**DIPLOMARBEIT**

Gesamtprojekt

**Webbasiertes Managementsystem für Klassenkonferenzen**

Betreuer/Betreuerin:

**Spezifische Themenstellung Christian Höller**

Christian Höller 5AHWII Prof. MSc Johannes Egger

**Spezifische Themenstellung Elias Werth**

Elias Werth 5AHWII Prof. MSc Johannes Egger

Ausgeführt im Schuljahr 2019/20 von: Christian Höller und Elias Werth

Abgabevermerk:

Datum: übernommen von:

**DIPLOMARBEIT**

**DOKUMENTATION**

|  |  |
| --- | --- |
| Namen der  Verfasser/innen | Christian Höller  Elias Werth |
| Jahrgang  Schuljahr | 5. Jahrgang  2019/20 |
| Thema der Diplomarbeit | Webbasiertes Managementsystem für Klassenkonferenzen |
| Kooperationspartner | HTL Vöcklabruck |

|  |  |
| --- | --- |
| Aufgabenstellung | Die Schule verfügt über eine Anwendung zur Steuerung von Klassenkonferenzen. Dieses System ist jedoch veraltet und daher soll ein neues erstellt werden. Gefordert sind neue Features und ein entsprechend modernes Design. |

|  |  |
| --- | --- |
| Realisierung | Um die Anforderungen der Schule umzusetzen, wurde die Seite neu aufgebaut und mit einem entsprechend modernen Design versehen.  Die Seite wurde in C# mit HTML und JavaScript realisiert. Dies ermöglicht eine übersichtliche Darstellung und einen reibungslosen Ablauf der Konferenz. |

|  |  |
| --- | --- |
| Ergebnisse | Durch den übersichtlichen Aufbau ist es den Lehrern möglich, die Seite leichter zu bedienen. Mithilfe der Nutzerauthentifizierung kann es zu keinem Missbrauch der Rechte kommen. |

|  |  |
| --- | --- |
| Typische Grafik, Foto etc.  (mit Erläuterung) | Abbildung – Nutzer-Ansicht  Abbildung 2 – Moderator-Ansicht    Abbildung – MS Anmeldung |

|  |  |
| --- | --- |
| Teilnahme an Wettbewerben, Auszeichnungen | Keine Teilnahmen |

|  |  |
| --- | --- |
| Möglichkeiten der Einsichtnahme in die Arbeit | Schulbibliothek |

|  |  |  |
| --- | --- | --- |
| Approbation  (Datum / Unterschrift) | Prüfer/Prüferin | Direktor/Direktorin  Abteilungsvorstand/Abteilungsvorständin |

**DIPLOMA THESIS**

**Documentation**

|  |  |
| --- | --- |
| Author(s) | Christian Höller  Elias Werth |
| Form  Academic year | 5th Form  2019/2020 |
| Topic | Web-based management system for class conferences |
| Co-operation partners | HTL Vöcklabruck |

|  |  |
| --- | --- |
| Assignment of tasks | The school has an application for controlling class conferences. However, this system is out of date and a new one should be created. New features and a correspondingly modern design are required. |

|  |  |
| --- | --- |
| Realisation | In order to implement the requirements of the school, the page was rebuilt and given a modern design.  The page was realized in C # with HTML and JavaScript. This enables a clear presentation and a smooth running of the conference. |

|  |  |
| --- | --- |
| Results | The clear structure enables teachers to use the site more easily. With user authentication, rights cannot be misused. |

|  |  |
| --- | --- |
| Illustrative graph, photo  (incl. explanation) | Abbildung – User View  Abbildung – Moderator View      Abbildung – MS Login |

|  |  |
| --- | --- |
| Participation in competitions  Awards | No Participitaion |

|  |  |
| --- | --- |
| Accessibility of  Diploma Thesis | School library |

|  |  |  |
| --- | --- | --- |
| Approval  (Date / Signature) | Examiner | Head of College / Department |

**EIDESSTATTLICHE ERKLÄRUNG**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Vöcklabruck, am TT.MM.JJJJ Verfasser / innen:

Christian Höller  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Elias Werth  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Inhaltsverzeichnis

1 Allgemeines 1

1.1 Vorstellung des Projektteams 1

1.1.1 Diplomanden 1

1.1.2 Vorstellung der Betreuer 4

1.2 Aufgabenstellung der Diplomarbeit 5

1.3 Ausgangssituation 5

1.4 Individuelle Zielsetzungen 6

1.4.1 Zielsetzung Höller 6

1.4.2 Zielsetzung Werth 8

2 Verwendete Programme und Tools 11

2.1 Visual Studio 11

2.2 DB Browser for SQLite 11

2.3 Git 11

2.4 GitHub 12

2.5 JSON-Formatierung 12

2.6 NuGet-Pakete 13

2.6.1 SQLite 13

2.6.2 ASP.net SignalR 13

2.6.3 Newtonsoft.JSON 13

2.6.4 AzureAD 14

2.6.5 Bootstrap 14

3 Sprachen 15

3.1 C# 15

3.2 HTML 15

3.3 CSS 16

3.4 jQuery 16

3.5 JavaScript 17

3.6 MySQL 17

4 Realisierung 18

4.1 Datenspeicherung 18

4.1.1 Speicherung der Konferenzablaufdaten 18

4.1.2 Speicherung der Besprechungsdaten 20

4.2 Datensynchronisierung zwischen Clients 22

4.3 Modellklassen 25

4.3.1 Lehrer 25

4.3.2 Klasse 25

4.3.3 Reihenfolge 25

4.4 Klassen 26

4.4.1 Allgemeine Informationen 26

4.4.2 Datenbankverbindung 29

4.5 Raumauswahl 32

4.6 Steuerung der Konferenz 33

4.6.1 Laden der Seite 34

4.6.2 Informationsaufruf 35

4.6.3 Moderator Informationen 36

4.6.4 Allgemeine Informationen 39

4.6.5 Überschneidungen 44

4.6.6 Steuern der Konferenz 46

4.7 Nutzeransicht 49

4.7.1 Laden der Räume 50

4.7.2 Laden der Informationen 52

4.8 Ansicht Administrator 54

4.8.1 Konferenz zurücksetzen 54

4.8.2 In Datenbank schreiben 55

4.8.3 Bestätigen der Aktionen 56

4.8.4 Registrierung im Azure-Portal 57

4.8.5 Konfiguration der Applikation 58

4.8.6 Implementierung des Logins 60

4.8.7 Weiterleitung nach erfolgreicher Authentifizierung 63

4.9 Progressive Web-App (PWA) 65

4.9.1 Service Worker 65

4.9.2 Manifest 67

4.10 Aufrufen von Lehrkräften 71

4.10.1 Hinzufügen von Aufruf-Buttons 71

4.10.2 Senden des Aufrufs 72

4.10.3 Speichern der Aufrufe 73

4.10.4 Empfangen des Aufrufs 74

4.10.5 Visualisierung der Benachrichtigung 75

4.10.6 Text-to-Speech 77

4.11 Design der Web-App 78

5 Zusammenfassung 79

6 Literaturverzeichnis 80

7 Abbildungsverzeichnis 82

8 Codeverzeichnis 83

ANHANG A - Terminplan 85

ANHANG B - Begleitprotokolle 86

# Allgemeines

## Vorstellung des Projektteams

### Diplomanden

#### Ein Bild, das Person, Schlips, Mann, Wand enthält. Automatisch generierte BeschreibungChristian Höller

Abbildung - Christian Höller

Klasse: 5AHWII

Geburtsdatum: 03.10.2000

Wohnort: Attnang-Puchheim

E-Mail: christian.hoeller@htlvb.at

Ausbildungszweig: Betriebsinformatik

**Beruflicher Werdegang**

**August 2019 - Verfahrenstechnik**

Lenzing AG

Arbeit im Schichtbetrieb zum eigenständigen Messen von Laugen- und Viskosewerten. Diese Werte wurden anschließend analysiert und auf Richtigkeit überprüft.

**August 2018 – Schlosserei**

Lenzing AG

Hilfskraft in der Schlosserei für Reparaturarbeiten, Assistieren bei Schweißarbeiten und Transport.

**August 2017 – Büroarbeit**

MIBA Gleitlager Austria GmbH

Erstellen von Arbeitsmappen über Produktionsmaschinen; Überarbeiten von bereits vorhandenen Dokumenten.

**Ausbildung**

**Seit 2016 – HTL Vöcklabruck – Wirtschaftsingenieure: Betriebsinformatik**

Maturatermin Juni 2020

**2014 – 2016 – HTL Wels – Mechatronik**

**2010-2014 – Bundesrealgymnasium Schloss Wagrain**

Deutsch, Mathematik, Englisch, Religion

#### Elias Werth

Abbildung - Elias Werth

Klasse: 5AHWII

Geburtsdatum: 05.05.2001

Wohnort: Edt bei Lambach

E-Mail: elias.werth@htlvb.at

Ausbildungszweig: Betriebsinformatik

**Beruflicher Werdegang**

**August 2019 – Netzwerktechnik**

BRP-Rotax GmbH & Co KG

Als Hilfskraft für die Netzwerktechniker der Firma über ein Monitoringsystem IP-Adressen vergeben und Gruppen der Device verwalten; Aufbauen von Workstations in der Produktion sowie in anderen Abteilungen.

**August 2019 – Informationstechnologie**

eww Gruppe

Instandhaltung von alten Dokumenten durch Scannen und Abspeichern in dem jeweiligen Ordner; Unterstützung der Außendienstmitarbeiter durch das Festhalten der Maße auf dem jeweiligen Plan.

**Ausbildung**

**Seit 2015 – HTL Vöcklabruck – Wirtschaftsingenieure: Betriebsinformatik**

Maturatermin Juni 2020

**2011-2015 – Sporthauptschule Lambach**

Deutsch, Mathematik, Englisch, Religion

### 1.1.2 Vorstellung der Betreuer

Abbildung - MSc BSc Johannes Egger

**Prof. MSc BSc Johannes Egger**

Ein großes Dankeschön geht an Herrn Prof. Egger, welcher uns tatkräftig bei unserer Arbeit unterstützt hat. Er stand uns bei Fragen und herausfordernden Aufgaben zur Seite, wofür wie uns herzlich bedanken möchten.

## Aufgabenstellung der Diplomarbeit

Aufgabe der Diplomarbeit war es, eine neue Website zu erstellen, welche eine Nutzerauthentifizierung mit Hilfe Microsoft, eine einfache Steuerung der Konferenz und ein modernes Design aufweist.

## Ausgangssituation

Bei der alten Anwendung existiert nur eine Ansicht, wodurch es jedem Nutzer ermöglicht war, die Konferenz zu steuern. Dieses Problem wird durch die Authentifizierung von Nutzern und der dazugehörigen Rechteverteilung gelöst. Folglich ergeben sich eine Nutzer-Ansicht und eine Moderator-Ansicht.

Durch ein modernes und ansprechendes Design wird eine übersichtliche Darstellung der Konferenz erzielt.

## Individuelle Zielsetzungen

### Zielsetzung Höller

#### Datenbankstruktur

Um die Daten möglichst kompakt und übersichtlich zu speichern, musste eine geeignete Datenbankstruktur festgelegt werden. Diese musste noch vor den verschiedenen Nutzeransichten erstellt werden, da die komplette Anwendung auf dieser Datenbank beruht.

#### Moderator-Ansicht

Eines der wichtigsten Anliegen der Schule war es, die Daten für eine Besprechung auf jeweils einer eigenen Seite darzustellen. Somit gibt es für jede Besprechung in einem Raum auch eine eigene Seite. Wie auf der Abbildung zu sehen ist, werden bei der alten Anwendung beide Besprechungen auf einer Seite angezeigt. Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 10 – Screenshot alte Anwendung

Im Gegensatz zu den Buttons „weiter“ und „zurück“, soll die Konferenz nur mehr mit einem zentralen Button gesteuert werden. Sobald der Moderator die Konferenz startet, soll die erste Klasse besprochen werden. Anschließend soll per Knopfdruck automatisch die nächste Klasse und deren zugehörige Daten geladen werden.

Durch die Verlagerung der Besprechungen auf je eine eigene Seite, war es eine große Hürde den Datenaustausch zwischen den Clients zu gewährleisten. Da sich die Überschneidungen der Lehrer ändern, sobald eine neue Klasse geladen wird, müssen diese Daten an den anderen Client in der zweiten Klasse übermittelt werden, ohne dass dieser die Website neu laden muss.

#### Nutzer-Ansicht

Die Nutzer-Ansicht existierte in der alten Anwendung noch nicht und sollte eine Ansicht der Konferenz für Lehrer ermöglichen. Der Nutzer sollte eine Ansicht beider Klassen auf einer Seite haben. Die abgeschlossenen und nächsten Klassen jedes Raumes werden ebenfalls angezeigt.

Grundvoraussetzung für die Ansicht war das automatische Aktualisieren der Daten. Somit werden die Daten für die besprochene Klasse stets aktuell auf der Website dargestellt.

### Zielsetzung Werth

#### Authentifizierung mittels Azure AD

Im Gegensatz zum alten Managementsystem soll das neue Programm die Möglichkeit besitzen, sich anzumelden. Mithilfe von Microsoft soll dies realisiert werden. Der Grund, warum hierfür ein externer Login implementiert wird, ist, dass bereits jede einzelne Lehrkraft über einen Office-365-Account (htlvb-Email) verfügt.

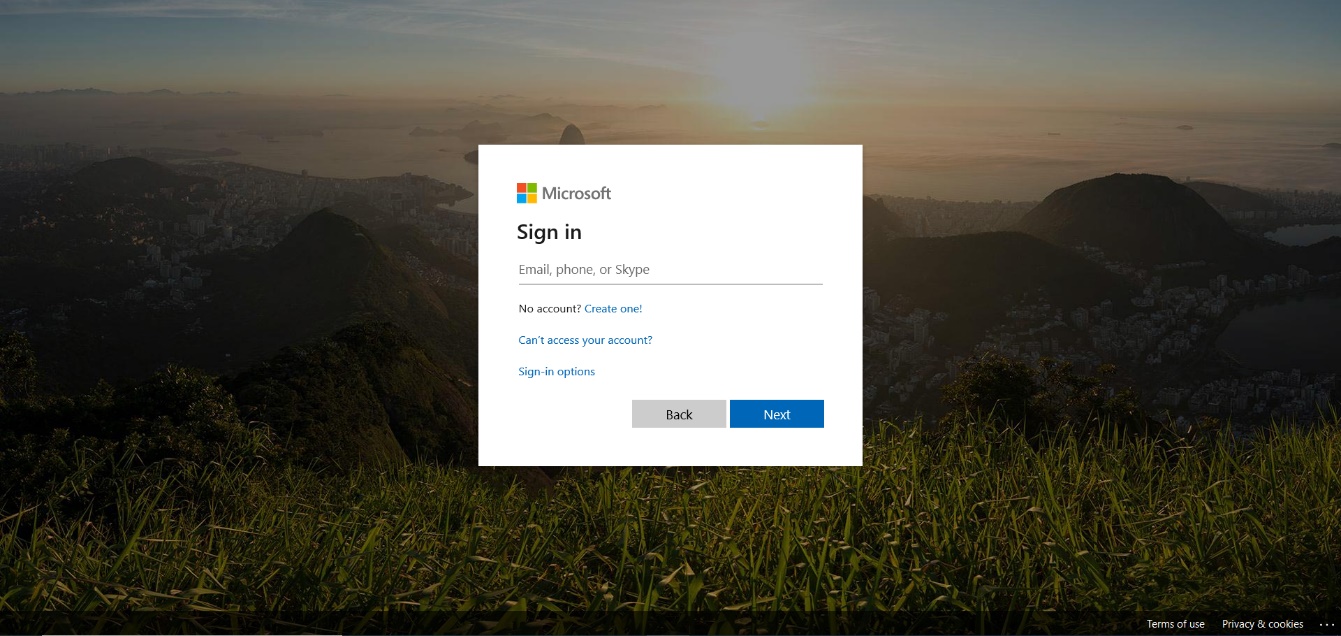
Das Prinzip hinter der Authentifizierung ist also folgendes: Ohne Anmeldung im neuen Managementsystem darf es den Lehrerinnen und Lehrern nicht möglich sein, eine Einsicht auf den Status der Konferenz zu bekommen bzw. den Ablauf steuern zu können. Die Lehrkraft wird also mehr oder weniger gezwungen sich mit ihrem MS-Konto anzumelden. Für die Anmeldung selbst wird man auf die typische Microsoft-Anmeldeseite weitergeleitet und nachdem man sich erfolgreich eingeloggt hat, wird man wieder zurück auf das Managementsystem gelenkt.

Abbildung – Microsoft Login

<https://login.microsoftonline.com/>

Der wohl größte Vorteil, der sich durch die Verwendung des Office-365-Konto ergibt, ist der, dass man nun Rollen vergeben kann. Durch eine Überprüfung der Rechte bzw. des Titels nach dem Login, lässt sich leicht feststellen, ob der Benutzer nun zur Moderator-Ansicht oder zur Nutzer-Ansicht weitergeleitet werden soll.

#### Progressive Web App

Der Grund, warum sich eine PWA hier sehr gut anbietet, ist einfach: Für Lehrkräfte ist es um einiges leichter und angenehmer, wenn sie den Status der Konferenz mühelos über ihr Mobiltelefon abrufen können. Eine PWA bietet zudem noch viel mehr Vorteile. Grundsätzlich vereint eine progressive Web App die Eigenschaften mobiler Websites und nativen Apps. Somit passt sich die Website an das Gerät an, über das sie aufgerufen wird. Bugs der Webpage können somit vorgebeugt werden.

Das bedeutet für das Managementsystem der Klassenkonferenzen, dass das Design entsprechend angepasst werden muss. Dem Benutzer soll es auch möglich sein, die Webseite wie eine native App auf dem Desktop des Smartphones zu installieren.

#### Aufrufen von Lehrern

Ein Moderator (Abteilungsvorstand) soll die Möglichkeit haben, nach Bedarf fehlende Lehrkräfte aufrufen zu können. Dieser hat in der Moderator-Ansicht eine Übersicht über alle Lehrer, die bei der aktuellen Klasse anwesend sein sollten. Wenn der Moderator nach der Anwesenheitskontrolle der Lehrkräfte feststellt, dass ein oder mehrere Personen fehlen, können diese mit einem einfachen Knopfdruck aufgerufen werden.

Bei der Lehrkraft, die aufgerufen wird, ist der Aufruf an sich eine einfache Meldung auf der Website. Dabei wird die Person höflich darauf hingewiesen, dass sie in dem Raum erwartet wird, in dem sie sein sollte. Zudem kann jeder normale Benutzer für sich selbst entscheiden, ob er Benachrichtigungen (Aufrufe) erhalten möchte oder nicht.

Text-to-Speech wird im Managementsystem als Feature implementiert, welches beim Aufrufen von Lehrkräften eine Rolle spielen soll. Sobald die ausgewählte Person aufgerufen wird, soll mittels JavaScript, die Benachrichtigung automatisch vorgelesen werden. Ziel ist dabei die Sicherstellung, dass die Lehrkraft den Aufruf auch tatsächlich wahrnimmt.

#### Design

Das Design soll nicht nur ansprechend und modern sein, die Lehrkraft soll sich vor allem in der Umgebung wohl fühlen. Die Steuerung der Konferenzen muss sich leicht anfühlen und die Daten in der Nutzer-Ansicht sollen angemessen dargestellt werden. Die Themenfarbe der Website bezieht sich auf die Farben der HTL Vöcklabruck.

# Verwendete Programme und Tools

## Bildergebnis für visual studio logoVisual Studio

Visual Studio ist eine von dem Unternehmen Microsoft angebotene integrierte Entwicklungsumgebung für verschiedene Hochsprachen.

Abbildung 12 - Visual Studio

(N.N., Wikipedia, 2020)

(N.N., Wikipedia, 2020)

## DB Browser for SQLite

DB Browser ist ein visuelles Programm, welches das Erstellen und Bearbeiten von SQLite-Datenbankdateien ermöglicht.

