

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


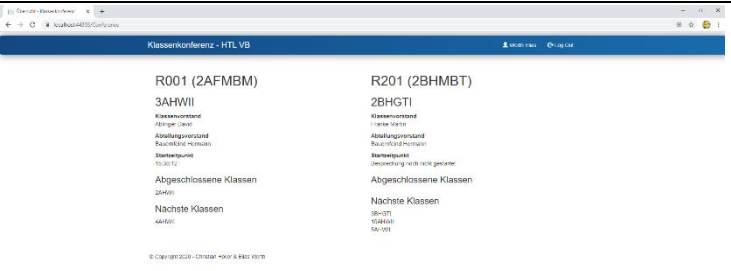
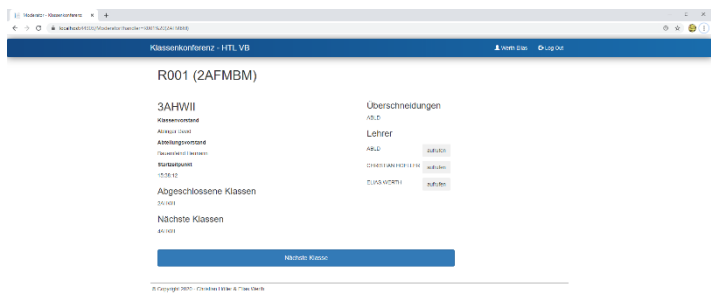
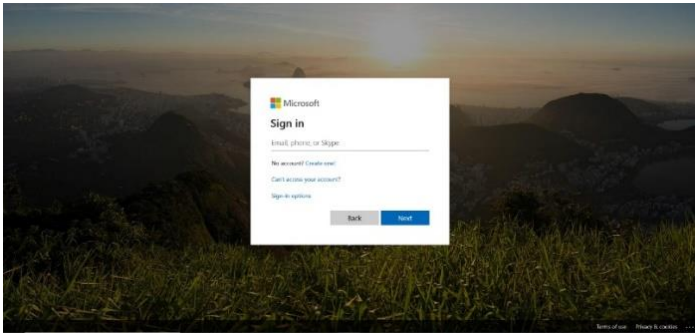
DOKUMENTATION


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.
------------	---

	HTBLA Vöcklabruck Höhere Lehranstalt für Wirtschaftsingenieurwesen Ausbildungsschwerpunkt Betriebsinformatik	Reife- und Diplomprüfung
<p>Typische Grafik, Foto etc. (mit Erläuterung)</p>	<div data-bbox="699 275 1433 544">  <p><i>Abbildung 1 – Nutzer-Ansicht</i></p> </div> <div data-bbox="722 723 1436 1014">  <p><i>Abbildung 2 – Moderator-Ansicht</i></p> </div> <div data-bbox="730 1265 1428 1597">  <p><i>Abbildung 3 – Microsoft Anmeldung</i></p> </div>	
Teilnahme an Wettbewerben, Auszeichnungen	Keine Teilnahmen	
Möglichkeiten der Einsichtnahme in die Arbeit	Schulbibliothek	

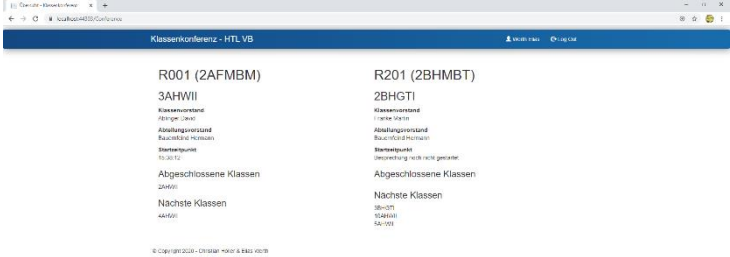
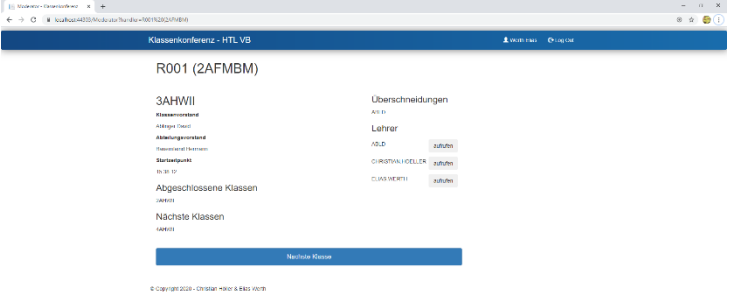
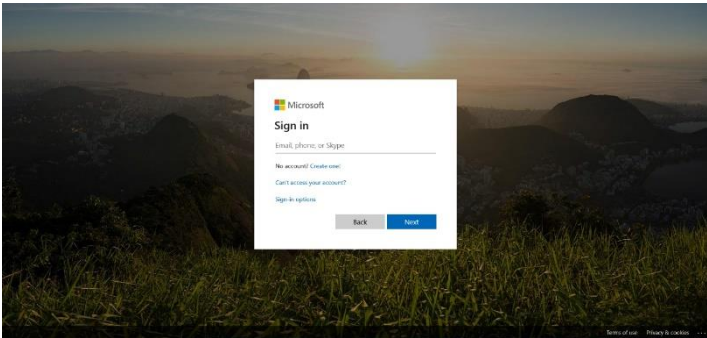
	HTBLA Vöcklabruck Höhere Lehranstalt für Wirtschaftsingenieurwesen Ausbildungsschwerpunkt Betriebsinformatik	Reife- und Diplomprüfung
---	--	-------------------------------------

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

DIPLOMA THESIS

DOCUMENTATION

Author(s)	Christian Höller Elias Werth
Form Academic year	5 th 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.

<p>Illustrative graph, photo (incl. explanation)</p>	 <p><i>Abbildung 4 – User View</i></p>  <p><i>Abbildung 5 – Moderator View</i></p>  <p><i>Abbildung 6 – MS Login</i></p>
<p>Participation in competitions</p> <p>Awards</p>	<p>No Participaion</p>

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	Bootstrap.....	12
2.7	jQuery	13
2.8	Razor	13
2.9	ToastR	13
2.10	NuGet-Pakete	14
2.10.1	SQLite	14
2.10.2	ASP.net SignalR	14
2.10.3	Newtonsoft.JSON.....	14
2.10.4	AzureAD.....	15

2.10.5	WebEssentials.AspNetCore.PWA	15
3	Sprachen	16
3.1	C#	16
3.2	HTML	16
3.3	CSS.....	17
3.4	JavaScript	17
3.5	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	49
4.7.2	Laden der Informationen	51
4.8	Ansicht Administrator	53
4.8.1	Konferenz zurücksetzen	53
4.8.2	In Datenbank schreiben.....	54
4.8.3	Bestätigen der Aktionen	55
4.9	Authentifizierung mittels Microsoft AzureAD	56
4.9.1	Registrierung im Azure-Portal	56
4.9.2	Konfiguration der Applikation.....	57
4.9.3	Implementierung des Logins	60
4.9.4	Weiterleitung nach erfolgreicher Authentifizierung	63
4.10	Progressive Web-App (PWA)	66
4.10.1	Service Worker.....	66
4.10.2	Manifest.....	68
4.11	Aufrufen von Lehrkräften	73
4.11.1	Hinzufügen von Aufruf-Buttons	73
4.11.2	Senden des Aufrufs mit SignalR	74
4.11.3	Speichern der Aufrufe	75
4.11.4	Empfangen des Aufrufs.....	76
4.11.5	Visualisierung der Benachrichtigung	78
4.11.6	Text-to-Speech.....	80
4.12	Design und Layout der Web-App.....	81
4.12.1	Navigationsmenü	81
4.12.2	Footer.....	83
4.12.3	Design der Nutzeransicht.....	84
4.12.4	Design der Moderatoransicht	86

5	Literaturverzeichnis	88
6	Abbildungsverzeichnis	92
7	Codeverzeichnis	94
ANHANG A	– Terminplan	97
ANHANG B	- Begleitprotokolle	98
ANHANG C	Tätigkeitsprotokoll	99

1 Allgemeines

1.1 Vorstellung des Projektteams

1.1.1 Diplomanden

1.1.1.1 Christian Höller

Klasse: 5AHWII

Geburtsdatum: 03.10.2000

Wohnort: Attnang-Puchheim

E-Mail: christian.hoeller@htlvb.at

Ausbildungszweig: Betriebsinformatik



*Abbildung 7 - Christian
Höller*

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

1.1.1.2 Elias Werth

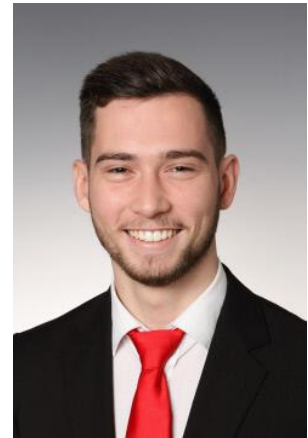
Klasse: 5AHWII

Geburtsdatum: 05.05.2001

Wohnort: Edt bei Lambach

E-Mail: elias.werth@htlvb.at

Ausbildungszweig: Betriebsinformatik



*Abbildung 8 - Elias
Werth*

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

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 wir uns herzlich bedanken möchten.



