oder im deutschen auch "Akkordeon-Anzeige".



Bew.Sport	8,6 ^
05.10.2022	8
07.12.2022	9/10
14.12.2022	8
21.12.2022	9
<button>Details</button>	
BWL	9 v
D	7,9 v

Bild 3.5: Screenshot Expansion Panel

Diese wird so benannt, da die einzelnen Abschnitte mit einem Klick weiter auseinander gezogen werden können, um mehr Daten anzuzeigen. Im Bild 3.5 kann man eine solche Anzeige sehen. Es kann genutzt werden, um wichtige Daten im äußeren Bereich anzuzeigen und Details im aufklappbaren Bereich darzustellen.

<sup>16</sup> <https://material.angular.io/components/table/overview> (Stand: 29.03.23)

Um eine solche Anzeige zu erstellen, benötigt man einen ähnlichen Ansatz wie beim Material-Table. Man verwendet einfach das HTML-Element `<mat-accordion>`.<sup>17</sup>

```
<mat-accordion class="grade-accordion" multi>
  <mat-expansion-panel *ngFor="let item of dataSource.connect() | async">
    <mat-expansion-panel-header>
      <mat-panel-title>
        {{ item.lesson.subjects }}
      </mat-panel-title>
      <mat-panel-description [class]="item.averageMark | colorMark">
        {{ (item.averageMark || '' | number: '0.0-1') || '' }}
      </mat-panel-description>
    </mat-expansion-panel-header>
    <div class="item-body">
      ...
    </div>
  </mat-expansion-panel>
</mat-accordion>
```

Code 11: Einfache Akkordeon-Darstellung

### 3.4.3.c Cards

Zu guter Letzt benötigen wir noch eine Anzeige, die die Durchschnittsnote anzeigen kann. Dabei denke ich an eine Art Karte, die aus dem Hintergrund heraussteicht. Siehe da, Angular Material hat auch für dieses Problem eine Lösung: Material-Cards. Das sind simple Container zur Darstellung von wichtigen Kennzahlen oder Informationen.<sup>18</sup>

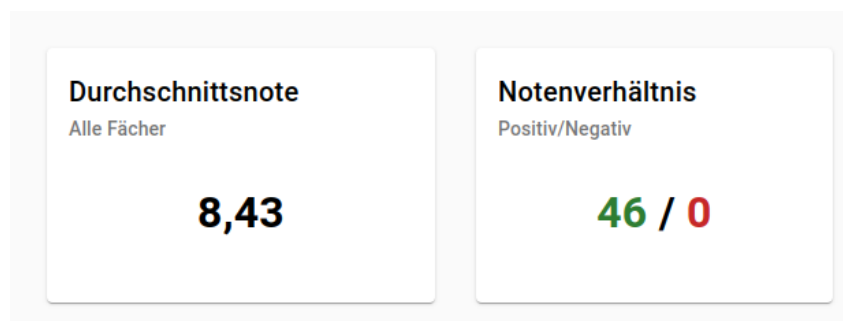


Bild 3.6: Screenshot Material Card

<sup>17</sup> <https://material.angular.io/cdk/accordion/overview> (Stand: 29.03.23)

<sup>18</sup> <https://material.angular.io/components/card/overview> (Stand: 29.03.23)

Ich glaube, wir haben nun verstanden, wie wir diese Komponenten verwenden.

```
<mat-card class="avg-mark">
  <mat-card-header>
    <mat-card-title>Average Mark</mat-card-title>
    <mat-card-subtitle>All subjects</mat-card-subtitle>
  </mat-card-header>
  <mat-card-content>
    <div class="mark">
      <span>{{
        this.averageMark | number: '1.0-2'
      }}</span>
    </div>
  </mat-card-content>
</mat-card>
```

*Code 12: Material Card*

### 3.4.4 Details der Noten

Wie bereits angedeutet, gibt es eine detaillierte Ansicht für jedes Fach. Dazu baue ich eine neue Komponente und erstelle eine Route dafür. Dabei darf ich den Guard nicht vergessen. Sobald die Komponente eingebunden ist, rufe ich die Daten ab, um sie zu nutzen. Oben werden wieder einige Kennzahlen dargestellt. Der interessante Teil dieser Komponente sind die Trenddaten.

#### 3.4.4.a NGX-Chart

Trenddaten lassen sich gut mit einem Diagramm darstellen. Ich habe mich bei einer von der Open Source Community entwickelten Bibliothek bedient. Diese ist speziell für Angular entwickelt und mit ihr kann man einfach und schnell alle möglichen Diagramme erstellen.