(N.N., SQLiteBrowser, 2020)

Abbildung 13 - SQLite DB Browser

(N.N., Twitter, 2020)

## Ein Bild, das Schild, Uhr, Zeichnung enthält. Automatisch generierte BeschreibungGit

Abbildung 14 – git

(N.N., Wikipedia, 2020)

Git ist ein kostenloses Open-Source Versionskontrollsystem, welches entwickelt wurde, um das gleichzeitige Arbeiten an einem Projekt, unabhängig vom Standort, zu ermöglichen.

Mithilfe von Git können Versionen eines Projekts gespeichert werden und bei Bedarf auch ältere Versionen abgerufen werden.

## GitHub

Abbildung 15 – GitHub

(N.N., Wikimedia Commons, 2020)

Auf GitHub können Entwickler ihre Projekte speichern und verwalten. Zudem werden die Projekte versioniert, wodurch jederzeit zu einer älteren Version zurückgestuft werden kann. Namensgebend für GitHub war das Versionsverwaltungssystem Git.

## JSON-Formatierung

JSON (JavaScript Object Notation) ist ein sprachenübergreifendes Textformat, welches den Austausch von strukturierten Daten ermöglicht. JSON basiert auf der JavaScript-Sprache und ist sehr einfach zu lesen.

Abbildung 16 - JSON

(N.N., Wikipedia, 2020)

*(N.N., Cloutcomputing-insider, 2020)*

## Bootstrap

Bootstrap ist ein freies Frontend-CSS-Framework. Es enthält auf HTML und CSS basierende Gestaltungsvorlagen für Typografie, Formulare, Buttons, Tabellen, Grid-Systeme, Navigations- und andere Oberflächengestaltungselemente sowie zusätzliche, optionale JavaScript-Erweiterungen.

Abbildung 17 – Bootstrap

(N.N., Wikipedia, 2020)

*(N.N., Wikipedia, 2020)*

## jQuery

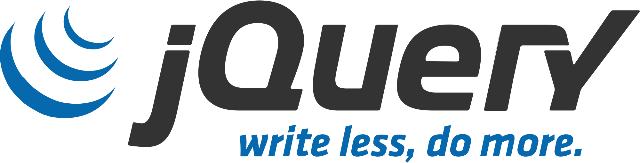
jQuery ist eine schnelle, kleine und funktionsreiche JavaScript-Bibliothek. Mit einer benutzerfreundlichen API, die in einer Vielzahl von Browsern funktioniert, werden Dinge wie das Durchlaufen und Bearbeiten von HTML-Dokumenten, die Ereignisbehandlung, die Animation und Ajax erheblich vereinfacht. Mit einer Kombination aus Vielseitigkeit und Erweiterbarkeit hat jQuery die Art und Weise verändert, wie Millionen von Menschen JavaScript schreiben.

Abbildung 18 – Jquery

(N.N., Wikipedia, 2020)

*(N.N., jQuery, 2020)*

## Razor

Razor ermöglicht serverbasierten Code in Webseiten einzubetten. Razor ist somit keine Programmiersprache, sondern es handelt sich um eine serverseitige Auszeichnungssprache.

*(N.N., Tutorialspoint, 2020)*

## NuGet-Pakete

### SQLite

SQLite ist ein Datenbanksystem, welches keinen eigenen Server benötigt, sondern lokal gespeichert wird. Das System wird vorwiegend für geringe Datenmengen verwendet und unterstützt einen Großteil der im SQL-92-Standard festgelegten SQL-Sprachbefehle.

Abbildung 19 – SQLite

(N.N., Wikipedia, 2020)

*(N.N., Wikipedia, 2020)*

### ASP.net SignalR

Abbildung 20 – SignalR

(N.N., dotnetstudio, 2020)

Ist eine Bibliothek für ASP.NET-Entwickler, die das Hinzufügen von Echtzeit Webfunktionen zu Anwendungen vereinfacht. Servercodeinhalte werden direkt an verbundene Clients übertragen, ohne dass diese die Daten erst anfordern müssen. Sie ermöglicht also das asynchrone Senden von Nachrichten an Clients.

*(N.N., Microsoft, 2020)*

### Newtonsoft.JSON

Ist eine Softwarebibliothek, welche das einfache und schnelle Serialiseren und Deserialiseren von Objekten ermöglicht. Mit über 100 Millionen Downloads (Stand: April 2020) ist diese Bibliothek die Nummer eins bei NuGet.

*(N.N., Newtonsoft, 2020)*

Abbildung 21 – Newtonsoft

(N.N., Newtonsoft, 2020)

### AzureAD

Die ASP.NET Core Azure Active Directory-Integration bietet Komponenten für die einfache Integration der Azure Active Directory-Authentifizierung in einer ASP.NET Core-Anwendung.

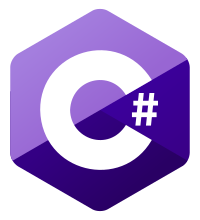
Abbildung 22 - Microsoft Azure

(N.N., Ceteris, 2020)

*(N.N., Nuget.org, 2020)*

# Sprachen

## C#

C# (ausgesprochen "See Sharp") ist eine einfache, moderne, objektorientierte und typsichere Programmiersprache. C# hat seine Wurzeln in der C-Sprachfamilie und wird C-, C++ -, Java- und JavaScript-Programmierern sofort bekannt sein.

*(N.N., Docs.Microsoft, 2020)*

Abbildung 23 - C#

(N.N., Wikipedia, 2020)

## HTML

Die Hypertext Markup Language (HTML, englisch für Hypertext-Auszeichnungssprache) ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt.

*(N.N., Wikipedia, 2020)*

Abbildung 24 – HTML

(N.N., Wikipedia, 2020)

## Bildergebnis für CsslogoCSS

Cascading Style, kurz CSS genannt, ist eine Stylesheet-Sprache für elektronische Dokumente und wird zusammen mit HTML auf fast jeder Website verwendet. Mithilfe von CSS können Schriftarten, Farben und andere Designelemente eingebunden werden

(N.N., Wikipedia, 2020)

Abbildung 25 – CSS

(N.N., Wikipedia, 2020)

## JavaScript

JavaScript (kurz JS) ist eine Skriptsprache, die für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und somit die Möglichkeiten von CSS und HTML zu erweitern.

Abbildung 26 – JavaScript

(N.N., Wikipedia, 2020)

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

Bild: <https://de.wikipedia.org/wiki/Datei:Unofficial_JavaScript_logo_2.svg>

## MySQL

MySQL ist ein frei verfügbares Open Source-Relational-Database-Management-System (RDBMS), das Structured Query Language (SQL) verwendet. MySQL ist eines der weltweit verbreitetsten relationalen Datenbankverwaltungssysteme.

Abbildung 27 – MySQL

(N.N., Wikipedia, 2020)

(N.N., Siteground, 2020)

(N.N., Wikipedia, 2020)

# Realisierung

## Datenspeicherung

Für das Speichern der allgemeinen Daten der Konferenz wurde JSON als Speicherformat ausgewählt. In einer JSON-Datei werden die Lehrer, Klassen und Räume mit den jeweils zugehörigen Daten gespeichert.

Zum Speichern des Konferenzstatus und Status der besprochenen Klassen wurde das Datenbanksystem SQLite herangezogen, da mit nur wenigen Datensätzen gearbeitet wird.

### Speicherung der Konferenzablaufdaten

Die Daten für den Ablauf der Konferenz werden in einer JSON-Datei gespeichert. Dazu wurde in dem Ordner „wwwroot“ ein neuer Ordner mit dem Namen „json“ erstellt. Darin befindet sich die JSON-Datei „conference-info.json“.

Die Datei besteht aus drei Objekten:

* Lehrer (teachers)
* Klassen (classes)
* Reihenfolge (order)

#### Lehrer

Ein Lehrer besteht aus einer ID und einem Namen. Als ID wird die E-Mail gespeichert und der Name besteht aus Nach- und Vornamen.

{

"id": "grug@htlvb.at",

"name": "Gruber Gerald"

},

Code 1 – JSON Lehrer

#### Klassen

Es werden Klassenname, Abteilungsvorstand, Klassenvorstand und die Lehrer, welche diese Klasse unterrichten, gespeichert.

Die Lehrer werden immer mit der ID abgespeichert, sodass es zu keinen Verwechslungen kommen kann.

{

"className": "3AHWII",

"headOfDepartment": "baue@htlvb.at",

"formTeacher": "abld@htlvb.at",

"teachers": [

"abld@htlvb.at",

"fram@htlvb.at",

"sacu@htlvb.at",

"soek@htlvb.at",

"grug@htlvb.at"

]

},

Code 2 - JSON Klassen

#### Reihenfolge

Die Reihenfolge besteht aus einem Raum und den Klassen. Das heißt, dass die Klassen, welche angeführt sind, in dem angegebenen Raum besprochen werden.

Die Klassen werden in der Reihenfolge besprochen, wie sie in der Datei gespeichert sind.

{

"room": "R001 (2AFMBM)",

"classes": ["2AHWII", "3AHWII", "4AHWII"]

},

Code 3 - JSON Reihenfolge

### Speicherung der Besprechungsdaten

Die Daten für den Konferenzstatus und der besprochenen Klassen werden in der Datenbankdatei „database\_conference.db“ gespeichert. Diese befindet sich im Projektordner „wwwroot/sqlite“.

#### Besprochene Klassen

Zum Abspeichern des Status, sowie der Start- und Endzeit der einzelnen Klassen wurde die Tabelle „General“ erstellt.

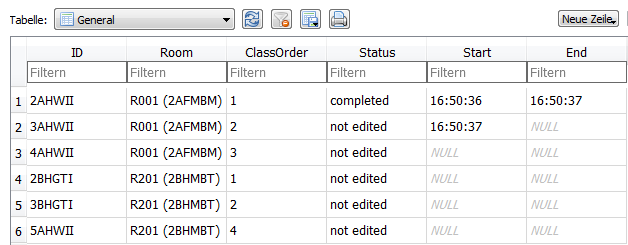


Abbildung 28 - SQLite Tabelle General

#### Status der Konferenz

In der Tabelle „State“ wird der aktuelle Konferenzstatus für die einzelnen Besprechungsräume gespeichert.

Es wird zwischen drei Werten für den Status der Konferenz unterschieden:

* Inactive – Die Konferenz wurde noch nicht gestartet.
* Running – Die Konferenz läuft gerade.
* Completed – Die Konferenz ist abgeschlossen.

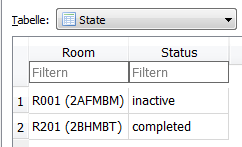


Abbildung 29 - SQLite Tabelle State

## Datensynchronisierung zwischen Clients

Da zwei Klassen gleichzeitig besprochen werden können, sind auch bis zu zwei Clients gleichzeitig auf der Moderator-Seite angemeldet. Um den Austausch von Informationen zwischen den Clients zu ermöglichen, kommt die Softwarebibliothek SignalR zum Einsatz.

Die Bibliothek ermöglicht es Informationen, wie Überschneidungen, stets aktuell an den anderen Client zu übergeben. Alle Daten der Anwendung werden mit SignalR gesendet.

#### Hinzufügen von SignalR

Solution Explorer 🡪 Rechtsklick auf Projekt 🡪 Add 🡪 Client-Side Library 🡪 Install

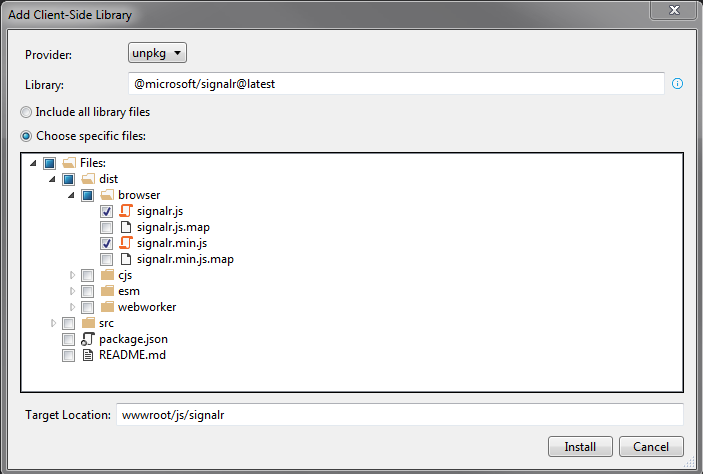


Abbildung 30 - Visual Studio Paket hinzufügen

Nachdem die Bibliothek installiert wurde, musste eine neue Klasse erstellt werden.

Diese Klasse wurde in dem Ordner „Hubs“ erstellt und trägt den Namen „mainHub“. Hier werden alle asynchronen Methoden, welche mit JavaScript kommunizieren, implementiert.

#### Einstellungen in der „Startup.cs“ Datei

In der Methode „ConfigureServices“ muss SignalR noch hinzugefügt werden. Dies geschieht mit „services.AddSignalR();“ am Ende der Methode.

Zum Schluss muss noch die Route zur Hub-Klasse festgelegt werden. Dazu wird in der Methode „Configure“ die Route folgend definiert:

app.UseSignalR(routes =>

{

routes.MapHub<MainHub>("/mainHub");

});

Code 4 - Configure Hub Route

#### JavaScript Implementierung

Nachdem die Hub-Klasse konfiguriert wurde, musste eine JavaScript-Datei erstellt werden. Diese JavaScript-Datei verwaltet die Kommunikation zum Hub.

Es wurden zwei JavaScript -Dateien im Ordner „wwwroot/js“ erstellt:

* Moderator\_view.js
* User\_view.js

Damit die HTML-Seite mit JavaScript kommunizieren und so SignalR verwenden kann, müssen die beiden Dateien jeweils in HTML verlinkt werden.

<script src="~/lib/signalr/dist/browser/signalr.js"></script>

<script src="~/js/moderator\_view.js"></script>

Code 5 - HTML moderator\_view.js script tag

<script src="~/lib/signalr/dist/browser/signalr.js"></script>

<script src="~/js/user\_view.js"></script>

Code 6 - HTML user\_view.js script tag

## Modellklassen

Um die erhaltenen JSON-Daten in ein Objekt umzuwandeln, wurden sogenannte Modellklassen erstellt.

Newtonsoft kann erhaltene JSON-Daten sofort in Objekte umwandeln. Um dies zu ermöglichen müssen der Klassenname und die zugehörigen Eigenschaften mit jenen aus den JSON-Daten übereinstimmen.

### Lehrer

public class Teacher

{

public string ID { get; set; }

public string Name { get; set; }

}

Code 7 - Modellklasse Teacher

### Klasse

public class MyClasses

{

public string ClassName { get; set; }

public string HeadOfDepartment { get; set;}

public string FormTeacher { get; set; }

public List<string> Teachers { get; set; }

}

Code 8 - Modellklasse MyClasses

### Reihenfolge

public class Order

{

public string Room { get; set; }

public List<string> Classes { get; set; }

}

Code 9 - Modellklasse Order

## Klassen

In diesem Kapitel werden die erstellten Klassen beschrieben. Diese Klassen ermöglichten sowohl eine Vermeidung von Code, als auch leichte Veränderungen von Variablen im Programm.

### Allgemeine Informationen

Zu Beginn wurde die Klasse „General“ in dem Ordner „Classes“ erstellt. Hier werden alle allgemeinen Informationen gespeichert.

Dazu zählen:

* Tabellennamen der Datenbank
* Pfadangaben innerhalb des Projekts
* JSON-Dateiinhalt in Form eines Strings

#### Pfadangaben

Um den Zugriff auf die Datenbank und die JSON-Datei zu ermöglichen, wurden die Pfade in Eigenschaften integriert.

private string path\_json;

private string path\_DB;

private string Path\_DB

{

get

{

if (path\_DB == null)

{

string root = "wwwroot";

string location = "sqlite";

string fileName = "database\_conference.db";

path\_DB = Path.Combine(

Directory.GetCurrentDirectory(),

root,

location,

fileName);

}

return path\_DB;

}

}

Code 10 –General Datenbank Pfadangabe

public string Path\_Json

{

get

{

if (path\_json == null)

{

string root = "wwwroot";

string location = "json";

string fileName = "conference-info.json";

path\_json = Path.Combine(

Directory.GetCurrentDirectory(),

root,

location,

fileName);

}

return path\_json;

}

}

Code 11 – General Pfad zur JSON-Datei

#### Tabellennamen

Die Tabellennamen wurden als statische Variablen deklariert und können mittels Eigenschaften aufgerufen werden.

private static readonly string tableNameGeneral = "General";

public string Table\_General

{

get

{

return tableNameGeneral;

}

}

Code 12 – General Tabellenname Eigenschaft

#### JSON-Dateiinhalt in Form eines Strings

Zum Lesen der einzelnen Objekte aus JSON wird die Datei in einen String umgewandelt. Hierbei ist es wichtig als Kodierung UTF-8 zu wählen, um Umlaute problemlos lesen zu können.

public string JsonString

{

get

{

if (jsonstring == null)

{

FileStream fileStream = new FileStream(Path\_Json, FileMode.Open);

StreamReader reader = new StreamReader(fileStream, Encoding.UTF8);

using (reader)

{

jsonstring += JsonConvert.DeserializeObject(reader.ReadToEnd());

}

}

return jsonstring;

}

}

Code 13 – General JSON Daten als String

### Datenbankverbindung

Da ständig Methoden zum Lesen und Schreiben von Datenbankinformationen verwendet werden, war es von Anfang an klar, dafür eine eigene Klasse zu implementieren.

In der Klasse wurden zwei Methoden implementiert:

* Query – zum Ausführen von Befehlen wie Update, Insert oder Delete
* Reader – zum Lesen von Datensätzen

Bei beiden Methoden war es wichtig, diese gegen SQL-Injections zu schützen. Daher werden die Parameter erst später in den SQL-Befehl eingesetzt.

*„SQL-Injection (dt. SQL-Einschleusung) ist das Ausnutzen einer Sicherheitslücke in Zusammenhang mit SQL-Datenbanken, die durch mangelnde Maskierung oder Überprüfung von Metazeichen in Benutzereingaben entsteht.“* (N.N., Wikipedia, 2020)

Eine parametrisierte Abfrage zum Aktualisieren des Zustandes einer Klasse könnte also folgendermaßen aussehen:

dB.Query($"UPDATE general SET status='completed' WHERE id=?",GetCurrentClassName());

Code 14 - Codebeispiel Entgegenwirken von SQL-Injections

#### Schreiben in die Datenbank

Zur Kommunikation mit der Datenbank wurden zwei Hilfsmethoden implementiert. Diese ermöglichen das Schreiben und Lesen der Daten aus der Datenbank.

Die Methoden „Query“ und „Reader“ kümmern sich um den Datenaustausch mit der Datenbank.

public int Query(string sqlstring, params object[] parametervalues)

{

using (var connection = new SQLiteConnection($"Data Source={general.PathDB}"))

{

connection.Open();

using (var command = new SQLiteCommand(sqlstring, connection))

{

command.CommandText = sqlstring;

if(parametervalues != null)

{

foreach (var param in parametervalues)

{

command.Parameters.Add(new SQLiteParameter() { Value = param });

}

}

return command.ExecuteNonQuery();

}

}

}

Code 15 – Methode Query

Die Methode setzt die Parameter in den SQL-Befehl ein und führt diesen aus. Zurückgegeben wird die Anzahl an betroffenen Zeilen.

public DataTable Reader(string sqlstring, params object[] parametervalues)

{

DataTable dt = new DataTable();

using (var connection = new SQLiteConnection($"Data Source={general.PathDB}"))

{

connection.Open();

using (var command = new SQLiteCommand(connection))

{

command.CommandText = sqlstring;

if (parametervalues != null)

{

foreach (var param in parametervalues)

{

command.Parameters.Add(new SQLiteParameter() { Value = param });

}

}

using (SQLiteDataReader reader = command.ExecuteReader())

{

dt.Load(reader);

}

}

}

return dt;

}

Code 16 – Methode Reader

Gleich wie bei „Query“ werden hier zuerst die Parameter eingesetzt. Anschließend wird der Lesebefehl ausgeführt wobei hier die Daten in einem DataTable gespeichert werden. Bei Abschluss des Lesevorgangs wird der DataTable zurückgegeben.