*Abbildung 9 - MSc
BSc Johannes Egger*

1.2 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.

1.3 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.

1.4 Individuelle Zielsetzungen

1.4.1 Zielsetzung Höller

1.4.1.1 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.

1.4.1.2 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.

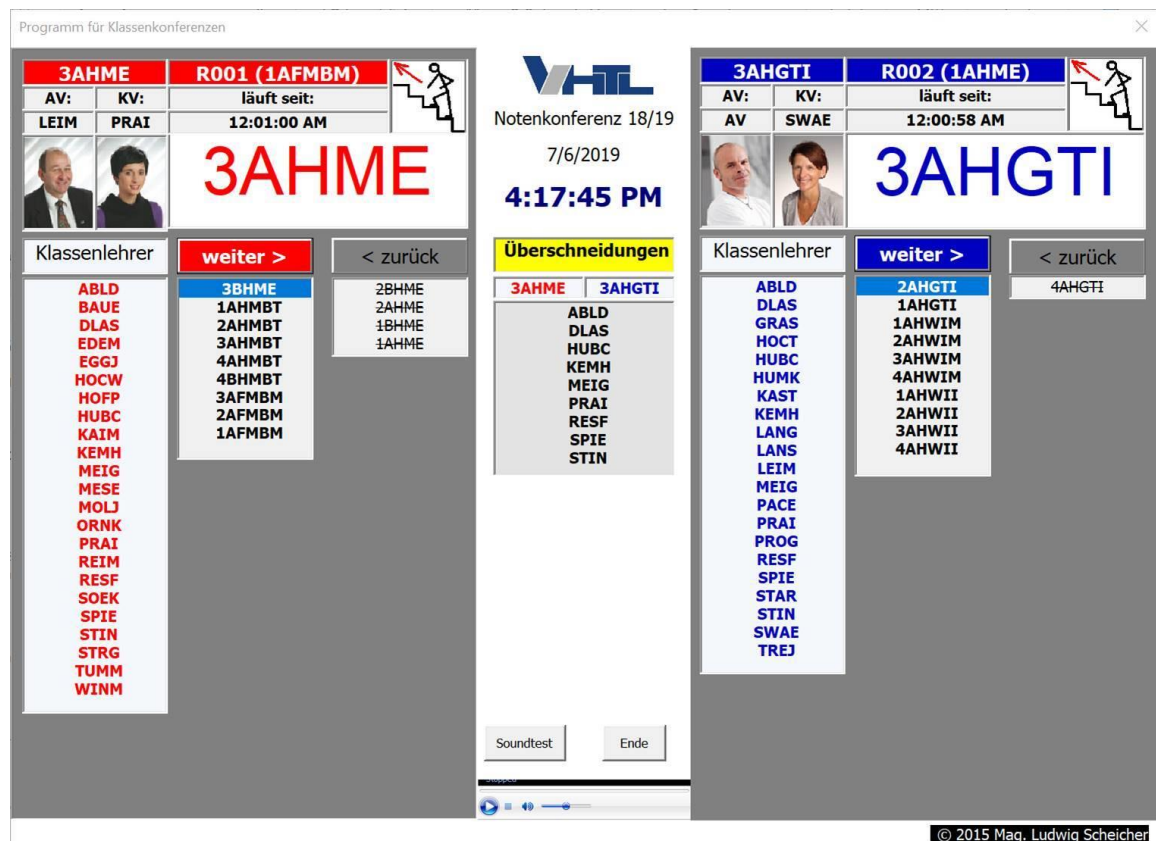


Abbildung 10 – Screenshot alte Anwendung

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.

1.4.1.3 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.

1.4.2 Zielsetzung Werth

1.4.2.1 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-E-Mail-Adresse) 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 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.

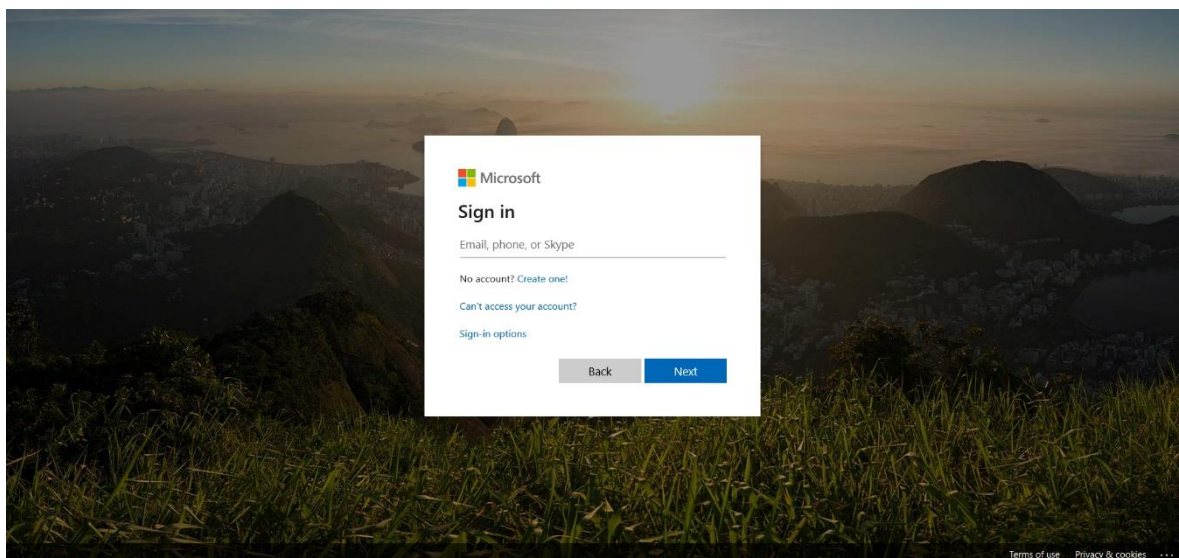


Abbildung 11 – Microsoft Login

Der wohl größte Vorteil, der sich durch die Verwendung des Office-365-Konto ergibt, ist der, dass man nun Rollen/Rechte vergeben kann. Durch eine Überprüfung dieser Rechte nach der Authentifizierung, lässt sich leicht feststellen, ob der Benutzer zur Moderatoransicht oder zur Nutzeransicht weitergeleitet werden soll.

1.4.2.2 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 Webseiten 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.



Abbildung 12 – PWA iPhone

1.4.2.3 Aufrufen von Lehrern

Ein Moderator (Abteilungsvorstand) soll die Möglichkeit haben, nach Bedarf fehlende Lehrkräfte aufrufen zu können. Dieser hat in der Moderatoransicht eine Übersicht über alle Lehrkräfte, 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 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.

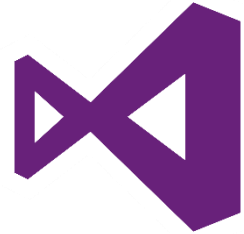
1.4.2.4 Design & Layout

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 Nutzeransicht sollen angemessen dargestellt werden. Das Layout im Allgemeinen soll sich stets an das Endgerät anpassen, um mögliche Bugs der Anzeige zu vermeiden. Die Themenfarbe der Website bezieht sich auf die Farben des HTL-Logos.

2 Verwendete Programme und Tools

2.1 Visual Studio

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

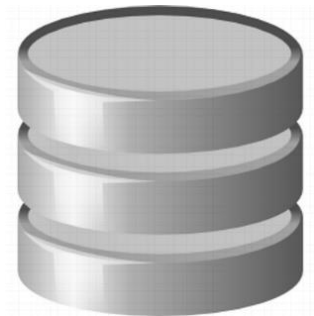


(N.N., Wikipedia, 2020)

Abbildung 13 - Visual Studio
(N.N., Wikipedia, 2020)

2.2 DB Browser for SQLite

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



(N.N., SQLiteBrowser, 2020)

Abbildung 14 - SQLite DB
Browser
(N.N., Twitter, 2020)

2.3 Git

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



Abbildung 15 – git
(N.N., Wikipedia, 2020)

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

2.4 GitHub

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.



Abbildung 16 – GitHub
(N.N., Wikimedia Commons, 2020)

2.5 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.

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

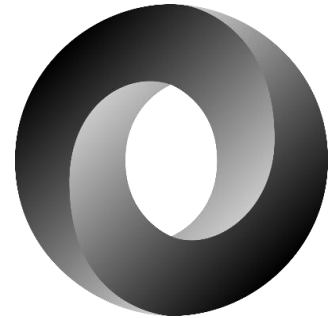


Abbildung 17 - JSON
(N.N., Wikipedia, 2020)

2.6 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.

(N.N., Wikipedia, 2020)



Abbildung 18 – Bootstrap
(N.N., Wikipedia, 2020)

2.7 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

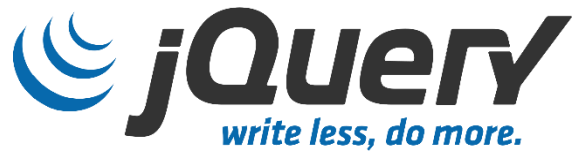


Abbildung 19 – jQuery

(N.N., Wikipedia, 2020)

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.