Um diese Bibliothek zu installieren, führen wir folgendes Kommando aus:

```
yarn add @swimlane/ngx-charts => yarn ist der Paketmanager für JS Projekte
```

Nachdem wir das Paket installiert haben, können wir das `NgxChartsModule` importieren und im Code verwenden.<sup>19</sup> Das sieht dann wie folgt aus:

```
<ngx-charts-line-chart
  #gradeChart
  [scheme]="colorScheme"
  [legend]="false"
  [showXAxisLabel]="false"
  [showYAxisLabel]="false"
  [xAxis]="true"
  [yAxis]="true"
  [xAxisTicks]="[]"
  [yAxisTicks]="[4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10]"
  [yScaleMax]="yAxisMax"
  [yScaleMin]="yAxisMin"
  [xAxisLabel]="''Index'"
  [yAxisLabel]="''Grade'"
  [results]="data"
>
</ngx-charts-line-chart>
```

Code 13: NGX-Chart HTML

Hier verwende ich eine Methode, um HTML-Attribute mit dem Code zu

<sup>19</sup> <https://swimlane.gitbook.io/ngx-charts/> (Stand: 06.04.23)

verbinden. Dies wird mit eckigen Klammern markiert. Viele Daten gebe ich einfach an, aber manche verweisen auch auf eine Variable im Code. Zum Beispiel sind die Noten als "data" im Code definiert.

Diese Daten müssen eine spezielle Struktur haben, damit die Bibliothek die Daten korrekt anzeigen kann. Um dieses bestimmte Format vorzugeben, habe ich ein sogenanntes Interface erstellt. In TypeScript wird dann überprüft, ob die Daten genau nach dieser Vorlage abgespeichert werden. Andernfalls wird ein Fehler ausgegeben.

```
interface ChartData {  
  name: string;  
  series: { name: string; value: number }[];  
}
```

*Code 14: NGX-Chart Data Interface*

Das Datenkonstrukt muss also wie oben angegeben gespeichert werden. Als erstes braucht es einen Namen für die Datenreihe. Danach kommt ein Array von Objekten, die jeweils einen Namen haben (x-Achse) und einen Wert (y-Achse). Jetzt folgt nur noch die Umwandlung der WebUntis Daten in diese Form. Ich habe das mit einer einfachen foreach-Schleife umgesetzt.

```
value.grades.forEach((grade, i) => {  
  if (grade.mark.markDisplayValue !== 0) {  
    this.data[0]['series'].push({  
      name: `${i + 1} - ${this.datePipe.transform(  
        this.webuntis.convertDate(grade.date.toString())  
      )}`,  
      value: grade.mark.markDisplayValue,  
    });  
  }  
});
```

*Code 15: NGX-Chart Data Transform*

Oben wird zuerst überprüft, ob die Note 0 ist. Falls das zutrifft, wird dieser Wert übersprungen. Danach werden bei der ersten Datenreihe im Array die Daten in der gewünschten Form hinzugefügt. Das Endergebnis sieht wie folgt aus:

**Trend**

Notenentwicklung



Bild 3.7: NGX-Chart

## 3.5 Design

Das Wichtigste bei einer Applikation ist heutzutage das Design. Wenn ein Benutzer die Applikation nicht nutzen kann, weil die Benutzeroberfläche zu kompliziert oder unübersichtlich ist, ist die Applikation nutzlos. Hinzu kommt, dass Benutzer zögern, eine Applikation zu nutzen, die nicht modern und ansprechend aussieht. Da ich jedoch kein erfahrener Frontend-Entwickler bin und nur begrenztes Wissen über das Design von Benutzeroberflächen habe, musste ich mir etwas anderes überlegen. Zum Glück gibt es großartige Tools, die Entwickler nutzen können, um mit wenig Aufwand und Wissen eine klare, übersichtliche und ansprechende Benutzeroberfläche zu gestalten. Erneut bietet Google ein Tool für Angular an. Dieses Tool enthält gängige UI-Komponenten, die in fast jeder Applikation vorkommen. Dadurch hat man Bausteine, mit denen man eine Applikation aufbauen kann.

### 3.5.1 Material Design

Dieses Werkzeug basiert auf den Designstudien von Google, die dazu verwendet wurden, ein Design-System für Entwickler zu erstellen. Der Name dieses Systems ist, wie auch im Titel erwähnt, *Material Design*.<sup>20</sup> Es gibt mehrere Konzepte, auf denen dieses System aufbaut.