## Raumauswahl

Ein Bild, das Schild, Pol, Zeichnung, blau enthält.

Automatisch generierte BeschreibungJeder Klassenraum in der Schule hat eine Raumbezeichnung und eine Klasse, welche dauerhaft in diesem Raum ist. Bevor der Moderator eine Ansicht der Konferenz erhält, muss er auswählen, in welcher Klasse er moderieren möchte.

Die Seitenansicht enthält zwei Knöpfe, welche zu der jeweiligen Ansicht weiterleiten.

Dazu wurde eine neue Razor Page im Ordner „Pages“ mit dem Namen „RoomSelection“ erstellt.

Abbildung 31 - Raumansicht

Die Klassenräume werden aus der JSON-Datei gelesen und zu einer Liste hinzugefügt. Diese Liste der Räume wird in einer Eigenschaft gespeichert. Die Razor Syntax ermöglicht eine „foreach“-Anweisung im HTML Code, welche eine einfache Ausgabe einer Liste ermöglicht.

@foreach (var order in Model.OrderList)

{

<**form** name="form" id="form" method="post" **asp-route-room**="@order.Room\_only">

<input type="submit" class="btn btn-info" value="@order.Room">

</**form**>

}

Code 17 - Raumausgabe in HTML

Nach Drücken eines beliebigen Knopfes wird man auf die jeweilige Seite weitergeleitet. Der Raum wird als Parameter in der URL an die Seite übergeben.

public IActionResult OnPost(string room)

{

return new RedirectToPageResult("Moderator", room);

}

Code 18 - Weiterleitung auf Moderator-Seite

## Steuerung der Konferenz

Abbildung 32 - Moderator Seite

Bei dieser Ansicht kann der Moderator die Konferenz steuern. Es werden Informationen zur aktuellen Klasse, die Lehrer und Überschneidungen angezeigt.

Wenn ein Lehrer nicht erscheint, kann dieser über den Knopf „ausrufen“ benachrichtigt werden. Die Steuerung der Konferenz gelingt über den Steuerungsbutton auf der Seite.

Bei dieser Ansicht kommt das erste Mal SignalR zum Einsatz. Das Gerüst der Seite besteht aus HTML-Code, in welchen mithilfe von JavaScript die gewünschten Informationen eingefügt werden.

### Laden der Seite

Wenn die Seite geladen oder neu gestartet wird, wird in der JavaScript-Datei „moderator\_view.js“ versucht eine Verbindung zum Hub herzustellen. Der Button zur Steuerung der Konferenz wird hier vorerst deaktiviert, sodass es zu keinem Klick vor einer hergestellten Verbindung kommen kann.

var connection = new signalR.HubConnectionBuilder().withUrl("/mainHub").build();

//Disable send button until connection is established

document.getElementById("sendButton").disabled = true;.

Code 19 - Verbindungsaufaufbau zum MainHub

Wenn die Verbindung steht (gekennzeichnet durch „connection.start().then…“) wird das HTML-Element mit dem Namen „room“ gesucht. In dieses Element wird anschließend der an die Seite übergebene Parameter für den Raum geschrieben.

connection.start().then(function () {

var currentroom = GetCurrentRoom();

General.WriteInElement("room", currentroom);

document.getElementById("sendButton").disabled = false;

connection.invoke("LoadModeratorPage", currentroom).catch(function (err) {

return console.error(err.toString());

});

}).catch(function (err) {

return console.error(err.toString());

});

Code 20 - Stehende Verbindung zum MainHub

Die Methode „GetCurrentRoom“ liest den mitgegebenen Wert des Raums aus der URL und gibt diesen als String zurück.

function GetCurrentRoom() {

return new URLSearchParams(window.location.search).get("handler");

}

Code 21 – JavaScript Methode GetCurrentRoom

### Informationsaufruf

Durch das „connection.invoke“ wird die asynchrone Hub-Methode „LoadModeratorPage()“ aufgerufen und der aktuelle Raum als Parameter übergeben.

In der Klasse „mainHub“ wird zuerst der aktuelle Raum in die Eigenschaft gespeichert. Anschließend werden die Daten für die verschiedenen Clients geladen.

public async Task LoadModeratorPage(string \_currentroom)

{

Currentroom = \_currentroom;

await LoadModeratorContent();

await LoadGeneralContent();

await LoadIntersections();

}

Code 22 - Methode LoadModeratorPage

### Moderator Informationen

Die Methode „LoadModeratorContent“ kümmert sich um das Laden der spezifischen Daten für den Moderator Client. Dazu zählen der Text des Steuerungsbuttons und die zugehörigen Lehrer der Klasse.

Zuerst wird ein neues JSON-Objekt erstellt. Diesem werden dann die Lehrer, die Überschneidungen, der aktuelle Raum und der Text für den Button hinzugefügt. Falls die Konferenz jedoch schon abgeschlossen ist, wird für die Lehrer nur ein leerer String hinzugefügt.

Die Informationen werden nur an jenen Client zurückgegeben, welcher die Methode aufgerufen hat.

public async Task LoadModeratorContent()

{

JObject content = new JObject();

var currentClassTeachers = new List<Teacher>();

var teachersString = string.Empty;

if (GetCurrentStateOfConference() != "completed")

{

var currentClassTeacherIds = GetClass(GetCurrentClassName()).Teachers;

foreach(var teacher in currentClassTeacherIds)

{

currentClassTeachers.Add(GetTeacher(teacher));

teachersString = JsonConvert.SerializeObject(currentClassTeachers);

}

}

content.Add("teachers", teachersString);

content.Add("buttonText", GetButtonText());

await Clients.Caller.SendAsync("ReceiveModeratorContent", content.ToString());

}

Code 23 – MainHub Methode LoadModeratorContent

Da meist nur mit den IDs der Lehrer gearbeitet wird, wurde eine eigene Methode erstellt, um das gesamte Lehrerobjekt zu erhalten. Man übergibt die ID und bekommt ein Teacher-Objekt mit ID und Name zurück.

private Teacher GetTeacher(string teacherId)

{

JObject jobject = JObject.Parse(general.JsonString);

JArray jTeachers = (JArray)jobject["teachers"];

var teachers = jTeachers.ToObject<List<Teacher>>();

return teachers.Find(teacher => teacher.ID == teacherId);

}

Code 24 – Methode GetTeacher

#### Text des Steuerungsbuttons

Durch die Methode „GetButtonText“ erhält man den Text für den Button zur Steuerung der Konferenz. Da sich der Anfangstext von „Konferenz starten“ auf einen anderen Wert ändert, muss dieser den entsprechend zutreffenden Text erhalten.

* „Konferenz starten“ – Die Konferenz wurde noch nicht gestartet.
* „Nächste Klasse“ – Die Konferenz läuft gerade.
* „Konferenz abgeschlossen“ – Die Konferenz wurde abgeschlossen.

private string GetButtonText()

{

switch (GetCurrentStateOfConference())

{

case "inactive":

text\_Conference\_State = "Konferenz starten";

break;

case "running":

text\_Conference\_State = "Nächste Klasse";

break;

case "completed":

text\_Conference\_State = "Konferenz abgeschlossen";

break;

}

return text\_Conference\_State;

}

Code 25 – Methode GetButtonText

#### Laden der Informationen in JavaScript

In JavaScript werden die Lehrer in eine Tabelle und der Text in den Button geschrieben.

connection.on("ReceiveModeratorContent", function (obj) {

var obj\_parsed = JSON.parse(obj);

document.getElementById("sendButton").value = obj\_parsed.buttonText;

WriteTeachersWithButtonsInTable(obj\_parsed.teachers);

});

Code 26 – JavaScript ReceiveModeratorContent

Mit JQuery wird zuerst die gesuchte Tabelle geleert. Wenn der übergeben Parameter leer ist, wird eine neue Tabellenreihe mit dem Wert „Keine Lehrer“ erstellt.

Andernfalls wird der übergeben Parameter geparst und für jedes Element eine neue Tabellenreihe mit einem Button erstellt. Der Button erhält als „onclick“-Attribut die Methode „callTeacher“ mit dem jeweiligen Index zugewiesen.

function WriteTeachersWithButtonsInTable(teacherArray) {

$("#teachers").empty(); //clear the table

if (teacherArray == "") {

$("#teachers").append("<tr><td>Keine Lehrer</td></tr>");

}

else {

var teacherData, buttonData;

var parsedArray = JSON.parse(teacherArray);

for (var i = 0; i < parsedArray.length; i++) {

var teacherID = parsedArray[i].ID;

var fullName = parsedArray[i].Name;

teacherData = "<td><p title='" + fullName + "'>" + getShorthandForTeacher(teacherID) + "</p></td>";

buttonData = "<td><button onclick='callTeacher(" + i + ")' class='btn btn-secundary' style='margin: 6px;'>ausrufen</button></td>";

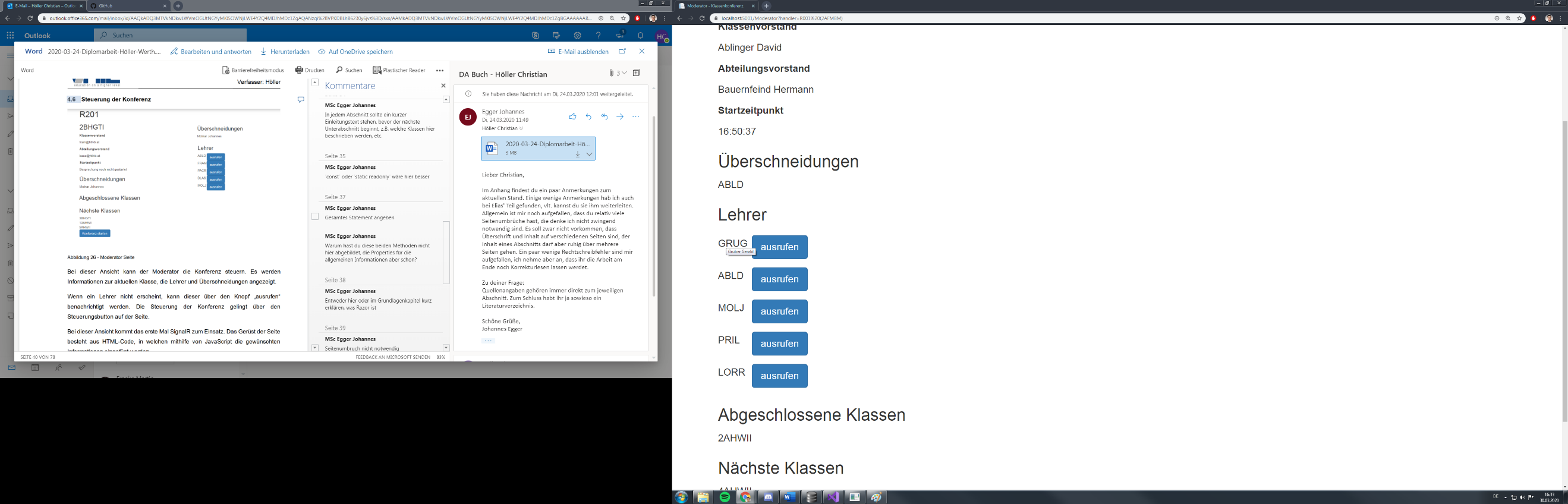
$("#teachers").append("<tr>" + teacherData + buttonData + "</tr>");

}

}

}

Code 27 – JavaScript WriteTeachersWithButtonsIntable

Zusätzlich wird im title-Attribut der vollständige Name des jeweiligen Lehrers angezeigt. Sobald man den Mauscursor auf dem gewünschten Lehrer positioniert, wird der gesamte Name angezeigt

Code 28 – Lehrer title-attribut

Die Methode „getShorthandForTeacher“ liefert das Lehrerkürzel. Hierbei werden von der ID nur die ersten vier Buchstaben genommen. Dieses Kürzel wird in Großbuchstaben zurückgegeben.

function getShorthandForTeacher(teacherID) {

return teacherID.split("@")[0].toUpperCase();

}

Code 29 – JavaScript Methode getShorthandForTeacher

### Allgemeine Informationen

In der Methode „LoadGeneralContent“ wird zuerst ein neues JSON-Objekt erstellt. Anschließend wird die aktuelle Klasse geladen, um zu bestimmen, welche Informationen geladen werden sollen.

Die Daten werden entsprechend dem Konferenzstatus geladen, da je nach Status andere Informationen angezeigt werden müssen.

public async Task LoadGeneralContent()

{

JObject information = new JObject();

var currentClass = GetClass(GetCurrentClassName());

string headOfDepartment = string.Empty;

string formTeacher = string.Empty;

switch (GetCurrentStateOfConference())

{

case "inactive":

formTeacher = JsonConvert.SerializeObject(GetTeacher(currentClass.FormTeacher));

headOfDepartment = JsonConvert.SerializeObject(GetTeacher(currentClass.HeadOfDepartment));

information.Add("room", Currentroom);

information.Add("classname", GetCurrentClassName());

information.Add("formTeacher", formTeacher);

information.Add("headOfDepartment", headOfDepartment);

information.Add("time", "Besprechung noch nicht gestartet");

break;

case "running":

DataTable dt = dB.Reader($"SELECT room, start FROM {general.Table\_General} WHERE ID = ? limit 1", GetCurrentClassName());

formTeacher = JsonConvert.SerializeObject(GetTeacher(currentClass.FormTeacher));

headOfDepartment = JsonConvert.SerializeObject(GetTeacher(currentClass.HeadOfDepartment));

information.Add("room", dt.Rows[0]["room"].ToString());

information.Add("time", dt.Rows[0]["start"].ToString());

information.Add("classname", GetCurrentClassName());

information.Add("formTeacher", formTeacher);

information.Add("headOfDepartment", headOfDepartment);

break;

case "completed":

information.Add("room", Currentroom);

information.Add("classname", "Alle Klassen abgeschlossen");

information.Add("formTeacher", JsonConvert.SerializeObject(new Teacher() { ID="-", Name="-"}));

information.Add("headOfDepartment", JsonConvert.SerializeObject(new Teacher() { ID = "-", Name = "-" }));

information.Add("time", "-");

break;

}

information.Add(new JProperty("classesCompleted", GetClassesCompleted()));

information.Add(new JProperty("classesNotEdited", GetClassesNotEdited()));

await Clients.All.SendAsync("ReceiveGeneralContent", information.ToString());

}

Code 30 - Hub Methode LoadGeneralContent Switch

Die abgeschlossenen und nächsten Klassen werden immer, unabhängig vom Konferenzstatus, zurückgegeben.

Letztendlich werden die Daten als JSON-String an JavaScript zurückgegeben.

information.Add(new JProperty("classesCompleted", GetClassesCompleted()));

information.Add(new JProperty("classesNotEdited", GetClassesNotEdited()));

await Clients.All.SendAsync("ReceiveGeneralContent", information.ToString());

}

Code 31 - Hub Methode LoadGeneralContent Klassen

#### Abgeschlossene Klassen

Zuerst wird überprüft, ob die Konferenz bereits abgeschlossen ist. Wenn dem so ist, werden alle Klassen, welche in der JSON-Order stehen, als JSON-String zurückgegeben.

Ansonsten wird die Position (Index) der aktuellen Klasse in der Liste der Reihenfolge bestimmt. Anschließend werden die Klassen bis zu dieser Position in ein JSON-Array geschrieben und dann als JSON-String zurückgegeben.

private string GetClassesCompleted()

{

var classes = GetOrderList().Find(order => order.Room == Currentroom).Classes;

if (GetCurrentStateOfConference() == "completed")

{

return new JArray(classes).ToString();

}

else

{

int index = classes.IndexOf(GetCurrentClassName());

return new JArray(classes.Take(index)).ToString();

}

}

Code 32 – Methode GetClassesCompleted

#### Nächste- Klassen

Diese Methode funktioniert ähnlich, nur dass die Klassen nach dem Index zurückgegeben werden.

Wenn die Konferenz abgeschlossen ist, oder die Informationen für die letzte Klasse in der Order-Liste geladen werden, wird dem JSON Array „Keine weiteren Klassen“ hinzugefügt und als JSON-String zurückgegeben.

private string GetClassesNotEdited()

{

JArray classesNotedited = new JArray();

var classes = GetOrderList().Find(order => order.Room == Currentroom).Classes;

if (GetCurrentStateOfConference() == "completed")

{

classesNotedited.Add("Keine weiteren Klassen");

}

else

{

int index = classes.IndexOf(GetCurrentClassName()) + 1; //the current class shouldn't be displayed

if (index == classes.Count)

{

classesNotedited.Add("Keine weiteren Klassen");

}

else

{

classesNotedited = new JArray(classes.GetRange(index, classes.Count - index));

}

}

return classesNotedited.ToString();

}

Code 33 – Methode GetClassesNotEdited

Da die Informationen an JavaScript in Form von JSON übergeben wird, muss dieser erst geparst werden. Anschließend werden die Informationen mithilfe von JQuery in das entsprechende Element geschrieben.

Die Werte für die abgeschlossenen und nächsten Klassen werden durch „WriteDataInTable“ der entsprechenden Tabelle zugeordnet.

connection.on("ReceiveGeneralContent", function (obj) {

var obj\_parsed = JSON.parse(obj);

if (obj\_parsed.room == GetCurrentRoom()) {

var formTeacher\_parsed = JSON.parse(obj\_parsed.formTeacher);

var headOfDepartment\_parsed = JSON.parse(obj\_parsed.headOfDepartment);

$("#classname").html(obj\_parsed.classname);

$("#formTeacher").html(formTeacher\_parsed.Name);

$("#headOfDepartment").html(headOfDepartment\_parsed.Name);

$("#time").html(obj\_parsed.time);

console.log(obj\_parsed.classesNotEdited);

WriteDataInTable("classesCompleted", JSON.parse(obj\_parsed.classesCompleted));

WriteDataInTable("classesNotEdited", JSON.parse(obj\_parsed.classesNotEdited));

}

});Code 34 – JavaScript ReceiveGeneralContent

Die Methode „WriteDataInTable” verwendet JQuery und löscht zuerst die vorhandenen Daten in der HTML-Tabelle. Anschließend können an die leere Tabelle die neuen Werte angehängt werden.

function WriteDataInTable(tablename, jsonArray) {

$("#" + tablename).empty();

var parsedArray = JSON.parse(jsonArray);

for (var i = 0; i < parsedArray.length; i++) {

$("#" + tablename).append("<tr><td>" + parsedArray[i] + "</td></tr>")

}

}

Code 35 – JavaScript Methode WriteDataInTable

### Überschneidungen

In der Methode „LoadIntersections“ werden die Überschneidungen, also jene Lehrer, welche zu Klassen in beiden Konferenzräumen zugehörig sind, geladen.

Abbildung 33 - Überschneidungen

Zuerst wird überprüft ob die Konferenz in einer der beiden Räume bereits abgeschlossen ist. Wenn ja, wird lediglich „Keine Überschneidungen“ in einem JSON-String zurückgegeben.

JArray jArrayIntersections = new JArray();

DataTable dt = dB.Reader($"Select ID from {general.Table\_General} WHERE Status='not edited' AND Room <> ? order by ClassOrder limit 1", Currentroom);

bool isOtherConferenceFinished = dt.Rows.Count == 0 ? true : false;

if (GetCurrentStateOfConference() == "completed" || isOtherConferenceFinished)

{

jArrayIntersections.Add("Keine Überschneidungen");

}

Code 36 – Überschneidungen Konferenzstatusüberprüfung

Wenn dies jedoch nicht der Fall ist, werden die Lehrer der parallellaufenden Klasse und die Lehrer der in diesem Raum besprochenen Klasse je in eine Liste geschrieben. Anschließend werden mithilfe des Linq-Befehls „Intersect“ die gemeinsamen Elemente in eine neue Liste geschrieben. Durch die Methode „GetTeacher“ wird für jede ID der Lehrer in eine Liste geschrieben.

string otherclassname = dt.Rows[0]["ID"].ToString();

MyClasses currentClass = GetClass(GetCurrentClassName());

MyClasses otherclass = GetClass(otherclassname);

List<string> otherClassTeacherIDs = otherclass.Teachers.Select(teacher => GetTeacher(teacher).ID).ToList();

List<string> currentClassTeacherIDs = currentClass.Teachers.Select(teacher => GetTeacher(teacher).ID).ToList();

List<string> intersectionIDs = otherClassTeacherIDs.Intersect(currentClassTeacherIDs).ToList();

List<Teacher> intersectedTeachers = intersectionIDs.Select(intersection => GetTeacher(intersection)).ToList();

jArrayIntersections = new JArray(JsonConvert.SerializeObject(intersectedTeachers));

Code 37 – Überschneidungen gemeinsame Lehrer

Zum Schluss wird die Liste in ein JSON-Array serialisiert und an alle Moderator-Clients zurückgegeben.