(N.N., jQuery, 2020)

2.8 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)

2.9 Toastr

Toastr ist eine JavaScript-Bibliothek für nicht blockierende Benachrichtigungen vom Typ Gnome/Growl. jQuery ist erforderlich. Ziel ist es, eine einfache Kernbibliothek zu erstellen, die angepasst und erweitert werden kann.

(N.N., GitHub, 2020)

2.10 NuGet-Pakete

2.10.1 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 20 – SQLite
(N.N., Wikipedia, 2020)

(N.N., Wikipedia, 2020)

2.10.2 ASP.net SignalR

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.



Abbildung 21 – SignalR
(N.N., dotnetstudio, 2020)

(N.N., Microsoft, 2020)

2.10.3 Newtonsoft.JSON

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

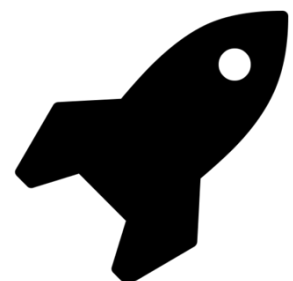


Abbildung 22 – Newtonsoft
(N.N., Newtonsoft, 2020)

(N.N., Newtonsoft, 2020)

2.10.4 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.

(N.N., NuGet, 2020)



Abbildung 23 - Microsoft Azure

(N.N., Ceteris, 2020)

2.10.5 WebEssentials.AspNetCore.PWA

Ist ein NuGet-Paket, welches für Unterstützung des Service Workers und Web-App-Manifests in ASP.NET Core-Projekten sorgt. Zusätzlich bietet es Offline-Unterstützung sowie clientseitiges Caching.

(N.N., NuGet, 2020)

3 Sprachen

3.1 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 24 - C#
(N.N., Wikipedia, 2020)

3.2 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 25 – HTML
(N.N., Wikipedia, 2020)

3.3 CSS

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 26 – CSS
(N.N., Wikipedia, 2020)

3.4 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.

(N.N., Wikipedia, 2020)



Abbildung 27 – JavaScript
(N.N., Wikipedia, 2020)

3.5 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.

(N.N., Siteground, 2020)

(N.N., Wikipedia, 2020)



Abbildung 28 – MySQL
(N.N., Wikipedia, 2020)

4 Realisierung

4.1 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.

4.1.1 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)

4.1.1.1 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

4.1.1.2 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

4.1.1.3 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

4.1.2 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“.

4.1.2.1 Besprochene Klassen

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

Tabelle: General Neue Zeile

	ID	Room	ClassOrder	Status	Start	End
	Filtern	Filtern	Filtern	Filtern	Filtern	Filtern
1	2AHWII	R001 (2AFMBM)	1	completed	16:50:36	16:50:37
2	3AHWII	R001 (2AFMBM)	2	not edited	16:50:37	NULL
3	4AHWII	R001 (2AFMBM)	3	not edited	NULL	NULL
4	2BHGTI	R201 (2BHMBT)	1	not edited	NULL	NULL
5	3BHGTI	R201 (2BHMBT)	2	not edited	NULL	NULL
6	5AHWII	R201 (2BHMBT)	4	not edited	NULL	NULL


Abbildung 29 - SQLite Tabelle General

4.1.2.2 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.

Tabelle:  State ▼

	Room	Status
	Filtern	Filtern
1	R001 (2AFMBM)	inactive
2	R201 (2BHMBT)	completed

Abbildung 30 - SQLite Tabelle State

4.2 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.

4.2.1.1 Hinzufügen von SignalR

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

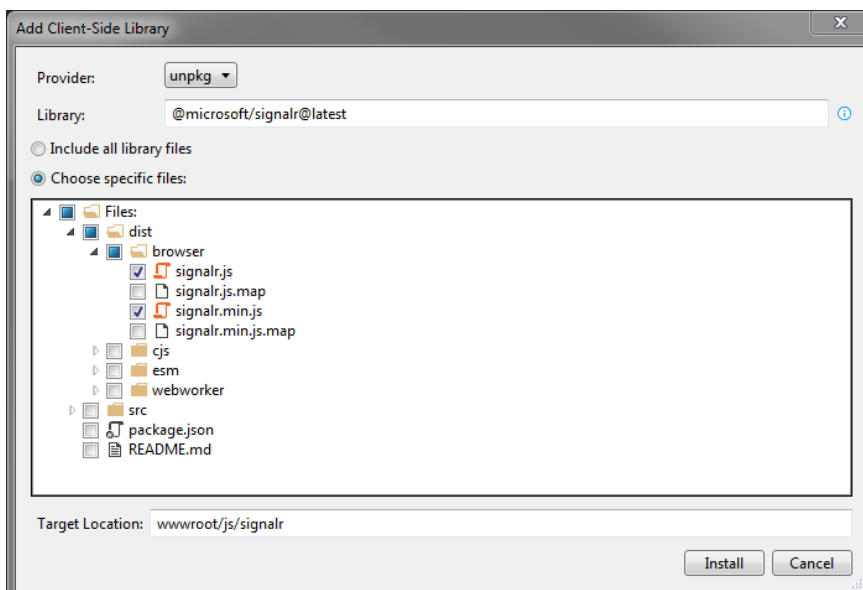


Abbildung 31 - Visual Studio Paket hinzufügen

Nachdem die Bibliothek installiert wurde, muss 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.

4.2.1.2 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

4.2.1.3 JavaScript-Implementierung

Nachdem die Hub-Klasse konfiguriert wurde, muss 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

4.3 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.

4.3.1 Lehrer

```
public class Teacher
{
    public string ID { get; set; }
    public string Name { get; set; }
}
```

Code 7 - Modellklasse Teacher

4.3.2 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

4.3.3 Reihenfolge

```
public class Order
{
    public string Room { get; set; }
    public List<string> Classes { get; set; }
}
```

Code 9 - Modellklasse Order

4.4 Klassen

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

4.4.1 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

4.4.1.1 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

4.4.1.2 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

4.4.1.3 JSON-Dateiinhalte 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

4.4.2 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

4.4.2.1 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[] parametervvalues)
{
    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 (parametervvalues != null)
            {
                foreach (var param in parametervvalues)
                {
                    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.

4.5 Raumauswahl

Jeder 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.

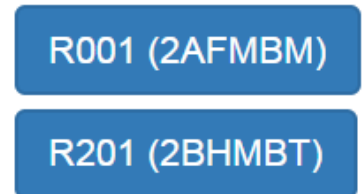


Abbildung 32 - Raumansicht

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">
    <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

4.6 Steuerung der Konferenz

Klassenkonferenz - HTL VB

Höller Christian Log Out

R001 (2AFMBM)

3AHWII

Klassenvorstand

Ablinger David

Abteilungsvorstand

Bauernfeind Hermann

Startzeitpunkt

20:03:29

Abgeschlossene Klassen

2AHWII

Nächste Klassen

4AHWII

Überschneidungen

ABLD

Lehrer

GRUG

ABLD

MOLJ

PRIL

LORR

Nächste Klasse

© Copyright 2020 - Christian Höller & Elias Werth

Abbildung 33 - 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 „ansufen“ 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 welchem mithilfe von JavaScript die gewünschten Informationen eingefügt werden.

4.6.1 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 - Verbindungsaufbau 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

4.6.2 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 der 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

4.6.3 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

4.6.3.1 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

4.6.3.2 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 geparkt und für jedes Element eine neue Tabellenreihe erstellt.

```
function WriteTeachersInTable(teacherArray) {

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

    if (teacherArray == "") {
        $("#teachers").append("<tr><td>Keine Lehrer</td></tr>");
    }
    else {
        var teacherData;
        var parsedArray = JSON.parse(teacherArray);

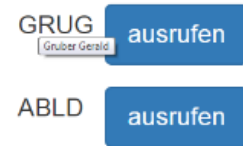
        for (var i = 0; i < parsedArray.length; i++) {
            var teacherID = parsedArray[i].ID;
            var fullName = parsedArray[i].Name;

            teacherData = "<td><p title='" + fullName + "'" +
getShorthandForTeacher(teacherID) + "</p></td>";
            $("#teachers").append("<tr>" + teacherData + "</tr>");
        }
    }
}
```

Code 27 – JavaScript WriteTeachersInTable

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

Lehrer



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

4.6.4 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

4.6.4.1 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 JSONArray(classes).ToString();
    }
    else
    {
        int index = classes.IndexOf(GetCurrentClassName());
        return new JSONArray(classes.Take(index)).ToString();
    }
}
```

Code 32 – Methode GetClassesCompleted

4.6.4.2 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()
{
    JSONArray classesNotedited = new JSONArray();
    var classes = GetOrderList().Find(order => order.Room == Currentroom).Classes;

    if (GetCurrentStateOfConference() == "completed")
    {
        classesNotedited.Add("Keine weiteren Klassen");
    }
    else
    {
        int index = classes.IndexOf(GetCurrentClassName()) + 1;
        if (index == classes.Count)
        {
            classesNotedited.Add("Keine weiteren Klassen");
        }
        else
        {
            classesNotedited = new JSONArray(classes.GetRange(index, classes.Count -
index));
        }
    }
    return classesNotedited.ToString();
}
```

Code 33 – Methode GetClassesNotEdited

Da die Informationen an JavaScript in Form von JSON übergeben wird, müssen diese erst geparkt 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

4.6.5 Überschneidungen

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

Überschneidungen

Molnar Johannes

Gruber Gerald

Abbildung 34 - 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();

JSONArrayIntersections = new JSONArray(JsonConvert.SerializeObject(intersectedTeachers));
```

Code 37 – Überschneidungen gemeinsame Lehrer

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

4.6.5.1 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.

4.6.6 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

4.6.6.1 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

4.6.6.2 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

4.7 Nutzeransicht

Klassenkonferenz - HTL VB

Höller Christian Log Out

R001 (2AFMBM)

3AHWII

Klassenvorstand
Ablinger David

Abteilungsvorstand
Bauernfeind Hermann

Startzeitpunkt
20:03:29

Abgeschlossene Klassen

2AHWII

Nächste Klassen

4AHWII

R201 (2BHMBT)

2BHGTI

Klassenvorstand
Franke Martin

Abteilungsvorstand
Bauernfeind Hermann

Startzeitpunkt
Besprechung noch nicht gestartet

Abgeschlossene Klassen

Nächste Klassen

3BHGTI
10AHWII
5AHWII

© Copyright 2020 - Christian Höller & Elias Werth

Abbildung 35 - 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.

4.7.1 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 geparkt 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

4.7.2 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("r1_room").innerHTML == obj_parsed.room) {  
        WriteUserViewInformation("r1_", obj_parsed);  
    }  
    else {  
        WriteUserViewInformation("r2_", 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.

