



Höhere Technische Bundeslehranstalt  
und Bundesfachschule  
im Hermann Fuchs Bundesschulzentrum

# Notenmanagment

## Projektarbeit

*ausgeführt im Schuljahr 2017/2018 von:*

Lukas Friedl (LF), 5AHELS  
Alexander Leimer (AL), 5AHELS

*Betreuer:*

Dipl.-Ing. Franz Matejka

17. April 2018

# Thema:

# Notenmanagment

## Subthemen und Bearbeiter:

### **Front End (Client)**

Alexander Leimer, 5AHELS

*Betreuer:* Dipl.-Ing. Franz Matejka

### **Back End (Server)**

Lukas Friedl, 5AHELS

*Betreuer:* Dipl.-Ing. Franz Matejka

## PROJEKTARBEIT DOKUMENTATION

<b>Verfasser/innen</b>	Lukas Friedl, Alexander Leimer
<b>Jahrgang Schuljahr</b>	5AHELS 2017/2018
<b>Thema des Projekts</b>	Notenmanagment
<b>Aufgabenstellung</b>	Die Aufgabe war eine Website zu erstellen, welche mit einem MySQL-Server kommuniziert, um die Noten der Schüler auszugeben. Es soll möglich sein die einzelnen Klassen auszugeben, sowie die Ergebnisse eines Schülers anzuzeigen, direkt nach einem Schüler zu suchen und neue Tests einzutragen. Eine weitere Erweiterung wäre die Einbindung einer Authentifizierung, sodass nicht jeder Schüler die Noten der anderen sieht und nur der Lehrer die neuen Tests eintragen kann.
<b>Realisierung</b>	Nach einem Brainstorming wurde beschlosssen die Javascript-Library Materialize zu verwenden, da sie neben Bootstrap eine sehr gute Visualisierung bietet. Die Webseite besteht aus einem HTML-File und einem JavaScript-File. Der Server selbst ist nur ein weiteres JavaScript-File, welches am Anfang des Testens mit einem Consolen-Befehl ausgeführt wird. Dazu muss am Computer bzw. in der Virtuellen Maschine NodeJS-Paket installiert werden. Zusätzlich müssen die Module express und mysql installiert werden. Die Datenbank selber besteht aus 5 Tabellen die miteinander verbunden sind.
<b>Ergebnisse</b>	Von der Grundidee sind die Ausgabe der Klassen und der einzelnen Test fertiggestellt worden. Auch die Kommunikation mit der Datenbank funktioniert hier. Um die Tests einzutragen wurde ein Navbar verwendet um zwischen den einzelnen Websites zu wechseln. Die Suche und das Hinzufügen der Tests sind derzeit nur im Front-End realisiert, die Verbindung mit der Datenbank wurde aus Zeitgründen noch nicht programmiert. Die Webseite beim Test hinzufügen ist zurzeit nur eine grobe Idee und noch nicht ausgereift. Bei der Suche und dem Hinzufügen wurde sich an das Design der vorherigen Version gehalten.

# Inhaltsverzeichnis

<b>Vorwort</b>	<b>v</b>
<b>Zusammenfassung</b>	<b>vi</b>
<b>1 Aufgabenstellung/Pflichtenheft</b>	<b>1</b>
1.1 Back-End . . . . .	1
1.2 Front-End . . . . .	1
<b>2 Bedienungsanleitung</b>	<b>2</b>
<b>3 Realisierte Lösung</b>	<b>7</b>
3.1 Datenbank Design . . . . .	7
3.2 API-Endpoints . . . . .	7
3.2.1 /api/get_all-classes . . . . .	7
3.2.2 /api/get_all-subjects . . . . .	7
3.2.3 /api/get_subjects-from-classes/:kid . . . . .	7
3.2.4 /api/get_classchecks/:kid/:fid . . . . .	7
3.2.5 /api/get_classchecks/:kid . . . . .	7
3.2.6 /api/get_results/:tid . . . . .	8
3.2.7 /api/get_all-subjects-from-student/:sid . . . . .	8
3.2.8 /api/post_test . . . . .	8
<b>Autoren</b>	<b>9</b>

# Vorwort

Die Aufgabenstellung wurde uns von unserem Betreuer gestellt und wir hatten einige Wochen Zeit um die bereits bestehende Webseite so umzuprogrammieren wie wir sie gerne haben möchten. Durch einige Termine und der Abgabe der Diplomarbeit sind uns leider einige Stunden ausgefallen, deswegen konnte das Projekt nicht im vollen Umfang aus programmiert werden. Die meiste Zeit hat danach die Erstellung der Tabellen gedauert, da es einige sind und man alle miteinander verbinden muss um für jeden Fall die richtigen Daten zu bekommen. Die Programmierung der Webseite und der Funktionen für die SQL-Aufrufe ging dann sehr rasch.