#### Laden der Informationen in JavaScript

Wenn die Clients die Nachricht erhalten, werden die Daten mithilfe der Methode „WriteIntersectionsInTable“ in die zugehörige Tabelle geschrieben. Wenn das Objekt an der Stelle null jedoch leer ist, wird mithilfe der Methode „WriteDataInTable“ eine entsprechende Meldung in die Tabelle geschrieben.

connection.on("ReveiveIntersections", function (obj) {

if (obj[0] == "") {

console.log("wazne");

var intersections = new Array();

intersections.push("Keine Überschneidungen");

WriteDataInTable("intersections", intersections)

}

else {

WriteIntersectionsInTable(obj);

}

});

Code 38 – JavaScript ReceiveIntersections

function WriteIntersectionsInTable(intersectionObject) {

$("#intersections").empty();

var intersections = JSON.parse(intersectionObject);

for (var i = 0; i < intersections.length; i++) {

$("#intersections").append("<tr><td><p title='" + intersections[i].Name + "'>" + getShorthandForTeacher(intersections[i].ID) + "</p></td></tr>");

}

}

Code 39 – JavaScript Methode WriteIntersectionsInTable

Diese Methode schreibt die Lehrer als Kürzel in die Überschneidungen.

### Steuern der Konferenz

Die Konferenz kann mit dem Button „sendButton“ gesteuert werden. Sobald die Konferenz einmal gestartet wurde, wird durch Drücken des Knopfes gleich die Besprechung für die nächste Klasse gestartet.

Die Hub-Methode „ConferenceAction“ aus der Klasse „mainHub“ wird aufgerufen, sobald der Knopf gedrückt wird. Hier wird der aktuelle Raum als Parameter übergeben.

document.getElementById("sendButton").addEventListener("click", function (event) {

connection.invoke("ConferenceAction", GetCurrentRoom()).catch(function (err) {

return console.error(err.toString());

});

event.preventDefault();

});

Code 40 – JavaScript Button-event

Wenn die Konferenz noch nicht gestartet wurde, wird die Methode „StartConference“ aufgerufen, ansonsten „NextClass“.

Anschließend werden die Informationen für den Moderator und die Informationen für den normalen Nutzer geladen.

public async Task ConferenceAction(string \_currentroom)

{

Currentroom = \_currentroom;

switch (GetCurrentStateOfConference())

{

case "inactive":

StartConference();

break;

case "running":

NextClass();

break;

}

await LoadModeratorPage(\_currentroom);

}

Code 41 – MainHub ConferenceAction

#### StartConference

In der Methode „StartConference“ wird zuerst die aktuelle Zeit in die Datenbank geschrieben. Dies geschieht über die Methode „WriteTimeInDatabase“.

Der übergebene Parameter „time“ wird in den SQL-Befehl eingesetzt. Zusätzlich wird noch der aktuelle Klassenname ermittelt, wodurch der Befehl vervollständigt wird.

private void WriteTimeInDatabase(string time)

{

DateTime date = DateTime.Now;

string timeonly = date.ToLongTimeString();

DB.Query($"UPDATE {General.Table\_General} set {time} = ? WHERE ID = ?", timeonly, GetCurrentClassName()) ;

}

Code 42 – MainHub WriteTimeInDatabase

Anschließend wird der Status der Konferenz auf „running“ gesetzt. Damit wird signalisiert, dass die Konferenz läuft.

Dies geschieht über die Methode „SetStateOfConference“, welche in die Datenbanktabelle „General“ den Status schreibt.

private void SetStateOfConference(string status)

{

DB.Query($"Update {General.TableStateOfConference} set Status = ? where Room = ?", status, Currentroom);

}

Code 43 – MainHub SetStateOfConference

#### NextClass

Zuerst wird der Abschlusszeitpunkt der aktuellen Klasse in die Datenbank geschrieben. Danach wird der Status der Klasse von „not edited“ auf „completed“, also auf abgeschlossen, gesetzt.

Falls keine neue Klasse mehr geladen wird, ist die Konferenz abgeschlossen. Wenn dies der Fall ist, wird der Status der Konferenz durch die Methode „SetStateOfConference“ auf „completed“ gesetzt und signalisiert somit den Abschluss der Konferenz in jenem Raum.

Wenn noch nicht alle Klassen abgeschlossen wurden, wird gleich die Startzeit für die nächste Klasse mit „WriteTimeInDataBase“ in die Datenbank geschrieben.

private void NextClass()

{

WriteTimeInDatabase("end");

DB.Query($"UPDATE {General.Table\_General} set Status='completed' WHERE ID = ?", GetCurrentClassName());

if (GetCurrentClassName() == null)

{

SetStateOfConference("completed");

}

else

{

WriteTimeInDatabase("start");

}

}

Code 44 – MainHub NextClass

## Nutzeransicht

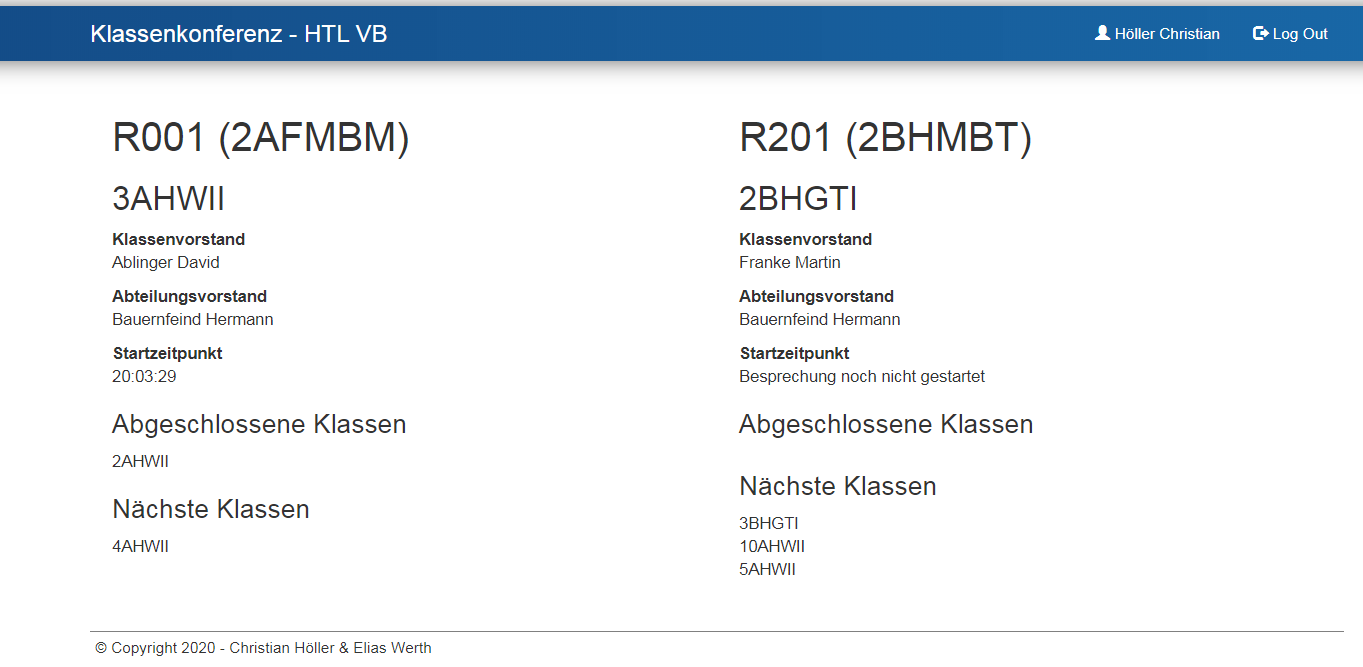
Auf der Nutzeransicht werden Informationen über die Konferenz in beiden Räumen angezeigt. Diese Ansicht kann von allen Lehrern aufgerufen werden und soll später auch auf der Leinwand in der Aula angezeigt werden.

Abbildung 34 - Nutzeransicht

### Laden der Räume

Sobald in der JavaScript-Datei „user\_view.js“ eine Verbindung mit dem Hub besteht, wird die Methode „LoadRooms“ aus der Klasse „mainHub“ aufgerufen.

connection.start().then(function () {

connection.invoke("LoadRooms").catch(function (err) {

return console.error(err.toString());

});

}).catch(function (err) {

return console.error(err.toString());

});

Code 45 – JavaScript Verbindungsaufbau zum MainHub

In der Klasse „mainHub.cs“ werden die beiden Order-Objekte durch die Methode „GetOrderList“ geladen. Diese Objekte werden zu einer Liste konvertiert und in Form eines JSON-Strings an den Client zurückgegeben.

public async Task LoadRooms()

{

JArray jOrder = new JArray(GetOrderList().Select(order => order.Room));

await Clients.All.SendAsync("ReceiveRooms", jOrder.ToString());

}

Code 46 – MainHub LoadRooms

In JavaScript wird die Liste geparst und die Räume mittels JQuery in das entsprechende HTML-Element geschrieben.

Anschließend werden mittels einer Schleife für jeden Raum die Informationen geladen. Dies geschieht über die Methode „LoadUserViewInfo“, welche den Raum als Parameter übergeben bekommt.

connection.on("ReceiveRooms", function (order) {

var order\_parsed = JSON.parse(order);

$("#c1\_room").html(order\_parsed[0]);

$("#c2\_room").html(order\_parsed[1]);

for (var i = 0; i < order\_parsed.length; i++) {

connection.invoke("LoadUserPageContent", order\_parsed[i]).catch(function (err) {

return console.error(err.toString());

});

}

});

Code 47 – JavaScript ReceiveRooms

### Laden der Informationen

Das Laden der Informationen funktioniert gleich wie bei der Moderator-Ansicht. Hier unterscheidet sich jedoch der Erhalt der Daten, da hier bestimmt wird, für welche Klasse die Informationen in HTML geschrieben werden.

Sobald das Objekt erhalten wurde, wird dieses getrennt und kann nun verwendet werden.

Um zu verhindern, dass die Informationen für den falschen Raum eingetragen werden, wird der Raum überprüft.

Da die Raumnamen bereits in HTML eingetragen wurden, wird einer dieser beiden Räume, mit dem Raumnamen des übergebenen Objekts verglichen. Wenn diese gleich sind, werden die Informationen für diesen Raum eingetragen. Ansonsten werden die Informationen für den anderen Raum geschrieben.

connection.on("ReceiveGeneralContent", function (myobject) {

var obj\_parsed = JSON.parse(myobject);

if (document.getElementById("c1\_room").innerHTML == obj\_parsed.room) {

WriteUserViewInformation("c1\_", obj\_parsed);

}

else {

WriteUserViewInformation("c2\_", obj\_parsed);

}

});

Code 48 – JavaScript ReceiveGeneralContent

Zu guter Letzt wird die Startsequenz und das Objekt an die Methode „WriteUserViewInformation“ übergeben. Die Startsequenz kennzeichnet, um welchen Raum es sich handelt. Wenn das Kürzel „r1\_“ ist, werden die Informationen für alle Objekte mit der Startsequenz „r1\_“ geschrieben.

Hier kommt wieder die Methode „WriteInElement“ zum Einsatz. Es werden die Werte für alle HTML-Elemente mit der entsprechenden Startsequenz gesetzt.

function WriteUserViewInformation(element, obj) {

var formTeacher\_parsed = JSON.parse(obj.formTeacher);

var headOfDepartment\_parsed = JSON.parse(obj.headOfDepartment);

$("#" + element + "room").html(obj.room);

$("#" + element + "classname").html(obj.classname);

$("#" + element + "formTeacher").html(formTeacher\_parsed.Name);

$("#" + element + "headOfDepartment").html(headOfDepartment\_parsed.Name);

$("#" + element + "time").html(obj.time);

WriteDataInTable(element + "classesCompleted", obj.classesCompleted);

WriteDataInTable(element + "classesNotEdited", obj.classesNotEdited);

}

Code 49 – JavaScript WriteUserViewInformation

Da die abgeschlossenen und nächsten Klassen mehrere Werte besitzen, werden diese in eine HTML-Tabelle geschrieben. Die Methode „WriteDataInTable“ bekommt den Elementnamen und die Klassen übergeben.

## Ansicht Administrator

Ein Bild, das rot, Schild enthält.

Automatisch generierte Beschreibung

Abbildung 35 – Konferenzeinstellungen

Diese Ansicht wurde zu Beginn rein aus Testzwecken erstellt, erwies sich später jedoch als durchaus hilfreich. Der Moderator hat zwei Buttons zum Regeln der Einstellungen für die Konferenz.

### Konferenz zurücksetzen

Mit diesem Button wurde das Zurücksetzen der Konferenz implementiert. Alle Daten werden auf Standard gesetzt, als hätte die Konferenz noch nicht gestartet. Diese Funktion erleichterte das Testen der Funktionen, da mit einem Klick ein Neustart initiiert werden konnte.

public void OnPostReset()

{

db.Query($"UPDATE {general.Table\_General} set Status = 'not edited', start=null, end=null");

db.Query($"Update {general.TableStateOfConference} set Status = 'inactive'");

}

Code 50 – Admin\_Settings OnPostReset

### In Datenbank schreiben

Dieser Button wurde erstellt, um Daten aus der JSON-Datei in die Datenbank zu schreiben. Alle Daten werden aus der JSON-Datei gelesen. Die Klassen werden in der richtigen Reihenfolge und mit dem Status „not edited“ in die Datenbank geschrieben. Die Tabelle für den Konferenzstatus wird mit den Werten der vorhandenen Räume gefüllt.

Hier ist anzumerken, dass die alten Tabellen („General“, „StateOfConference“), noch vor dem Befüllen, komplett gelöscht werden. Diese Tabellen werden dann mit den neuen Daten wieder erstellt.

public void OnPostSetJsonData()

{

JObject jobject = JObject.Parse(general.JsonString);

JArray jOrder = (JArray)jobject["order"];

List<Order> orderlist = jOrder.ToObject<List<Order>>();

DeleteEverythingFromDatabase();

foreach (var orderitem in orderlist)

{

string roomonly = orderitem.Room.Split(' ')[0];

int ordercounter = 1;

SetStateSettings(roomonly);

foreach (string classitem in orderitem.Classes)

{

db.Query($"INSERT INTO {general.Table\_General} (ID, Room, ClassOrder, Status) VALUES(?,?,?, 'not edited')", classitem, roomonly, ordercounter);

ordercounter++;

}

}

}

Code 51 – Administratoransicht Methode OnPostSetJSONData

### Bestätigen der Aktionen

Um vor unbeabsichtigten Änderungen zu schützen, wurde ein einfache Bestätigung der Aktion implementiert. Somit muss der Administrator, bevor die Änderung in kraft tritt, diese auch bestätigen

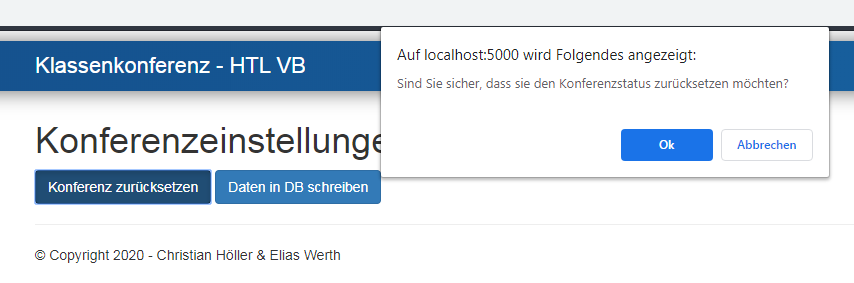


Abbildung 36 – Administratoransicht Bestätigen einer Aktion

Diese Mitteilungen wurden mit einem sogenannten „confirm“ in JavaScript erstellt. Ein „confirm“ ermöglicht es eine Mitteilung zu erstellen, und das Ergebnis der Abfrage zu erhalten. Also *True* wenn „Ok“ und *False* wenn „Abbrechen“ gedrückt wurde. In diesem Fall wird die Aktion nur ausgeführt, wenn die Abfrage bestätigt wurde.

## Authentifizierung mittels Microsoft AzureAD

### Registrierung im Azure-Portal

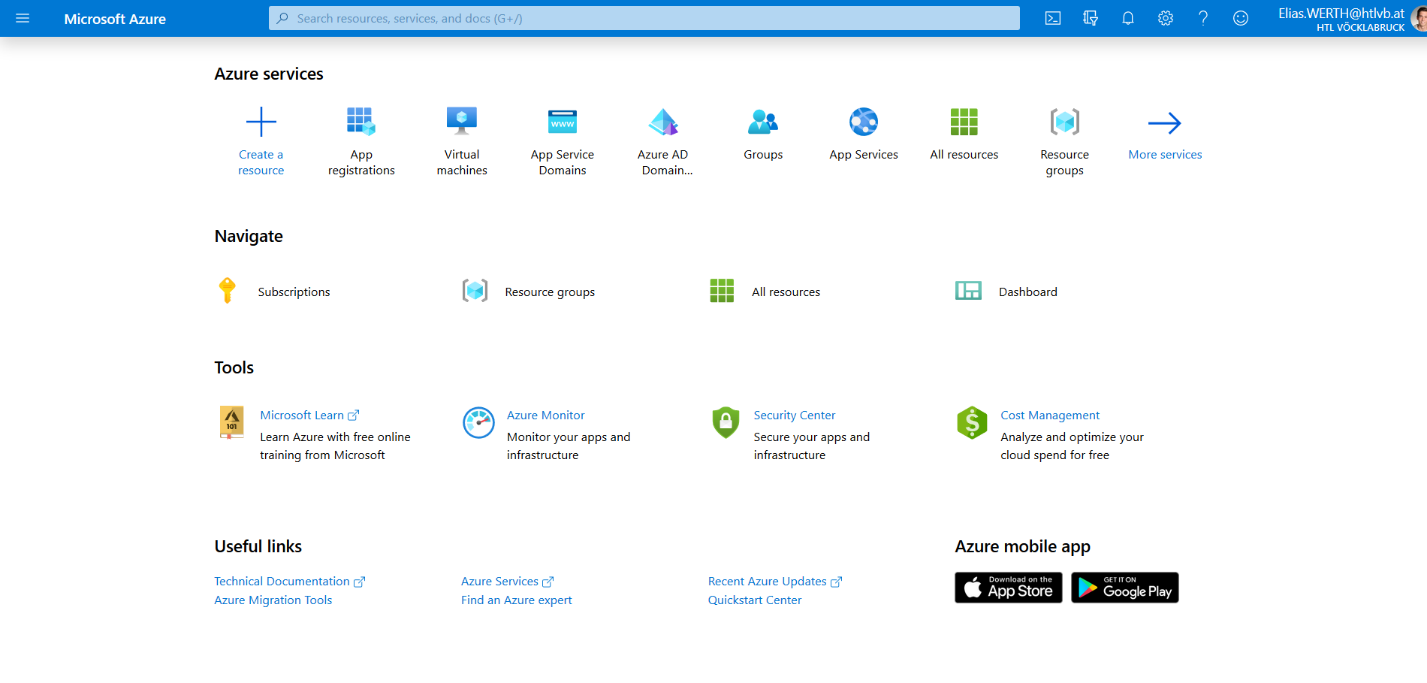
Bevor man fähig ist, den Office-Login zu implementieren, muss man seine Anwendung Azure-Portal registrieren. Im ersten Schritt navigiert man zu *portal.azure.com*.