```
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.

4.8 Ansicht Administrator

Konferenzeinstellungen

Konferenz zurücksetzen

Daten in DB schreiben

Abbildung 36 – 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.

4.8.1 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

4.8.2 In Datenbank schreiben

Dieser Button wurde erstellt, um Daten aus der JSON-Datei in die Datenbank zu schreiben. 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

4.8.3 Bestätigen der Aktionen

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

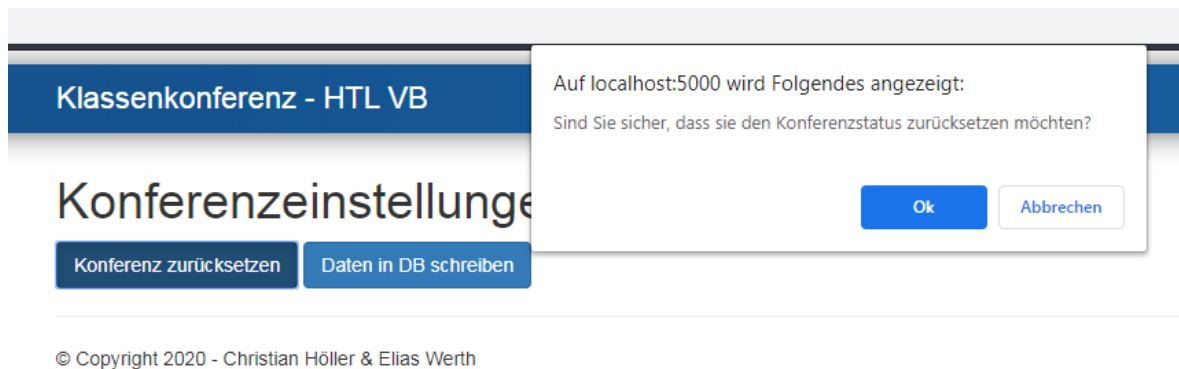


Abbildung 37 – Administratoransicht Bestätigen einer Aktion

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.

4.9 Authentifizierung mittels Microsoft AzureAD

Ein wichtiger Teil der Aufgabenstellung ist die Authentifizierung. Mithilfe von Microsoft AzureAD wird dies realisiert. Die Authentifizierung umfasst nicht nur den eigentlichen Login, sondern auch die Rechteverteilung und die dazugehörige Weiterleitung auf die entsprechende Ansicht.

4.9.1 Registrierung im Azure-Portal

Bevor man fähig ist, den Office-Login zu implementieren, muss man seine Anwendung im Azure-Portal registrieren. Im ersten Schritt navigiert man zu *portal.azure.com* (s. Abb. 38).

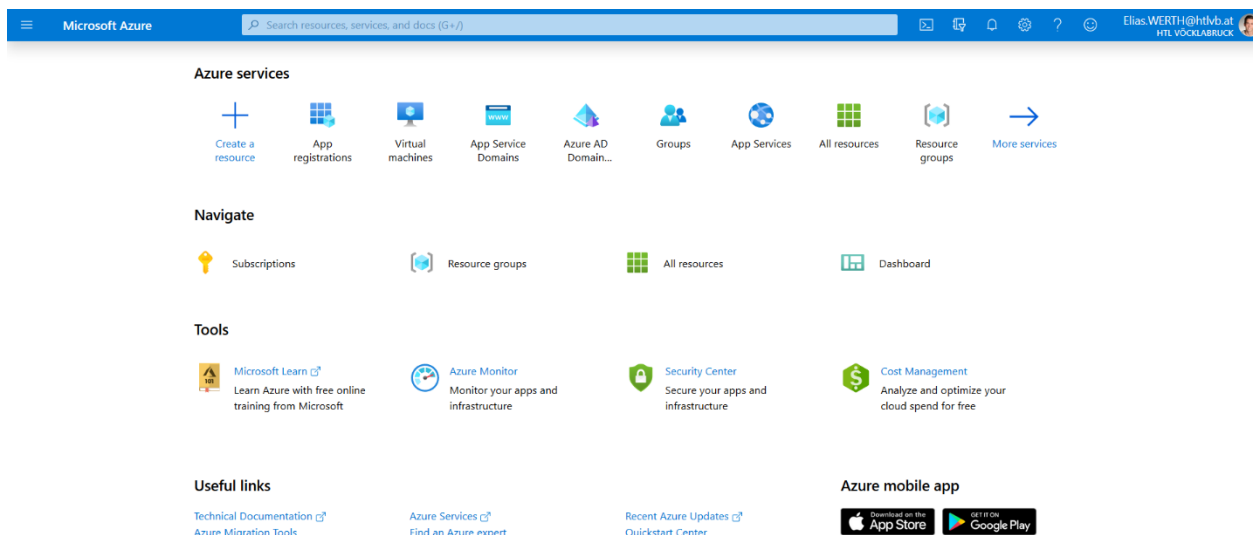


Abbildung 38 – Homepage Azure Portal

Unter dem Reiter „App registrations“ (s. Abb. 38) 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 (s. Abb. 39) muss man nur den Namen der Applikation eingeben und schon ist die Registrierung abgeschlossen.

[Home](#) > [App registrations](#) > Register an application

Register an application

* Name

The user-facing display name for this application (this can be changed later).

Supported account types

Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (HTL Vöcklabruck only - Single tenant)
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

By proceeding, you agree to the [Microsoft Platform Policies](#) 

[Register](#)

Abbildung 39 – Registrierung einer Anwendung

4.9.2 Konfiguration der Applikation

Nach der Registrierung ist man sofort in der Lage seine Anwendung zu konfigurieren, denn man bekommt darauf all die wichtigen Informationen (s. Abb. 40), die man dazu braucht.

Display name : [Managementsystem für Klassenkonferenzen](#)
Application (client) ID : e6aeade9-43f0-4743-a205-3f0397188d51
Directory (tenant) ID : 81de7086-f6b3-4e4b-9faf-18d4a406e66d
Object ID : ac081cef-cdd6-4593-8df6-528320259034

Abbildung 40 - Anwendungsinformationen

Besonders die angegebenen IDs (s. Abb. 40) 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. (N.N., Oauth, 2020)*
- 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. (N.N., Microsoft, 2020)*

Die Quickstart-Guides dieser Seite bieten eine perfekte Schritt-für-Schritt-Anleitung an, die für das Managementsystem hervorragend funktionieren. Zusätzlich wird hier (s. Abb. 41) auch erklärt, wie die Authentifizierung funktioniert.

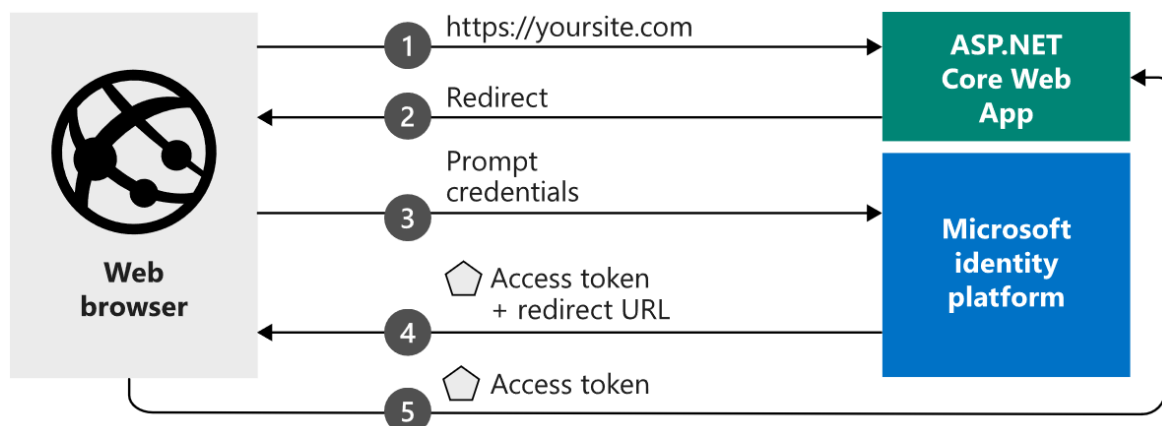


Abbildung 41 – Prinzip der Authentifizierung

Im ersten Schritt muss im Azure-Portal der Redirect-URL konfiguriert werden (s. Abb. 42). Ohne den richtigen Redirect-URL ist es nicht möglich, die Anwendung nach dem Login bzw. Logout zurückzuleiten, vorausgesetzt die Authentifizierungsantwort (Token) ist erfolgreich.