# Zusammenfassung

Diese Projektarbeit beschreibt das Projekt Notenmanagment, welches eine Kombination aus einer Webseite und einem MySQL-Servers darstellt. Das Projekt ist ein reines Softwareprojekt und hat keinen Hardwareteil. Kurz gesagt, soll das Projekt Daten von Test aus der Datenbank auslesen und auch neue Daten eintragen. Es werden Daten der Tests und der einzelnen Schüler von der Datenbank abgerufen und ausgegeben. Um direkt nach einem Schüler zu suchen sollte eine Such-Funktion implementiert werden um schnell den gesuchten Schüler zu finden dies wurde aber aus Zeitgründen nicht programmiert. Ebenfalls war für die Programmierung der Authentifizierung die Zeit zu knapp. Die Webseite ist für alle Geräte optimiert und skaliert (Responsive Design).

# 1 Aufgabenstellung/Pflichtenheft

## 1.1 Back-End

Vorgegebene Schülerdaten in die Datenbank einlesen  
Funktionen zur Kommunikation mit der Webseite programmieren  
Alle Daten sind in einer Datenbank zu speichern  
Der Zugriff erfolgt über einen Node.js Server  
Tests können editiert und gelöscht werden  
Testergebnisse können editiert werden Für eine einzelne Klasse  
Für ein bestimmtes Fach

## 1.2 Front-End

Programmierung der grafischen Oberfläche  
Für eine einzelne Klasse  
Für ein bestimmtes Fach  
Alle Tests von einer Klasse werden angezeigt  
Tests können editiert und gelöscht werden  
Testergebnisse können editiert werden  
Mehrere HTML-Seiten

## 2 Bedienungsanleitung

Als Erstes muss Node.js installiert werden:

<https://nodejs.org/en/download/>

hier kann Node.js heruntergeladen werden.

Jetzt wird das Projekt von GitHub heruntergeladen.