Abbildung – Homepage Azure Portal

Unter dem Reiter „App registrations“ gelangt man auf eine Seite, auf der man vor allem seine bereits registrierten Applikationen verwalten, aber auch mit dem Button „New registration“ eine neue Anwendung registrieren kann. Hier muss man nur den Namen der Applikation eingeben und schon ist die Registrierung fertig.

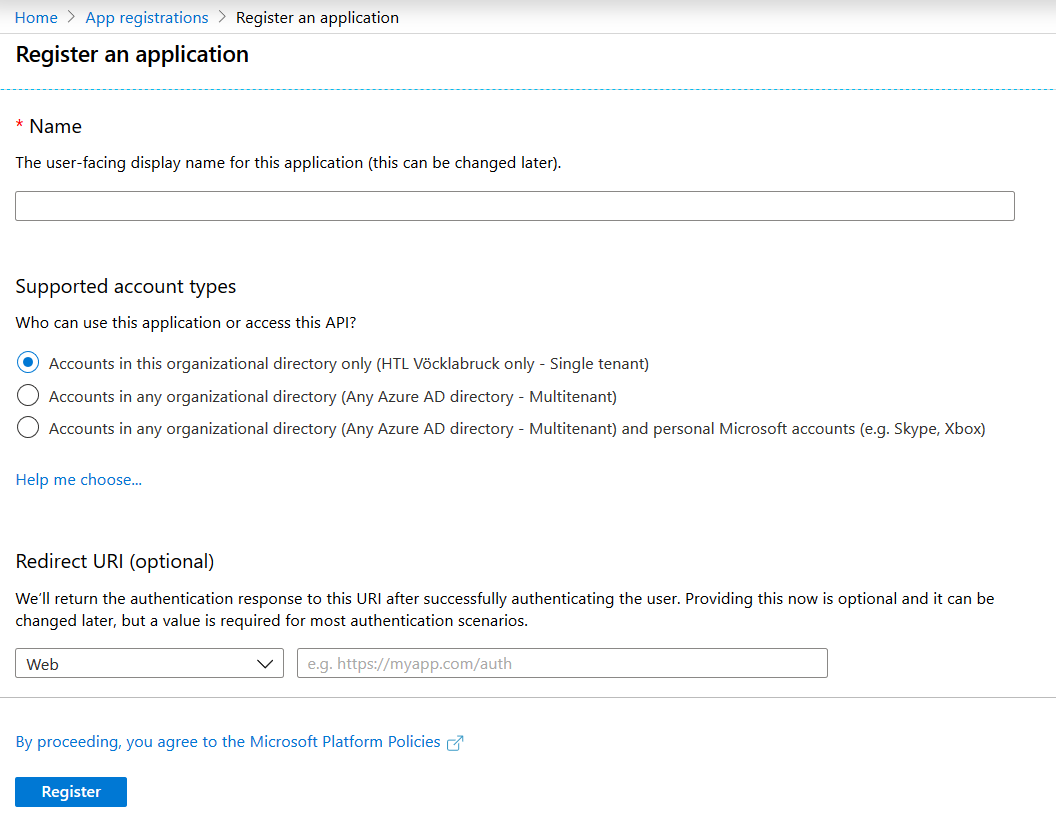


Abbildung – Registrierung einer Anwendung

### Konfiguration der Applikation

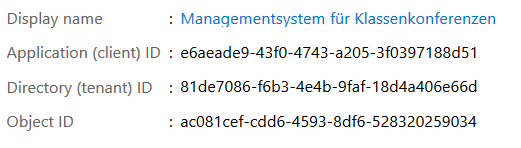
Nach der Registrierung ist man sofort in der Lage seine Anwendung zu konfigurieren, denn man bekommt darauf all die wichtigen Informationen, die man dazu braucht.

Abbildung - Anwendungsinformationen

Besonders die angegebenen IDs sind essenziell für eine korrekt Verbindung zwischen Anwendung und Azure-Portal:

* Application (client) ID
  + Die Client ID ist eine öffentliche Kennung für Anwendungen. Obwohl sie öffentlich ist, sollte sie nicht an Dritte weitergegeben werden. Ebenfalls muss die Client ID einzigartig sein.
  + <https://www.oauth.com/oauth2-servers/client-registration/client-id-secret/>
* Directory (tenant) ID
  + Die Tenant ID ist ein globally unique identifier (GUID). Diese ID gibt an, in welcher Azure-AD-Instanz sich die Anwendung befindet.
  + <https://microsoft.github.io/AzureTipsAndTricks/blog/tip153.html>

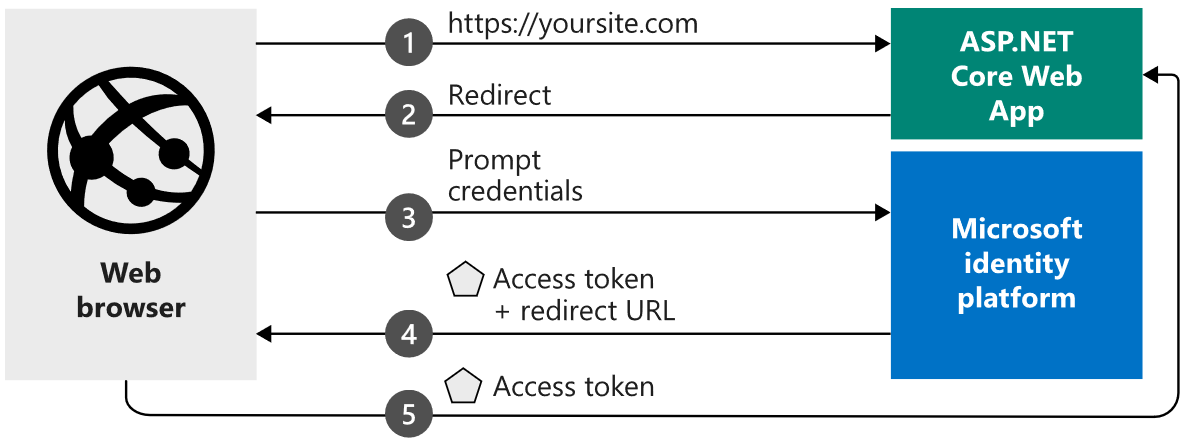
Die Quickstart-Guides dieser Seite bieten eine perfekte Schritt-für-Schritt-Anleitung an, die für das Managementsystem für Klassenkonferenzen hervorragend funktionieren. Zusätzlich wird hier auch erklärt wie die Authentifizierung funktioniert.

Abbildung – Prinzip der Authentifizierung

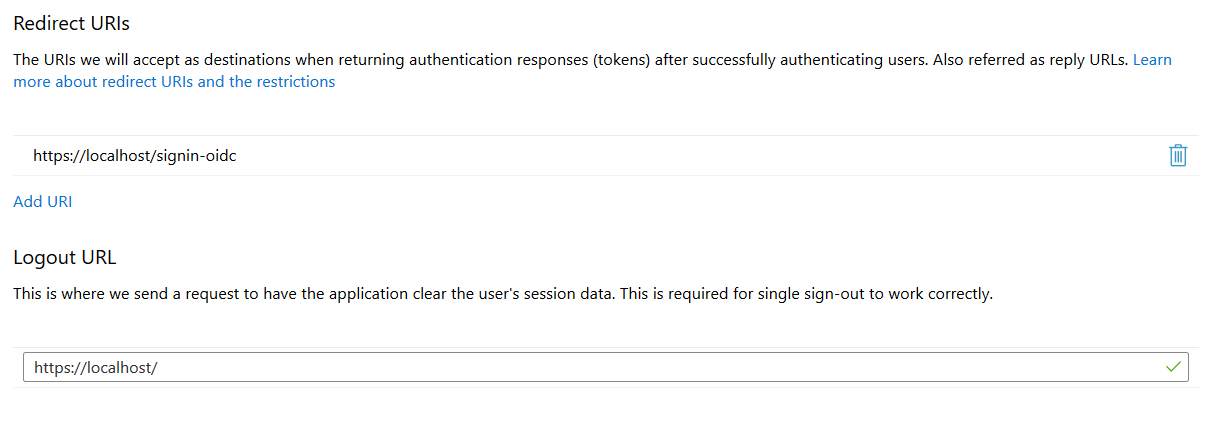
Im ersten Schritt muss im Azure-Portal der Redirect-URL konfiguriert werden. Ohne den richtigen Redirect-URL ist es nicht möglich, die Anwendung nach dem Login bzw. Logout zurückzuleiten, vorausgesetzt die Authentifizierungsantwort (Token) ist erfolgreich.

Abbildung – Redirect URIs

Der Port ist in diesem Fall nicht weiter wichtig. Essenziell ist nur der *localhost* und vor allem das *http****s****://*. Dasselbe gilt auch für den Logout-URL. Die Endung *signin-oidc* ist der Standardwert der OIDC-Client-Middleware und ist wesentlich für die korrekte Umleitung.

Im nächsten großen Schritt wird die appsettings.json-Datei konfiguriert. Hierbei werden alle zentralen AzureAD-Informationen eingetragen. Dazu gehören die Instanz (hier: Microsoft), die Domäne, auf der die Anwendung läuft und die bereits kennengelernten IDs.

"AzureAD": {

"Instance": "https://login.microsoftonline.com",

"Domain": "htlvb.at",

"ClientId": "e6aeade9-43f0-4743-a205-3f0397188d51",

"TenantId": "81de7086-f6b3-4e4b-9faf-18d4a406e66d"

},

### Implementierung des Logins

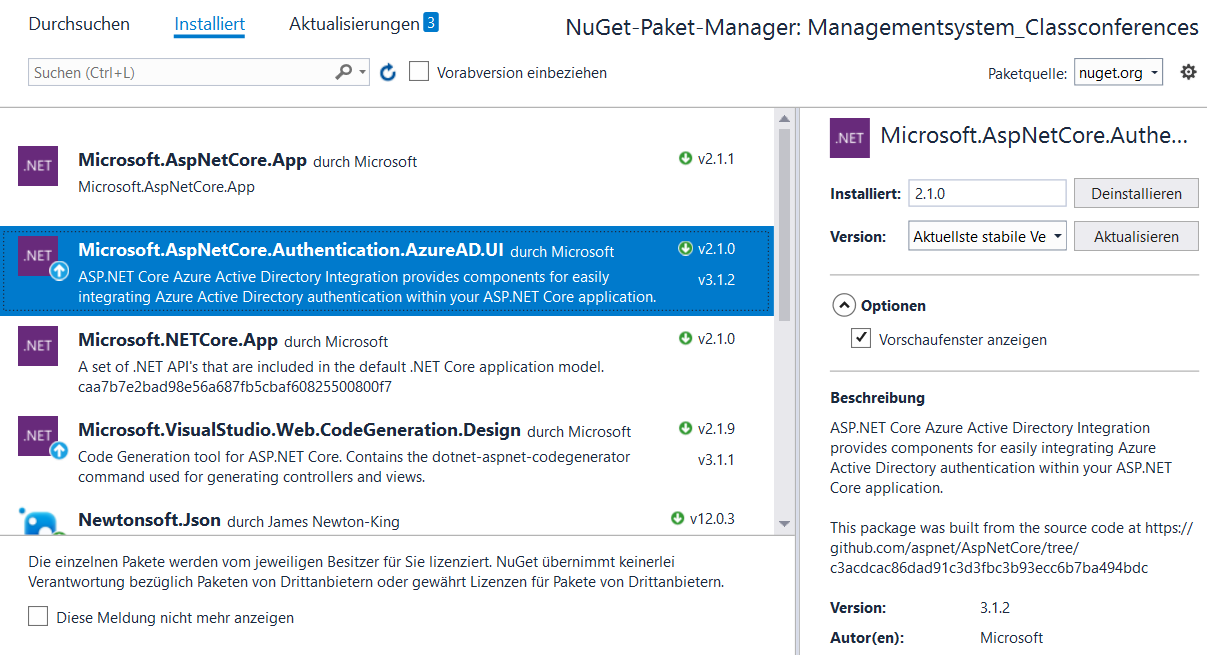
Bei der Implementierung von Microsoft Azure AD wird der Login korrekt in die Anwendung eingebaut. Der erste und wichtigste Schritt findet in der Startup.cs-Klasse statt, doch bevor man dazu fähig ist, braucht man das entsprechende NuGet-Paket.

Abbildung – NuGet AzureAD

Nach der Installation ist man nun im Stande, mit dem entsprechenden Code, AzureAD zu implementieren.

using Microsoft.AspNetCore.Authentication.AzureAD.UI;

using Microsoft.AspNetCore.Authentication;

[...]

public void ConfigureServices(IServiceCollection services)

{

[...]

services.AddAuthentication(AzureADDefaults.AuthenticationScheme

.AddAzureAD(options => Configuration.Bind("AzureAD", options));

services.Configure<OpenIdConnectOptions>(AzureADDefaults.OpenIdScheme,

options =>

{

options.Authority = options.Authority + "/v2.0/";

options.TokenValidationParameters.ValidateIssuer = false;

});

[...]

}

[...]

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

[...]

app.UseAuthentication();

[...]

}

Mit using Microsoft.AspNetCore.Authentication.AzureAD.UI; wird die Klasse AzureADDefaults aktiviert. Das Using kann man mit der Installation des NuGet-Pakets hinzugefügt werden.

Mit app.UseAuthentication(); wird die Authentifizierungsmiddleware zum spezifischen IApplicationBuilder hinzugefügt, was wiederum die benötigten Authentifizierungsfähigkeiten aktiviert.

Der nächste und letzte wichtige Schritt ist die Implementierung des eigentlichen Buttons in der \_Layout.cshtml-Page. Vorgesehen ist dafür ein Button in der rechtsbündigen Navbar mit entsprechendem Icon.

<div class="navbar-collapse collapse">

@using System.Security.Claims

@if (User.Identity.IsAuthenticated)

{

var identity = User.Identity as ClaimsIdentity;

string preferredUsername = identity.Claims.FirstOrDefault(c => c.Type ==

"name")?.Value;

<ul class="nav navbar-nav navbar-right">

<li><a href="https://www.office.com/"><span class="glyphicon glyphicon-

user"></span> @preferredUsername</a></li>

<li><**a** **asp-area**="AzureAD" **asp-controller**="Account" **asp-**

**action**="SignOut"><span class="glyphicon glyphicon-log-out"></span> Log

Out</**a**></li>

</ul>

}

else

{

<ul class="nav navbar-nav navbar-right">

<li>

<**a** **asp-area**="AzureAD" **asp-controller**="Account" **asp-action**="SignIn">

<span class="glyphicon glyphicon-log-in"></span> Login

</**a**>

</li>

</ul>

}

</div>

Zuerst wird überprüft, ob der Benutzer eingeloggt ist. Der Login-Button soll angezeigt werden, wenn er nicht angemeldet ist. Der Logout-Button, wenn er angemeldet ist.

Hinzu kommt, dass Vor- und Nachname links neben dem Button angezeigt werden soll, wenn er eingeloggt ist. Mit ClaimsIdentity werden einige essenzielle Daten des Accounts gesammelt. Danach wird nur noch der preferredUsername im Listitem ausgegeben. Mit einem Klick auf diesen Namen gelangt der Benutzer sofort zur Office-365-Startseite.

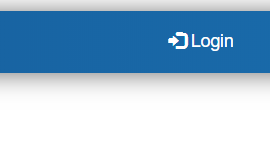
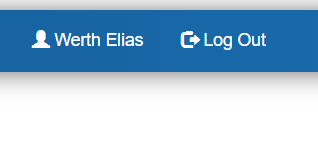
Nun ist die Applikation fertig konfiguriert und der Login vollständig implementiert. Der Authentifizierung funktioniert nun erfolgreich.

Abbildung – Logout-Button

Abbildung – Login-Button

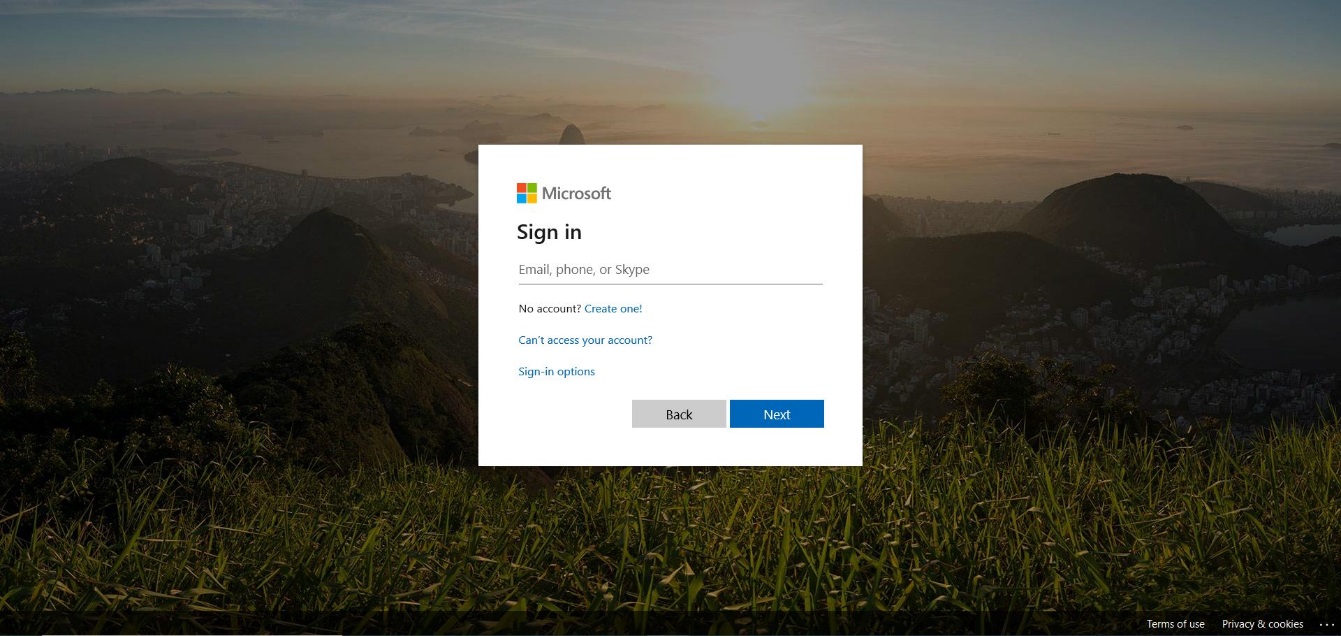


Abbildung – Microsoft Anmeldung

### Weiterleitung nach erfolgreicher Authentifizierung

Ein weiter wichtiger Teil des Logins, der auf keinen Fall vergessen werden darf, ist die Weiterleitung. Die Idee dabei ist folgende: Der Moderator soll auf die Seite für die Raumauswahl gelangen und der normale Benutzer soll zur Konferenzübersicht gelangen. Zusätzlich wird der Admin (Johannes Egger) zu der Seite für die Admineinstellungen weitergeleitet. Als Moderatoren zählen in diesem Fall die Abteilungsvorstände. Alle anderen Lehrkräfte haben kein Zugriffsrecht auf die Moderatorseite.

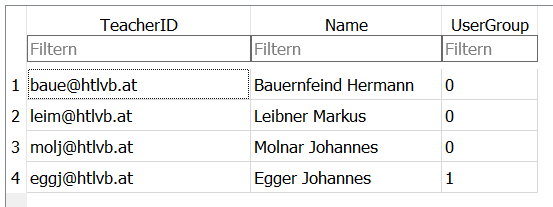
Der erste Schritt besteht daraus, eine neue Tabelle in der Datenbank anzulegen. Mit dem Namen „UserRights“ bestimmt diese, welche Rechte die jeweilige Lehrkraft hat. Die Spalten sind TeacherID, Name und UserGroup. Die TeacherID ist in diesem Fall die E-Mail-Adresse und die UserGroup bestimmt welches Recht die Lehrkraft besitzt. Die 0 repräsentiert hierbei die Abteilungsvorstände und die 1 steht für den Admin. Die Spalte „Name“ dient nur zur Vervollständigung.