Eines davon ist das Konzept, dass sich die Elemente wie im realen Leben verhalten sollen. Eine gute Allegorie dafür ist ein Blatt Papier. Material Design versucht, alles so darzustellen, als wäre es nur ein Blatt Papier im dreidimensionalen Raum. Dabei wird auch die Tiefe eines Elements beachtet. Man kann ein Element weiter nach vorne bewegen, um es hervorzuheben. Dabei wirft es auch einen Schatten auf die hinteren Objekte. Es ist wichtig zu beachten, dass dieses Element keine Dicke hat, genau wie ein Blatt Papier.<sup>21</sup> Der Effekt ist eine klare und vertraute Oberfläche, die gleichzeitig modern und übersichtlich ist.

Ein weiteres Konzept sind die Farben. Farben sollten nicht beliebig gewählt werden, sondern sollten erstens mit der Unternehmensidentität übereinstimmen und zweitens Wichtiges hervorstechen lassen, während sie

---

20 <https://m3.material.io> (Stand: 06.04.23)

21 <https://m2.material.io/design/environment/surfaces.html> (Stand: 06.04.23)



gleichzeitig lesbar und sichtbar bleiben. Dazu gibt es in Material Design eine Hauptfarbe und eine Nebenfärb, um die Applikation zu personalisieren. Außerdem gibt es mehrere Meldefärbn für verschiedene Zwecke wie z.B. Knöpfe, Fehlermeldungen, Warnungen und vieles mehr.<sup>22</sup>

Der letzte wichtige Bereich ist die Typographie, der jeglichen lesbaren Text und dessen Darstellung betrifft. Hierbei werden sogenannte Fonts verwendet, die festlegen, wie eine Schriftart aussehen soll. Anders ausgedrückt, es handelt sich um die Auswahl der Schriftart. Auch bei der Typographie ist es wichtig, dass sie nicht willkürlich gewählt wird, sondern gründlich überdacht wird. Sie sollte sowohl zur Applikation passen als auch lesbar bleiben. Dabei können verschiedene Einstellungen für jeden Font angepasst werden, wie beispielsweise der Buchstabenabstand, die Größe sowie ob es fett oder kursiv dargestellt wird.<sup>23</sup>

### 3.5.2 Angular und Material

Das Konzept ist schön und gut, aber wie setzt man es in der Praxis um? Dafür gibt es eine Bibliothek, die diese vorgefertigten Elemente bereitstellt. Diese muss man lediglich installieren und importieren, und schon kann sie verwendet werden. Wie ich bereits in Kapitel 3.4.3 gezeigt habe, kann man diese Elemente nutzen, um das Material Design in der eigenen Applikation umzusetzen.

Angular verwendet standardmäßig SCSS, eine Erweiterungssprache für CSS, ähnlich wie TypeScript eine Erweiterungssprache für JavaScript ist. Diese Sprachen werden in der Webentwicklung für die Gestaltung von Webseiten verwendet. SCSS bietet zusätzliche Funktionen wie verschachtelte Verweise, Variablen, Module, Mixins und Vererbung, die den Entwicklungsprozess erheblich vereinfachen.<sup>24</sup>

Mit Hilfe dieser Werkzeuge habe ich schließlich die Benutzeroberfläche für meine Applikation erstellt.

---

22 <https://m2.material.io/design/color/the-color-system.html> (Stand: 06.04.23)

23 <https://m2.material.io/design/typography/the-type-system.html> (Stand: 06.04.23)

24 <https://sass-lang.com/documentation/> (Stand: 06.04.23)

## 4 Deployment

Sobald man soweit ist, die Applikation zu veröffentlichen, stellt sich die Frage, wie man diese nun auf den Server bekommt. Keine Angst, man muss dort nicht Angular installieren. Die Applikation kann wie jede andere Website installiert werden. Davor muss sie jedoch kompiliert werden. Dies kann ganz einfach durch folgenden Befehl geschehen:

```
yarn nx build => Wird in den Ordner ./dist/apps kompiliert
```

Diesen Ordner kann man dann lediglich auf den Server kopieren.