<https://github.com/EnergyFussl/notenmanagment>

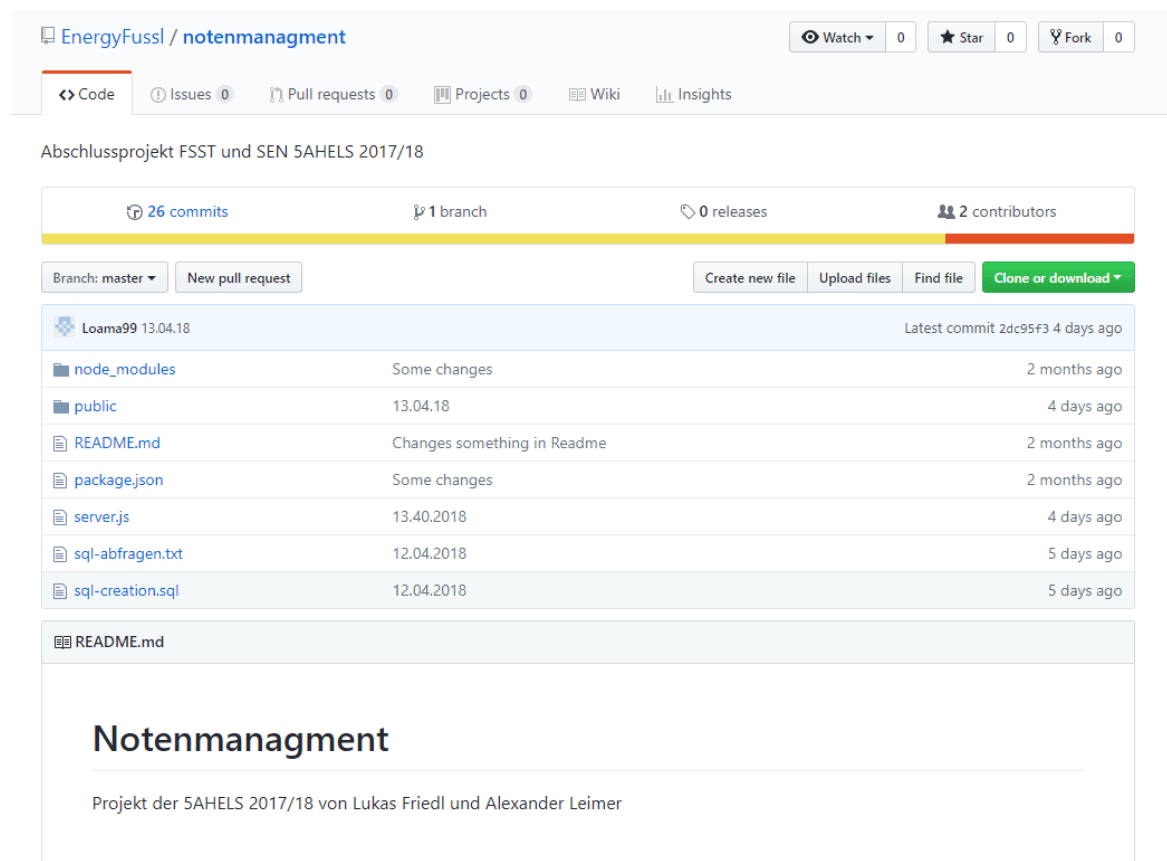


Abbildung 2.1: Github

Dazu einfach auf den Download-Button drücken. Danach wird das heruntergeladene Projekt entpackt und in den gewünschten Ordner verschoben.



Dann im Ordner Shift + Rechte Maustaste drücken und PowerShell öffnen:  
(Es können auch andere Programme verwendet werden, das ist eine einfache Lösung für Windows)

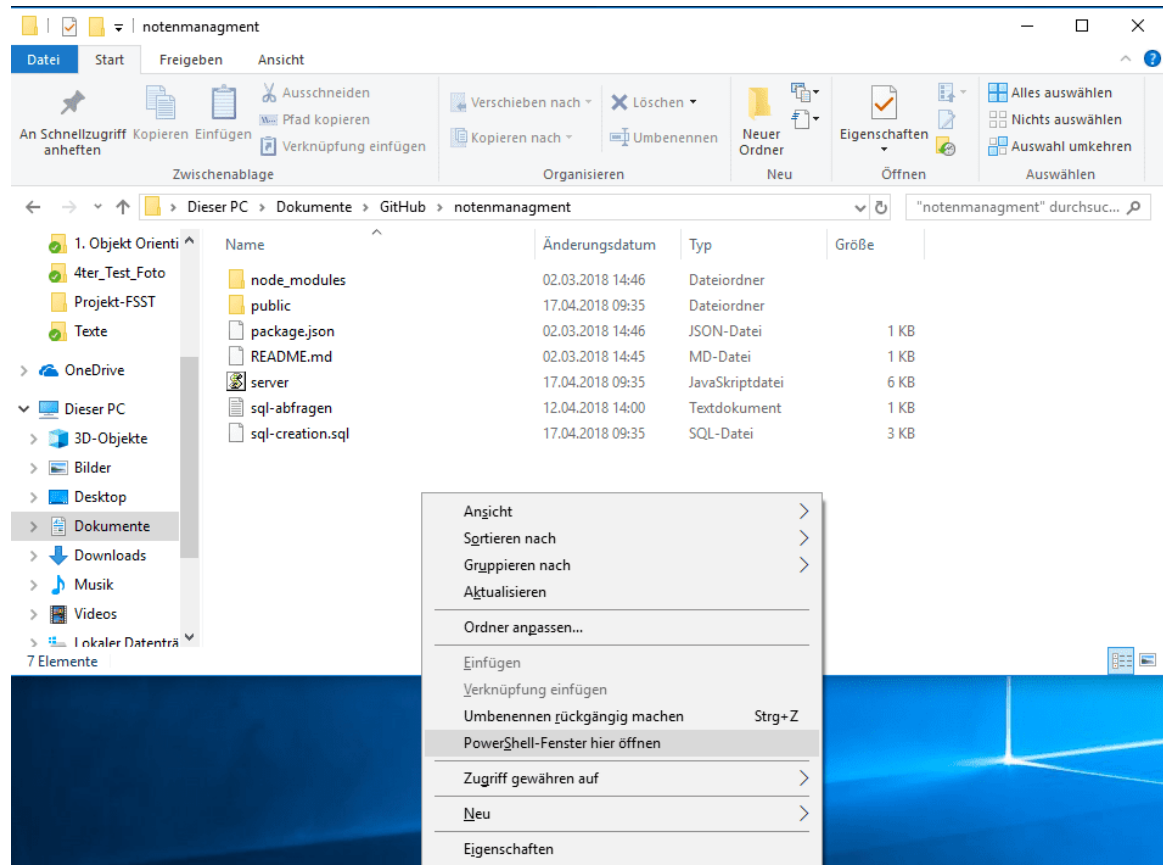
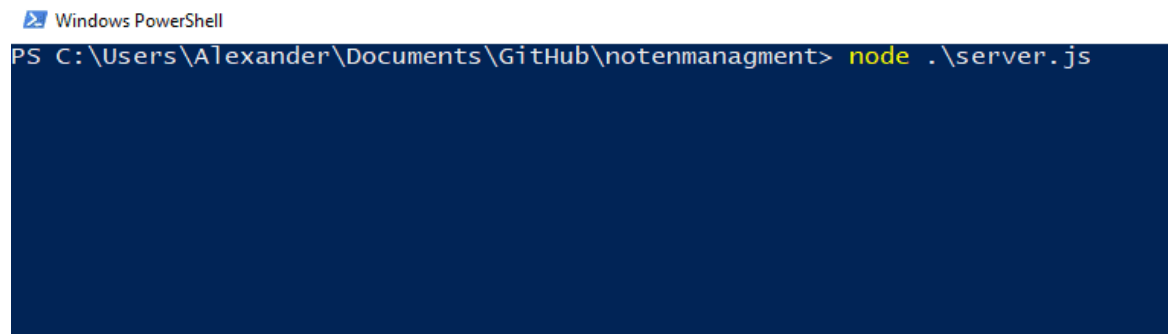