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating users. Also referred as reply URLs. [Learn more about redirect URIs and the restrictions](#)



[Add URI](#)

Logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.



Abbildung 42 – Redirect URIs

Der Port ist in diesem Fall nicht weiter wichtig. Essenziell ist nur der *localhost* und vor allem das *https://*. Dasselbe gilt auch für den Logout-URL. Die Endung *signin-oidc* ist der Standardwert der *OIDC-Client-Middleware* (N.N., GitHub, 2020) 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 (s. Abb. 40).

```
"AzureAD": {
  "Instance": "https://login.microsoftonline.com",
  "Domain": "htlvb.at",
  "ClientId": "e6aeade9-43f0-4743-a205-3f0397188d51",
  "TenantId": "81de7086-f6b3-4e4b-9faf-18d4a406e66d"
},
```

Code 52 – AzureAD appsettings

4.9.3 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 (s. Abb. 43).

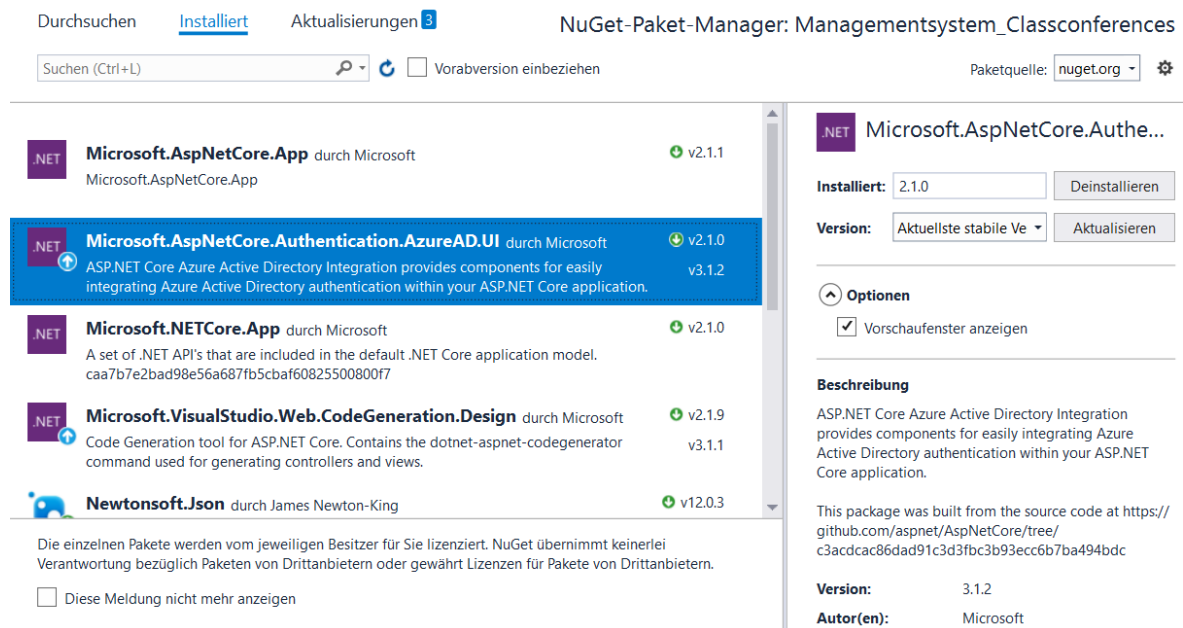


Abbildung 43 – NuGet AzureAD

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();
    [...]
}
```

Code 53 – Implementation in der Startup-Klasse

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

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 die `_Layout.cshtml`-Page. Vorgesehen ist ein Button mit entsprechendem Icon rechtsbündig im Navigationsmenü.

```
<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>
```

Code 54 – Implementation in die Navigationsleiste

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 sollen, wenn der Benutzer 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 zur Office-365-Startseite.

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



Abbildung 46 – Login-Button

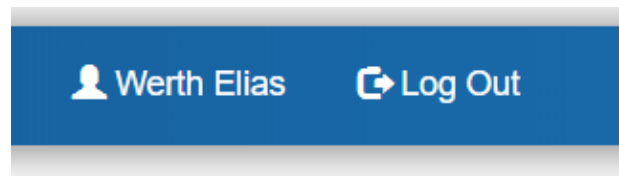


Abbildung 45 – Logout-Button

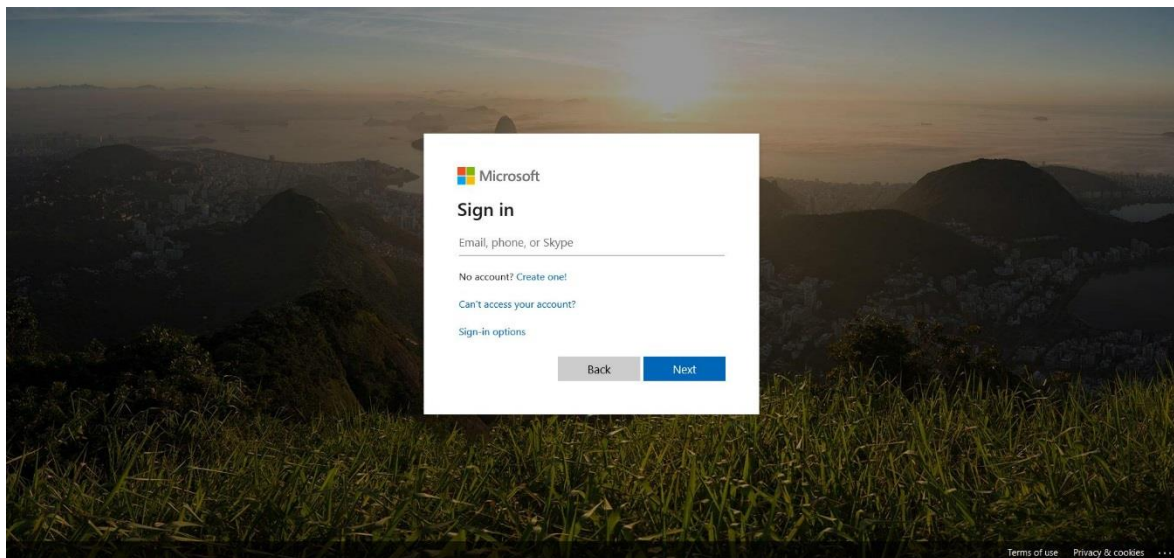


Abbildung 44 – Microsoft Anmeldung

4.9.4 Weiterleitung nach erfolgreicher Authentifizierung

Ein weiterer 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 der Raumauswahl und der normale Benutzer soll zur Konferenzübersicht gelangen. Zusätzlich wird der Administrator (Johannes Egger) zu der Seite für die Administratoreinstellungen 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

	TeacherID	Name	UserGroup
	Filtern	Filtern	Filtern
1	baue@htlvb.at	Bauernfeind Hermann	0
2	leim@htlvb.at	Leibner Markus	0
3	molj@htlvb.at	Molnar Johannes	0
4	eggj@htlvb.at	Egger Johannes	1

Abbildung 47 – Tabelle UserRights

Lehrkraft besitzt. 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.

Im nächsten Schritt wird die Abfrage nach den Rechten ins Programm implementiert. Das Wichtigste spielt sich dabei in der `Index.cshtml.cs` in der `OnGet()`-Methode ab. Diese Methode wird jedes Mal aufgerufen, wenn die Index-Seite neu lädt (also auch nach der Anmeldung).