### 4.1 Hosting

Damit die Applikation ordnungsgemäß funktioniert, müssen noch einige Einstellungen auf dem Server vorgenommen werden. In diesem Beispiel verwenden wir einen Apache Web-Server. Für andere Konfigurationen siehe bitte die Angular-Dokumentation. Wenn man den Ordner untersucht, wird man feststellen, dass es nur eine HTML-Datei gibt. Nun stellt sich die Frage, wohin die einzelnen Seiten der Applikation verschwunden sind. Keine Sorge, sie sind immer noch da. Angular ändert mithilfe des DOMs die HTML-Datei, um mehrere Seiten anzuzeigen. Dies erfordert jedoch eine spezielle Konfiguration am Web-Server. Bei Apache sieht diese Konfiguration wie folgt aus:

```
RewriteEngine On
# If an existing asset or directory is requested go to it as it is
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -f [OR]
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -d
RewriteRule ^ - [L]

# If the requested resource doesn't exist, use index.html
RewriteRule ^ /index.html
```

Code 16: Apache Konfiguration

Diese Konfiguration sorgt dafür, dass alle Anfragen auf `index.html` umgeleitet werden. Dadurch wird verhindert, dass ein Link zu einer nicht existierenden Datei führt. Zusätzlich muss sichergestellt werden, dass Bilder und Ressourcen nicht auf `index.html` umgeleitet werden.<sup>25</sup>

In meiner Applikation habe ich mehrere Sprachen mithilfe des I18n-Moduls

---

<sup>25</sup> <https://angular.io/guide/deployment> (Stand: 06.04.23)

eingebettet. Da ich jedoch nur Deutsch auf dem Server benötige, kopiere ich nur den entsprechenden Ordner. Dazu muss auch in der `index.html` Datei `<base href="/de/">` zu `<base href="/">` geändert werden. Danach funktioniert die App wie gewünscht.

#### 4.1.1 Reverse-Proxy

Ein weiteres Detail, das ich bisher nicht erwähnt habe, ist, dass der Browser standardmäßig keine API-Verbindungen zu einer anderen Domäne zulässt, es sei denn, dies ist auf der API-Seite explizit definiert. Da ich jedoch nicht verlangen kann, dass WebUntis ihre Konfiguration ändert, musste ich kreativ werden. Ich habe einen Reverse Proxy auf demselben Server installiert, auf dem meine App läuft, und diesen auf die API weitergeleitet. Dadurch denkt der Browser, dass er sich mit derselben Domäne verbindet und erlaubt letztendlich den Zugriff. Dies kann auch mit einer Apache-Konfiguration erreicht werden:

```
ProxyRequests Off
ProxyPassReverse "/api/" "https://mese.webuntis.com/"
ProxyPass "/api/" "https://mese.webuntis.com/"
```

*Code 17: Apache Proxy Configuration*

Diese Konfiguration bewirkt, dass, wenn die Applikation auf den Pfad `/api/` zugreift, die Anfrage an `mese.webuntis.com` weitergeleitet wird und die Antwort an den Browser zurückgeschickt wird. Dadurch kann die sogenannte CORS-Policy umgangen werden.

## 4.2 PWA

Progressive Web Apps sind eine vergleichsweise neue Technologie im Internet, die Vorteile von lokal installierten Apps bieten. Sie kombinieren die Flexibilität von Web-Apps mit den Vorteilen von lokal installierten Apps. Mittlerweile werden sie von Apple, Android, Windows und Linux unterstützt.<sup>26</sup> Daher ist es sinnvoll, meine App auch als Progressive Web App anzubieten. In Angular ist es einfach, dies umzusetzen, indem man eine entsprechende Bibliothek installiert und einige Einstellungen vornimmt.<sup>27</sup>

<sup>26</sup> [https://developer.mozilla.org/en-US/docs/Web/Progressive\\_web\\_apps](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps) (Stand: 06.04.23)

<sup>27</sup> <https://angular.io/guide/service-worker-intro> (Stand: 06.04.23)

## 5 Endergebnis

Sobald alle Teile zusammengefügt wurden, stellt sich jedoch die Frage, wie man das Programm tatsächlich verwendet. Hier kommt die Perspektive des Endbenutzers ins Spiel. Für ihn muss das Programm intuitiv bedienbar und verständlich sein. Ich habe versucht, das so gut wie möglich zu erreichen, aber trotzdem können noch Fragen auftreten. Deshalb biete ich hier eine kleine Hilfe an.