Abbildung 2.2: PowerShell

Nun in der PowerShell diesen Befehl eingeben um den Server zu starten:



```
Windows PowerShell
PS C:\Users\Alexander\Documents\GitHub\notenmanagment> node .\server.js
```

Abbildung 2.3: Server starten

Startseite bei Aufruf der Webseite:

Bei drücken des Home-Buttons wird immer diese Webseite aufgerufen.

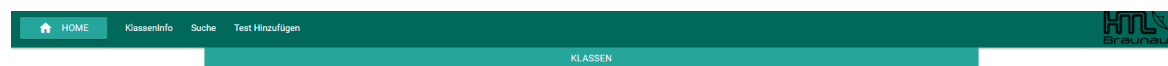


Abbildung 2.4: Startseite

Nach dem Klicken auf Klassen werden alle Klassen der HTL angezeigt:



1AHELS	2AHELS	3AHELS	4AHELS	5AHELS
1BHELS	2BHELS	3BHELS	4BHELS	5BHELS
1CHELS	2CHELS	3CHELS	4CHELS	5CHELS
1DHELS	2DHELS	3DHELS	4DHELS	5DHELS
1AHMEA	2AHMEA	3AHMEA	4AHMEA	5AHMEA
1BHMEA	2BHMEA	3BHMEA	4BHMEA	5BHMEA
1AHET	2AHET	3AHET	4AHET	5AHET
1AFEL	2AFEL	3AFEL	4AFEL	

Abbildung 2.5: Startseite

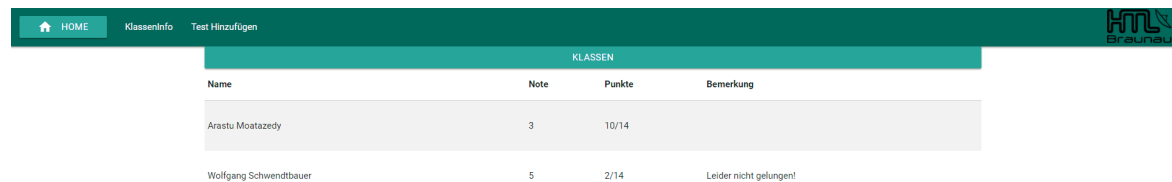
Nun kann man seine Klasse auswählen, indem man auf den Namen klickt:  
Dann werden alle Tests dieser Klasse angezeigt. Die Tests sind nach dem Datum sortiert, das heißt die neuesten Tests stehen als erstes:



KLASSEN				
Typ	Fach	Thema	Datum	
SMÜP	FSST	Amplitudenmodulation	09-01-2018	<a href="#">Ansehen</a>
Test	FSST	Netzwerkverfahren	13-03-2018	<a href="#">Ansehen</a>
Test	SEN	SQL und Node.js	13-02-2018	<a href="#">Ansehen</a>

Abbildung 2.6: Klassen

Jetzt kann man auf den gewünschten Test klicken um die genauen Ergebnisse zu sehen:



KLASSEN			
Name	Note	Punkte	Bemerkung
Arastu Mostazedzy	3	10/14	
Wolfgang Schwendtbauer	5	2/14	Leider nicht gelungen!

Abbildung 2.7: Schueler

Klickt man auf Test Hinzufügen kann man Klasse, Fach, Beschreibung und das Datum eintragen und so den Test in die Datenbank hinzufügen:

KLASSEN

1AHELS

Fach

FSST

Beschreibung

C++

13.04.2018

SPEICHERN >

Abbildung 2.8: Test

Dann kann man für jeden Schüler die Noten eintragen und die Ergebnisse in der Datenbank speichern. Danach wird ein alert-Fenster aufgerufen welches die Speicherung der Daten bestätigt. Klickt man auf OK wird man auf die Homepage weitergeleitet

Name	Note	Punkte	Bemerkung
Arestu Mostazedy	2	14	
Wolfgang Schwendtbauer	1	14,5	1-
Leon Stempfer			gefehlt

SPEICHERN >

Abbildung 2.9: Test

## 3 Realisierte Lösung

### 3.1 Datenbank Design

Wir haben insgesamt 5 Datenbank Tables angelegt mit folgenden Namen:

- subjects für die einzelnen Fächer
- classes für die einzelnen Klassen
- checks für die einzelnen Tests
- students für die einzelnen Schüler
- results für die einzelnen Testergebnisse

### 3.2 API-Endpoints

Des Weiteren wurden insgesamt 6 API-Endpoints implementiert.

#### 3.2.1 /api/get\_all-classes

Dieser Endpoint gibt alle Klassen die in der Datenbank vorhanden sind zurück.

#### 3.2.2 /api/get\_all-subjects

Dieser Endpoint gibt alle Fächer die in der Datenbank vorhanden sind zurück.

#### 3.2.3 /api/get\_subjects-from-classes/:kid

Dieser Endpoint gibt anhand der **KlassenID** alle Fächer einer einzigen Klasse zurück.

#### 3.2.4 /api/get\_classchecks/:kid/:fid

Dieser Endpoint gibt anhand der **KlassenID** und der **FachID** alle Tests der Klasse unter Berücksichtigung des Faches zurück.

#### 3.2.5 /api/get\_classchecks/:kid

Dieser Endpoint gibt anhand der **KlassenID** alle Tests der Klasse zurück.

### 3.2.6 /api/get\_results/:tid

Dieser Endpoint gibt anhand der **TestID** alle Ergebnisse des angegebenen Tests zurück.

### 3.2.7 /api/get\_all-subjects-from-student/:sid

Dieser Endpoint gibt anhand der **SchülerID** alle Fächer eines Schüler zurück. Dieser Endpoint wurde aber nicht fertig gestellt und daher auskommentiert.

### 3.2.8 /api/post\_test

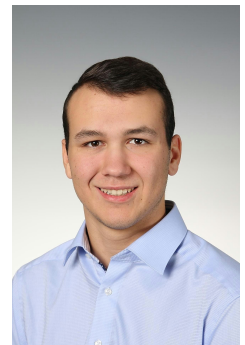
Dieser Endpoint erstellt einen neuen und fügt anschließend die Testresultate in den richtigen MySQL Table ein unter Berücksichtigung der neu erstellten **TID** (TestID). Die Daten werden vom Javascript des Clients über den Body übertragen daher auch die Verwendung des **post** anstatt **get**.

# Autoren

## Lukas Friedl

*Anschrift:* 5303, Thalgau  
Österreich

*E-Mail:* lukas.friedl@htl-braunau.at



## Alexander Leimer

*Anschrift:* 4822, Bad Goisern am Hallstättersee  
Österreich

*E-Mail:* alexander.leimer@htl-braunau.at