```
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;
```

Code 55 – Abfrage der Rechte

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 `null` zurückgegeben. Andernfalls wird im nächsten Schritt, wie bei der Authentifizierung, mit `ClaimsIdentity` die E-Mail-Adresse der angemeldeten

Lehrkraft festgestellt. 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, 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 Konferenzübersicht (`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`).

4.10 Progressive Web-App (PWA)

Um die ASP.NET Core Website in eine PWA (s. Kapitel 1.4.2.2) umzuwandeln, sind einige Schritte notwendig. Der erste und wohl wichtigste Schritt besteht daraus, einen Service Worker und ein Manifest in die Anwendung zu implementieren.

4.10.1 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 vereinfacht das Caching 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.

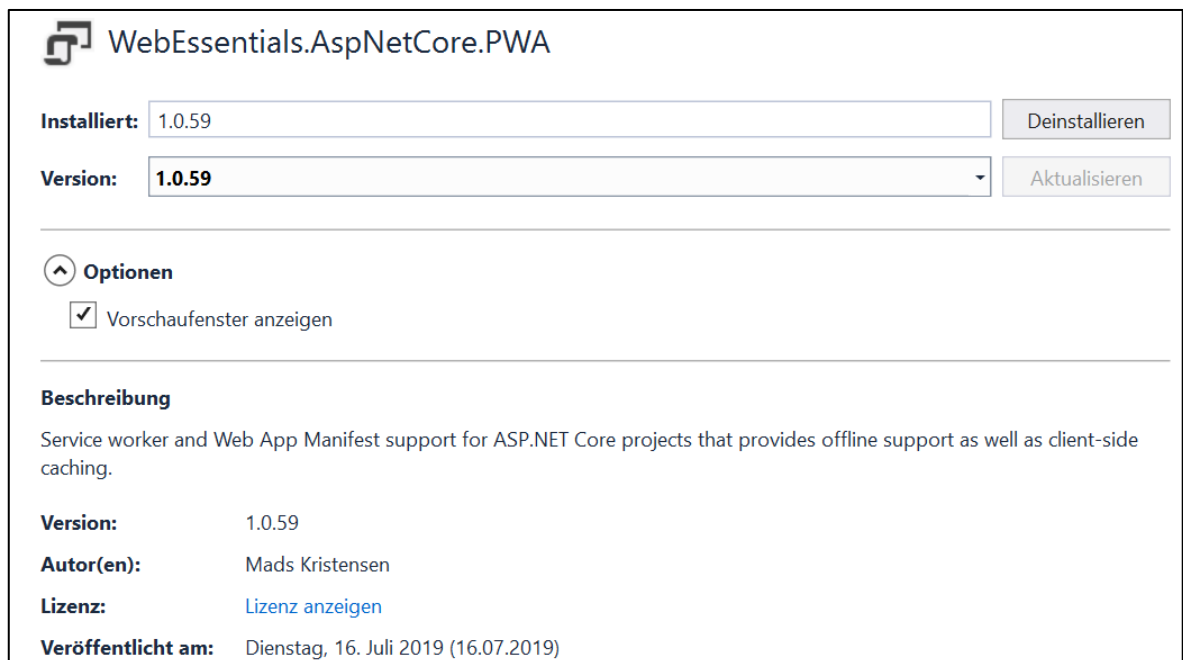


Abbildung 48 – Nuget WebEssentials for PWA

Dieses NuGet-Paket erstellt nicht nur den Service Worker, es kümmert sich zusätzlich 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();
    [...]
}
```

Code 56 – Hinzufügen der PWA in die Startup-Klasse

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
  }
}
```

Code 57 – PWA-Einstellungen

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. (N.N., Elmah, 2020)

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. Wenn die neueste Version nicht verfügbar ist, wird auf den Cache zurückgegriffen. (N.N., Elmah, 2020)

`routeToPreCache` gibt an, welche Seiten vor dem Cachen gespeichert werden sollen. Dies bedeutet, dass sie dem Cache hinzugefügt werden, sobald der Service Worker registriert ist. Dies ist besonders nützlich, um alle Unterseiten auf der Webpage zwischenzuspeichern, damit der Benutzer diese besuchen kann, während er offline ist. (N.N., Elmah, 2020)

`registerServiceWorker` und `registerWebmanifest` dienen zum Deaktivieren einzelner Funktionen im Service. (N.N., Elmah, 2020)

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.

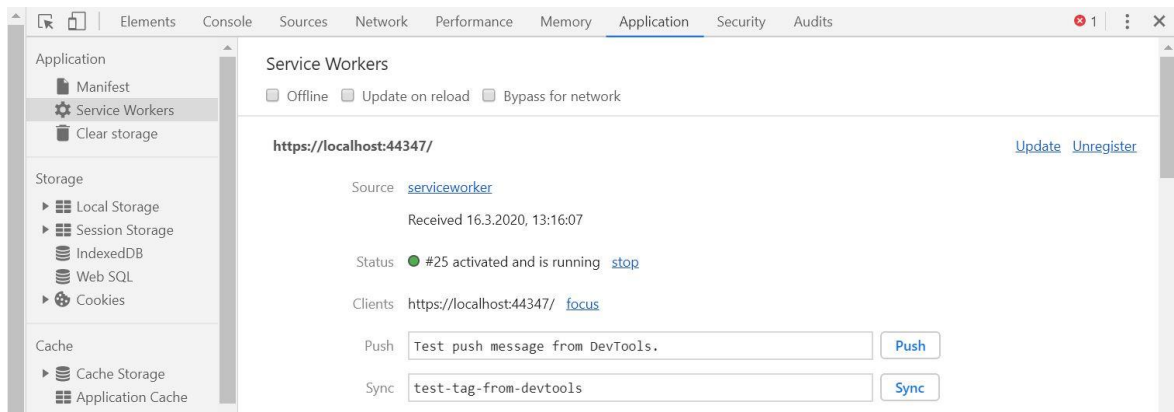


Abbildung 49 – DevTools Service Worker

4.10.2 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 dem Browser auch, wie er sich verhalten soll, wenn die Anwendung am Mobiltelefon oder Desktop des Benutzers installiert wird.

Das Erstellen des Manifests ist keine große Herausforderung, jedoch muss beachtet werden, dass sich die JSON-Datei im „wwwroot“-Ordner befindet. Der Name dieser Datei spielt 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"
}
  
```

Code 58 – Inhalt des Manifests

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. (N.N., web.dev, 2020)

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 wirkt 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. (N.N., web.dev, 2020)

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 sind. Die benötigten Icons müssen sich in dem Ordner „images“ der Projektmappe befinden. (N.N., web.dev, 2020)

Die `start_url` ist erforderlich und teilt dem Browser mit, wo die Anwendung beim Ausführen gestartet werden soll. Es verhindert, dass die App auf der Seite gestartet wird, auf der sich der Benutzer befand, als er die App zu seinem Startbildschirm hinzufügte. (N.N., web.dev, 2020)

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-Screen zu sehen ist das Icon. (N.N., web.dev, 2020)

Die `theme_color` ist ein einfaches Attribut, das bestimmt, welche Farbe die Toolbar annimmt. Wie der Name schon sagt, ist es das Farbschema der Seite. (N.N., web.dev, 2020)