### 5.1 Zugang und Login

Als erstes braucht man einen Zugang zum Programm. Dazu muss man ein Konto direkt bei WebUntis haben, das der jeweiligen Schule gehört. Ohne einen solchen Benutzer hat man keinen Zugriff auf das Programm. Wenn man Schüler ist, sollte man zu Beginn des Schuljahres den Zugang für WebUntis erhalten haben. Nun stellt sich die Frage, wie man die Anwendung installiert. Zur Erinnerung: Es handelt sich um eine Web-Anwendung, die im Web bereitgestellt wird. Wie das genau funktioniert, habe ich im Kapitel 4.1 genauer erklärt. In jedem Fall kann man die Anwendung mit jedem gängigen Internet-Browser wie Google Chrome, Firefox oder Safari öffnen. Das bedeutet, dass die Anwendung auf allen Geräten

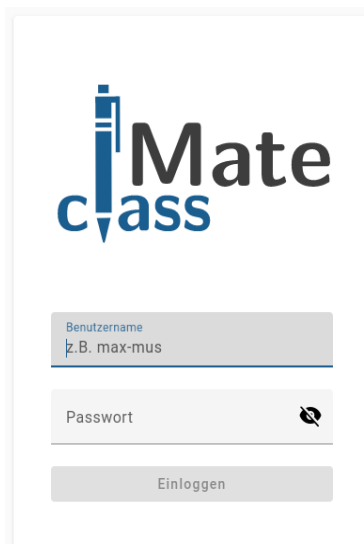


Bild 5.1: Screenshot  
Anmeldeformular

geöffnet werden kann. Der erste Schritt ist, den Browser zu öffnen und die Anwendung über folgende Adresse zu öffnen: <https://noten-tschuggmall.eu/>. Diese Instanz gehört zu meiner Schule. Daher muss man auch Schüler an dieser Schule sein. Sollte es sich um eine andere Schule handeln, muss man entweder seine eigene Instanz auf einem Server bereitstellen oder die Schule um Unterstützung bitten, eine Instanz bereitzustellen. Wenn man dann auf der richtigen Seite gelandet ist, sieht man das rechts abgebildete Anmeldefenster. Dort muss man den selben Benutzernamen und dasselbe Passwort wie bei WebUntis eingeben, um sich einzuloggen.

## 5.2 Notenübersicht

Sobald man sich erfolgreich angemeldet hat, bekommt man eine Übersicht über alle Noten, die im aktuellen Schuljahr vergeben wurden. Ganz oben auf der Seite befindet sich eine Reihe von Kärtchen, die vor allem zur Anzeige einzelner Kennzahlen dienen. Als erstes wird die Durchschnittsnote angezeigt, die sich aus der Summe aller Noten geteilt durch die Anzahl der Noten ergibt. Dies zeigt das arithmetische Mittel aller Noten an. Das nächste Kärtchen zeigt das Verhältnis zwischen positiven (grünen) und negativen (roten) Noten an. Hierbei werden alle Noten unterhalb der Note 6 als negativ angesehen. Das letzte Kärtchen zeigt die zuletzt hinzugefügten Noten an. Diese Anzeige ist dazu da, um nach neuen Noten Ausschau zu halten. Zuletzt gibt es eine Tabelle, in der alle Noten aufgelistet sind.

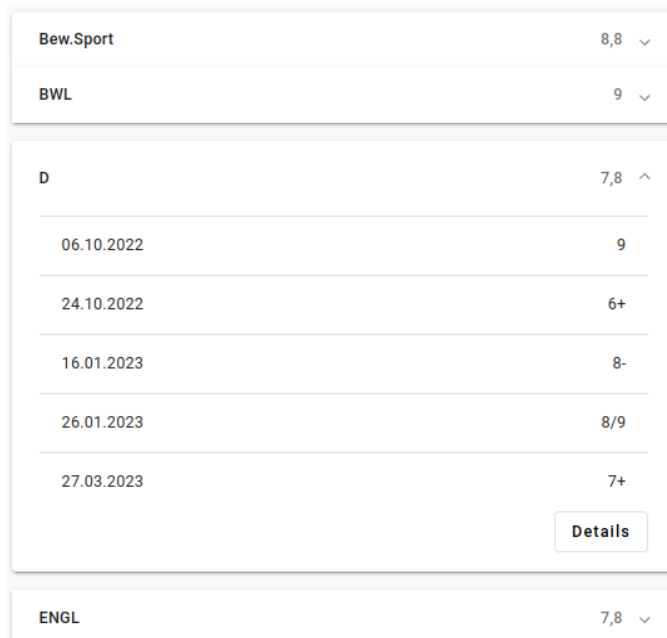
<b>Durchschnittsnote</b> Alle Fächer <b>8,36</b>		<b>Notenverhältnis</b> Positiv/Negativ <b>57 / 0</b>		<b>Kürzlich hinzugefügte Noten</b>	
				IT	7/8 19.04.2023
				GE	9 17.03.2023
				Bew.Sport	9/10 12.04.2023