Abbildung – Tabelle UserRights

Im nächsten Schritt wird die Abfrage nach den Rechten ins Programm implementiert. Das Wichtigste spielst dabei in der Index.cshtml.cs in der OnGet()-Methode ab, denn diese Methode wird jedes Mal abgerufen, wenn die Index-Seite neu lädt (also auch nach dem Login).

if (User.Identity.IsAuthenticated)

{

var identity = User.Identity as ClaimsIdentity;

string teacherId = identity.Claims.FirstOrDefault(c => c.Type ==

"preferred\_username")?.Value;

DataTable TeacherRight = DB.Reader($"SELECT UserGroup from

{general.TableUserRights} WHERE TeacherID LIKE ? LIMIT 1", teacherId);

if (TeacherRight.Rows.Count != 0)

{

if (Convert.ToInt64(TeacherRight.Rows[0][0]) == 0)

{

return new RedirectToPageResult("roomselection");

}

else return new RedirectToPageResult("admin\_settings");

}

else return new RedirectToPageResult("conference");

}

else return null;

Mit User.Identity.IsAuthenticated wird im ersten Schritt überprüft, ob der Benutzer überhaupt angemeldet ist. Im Falle eines Users, welcher nicht eingeloggt ist, wird lediglich ein null zurückgegeben. Andernfalls wird im nächsten Schritt, wie bei der Authentifizierung, mit ClaimsIdentity die E-Mail der angemeldeten Lehrkraft ermittelt. Im Anschluss setzt man eine SQL-Select-Anweisung ein, in der man die „UserGroup“ der Lehrkraft ermittelt. Die Tabelle „UserRights“ wurde schon im Vorfeld zu der Klasse General.cs hinzugefügt und kann so mit general.TableUserRights abgerufen werden.

Natürlich stehen nur Moderatoren und Admins in der Tabelle. Das bedeutet, dass wenn sich eine Lehrkraft ohne jegliche Rechte anmeldet, der DataTable leer bleibt. Genau dies wird in der ersten if-Abfrage überprüft. Wenn es wahr ist, dass der DataTable leer ist, wird die Lehrkraft mit RedirectToPageResult zur Konferenz-Page (conference) weitergeleitet.

Falls in den DataTable etwas geschrieben wurde, bedeutet dies, dass sich entweder ein Moderator oder ein Admin angemeldet hat. Genau das wird in der nächsten if-Abfrage überprüft und auch hier wieder mit RedirectToPageResult auf die entsprechende Seite weitergeleitet (roomselection bzw. admin\_settings).

## Progressive Web-App (PWA)

Um die ASP.NET Core Website in eine PWA umzuwandeln sind einige Schritte notwendig. Der erste und wohl wichtigste Schritt besteht daraus, einen **Service Worker** und ein **Manifest** in die Anwendung zu implementieren.

### Service Worker

Der Service Worker ist eine Art von Web Worker. Im Wesentlichen ist es ein JavaScript-File, das im Hintergrund und unabhängig vom Browser-Thread arbeitet. Der Service Worker macht Caching einfach und definiert Regeln, wenn bestimmter Inhalt der Seite nicht vorhanden ist.

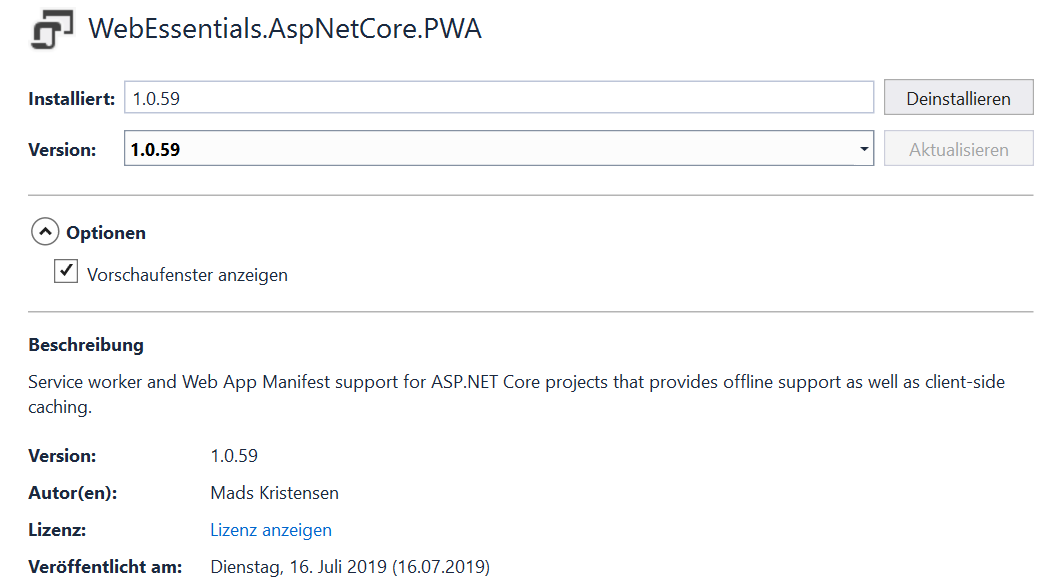
Um den Service Worker kümmert sich mehr oder weniger das NuGet-Paket „WebEssentials.AspNetCore.PWA“, entwickelt von Mads Kristensen.

Abbildung – Nuget WebEssentials for PWA

Dieses NuGet erstellt nicht nur den Service Worker, es kummert sich auch um dessen Implementierung in die Anwendung. Erwähnenswert ist hierbei, dass der Service Worker im Programm nicht einsehbar ist, und somit nicht verändert werden kann.

Ein essenzieller Schritt, der gemacht werden muss, bevor der Service Worker funktionieren kann, ist die Einbindung der PWA in die Startup.cs-Klasse.

public void ConfigureServices(IServiceCollection services)

{

[...]

services.AddProgressiveWebApp();

[...]

}

Durch diesen Service wird die Referenz zum Service Worker erstellt und die Nutzung des Service Workers wird dadurch garantiert.

Weitere Einstellungen bezüglich der PWA werden in der appsettings.json-Datei definiert.

{

[...]

"pwa": {

"cacheId": "Worker 1.1",

"strategy": "cacheFirstSafe",

"routesToPreCache": "/Index, /Conference",

"registerServiceWorker": true,

"registerWebmanifest": true

}

}

Das Merkmal cacheId definiert eine eindeutige ID für die aktuelle Version der Seite und wird im Service Worker verwendet, um zu überprüfen, ob der Benutzer derzeit über die neueste Version der Seite verfügt.

Mit strategy wählt man unter einigen vordefinierten Profilen eine Strategie für den Service Worker aus. Die Strategie cacheFirstSafe ist die Standardstrategie, mit der jede Ressource zwischengespeichert wird und sichergestellt wird, dass beim ersten Versuch über das Netzwerk auf Doctype text/html zugegriffen wird, um die neueste Version abzurufen. Wenn dies nicht verfügbar ist, wird auf den Cache zurückgegriffen.

routeToPreCache gibt an, welche Seiten vor dem cachen gespeichert werden sollen. Dies bedeutet, dass sie dem Cache hinzugefügt werden, sobald der Service Worker registriert wurde. Dies ist besonders nützlich, um alle Unterseiten auf der Webpage zwischenzuspeichern, damit der Benutzer diese besuchen kann, während er offline ist.

registerServiceWorker und registerWebmanifest dienen zum Deaktivieren einzelner Funktionen im Service.

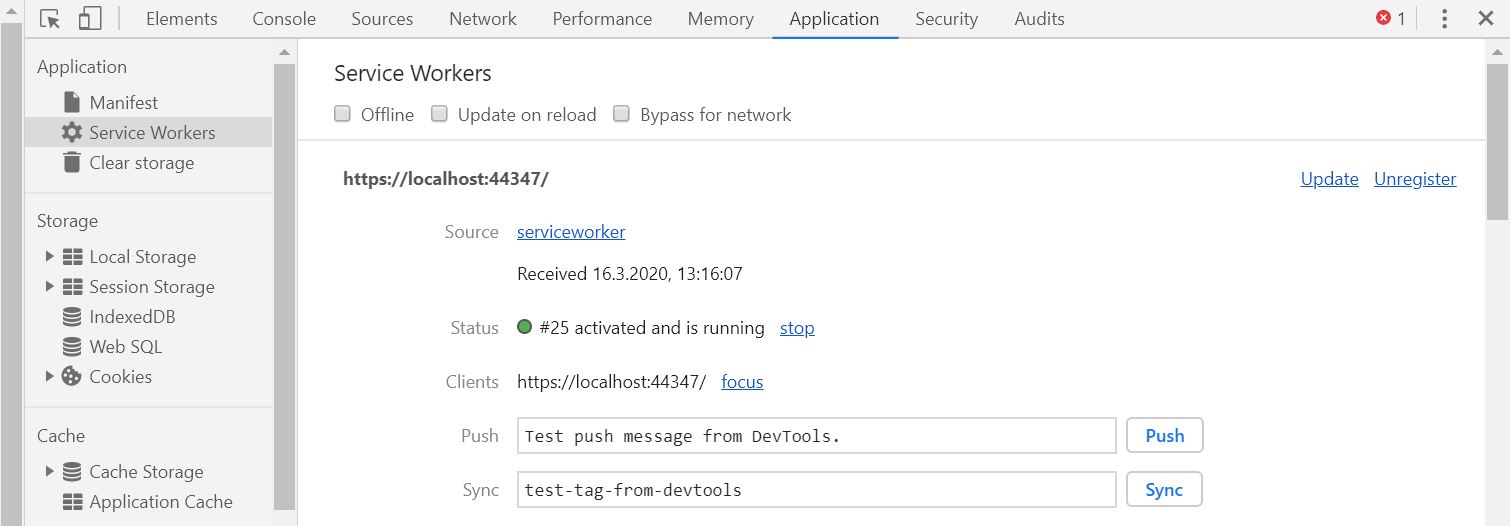
Auf dem jetzigen Stand der PWA ist es bereits möglich zu sehen ob der Service Worker aktiv ist und läuft. Dies kann man in den Chrome DevTools unter Application > Service Worker einsehen.

Abbildung – DevTools Service Worker

### Manifest

Ausständig bleibt nun nur noch das Erstellen eines Web-App-Manifests. Das Manifest ist ein JSON-File, das dem Browser wichtige Information der Progressive Web-App übermittelt. Es sagt den Browser auch, wie es sich verhalten soll, wenn es am Mobiltelefon des Benutzers installiert wird.

Das Erstellen des Manifests ist keine große Herrausforderung, jedoch muss beachtet werden, dass sich die JSON-Datei im wwwroot-Ordner befindet. Der Name dieser Datei spielt für gewöhnlich keine entscheidende Rolle, jedoch ist eine sprechende Bezeichnung zu bevorzugen. Hier wurde der Name manifest.json gewählt.

Der Inhalt des Manifests ist folgendermaßen strukturiert:

{

"name": "Managementsystem für Klassenkonferenzen",

"short\_name": "Klassenkonferenzen",

"description": "Das Managementsystem der HTL VB für Klassenkonferenzen",

"display": "standalone",

"icons": [

{

"src": "/images/logox192.png",

"type": "image/png",

"sizes": "192x192"

},

{

"src": "/images/logox512.png",

"type": "image/png",

"sizes": "512x512"

}

],

"start\_url": "/",

"color": "#134883",

"background\_color": "#2b2b2b",

"theme\_color": "#134883"

}

Die Eigenschaft name ist der volle Name der Anwendung. Solange name zur Verfügung steht, ist es nicht unbedingt notwendig, die Eigenschaft short\_name auch in das Manifest einzubinden. Nichtsdestotrotz hat man sich hier dafür entschieden, denn short\_name wird verwendet, wenn der Benutzer die Web-App auf seinem Homescreen bzw. Desktop installiert.

Mit display wird definiert welches User Interface für den Browser gewählt wird. Das Item standalone öffnet die Web-App so, dass sie wie eine eigenständige native App aussieht und sich auch so anfühlt. Die App wird in einem eigenen Fenster ausgeführt, das vom Browser getrennt ist, und verbirgt Standardelemente der Browser-Benutzeroberfläche wie die URL-Leiste.

Die icons-Eigenschaft ist ein Array von Bildern. Jedes Objekt braucht eine src (Quelle), die sizes (Abmessungen) und den type (Art) des Objekts. Für Chrome braucht man mindestens ein 192x192pixel-Icon und ein 512x512pixel-Icon. Falls nur diese zwei Icons vorhanden sind, skaliert Chrome die Icons automatisch auf die Größe, die für das Gerät geeignet ist. Die benötigten Icons müssen sich in dem Ordner „images“ der Projektmappe befinden.

Die start\_url ist erforderlich und teilt dem Browser mit, wo Ihre Anwendung beim Ausführen gestartet werden soll, und verhindert, dass die App auf der Seite gestartet wird, auf der sich der Benutzer befand, als er Ihre App zu seinem Startbildschirm hinzufügte.

Die Eigenschaften color und background\_color hängen mit dem Splash-Screen zusammen. Der Splash-Screen wird angezeigt, wenn die App zum ersten Mal am Mobilgerät gestartet wird. Es ist also nichts weiter als ein grafischer Platzhalter, der solange zu sehen ist, wie die App zum Laden benötigt. Die color ist in diesem Fall die Schriftfarbe, in der der Titel der Anwendung angezeigt wird und die background\_color ist die Hintergrundfarbe des Splash-Screens. Ebenfalls am Splash-Screens zu sehen am Splashscreen ist das Icon.

Die theme\_color ist ein einfaches Attribut, dass bestimmt, welche Farbe der Toolbar bestimmt. Wie der Name schon sagt, ist es das Farbschema der Seite.

Das Manifest ist nun komplett und der letzte essenzielle Schritt ist es, das Manifest mit einem <link>-Tag zu verlinken. Dies geschieht im <head> der \_Layout.cshtml-Page.

<**environment** **include**="Development">

[...]

<link rel="manifest" href="~/manifest.json" />

[...]

</**environment**>

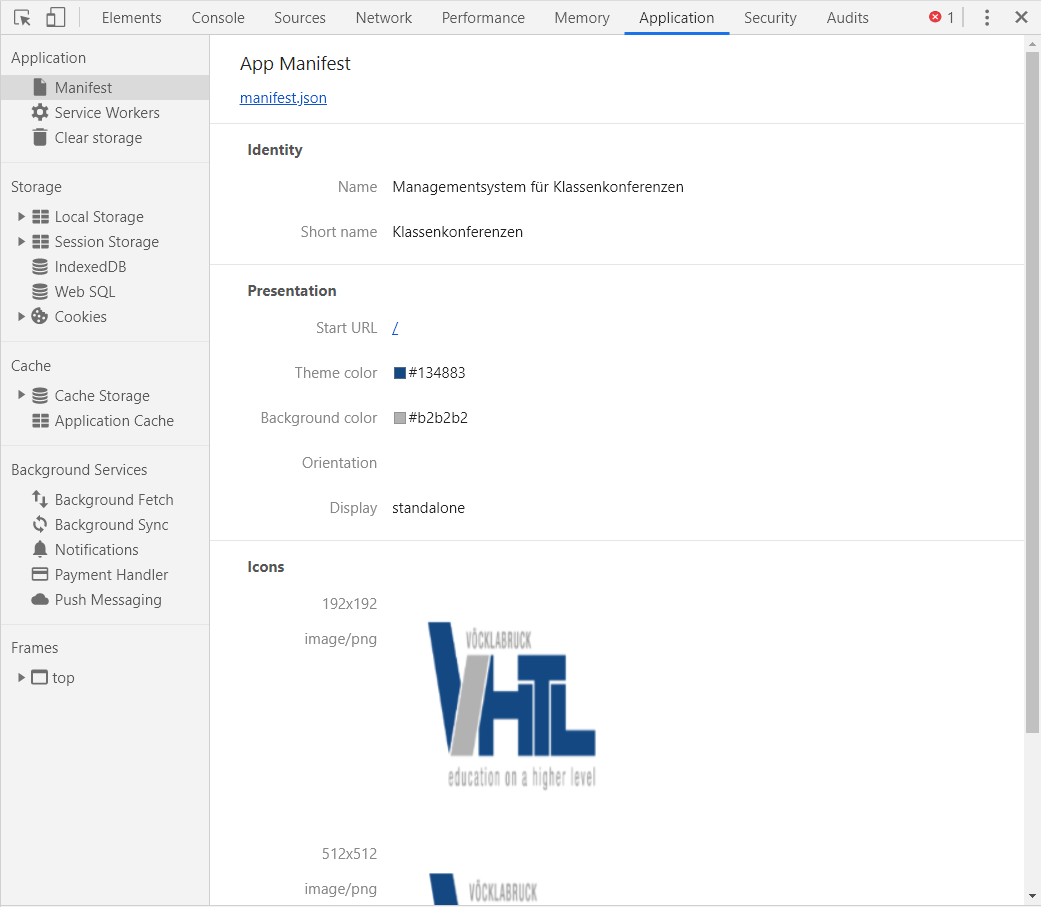
Danach ist die Progressive Web App startklar und die Anwendung ist bereit zum Testen. Mithilfe der Chrome DevTools hat man auch hier die Möglichkeit unter Application > Manifest alle wichtigen Details einzusehen.

Abbildung – DevTools Manifest

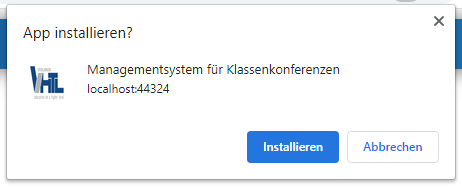
Auch die Installation der Instanz ist nun bereit. Für Benutzer, die über den PC auf das Managementsystem gelangen, wird die Installation direkt nach dem Abruf der Webseite (in Chrome) vorgeschlagen.

Abbildung - Installationsbenachrichtigung

Und auch schon direkt nach dem Mausklick auf den Installations-Button öffnet sich die installierte Progressive Web App. Auch das Icon ist von jetzt an am Desktop sichtbar.

Abbildung – Installierte Version

## Aufrufen von Lehrkräften

Die Moderatoren der einzelnen Konferenzen sollen die Möglichkeit haben, eine Lehrkraft aufzurufen. Dies ist notwendig, um Abwesenheiten vorzubeugen. In einem Szenario, in dem der Moderator feststellt, dass sich ein oder mehrere Lehrkräfte nicht im Raum befinden, muss der Moderator der Klassenkonferenz nur einen Button drücken und die fehlende Person wird sofort über ihr Smartphone erinnert, dass sie erwartet wird.

### Hinzufügen von Aufruf-Buttons

Der Moderator soll für jeden Eintrag in der Liste der Lehrkräfte dieser Klasse einen eigenen Button haben. Praktisch gesehen muss nur eine neue Spalte in die Tabelle eingefügt und mit den Buttons befüllt werden. Die Methode, um alle Lehrkräfte aufzuzählen besteht bereits, also muss diese nur noch erweitert werden. Dies geschieht in der moderator\_view.js-Datei.

function WriteTeachersWithButtonsInTable(teacherArray) {

$("#teachers").empty();

if (teacherArray == "") {

$("#teachers").append("<tr><td>Keine Lehrer</td></tr>");

}

else {

var teacherData, buttonData;

var parsedArray = JSON.parse(teacherArray);

for (var i = 0; i < parsedArray.length; i++) {

var teacherID = parsedArray[i].ID;

var fullName = parsedArray[i].Name;

teacherData = "<td><p title='" + fullName + "'>" +

getShorthandForTeacher(teacherID) + "</p></td>";

buttonData = "<td><button onclick='callTeacher(" + i + ")'

class='btnbtn-secundary'>ausrufen</button></td>";

$("#teachers").append("<tr>" + teacherData + buttonData + "</tr>");

}

}

}

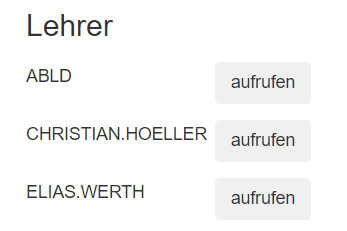
Zuallererst wurde der Name der Funktion umgeändert, um die sprechende Bezeichnung aufrecht zu halten. Im nächsten Schritt wird, wie bereits erwähnt, die Funktion erweitert. Als Hilfestellung wird hierbei die Variable buttonData herangezogen. In diese wird der HTML-Code für die Buttons innerhalb der Tabelle codiert. Zu beachten gilt hierbei, dass die Buttons selbstverständlich ein OnClick-Event benötigen, die die Funktion aufruft, in der der eigentliche Aufruf der verlangten Person stattfindet.