Das Manifest ist nun komplett und der letzte essenzielle Schritt ist, 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>
```

Code 59 – Verlinkung des Manifests

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.

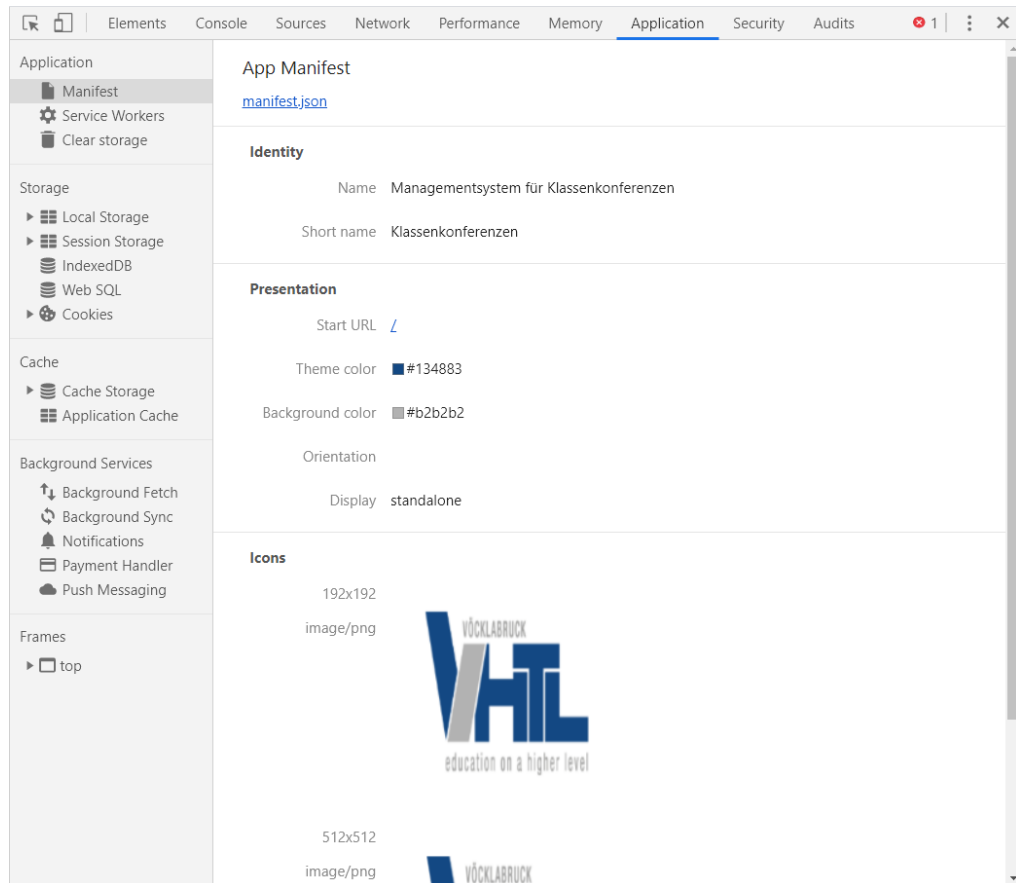


Abbildung 50 – DevTools Manifest

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

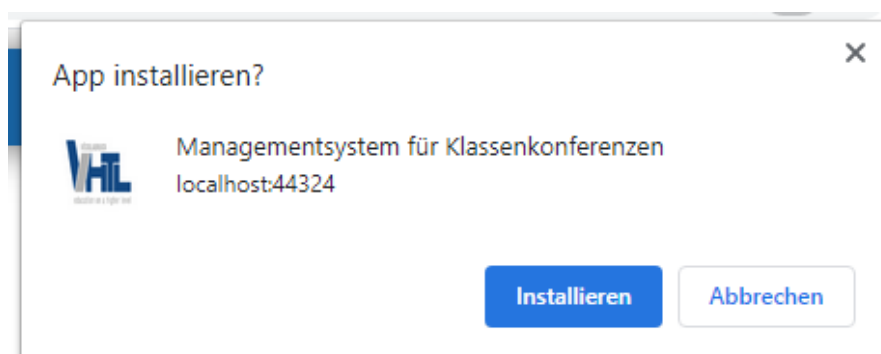


Abbildung 51 - Installationsbenachrichtigung

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.

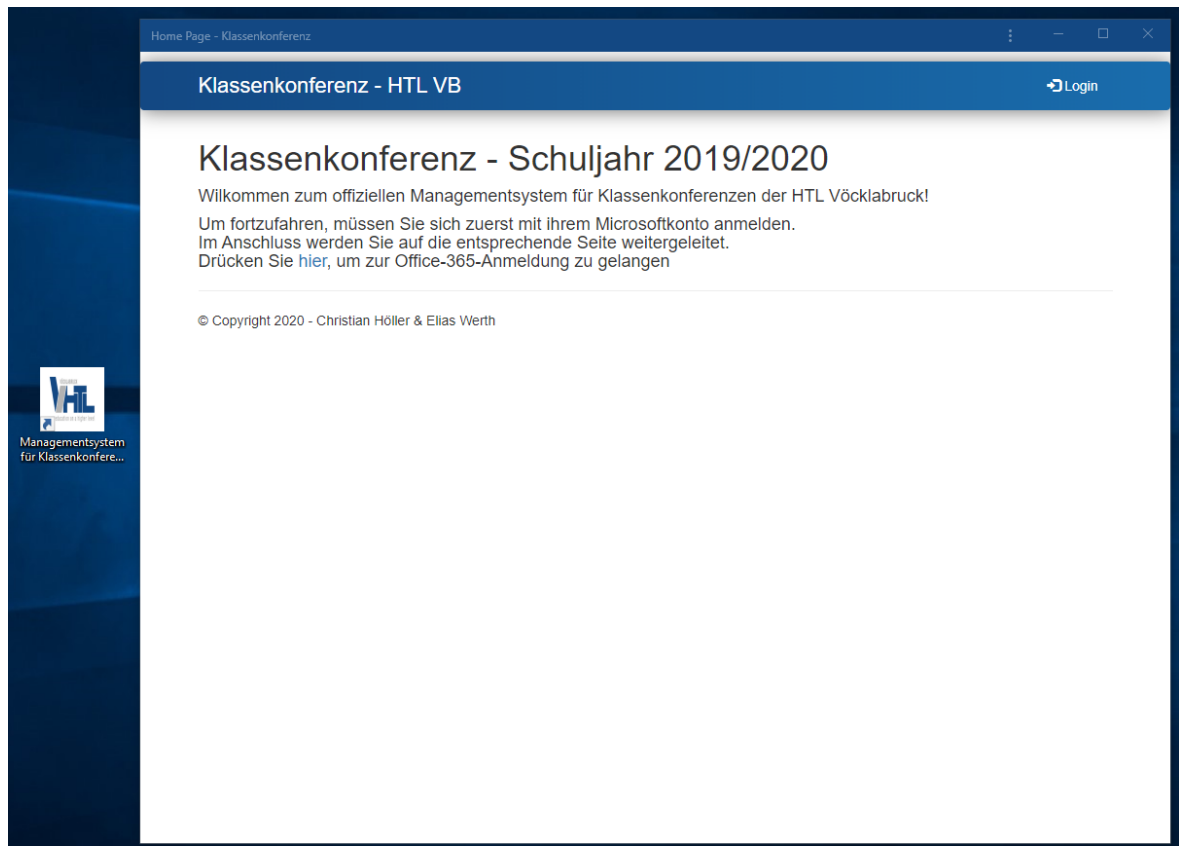


Abbildung 52 – Installierte Version

4.11 Aufrufen von Lehrkräften

Die Moderatoren der beiden Räume 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.

4.11.1 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 (s. Kapitel 4.6.3.2, Code 27). Somit 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='btn btn-secondary'>ausrufen</button></td>";

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

Code 60 – Methode *WriteTeachersWithButtons*

Zuallererst wird 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 ein `OnClick`-Event benötigen, die die Funktion aufruft, in der der eigentliche Aufruf der fehlenden Person stattfindet.

Lehrer

ABLD	<input type="button" value="aufrufen"/>
CHRISTIAN.HOELLER	<input type="button" value="aufrufen"/>
ELIAS.WERTH	<input type="button" value="aufrufen"/>

Abbildung 53 – Aufruf-Buttons

4.11.2 Senden des Aufrufs mit `SignalR`

Die Funktion, die durch den Klick auf den Button aufgerufen werden soll, wird auch in der `moderator_view.js`-Datei programmiert. 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());
    });
}
```

Code 61 – Methode `callTeacher`

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. Voraussetzung dafür ist selbstverständlich die Verbindung zum `MainHub`. Essenziell sind die Variablen, die mitgesendet werden. `indexOfCalledTeacher` ist die Variable, die bereits der jetzigen Funktion mitgegeben wurde. `moderatorID` ist E-Mail-Adresse des Moderators, der die Lehrkraft aufruft. Diese Variable wurde in der Zeile zuvor ermittelt. Mit `GetCurrentRoom()` wird der aktuelle Raum, in der die Konferenz stattfindet, abgerufen.

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 _currentroom)
{
    Currentroom = _currentroom;
    var currentClass = GetClass(GetCurrentClassName());
    var teacherToCall = currentClass.Teachers[indexOfCalledTeacher];
    var message = $"Sie werden in Raum {Currentroom} erwartet";

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

Code 62 – MainHub SendTeacherCall

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

4.11.3 Speichern der Aufrufe

Teil der Aufgabenstellung ist es, jeden einzelnen Aufruf für statistische Zwecke in eine eigene Tabelle zu speichern. In dieser Tabelle sollen der Moderator und die Lehrkraft, die ausgerufen wurde, der Zeitpunkt des Aufrufs und die Klasse, die in diesem Moment besprochen wurde, gesichert werden.

Tabelle: TeacherCall

	Moderator	Teacher	Time	Class
	Filtern	Filtern	Filtern	Filtern
1	baue@htlvb.at	abld@htlvb.at	14:22:34	4AHWII
2	baue@htlvb.at	edem@htlvb.at	14:23:12	4AHWII
3	leim@htlvb.at	matt@htlvb.at	15:16:43	3AHGTI
4	molj@htlvb.at	abld@htlvb.at	15:55:52	1AHME

Abbildung 54 – 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);
```

Code 63 – SQL-Befehl für Tabelle TeacherCall

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

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.

4.11.4 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>
```

Code 64 – HTML-Code Einverständniserklärung

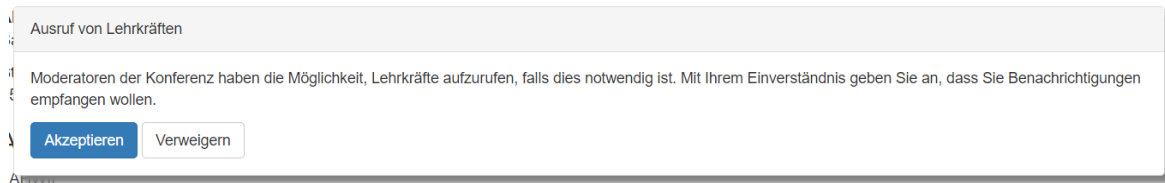


Abbildung 55 – Einverständniserklärung für Aufrufe

Je nach Entscheidung des Nutzers wird eine der folgenden Funktionen ausgeführt.

```
var allowNotifications = false;

function yesToNotifications() {
    allowNotifications = true;
    document.getElementById('notiContent').style.visibility = "hidden";
}

function noToNotifications() {
    allowNotifications = false;
    document.getElementById('notiContent').style.visibility = "hidden";
}
```

Code 65 – Akzeptieren/Verweigern von Benachrichtigungen

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 der Benutzer die Benachrichtigungen akzeptiert hat und ein Aufruf empfangen wird, muss nun geklärt werden, ob der Aufruf für ihn gedacht ist. Dafür wird die „TeacherID“ (E-Mail-Adresse der aufgerufenen Lehrkraft) mit der „UserID“ (E-Mail-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);
}
```

Code 66 – Erlangen der UserID

Platziert wird dieser Code in der `conference.cshtml`-Page. Mit `using 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-Adresse ist.

Die Abfrage, die klärt, ob der Aufruf an diesen Client gesendet werden muss, befindet sich in der `user_view.js`-Datei und sieht folgendermaßen aus:

```
connection.on("ReceiveTeacherCall", function (teacherID, message) {  
    if (allowNotifications == true) {  
        var userID = $("#userID").val();  
        if (teacherID == userID.toLowerCase()) {  
            //Platz für die Visualisierung  
        }  
    }  
});
```

Code 67 – Empfangen des Aufrufs

Der Grund, warum die „UserID“ mit `toLowerCase()` in die Kleinschreibweise umgewandelt wird, ist, um die IDs einheitlich zu machen und einen genauen Vergleich zu garantieren.

4.11.5 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.

Um eine Toast-Notifikation in das Programm einzubinden, sind 3 wichtige Schritte notwendig. Die ersten 2 Schritte müssen auf der `_Layout.cshtml`-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>
```

Code 68 – Verlinkung der CSS-Datei

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>
```

Code 69 – Verlinkung der JavaScript-Datei

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 (s. Kapitel 4.11.4, Code 67).