Fach							Durchschnitt
Bew.Sport	8	9/10	8	9	9/10		8,8
BWL	8/9	10-	9-				9
D	9	6+	8-	8/9	7+		7,8
ENGL	8	8	8	7/8	7	8	7,8
FT							/
GE	8/9	9/10	9	9			9
Inst.W.	9-	9	9/10	9+	8+		9
IT	7-	7-	8	8	8-	7/8 7/8	7,5
M	10-	9/10	9	9	9/10		9,4
P.Manag.	8+	8/9	9/10	9-	9-	9/10	8,9
Projekt	8+	8	8	8	8/9	8/9	8,2
R							/
T.Physik	6/7	6	6	10	9+		7,6

Bild 5.2: Screenshot Übersicht

Im Vergleich zu den anderen Kärtchen hat die Tabelle eine besondere Eigenschaft. Wenn der Bildschirm zu klein wird, verwandelt sie sich in eine Akkordeon-Anzeige. Das bedeutet, dass man auf dem Smartphone zuerst nur eine Reihe von Fächern und deren Durchschnittsnote sieht. Wenn man auf die



Bew.Sport	8,8
BWL	9
D	7,8
06.10.2022	9
24.10.2022	6+
16.01.2023	8-
26.01.2023	8/9
27.03.2023	7+
Details	
ENGL	7,8

Zeile klickt, öffnet sich der Reiter und die einzelnen Noten werden angezeigt. Darunter befindet sich ein Button, der auf die Details verlinkt. Bei einem größeren Bildschirm hingegen werden alle Noten sofort angezeigt. Um das Datum der Note zu sehen, muss man den Mauszeiger auf die Note zeigen, dann wird ein weiteres Feld mit dem Datum angezeigt. Um Details anzuzeigen, muss man auf die jeweilige Zeile drücken.

Bild 5.3: Screenshot Akkordeon

## 5.3 Details

Sobald man auf ein Fach klickt, gelangt man auf die Detailseite. Dort sieht man die gleichen Kennzahlen wie auf der Übersicht, nur sind sie auf das jeweilige Fach bezogen. Was sofort ins Auge sticht, ist das Liniendiagramm darunter. Es zeigt den Notenverlauf des jeweiligen Fachs an. Wenn man nun auf einen Datenpunkt klickt, zeigt es die Note mit dem entsprechenden Datum an. Bei näherer Betrachtung sieht man auch den Mittelwert-Rechner. Dieser ist dazu da, um den zukünftigen Mittelwert zu berechnen. Man startet mit dem aktuellen Mittelwert und kann dann die nächste erwartete Note eingeben, um zu sehen, wie sich das auf den Mittelwert auswirkt. Ein Beispiel wäre, wenn man beim nächsten Test eine 9 erreichen will, um beispielsweise einen Zielmittelwert von 8 zu erreichen. Wenn man jedoch feststellt, dass dies nicht ausreicht, kann man eine weitere gewünschte Note eingeben oder die letzte Note mit dem Löschesymbol neben den Noten löschen und austauschen.