Abbildung – Aufruf-Buttons

### Senden des Aufrufs

Die Funktion, die durch den Click auf den Button aufgerufen werden soll, wird auch in der moderator\_view.js-Datei. Der Name der Funktion lautet callTeacher und die mitgegebene Variable ist der Index der Lehrkraft, die aufgerufen werden soll.

function callTeacher(indexOfCalledTeacher) {

var moderatorID = $("#moderatorID").val();

connection.invoke("SendTeacherCall", indexOfCalledTeacher, moderatorID, GetCurrentRoom()).catch(function (err) {

return console.error(err.toString());

});

}

Sobald der Moderator den Button betätigt, wird die Funktion callTeacher() aufgerufen. Das Ziel dieser Funktion ist der Aufruf einer weiteren Funktion mit dem Namen SendTeacherCall, die sich im MainHub befindet. Vorrausetzung dafür ist selbstverständlich die Verbindung zum MainHub. Essenziell sind die Variablen, die mitgesendet werden. Index ist die Variable, die bereits der jetzigen Funktion mitgegeben wurde. moderatorID ist Email-Adresse der Lehrkraft, die aufgerufen wird. Diese Variable wurde in der Zeile zuvor ermittelt. Mit GetCurrentRoom() wird der aktuelle Raum, in der die Konferenz stattfindet, aufgerufen.

Wenn durch connection.invoke erfolgreich an den MainHub gesendet wurde, wird die Funktion SendTeacherCall ausgeführt. Vom MainHub aus kann nun auf alle verbundenen Clients gesendet werden.

public async Task SendTeacherCall(int indexOfCalledTeacher, string moderatorID, string \_currenroom)

{

Currentroom = \_currenroom;

var currentClass = GetClass(GetCurrentClassName());

var teacherToCall = currentClass.Teachers[indexOfCalledTeacher];

var message = $"Sie werden in Raum {Currentroom} erwartet";

await Clients.All.SendAsync("ReceiveTeacherCall", teacherToCall, message);

}

Der Sinn dieser Methode ist, den Ausruf an alle verbundenen Clients zu senden. Mit der Variable teacherToCall gibt man die Lehrkraft an, an die diese Benachrichtigung gerichtet ist. Dass bedeutet, dass jeder Client zuerst überprüfen muss, ob diese Nachricht für ihn bestimmt ist. Ebenfalls wichtig ist die Nachricht selbst, in der auch der aktuelle Raum geschrieben steht.

### Speichern der Aufrufe

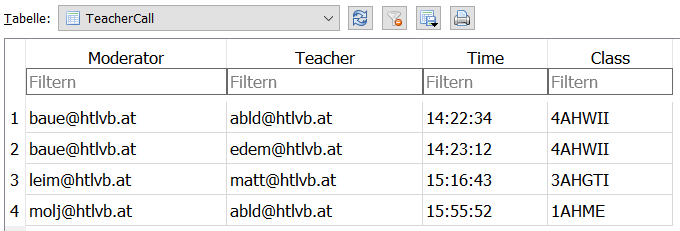
Teil der Aufgabenstellung ist es auch, jeden einzelnen Ausruf für statistische Zwecke in eine eigene Tabelle zu speichern. In dieser Tabelle soll der Moderator und die Lehrkraft, die dieser aufgerufen hat, der Zeitpunkt zu dem aufgerufen wurde und die Klasse, die in diesem Moment besprochen wurde.

Abbildung – Tabelle TeacherCall

Erstellt wird diese Tabelle in der gewohnten Datenbank der Konferenz. Als Name wurde TeacherCall gewählt. Befüllt wird diese Tabelle mit einem einfachen SQL-Command.

dB.Query($"INSERT INTO {general.TableTeacherCall}(Moderator, Teacher, Time, Class)

VALUES(?,?,?,?)", moderatorID, teacherToCall, DateTime.Now.ToLongTimeString(), currentClass.ClassName);

Den Namen der Tabelle erlangt man hierbei durch die General.cs-Klasse, in der die Namen aller Tabellen als Variablen gespeichert sind. Codiert wird dieser Befehl in der SendTeacherCall-Funktion.

Auf der Webpage des Managementsystems ist diese Tabelle nicht zu sehen. Wie bereits erwähnt, werden diese Daten nur für eine statistische Auswertung der Konferenz gesichert.

### Empfangen des Aufrufs

Bevor die Möglichkeit besteht, Aufrufe entgegenzunehmen, muss der Benutzer um Erlaubnis gefragt werden. Sobald man auf die Übersichtsseite der Konferenz gelangt, wird jede Lehrkraft gefragt, ob sie damit einverstanden ist, Benachrichtigungen zu empfangen. Realisiert wird diese Einverständniserklärung mit einem einfachen Panel in der Conference.cs.html-Page.

<div id="notiContent" class="panel panel-default" style="visibility: visible;">

<div class="panel-heading">Ausruf von Lehrkräften</div>

<div class="panel-body">

<p>Moderatoren der Konferenz haben die Möglichkeit, Lehrkräfte aufzurufen,

falls dies notwendig ist. Mit Ihrem Einverständnis geben Sie an, dass

Sie Benachrichtigungen empfangen wollen.</p>

<button class="btn btn-primary"

Onclick="yesToNotifications()">Akzeptieren</button>

<button class="btn btn-default"

Onclick="noToNotifications()">Verweigern</button>

</div>

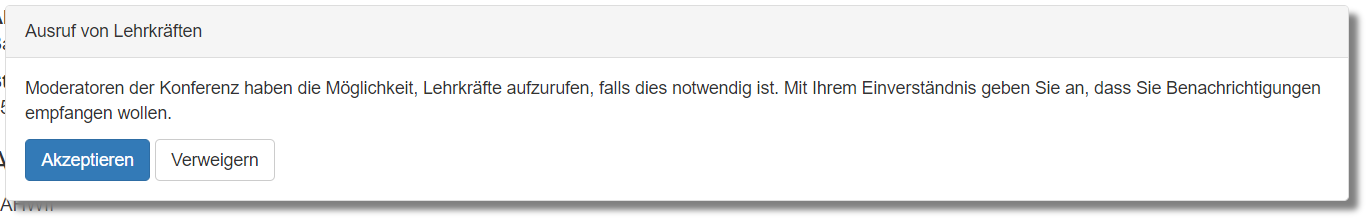
</div>

Abbildung – Einverständniserklärung für Aufrufe

Je nach Entscheidung des Nutzers wird eine Funktion ausgeführt.

var allowNotifications = false;

function yesToNotifications() {

allowNotifications = true;

document.getElementById('notiContent').style.visibility = "hidden";

}

function noToNotifications() {

allowNotifications = false;

document.getElementById('notiContent').style.visibility = "hidden";

}

Die Variable allowNotifications speichert die Auswahl der Lehrkraft und kann zu einem späteren Zeitpunkt überprüft werden. Gemeinsam haben diese beiden Funktionen, dass sie nach dem setzen der Variable, das Panel unsichtbar machen.

Nachdem ein Aufruf empfangen wird und sich der Benutzer für die Benachrichtigungen entschieden hat, muss nun geklärt werden ob der Aufruf für ihn gedacht ist. Dafür wird die TeacherID (Email-Adresse der gesuchten Lehrkraft) mit der UserID (Email-Adresse des eingeloggten Benutzers) verglichen. Um die UserID zu erlangen ist folgender Code notwendig:

@{

@using System.Security.Claims;

var identity = User.Identity as ClaimsIdentity;

string userID = identity.Claims.FirstOrDefault(c => c.Type ==

"preferred\_username")?.Value;

@Html.HiddenFor(x => userID);

}

Platziert wird dieser Code in der Conference.cshtml-Page. Mit System.Security.Claims kann man ClaimsIdentity anwenden. Dies sammelt essenzielle Daten über den eingeloggten Benutzer. Darunter auch den preferred\_username, der in diesem Fall die E-Mail ist.

Die Abfrage, die klärt, ob der Aufruf an diesen Client gesendet werden muss, sieht folgendermaßen aus:

connection.on("ReceiveTeacherCall", function (teacherID, message) {

if (allowNotifications == true) {

var userID = $("#userID").val();

if (teacherID == userID.toLowerCase()) {

//Platz für die Visualisierung

}

}

});

Der Grund, warum die UserID in den Lower Case umgewandelt wird, ist, um die IDs einheitlich zu machen und einen genauen Vergleich zu garantieren.

### Visualisierung der Benachrichtigung

Nach dem der Code für die Benachrichtigung geschrieben wurde, muss diese nur noch dargestellt werden. Gewünscht ist eine Push-Benachrichtigung, auf der Übersichtsseite.

Um keine Zeit mit dem Programmieren dieser Push-Benachrichtigung zu verschwenden, hat man sich dazu entschieden, eine bereits bestehende Zusatz-Bibliothek zu verwenden. Als beste Lösung stellte sich Toastr heraus. „toastr ist eine Javascript-Bibliothek für nicht blockierende Benachrichtigungen vom Typ Gnome/Growl.“ (<https://codeseven.github.io/toastr/>)

Um eine Toast-Notifikation in das Programm einzubinden, sind 3 wichtige Schritte notwendig. Die ersten 2 Schritte müssen auf der Layout-Seite in der Entwicklungsumgebung erfolgen.

<**environment** **include**="Development">

<link rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/2.0.1/css/toastr.css" />

</**environment**>

Mithilfe des <link>-Tags wird auf die css-Datei der Toastr-Bibliothek verwiesen.

<**environment** **include**="Development">

<script

src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/2.0.1/js/toastr.js">

</script>

</**environment**>

Im zweiten Schritt wird auf die Javascript-Datei der Bibliothek verwiesen.

Der letzte Schritt ist die eigentliche Darstellung des Toasts. Dies geschieht im user\_view.js-File, genau dort, wo bis jetzt Platz gelassen wurde.

toastr.options = {

"closeButton": true,

"positionClass": "toast-bottom-right",

"onclick": null,

"showDuration": "300",

"hideDuration": "1000",

"timeOut": "60000",

"extendedTimeOut": "5000",

"showEasing": "swing",

"hideEasing": "linear",

"showMethod": "fadeIn",

"hideMethod": "fadeOut"

};

toastr.info(message, "Ausruf");

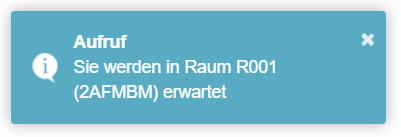
Zuerst werden die Eigenschaften der Notifikation definiert. Unter andern kann man hier einstellen, wo der Toast erscheinen soll und wie lange dieser ersichtlich bleibt. Anschließend wird mit toastr.info() der Zweck der Benachrichtigung festgelegt. „Info“ steht hierbei für eine Information. Weitere Optionen wären .success, .warning oder .error. Mitgegeben werden hier nur die eigentliche Nachricht und der Titel dieser Benachrichtigung. Mehr braucht es nicht um die Push-Benachrichtigung in die Anwendung einzubinden.

Abbildung - Toast

### Text-to-Speech

Als nettes Feature hat man sich dafür entschieden, die Nachrichten automatisch vorlesen zu lassen, in dem Moment, in dem diese eingeht. Grund dafür ist, dass kein Lehrer die Benachrichtigung übersehen kann.

Der JavaScript-Code dafür wird direkt unter dem Toast geschrieben.

const utterThis = new SpeechSynthesisUtterance(message);

utterThis.lang = 'de-AT';

speechSynthesis.speak(utterThis);

Im ersten Schritt wird eine Variable angelegt, in der die Äußerung der SpeechSysthesis-Schnittstelle hinterlegt wird. „Die SpeechSynthesis-Schnittstelle der Web Speech API ist die Controller-Schnittstelle für den Sprachdienst. Sie kann genutzt werden um Informationen über die Synthesestimmen, die auf dem Gerät verfügbar sind, zu erhalten. Außerdem um die Sprache zu starten, zu pausieren und andere Befehle auszuführen.“

(<https://developer.mozilla.org/de/docs/Web/API/SpeechSynthesis>)

Mit utterThis.lang definiert man die Sprache, in der gesprochen werden soll. Ohne diesem Attribut wäre die Sprache vom Browser abhängig, dass für dieses Projekt nicht geeignet wäre.

Um die Nachricht nun abzuspielen, wird die Methode .speak() herangezogen, der die Variable utterThis mitgegeben wird.

## Design und Layout der Web-App

Das Design und das Layout spielen eine große Rolle in der Anwendung. Teil der Aufgabenstellung ist es, das System optisch ansprechender zu machen. Ein modernes Design und ein Layout, dass sich an das Endgerät anpasst, sind die Hauptaufgaben dieses Kapitels. Bootstrap und CSS helfen hierbei am meisten.

### Navigationsbar und Footer

#### Navigationsbar

Das Design der Navigationsbar soll einem direkt auffallen. Für diesen Effekt sorgt ein Farbenverlauf. Als Themenfarbe wurde hier die Hauptfarbe der Schule gewählt. Der Verlauf geht von links nach recht und wird stets ein wenig heller.

Bootstrap vereinfacht die ganze Sache. Mit einer Auswahl an unzähligen, bereits vordefinierten Klassen ist Bootstrap in dieser Hinsicht eine große Unterstützung. Folgender Code befindet sich in der \_Layout.cshtml-Datei.

<nav class="navbar navbar-inverse navbar-custom">

[…]

</nav>

Die Klassen navbar und navbar-inverse sind ein Teil von Bootstrap. Die Klasse navbar-custom im Gegensatz ist eine Eigenkreation. Diese sorgt für den bereits erwähnten Verlauf. Der Code dafür wird in die site.css-Datei programmiert.

.navbar-custom {

width: auto;

margin-top: 10px;

margin-bottom: 10px;

background-color: #134883;

background: -moz-linear-gradient(left, rgba(19,72,131,1) 0%,

rgba(22,139,229,0.71) 99%, rgba(22,139,229,0.71) 100%); /\* FF3.6+ \*/

background: -webkit-gradient(linear, left top, right top, color-

stop(0%,rgba(19,72,131,1)), color-stop(99%,rgba(22,139,229,0.71)), color-stop(100%,rgba(22,139,229,0.71))); /\* Chrome,Safari4+ \*/

background: -webkit-linear-gradient(left, rgba(19,72,131,1)

0%,rgba(22,139,229,0.71) 99%,rgba(22,139,229,0.71) 100%); /\*Chrome10+,Safari5+\*/

background: -o-linear-gradient(left, rgba(19,72,131,1) 0%,rgba(22,139,229,0.71)

99%,rgba(22,139,229,0.71) 100%); /\* Opera 11.10+ \*/

background: linear-gradient(to right, rgba(19,72,131,1) 0%,rgba(22,139,229,0.71)

99%,rgba(22,139,229,0.71) 100%); /\* W3C \*/

border: none;

overflow: hidden;

box-shadow: 1px 1px 20px #737373;

}

 Das resultierende Navigationsbar sieht nach Gebrauch dieses Codes folgendermaßen aus:

Abbildung - Navigationsbar

Wie bereits erwähnt muss sich das Layout auch an das dementsprechende Endgerät anpassen. Das bedeutet, sobald sich die Weite des Bildschirmes verändert, verändert sich auch die Navigationsleiste.

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse"

data-target=".navbar-collapse">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand">Klassenkonferenz - HTL VB</a>

</div>

<div class="navbar-collapse collapse">

//Code des Login-Buttons

</div>

Dies ist eine Lösung von Bootstrap. Im Prinzip erfolgt hierbei nichts anderes, als dass der Inhalt der Navigationsleiste mit den Iconbars ersetzt wird, sobald die Bildschirmweite nicht mehr ausreicht.

Abbildung – Navigationsbar Toggle

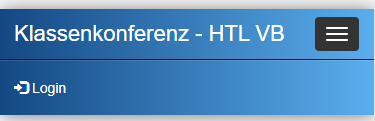
Der Login-Button ist naturlich nicht verloren. Durch einen Klick auf den Toggle-Button fächert sich das Menü nach unten aus.

Abbildung – Navigationsbar Toggle ausgefächert

#### Footer

Der Sinn des Footer ist nichts weiter, als das Copyright bekannt zu geben. Der Footer sollte logischer Weise immer am Fuße der Webseite sein. Getrennt wird dieser vom Inhalt durch eine Border quer über die Seite. Codiert wird der Footer in \_Layout.cshtml.

<div class="container body-content">

@RenderBody()

<footer class="footer">

<p>&copy; Copyright @DateTime.Now.Year - Christian Höller & Elias Werth</p>

</footer>

</div>

Die Css-Klasse footer sorgt hier für eine optimale Darstellung und eine Anzeige ohne Bugs.

.footer {

float: left;

width: 100%;

margin-top: 25px;

padding: 5px;

border-top: 1px solid gray;

}

Der fertige Footer sieht wie folgt aus:

Abbildung - Footer

### Design der Nutzeransicht

Das Ziel der Nutzeransicht ist es, eine Übersicht über den Stand der Konferenz zu bieten. Das bedeutet, dass von beiden Besprechungen der Status so gut als möglich dargestellt werden muss. Die Nutzeransicht beinhaltet auch das Panel mit der Frage nach den Aufrufen.

Im ersten Schritt wird jeder Raum und die dazugehörigen Informationen in eine eigene <div> gesetzt, um sie später einzeln bearbeiten zu können.

<body>

<div id="room1" class="left-side">

//Code des ersten Raums

</div>

<div id="room2" class="right-side">

//Code des zweiten Raums

</div>

</body>

Die beiden Klassen der Divisions sind selbst gecodet und helfen bei der Anordnung.

.left-side {

float: left;

width: 50%;

padding: 20px;

}

.right-side{

float: left;

width: 50%;

padding: 20px;

}

Durch den float-Wert und der aufteilung auf 50/50 gelingt es, die beiden Anzeigen nebeneinander zu platzieren. Der Padding-Wert ist ein Zusatzattribut, das Überschneidungen der Elemente vorbeugt.

Das Design des Panels kann glücklicherweise Bootstrap übernehmen.

<div id="notiContent" class="panel panel-default" style="visibility: visible;">

<div class="panel-heading">Ausruf von Lehrkräften</div>

<div class="panel-body">

</div>

</div>

Was Bootstrip nicht übernehmen kann, ist die Platzierung des Panels. Dieses sollte genau in der Mitte und über dem eigentlichen Inhalt der Seite platziert werden. Hierfür wird ein weiterer Css-Code geschrieben. Das ist auch der Grund, weshalb dem Panel eine ID vergeben wurde.