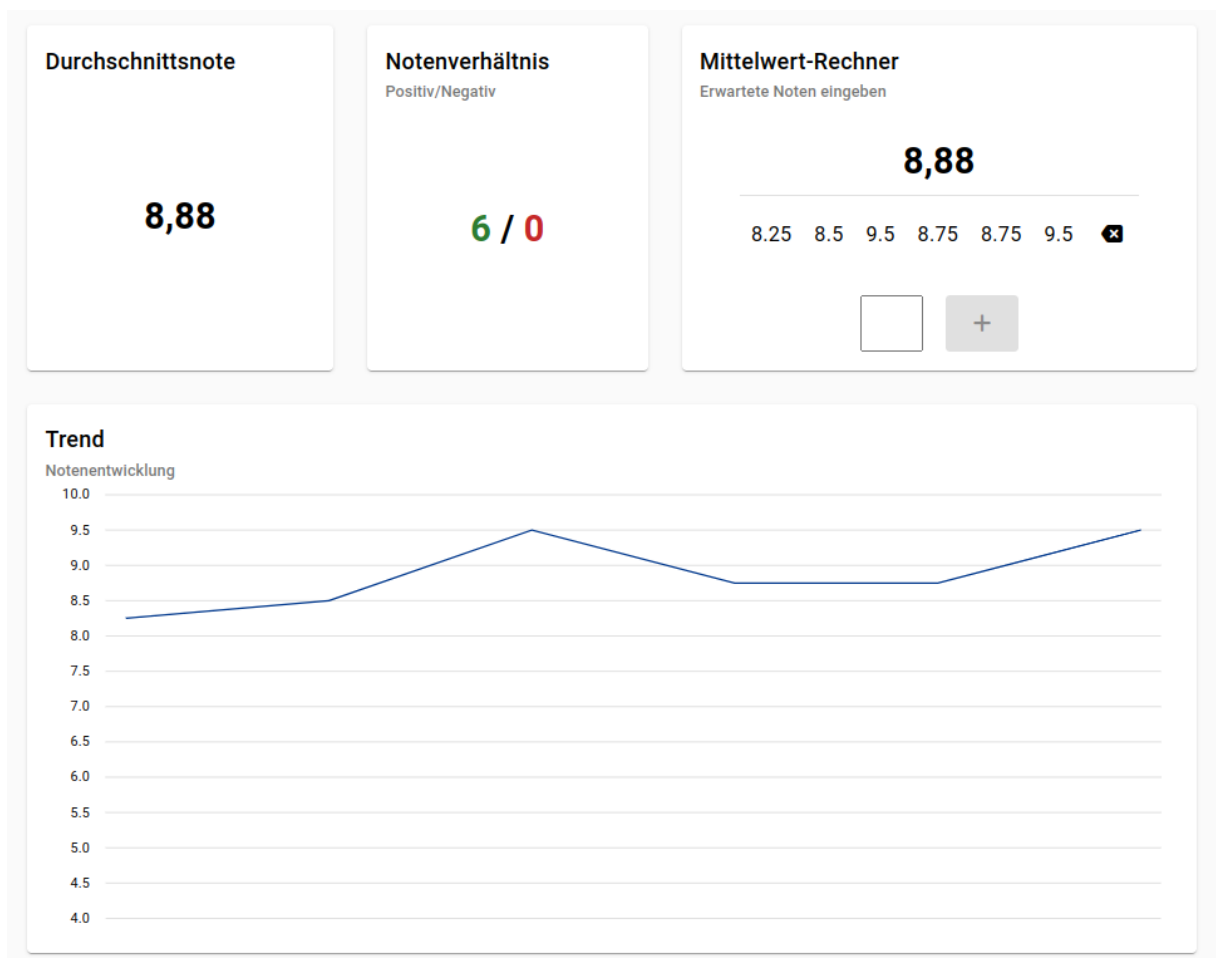


Bild 5.4: Screenshot Details

## 5.4 Installation

Richtig gehört: Installation. Die Applikation kann wie jede andere App installiert werden. Allerdings hängt dies vom Gerätetyp ab. Wenn man auf einem Desktop oder einem Android-Gerät ist, sieht man meistens ein Installationssymbol im Menü oder der Suchleiste. Auf dieses klickt man anschließend, um es zu installieren. Danach erscheint die App im Anwendungsmenü. Alternativ wird beim ersten Öffnen der Seite auf dem Smartphone eine Meldung unten angezeigt, die fragt, ob man die App dem Home-Bildschirm hinzufügen möchte. Bei einem Apple-Gerät ist das etwas anders. Dort muss man auf das Teilen-Symbol gehen und dann auf "Weitere" drücken, um den Knopf auf den Startbildschirm zu senden.

## 6 Fazit

Im Rahmen meiner Facharbeit zur Erstellung einer Web-App habe ich eine Menge dazugelernt. Dazu gehören die Verbesserung meiner Kenntnisse in Angular und des gesamten Entwicklerumfelds. Während des Projekts konnte ich praktische Erfahrungen in der Umsetzung sammeln. Dabei lag der Fokus auf der Programmierung, aber ich habe auch erstmals ein Projekt von Grund auf geplant, überprüft und erfolgreich abgeschlossen, wie es auch in der Arbeitswelt üblich ist.

Die Entwicklung erfolgte dabei in mehreren Schritten, beginnend mit der Planung und Recherche für das Projekt, gefolgt von der eigentlichen Umsetzung. Wobei mir dort einige Herausforderungen begegnet sind.

Der Schwerpunkt lag bei der Programmierung auf der Erstellung einer einfachen und übersichtlichen Benutzeroberfläche für die Notenübersicht. Dieses Ziel wurde innerhalb des vorgegebenen Zeitrahmens erreicht, wobei besonderer Wert auf Benutzerfreundlichkeit gelegt wurde.

Bei der Programmierung habe ich mich selbstverständlich an bewährte Best Practices gehalten, um nicht nur die Funktionalität der Applikation zu gewährleisten, sondern auch einen lesbaren und fehlerfreien Programmcode anzubieten.

### 6.1 Problematiken

Insgesamt verlief das Projekt gemäß meinen Vorstellungen. Es traten jedoch einzelne Herausforderungen auf, die ich lösen musste.

#### 6.1.1 Kommunikation mit der API

Das erste Problem bestand darin, die Anbindung an die API für die Web-App zu realisieren. Der Browser erlaubte mir nicht, mich mit einer externen API zu verbinden, was aus Sicherheitsgründen verständlich ist. Daher musste ich mich ausführlich mit der CORS-Policy auseinandersetzen, um mögliche Lösungen zu finden.



### **6.1.2 SSO und WebUntis**

Darauf kam mir schon das nächste Problem entgegen. Wie schaffe ich es, ohne mich erneut anzumelden, auf das Tool zuzugreifen? Leider bot WebUntis keine Möglichkeiten, dies auf einfache Weise zu realisieren. Daher musste ich darauf zurückgreifen, das Passwort im Browser zu speichern, um den Zugriff zu erleichtern.

## **6.2 Was verbirgt die Zukunft?**

Ich bin froh, dass die Applikation auch eingesetzt werden kann und somit das Leben der Schüler wieder um ein bisschen einfacher wurde. In Zukunft wird diese Applikation weiterhin ihren Dienst leisten und hoffentlich noch lange eingesetzt werden. Persönlich habe ich durch dieses Projekt eine Menge dazugelernt, was ich auch in der Arbeitswelt anwenden kann. Somit hat mir das Projekt nicht nur geholfen, mehr dazu zu lernen, sondern auch mich selbst ein bisschen besser auf die Arbeitswelt vorzubereiten.

Das Schönste an der Programmierung ist es zu sehen, dass das funktionierende Endprodukt letztendlich von den Benutzern gerne verwendet wird.

Raphael Amhof, 2023

## 7 Literaturverzeichnis

### Onlinequellen:

<https://angular.io/>

<https://de.wikipedia.org/>

<https://github.com/>

<https://www.typescriptlang.org/>

<https://material.angular.io/>

<https://swimlane.gitbook.io/>

<https://m3.material.io/>

<https://m2.material.io/>

<https://sass-lang.com/>

<https://developer.mozilla.org/>

<https://www.learnrxjs.io/>

## 8      **Abbildungsverzeichnis**

Für alle Bilder und Grafiken in diesem Dokument besitze ich alle Rechte:

Titelbild: ClassMate Logo

### **Abbildungen**

Bild 1: Grobziele.....	6
Bild 2: Soziales Umfeld.....	6
Bild 3: Projektstrukturplan.....	13
Bild 4: Navigationsdiagramm.....	25
Bild 5: Data Flow Diagramm.....	26
Bild 6: Netzwerkanalyse von WebUntis.....	27
Bild 7: API Zugriffsschema.....	27
Bild 8: Screenshot Expansion Panel.....	34
Bild 9: Screenshot Material Card.....	35
Bild 10: NGX-Chart.....	39
Bild 11: Screenshot Anmeldeformular.....	44
Bild 12: Screenshot Übersicht.....	45
Bild 13: Screenshot Akkordeon.....	46
Bild 14: Screenshot Details.....	47

## 9 Erklärung

Hiermit erkläre ich, dass ich die vorliegende Facharbeit zum Thema „Notendarstellung – Angular Applikation“ ohne fremde Hilfe erstellt habe. Ich versichere zudem, dass ich noch keine Arbeit zum selben Thema verfasst habe. Alle verwendeten Quellen wurden angegeben.

Pfalzen, 09.05.23

## **10 Anhang**

Github Repository: [github.com/Schlaumra/classMate](https://github.com/Schlaumra/classMate)