.panel {

position: fixed;

width: 70%;

left: 50%;

top: 35%;

margin-left: -35%;

box-shadow: 7px 7px 5px grey;

}

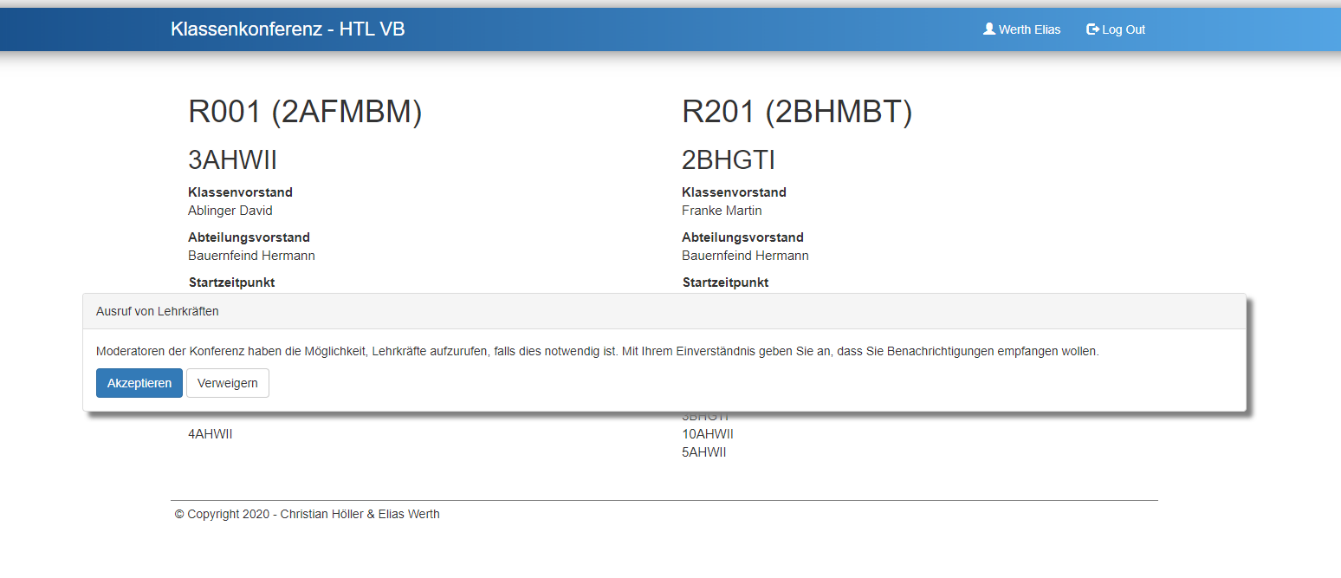


Abbildung – Nutzeransicht mit Einverständniserklärung

### Design der Moderatoransicht

In der Moderatoransicht gibt es einige Dinge zu beachten, wenn es zum Layout kommt. Zum einen darf kein Platzmangel entstehen, wenn alle Lehrkräfte und Überschneidungen geladen werden, zum anderen muss das Layout die Steuerung der Konferenz erleichtern.

Um dem Problem des Platzmangels entgegenzuwirken, hat man sich dazu entschieden, die Liste der Lehrkräfte und die Überschneidungen auf die rechte Seite zu verschieben.

<h1 id="room" style="padding-left:20px;"></h1>

<div class="left-side">

//Code der linken Seite

</div>

<div class="right-side">

//Code der rechten Seite

</div>

Genau wie bei der Nutzeransicht werden die beiden Seiten in eine Division gesetzt und mit den bereits codierten Klassen left-side und right-side versehen.

Da sich die Konferenz mit einem einzigen Button steuern lässt, ist es essenziell diesen hervorzuheben. Eine angemessene Vergrößerung dieses Knopfs kommt dabei sehr gelegen. Es ist nicht weiter zu machen, als den Button mit einer neuen Css-Klasse zu auszustatten.

<input type="button" class="btn btn-primary btn-moderator" id="sendButton"

value="Besprechung"/>

Der Css-Code für die btn-moderator -Klasse sieht folgendermaßen aus:

.moderator {

padding: 10px;

font-size: 17px;

margin: 20px;

width: 71%;

}

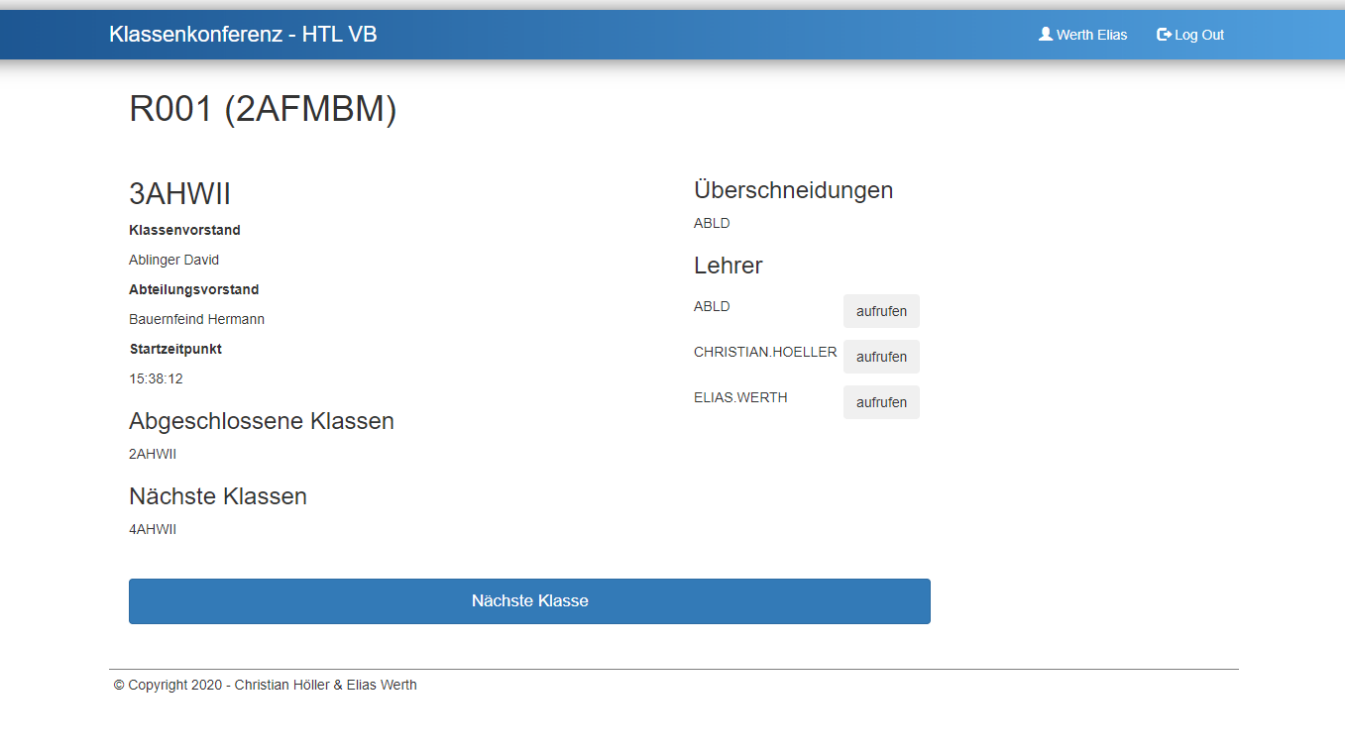


Abbildung - Moderatoransicht

# Zusammenfassung

In diesem Kapitel kann von den Schüler/innen gemeinsam eine Zusammenfassung eingefügt werden.

# Literaturverzeichnis

(kein Datum).

Bundesministerium für Bildung. (Jänner 2016). *Leitfaden für die Durchführung von Diplomarbeiten.* (Bundesministerium für Bildung, Hrsg.) Wien: Eigendruck.

N.N. (10. März 2020). *asdf*. Abgerufen am 10. März 2020 von asdf: https://login.microsoftonline.com/

N.N. (10. März 2020). *Ceteris*. Abgerufen am 10. März 2020 von Ceteris: https://www.ceteris.ag/innovative-bi/technologien/microsoft-azure/

N.N. (10. März 2020). *dotnetstudio*. Abgerufen am 10. März 2020 von dotnetstudio: https://blog.dotnetstudio.nl/connect-azure-ad-authenticated-signalr-hub-from-spfx/

N.N. (01. April 2020). *jQuery*. Abgerufen am 01. April 2020 von jQuery: https://jquery.com/

N.N. (10. März 2020). *Microsoft*. Abgerufen am 10. März 2020 von Microsoft: https://login.microsoftonline.com/

N.N. (01. April 2020). *Microsoft*. Abgerufen am 01. April 2020 von Microsoft: https://docs.microsoft.com/de-de/aspnet/signalr/overview/getting-started/introduction-to-signalr

N.N. (10. März 2020). *Newtonsoft*. Abgerufen am 10. März 2020 von Newtonsoft: https://www.newtonsoft.com/json

N.N. (01. April 2020). *Nuget.org*. Abgerufen am 01. April 2020 von Nuget.org: https://www.nuget.org/packages/Microsoft.AspNetCore.Authentication.AzureAD.UI/

N.N. (01. April 2020). *SQLiteBrowser*. Abgerufen am 01. April 2020 von SQLiteBrowser: https://sqlitebrowser.org/

N.N. (10. März 2020). *Twitter*. Abgerufen am 10. März 2020 von Twitter: https://twitter.com/sqlitebrowser/status/967968416236785665

N.N. (12. März 2020). *Wikimedia Commons*. Abgerufen am 12. März 2020 von Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Octicons-mark-github.svg

N.N. (10. März 2020). *Wikipedia*. Abgerufen am März 2020 von Wikipedia: https://de.wikipedia.org/wiki/Git

N.N. (10. März 2020). *Wikipedia*. Abgerufen am März 2020 von Wikipedia: https://de.wikipedia.org/wiki/SQLite

N.N. (10. März 2020). *Wikipedia*. Abgerufen am März 2020 von Wikipedia: https://de.wikipedia.org/wiki/C-Sharps

N.N. (10. März 2020). *Wikipedia*. Abgerufen am März 2020 von Wikipedia: https://de.wikipedia.org/wiki/Datei:Unofficial\_JavaScript\_logo\_2.svg

N.N. (10. März 2020). *Wikipedia*. Abgerufen am März 2020 von Wikipedia: https://de.wikipedia.org/wiki/Cascading\_Style\_Sheets

N.N. (10. März 2020). *Wikipedia*. Abgerufen am 10. März 2020 von Wikipedia: https://de.m.wikipedia.org/wiki/Datei:Visual\_Studio\_2013\_Logo.svg

N.N. (10. März 2020). *Wikipedia*. Abgerufen am 10. März 2020 von Wikipedia: https://de.wikipedia.org/wiki/JavaScript\_Object\_Notation

N.N. (10. März 2020). *Wikipedia*. Abgerufen am 10. März 2020 von Wikipedia: https://de.wikipedia.org/wiki/Bootstrap\_(Framework)

N.N. (10. März 2020). *Wikipedia*. Abgerufen am 10. März 2020 von Wikipedia: https://de.wikipedia.org/wiki/HTML5

N.N. (10. März 2020). *Wikipedia*. Abgerufen am 10. März 2020 von Wikipedia: https://de.wikipedia.org/wiki/Datei:JQuery-Logo.svg

N.N. (10. März 2020). *Wikipedia*. Abgerufen am 10. März 2020 von Wikipedia: https://de.wikipedia.org/wiki/MySQL

N.N. (8. 3 2020). *Wikipedia*. Abgerufen am 2020 von Wikipedia: https://de.wikipedia.org/wiki/SQLite

N.N. (14. März 2020). *Wikipedia*. Abgerufen am 14. März 2020 von Wikipedia: https://de.wikipedia.org/wiki/SQL-Injection

N.N. (01. 04 2020). *Wikipedia*. Abgerufen am 01. 04 2020 von Wikipedia: https://de.wikipedia.org/wiki/Visual\_Studio

# Abbildungsverzeichnis

[Abbildung 1 – Nutzer-Ansicht II](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654611)

[Abbildung 2 – Moderator-Ansicht II](#_Toc36654612)

[Abbildung 3 – MS Anmeldung II](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654613)

[Abbildung 4 – User View 5](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654614)

[Abbildung 5 – Moderator View 5](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654615)

[Abbildung 6 – MS Login 5](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654616)

[Abbildung 7 - Christian Höller 1](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654617)

[Abbildung 8 - Elias Werth 3](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654618)

[Abbildung 9 - MSc BSc Johannes Egger 4](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654619)

[Abbildung 10 – Screenshot alte Anwendung 6](#_Toc36654620)

[Abbildung 11 – Microsoft Login 8](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654621)

[Abbildung 12 - Visual Studio 11](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654622)

[Abbildung 13 - SQLite DB Browser 11](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654623)

[Abbildung 14 – git 11](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654624)

[Abbildung 15 – GitHub 12](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654625)

[Abbildung 16 - JSON 12](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654626)

[Abbildung 17 – Bootstrap 12](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654627)

[Abbildung 18 – Jquery 13](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654628)

[Abbildung 19 – SQLite 14](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654629)

[Abbildung 20 – SignalR 14](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654630)

[Abbildung 21 – Newtonsoft 14](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654631)

[Abbildung 22 - Microsoft Azure 15](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654632)

[Abbildung 23 - C# 16](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654633)

[Abbildung 24 – HTML 16](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654634)

[Abbildung 25 – CSS 17](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654635)

[Abbildung 26 – JavaScript 17](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654636)

[Abbildung 27 – MySQL 17](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654637)

[Abbildung 28 - SQLite Tabelle General 20](#_Toc36654638)

[Abbildung 29 - SQLite Tabelle State 21](#_Toc36654639)

[Abbildung 30 - Visual Studio Paket hinzufügen 22](#_Toc36654640)

[Abbildung 31 - Raumansicht 32](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654641)

[Abbildung 32 - Moderator Seite 33](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654642)

[Abbildung 33 - Überschneidungen 44](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654643)

[Abbildung 34 - Nutzeransicht 49](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654644)

[Abbildung 35 – Konferenzeinstellungen 54](#_Toc36654645)

[Abbildung 36 – Administratoransicht Bestätigen einer Aktion 56](#_Toc36654646)

[Abbildung 37 – Homepage Azure Portal 57](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654647)

[Abbildung 38 – Registrierung einer Anwendung 58](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654648)

[Abbildung 39 - Anwendungsinformationen 58](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654649)

[Abbildung 40 – Prinzip der Authentifizierung 59](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654650)

[Abbildung 41 – Redirect URIs 59](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654651)

[Abbildung 42 – NuGet AzureAD 60](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654652)

[Abbildung 43 – Logout-Button 62](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654653)

[Abbildung 44 – Login-Button 62](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654654)

[Abbildung 45 – Microsoft Anmeldung 63](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654655)

[Abbildung 46 – Tabelle UserRights 63](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654656)

[Abbildung 47 – Nuget WebEssentials for PWA 65](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654657)

[Abbildung 48 – DevTools Service Worker 67](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654658)

[Abbildung 49 – DevTools Manifest 70](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654659)

[Abbildung 50 - Installationsbenachrichtigung 70](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654660)

[Abbildung 51 – Installierte Version 71](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654661)

[Abbildung 52 – Aufruf-Buttons 73](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654662)

[Abbildung 53 – Tabelle TeacherCall 74](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654663)

[Abbildung 54 – Einverständniserklärung für Aufrufe 75](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654664)

[Abbildung 55 - Toast 78](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654665)

[Abbildung 56 - Navigationsbar 80](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654666)

[Abbildung 57 – Navigationsbar Toggle 80](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654667)

[Abbildung 58 – Navigationsbar Toggle ausgefächert 80](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654668)

[Abbildung 59 - Footer 81](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654669)

[Abbildung 60 – Nutzeransicht mit Einverständniserklärung 83](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654670)

[Abbildung 61 - Moderatoransicht 84](file:///C:\Users\elias\Desktop\DA\Diplomarbeit\DA-Buch\Diplomarbeit-Höller-Werth%20(8).docx#_Toc36654671)

# Codeverzeichnis

[Code 1 – JSON Lehrer 19](#_Toc36573186)

[Code 2 - JSON Klassen 19](#_Toc36573187)

[Code 3 - JSON Reihenfolge 19](#_Toc36573188)

[Code 4 - Configure Hub Route 23](#_Toc36573189)

[Code 5 - HTML moderator\_view.js script tag 24](#_Toc36573190)

[Code 6 - HTML user\_view.js script tag 24](#_Toc36573191)

[Code 7 - Modellklasse Teacher 25](#_Toc36573192)

[Code 8 - Modellklasse MyClasses 25](#_Toc36573193)

[Code 9 - Modellklasse Order 25](#_Toc36573194)

[Code 10 –General Datenbank Pfadangabe 26](#_Toc36573195)

[Code 11 – General Pfad zur JSON-Datei 27](#_Toc36573196)

[Code 12 – General Tabellenname Eigenschaft 27](#_Toc36573197)

[Code 13 – General JSON Daten als String 28](#_Toc36573198)

[Code 14 - Codebeispiel Entgegenwirken von SQL-Injections 29](#_Toc36573199)

[Code 15 – Methode Query 30](#_Toc36573200)

[Code 16 – Methode Reader 31](#_Toc36573201)

[Code 17 - Raumausgabe in HTML 32](#_Toc36573202)

[Code 18 - Weiterleitung auf Moderator-Seite 32](#_Toc36573203)

[Code 19 - Verbindungsaufaufbau zum MainHub 34](#_Toc36573204)

[Code 20 - Stehende Verbindung zum MainHub 34](#_Toc36573205)

[Code 21 – JavaScript Methode GetCurrentRoom 34](#_Toc36573206)

[Code 22 - Methode LoadModeratorPage 35](#_Toc36573207)

[Code 23 – MainHub Methode LoadModeratorContent 36](#_Toc36573208)

[Code 24 – Methode GetTeacher 37](#_Toc36573209)

[Code 25 – Methode GetButtonText 37](#_Toc36573210)

[Code 26 – JavaScript ReceiveModeratorContent 38](#_Toc36573211)

[Code 27 – JavaScript WriteTeachersWithButtonsIntable 38](#_Toc36573212)

[Code 28 – Lehrer title-attribut 39](file:///C:\Users\elias\Downloads\Diplomarbeit-Höller-Werth%20(7).docx#_Toc36573213)

[Code 29 – JavaScript Methode getShorthandForTeacher 39](#_Toc36573214)

[Code 30 - Hub Methode LoadGeneralContent Switch 40](#_Toc36573215)

[Code 31 - Hub Methode LoadGeneralContent Klassen 40](#_Toc36573216)

[Code 32 – Methode GetClassesCompleted 41](#_Toc36573217)

[Code 33 – Methode GetClassesNotEdited 42](#_Toc36573218)

[});Code 34 – JavaScript ReceiveGeneralContent 43](#_Toc36573219)

[Code 35 – JavaScript Methode WriteDataInTable 43](#_Toc36573220)

[Code 36 – Überschneidungen Konferenzstatusüberprüfung 44](#_Toc36573221)

[Code 37 – Überschneidungen gemeinsame Lehrer 45](#_Toc36573222)

[Code 38 – JavaScript ReceiveIntersections 45](#_Toc36573223)

[Code 39 – JavaScript Methode WriteIntersectionsInTable 46](#_Toc36573224)

[Code 40 – JavaScript Button-event 46](#_Toc36573225)

[Code 41 – MainHub ConferenceAction 47](#_Toc36573226)

[Code 42 – MainHub WriteTimeInDatabase 47](#_Toc36573227)

[Code 43 – MainHub SetStateOfConference 48](#_Toc36573228)

[Code 44 – MainHub NextClass 48](#_Toc36573229)

[Code 45 – JavaScript Verbindungsaufbau zum MainHub 50](#_Toc36573230)

[Code 46 – MainHub LoadRooms 50](#_Toc36573231)

[Code 47 – JavaScript ReceiveRooms 51](#_Toc36573232)

[Code 48 – JavaScript ReceiveGeneralContent 52](#_Toc36573233)

[Code 49 – JavaScript WriteUserViewInformation 53](#_Toc36573234)

[Code 50 – Admin\_Settings OnPostReset 54](#_Toc36573235)

[Code 51 – Administratoransicht Methode OnPostSetJSONData 55](#_Toc36573236)

###### - Terminplan

**[](https://www.google.at/imgres?imgurl=https://previews.123rf.com/images/daksel/daksel1111/daksel111100005/11137191-gelben-Aufkleber-beigef-gten-Zeichnung-pin-Lizenzfreie-Bilder.jpg&imgrefurl=https://de.123rf.com/clipart-vektorgrafiken/stift_icon.html&docid=hNG3z7d6vhg7fM&tbnid=AFQGbeFzQxalXM:&vet=10ahUKEwjZhtLw7IPUAhXJIJoKHbiQCo4QMwhbKDEwMQ..i&w=1300&h=1300&bih=822&biw=1593&q=pinnadel%20clipart&ved=0ahUKEwjZhtLw7IPUAhXJIJoKHbiQCo4QMwhbKDEwMQ&iact=mrc&uact=8)****⇨ Kurzeinführung MS Project.pdf**

Microsoft Project (Bestandteil der Microsoft Schüler-CD!) ist nur eine Möglichkeit, um professionelle Terminpläne zu erstellen! Welches Programm auch immer, ein Terminplan ist Pflicht!

###### - Begleitprotokolle

**⇨ Begleitprotokoll.pdf**