```
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");
```

Code 70 – Ausgabe des Toasts

Zuerst werden die Eigenschaften der Notifikation definiert. Unter anderem 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.

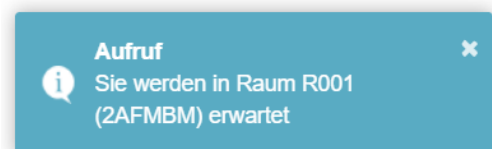


Abbildung 56 - Toast

4.11.6 Text-to-Speech

Als nettes Feature hat man sich dafür entschieden, die Nachrichten automatisch vorlesen zu lassen, in dem Moment, in dem diese eingeht.

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

```
const utterThis = new SpeechSynthesisUtterance(message);  
utterThis.lang = 'de-AT';  
speechSynthesis.speak(utterThis);
```

Code 71 – Text-to-Speech

Im ersten Schritt wird eine Variable angelegt, in der die Äußerung der SpeechSynthesis-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.* (N.N., Mozilla, 2020)

Mit `utterThis.lang` definiert man die Sprache, in der gesprochen werden soll. Ohne diesem Attribut wäre die Sprache vom Browser abhängig.

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

4.12 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.

4.12.1 Navigationsmenü

Das Design der Navigationsleiste 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 rechts 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>
```

Code 72 – HTML Navigationsmenü

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;
}
```

Code 73 – CSS-Klasse Farbenverlauf

Das resultierende Navigationsmenü sieht nach Einsatz dieses Codes folgendermaßen aus:



Abbildung 57 - Navigationsmenü

Wie bereits erwähnt, muss sich das Layout auch an das entsprechende 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>
```

Code 74 – HTML Navigationsmenü Toggle

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 58 – Navigationsbar Toggle

Der Login-Button ist natürlich nicht verloren. Durch einen Klick auf den Toggle-Button fächert sich das Menü nach unten aus (s. Abb. 59).



Abbildung 59 – Navigationsbar Toggle ausgefächert

4.12.2 Footer

Der Sinn der Fußzeile ist nichts weiter, als das Copyright bekannt zu geben. Der Footer sollte logischerweise 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>
  
```

Code 75 – HTML Footer

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;
}
  
```

Code 76 – CSS-Klasse Footer

Der fertige Footer sieht wie folgt aus:

© Copyright 2020 - Christian Höller & Elias Werth

Abbildung 60 - Footer

4.12.3 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 wie möglich dargestellt werden muss. Die Nutzeransicht beinhaltet auch das Panel mit der Einverständniserklärung (s. Kapitel 4.11.4).

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>
```

Code 77 – HTML Nutzeransicht

Die beiden Klassen der Divisionen (div) sind selbst gecodet und helfen bei der Anordnung.

```
.left-side {
  float: left;
  width: 50%;
  padding: 20px;
}
.right-side{
  float: left;
  width: 50%;
  padding: 20px;
}
```

Code 78 – Css-Klasse Nutzeransicht

Durch den `float`-Wert und der Aufteilung auf 50% zu 50% gelingt es, die beiden Anzeigen nebeneinander zu platzieren. Der `padding`-Wert ist ein Zusatzattribut, welches Überschneidungen der Elemente vorbeugt.

Das Design des Panels kann glücklicherweise Bootstrap übernehmen.

```
<div class="panel panel-default" style="visibility: visible;">
  <div class="panel-heading">Ausruf von Lehrkräften</div>
  <div class="panel-body">
  </div>
</div>
```

Code 79 – HTML Einverständniserklärung

Was Bootstrap 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.

```
.panel {
  position: fixed;
  width: 70%;
  left: 50%;
  top: 35%;
  margin-left: -35%;
  box-shadow: 7px 7px 5px grey;
}
```

Code 80 – CSS-Klasse Panel



Abbildung 61 – Nutzeransicht mit Einverständniserklärung

4.12.4 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 der rechten Seite zu platzieren.

```
<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>
```

Code 81 – HTML Moderatoransicht

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 Buttons kommt dabei sehr gelegen. Es ist nichts weiter zu machen, als den Button mit einer neuen CSS-Klasse auszustatten.

```
<input type="button" class="btn btn-primary btn-moderator" id="sendButton"
value="Besprechung"/>
```

Code 82 – HTML Moderator-Button

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

```
.btn-moderator {
    padding: 10px;
    font-size: 17px;
    margin: 20px;
    width: 71%;
}
```

Code 83 – CSS-Klasse Moderator-Button

Klassenkonferenz - HTL VB Werth Elias Log Out

R001 (2AFMBM)

3AHWII

Klassenvorstand
Ablinger David

Abteilungsvorstand
Bauernfeind Hermann

Startzeitpunkt
15:38:12

Abgeschlossene Klassen
2AHWII

Nächste Klassen
4AHWII

Überschneidungen
ABLD

Lehrer
ABLD [aufrufen](#)
CHRISTIAN.HOELLER [aufrufen](#)
ELIAS.WERTH [aufrufen](#)

Nächste Klasse

© Copyright 2020 - Christian Höller & Elias Werth

Abbildung 62 - Moderatoransicht

5 Literaturverzeichnis

- N.N. (10. März 2020). *Ceteris*. Abgerufen am 10. März 2020 von Ceteris:
<https://www.ceteris.ag/innovative-bi/technologien/microsoft-azure/>
- N.N. (01. April 2020). *Cloutcomputing-insider*. Abgerufen am 01. April 2020 von
Cloutcomputing-insider: [https://www.cloudcomputing-insider.de/was-ist-
json-a-704909/](https://www.cloudcomputing-insider.de/was-ist-json-a-704909/)
- N.N. (01. April 2020). *Docs.Microsoft*. Abgerufen am 01. April 2020 von
Docs.Microsoft: [https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-
csharp/](https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/)
- 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/](https://blog.dotnetstudio.nl/connect-azure-ad-authenticated-signalr-hub-from-spfx/)
- N.N. (23. März 2020). *Elmah*. Abgerufen am 23. März 2020 von Elmah:
[https://blog.elmah.io/turning-an-aspnet-core-website-into-a-progressive-
web-app-pwa/](https://blog.elmah.io/turning-an-aspnet-core-website-into-a-progressive-web-app-pwa/)
- N.N. (30. Februar 2020). *GitHub*. Abgerufen am 30. Februar 2020 von GitHub:
<https://github.com/CodeSeven/toastr>
- N.N. (20. Februar 2020). *GitHub*. Abgerufen am 20. Februar 2020 von GitHub:
<https://github.com/openiddict/openiddict-core/issues/35>
- N.N. (01. April 2020). *jQuery*. Abgerufen am 01. April 2020 von jQuery:
<https://jquery.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](https://docs.microsoft.com/de-de/aspnet/signalr/overview/getting-started/introduction-to-signalr)
- N.N. (2. Februar 2020). *Microsoft*. Abgerufen am 2. Februar 2020 von Microsoft:
<https://microsoft.github.io/AzureTipsAndTricks/blog/tip153.html>

- N.N. (31. März 2020). *Mozilla*. Abgerufen am 31. März 2020 von Mozilla:
<https://developer.mozilla.org/de/docs/Web/API/SpeechSynthesis>
- N.N. (10. März 2020). *Newtonsoft*. Abgerufen am 10. März 2020 von Newtonsoft:
<https://www.newtonsoft.com/json>
- N.N. (2. April 2020). *NuGet*. Abgerufen am 2. April 2020 von NuGet:
<https://www.nuget.org/packages/WebEssentials.AspNetCore.PWA/>
- N.N. (01. April 2020). *NuGet*. Abgerufen am 01. April 2020 von NuGet:
<https://www.nuget.org/packages/Microsoft.AspNetCore.Authentication.AzureAD.UI/>
- N.N. (2. Februar 2020). *Oauth*. Abgerufen am 2. Februar 2020 von Oauth:
<https://www.oauth.com/oauth2-servers/client-registration/client-id-secret/>
- N.N. (01. April 2020). *Siteground*. Abgerufen am 01. April 2020 von Siteground:
<https://www.siteground.com/tutorials/php-mysql/mysql/>
- N.N. (01. April 2020). *SQLiteBrowser*. Abgerufen am 01. April 2020 von SQLiteBrowser: <https://sqlitebrowser.org/>
- N.N. (01. April 2020). *Tutorialspoint*. Abgerufen am 01. April 2020 von Tutorialspoint:
https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_razor.htm
- N.N. (10. März 2020). *Twitter*. Abgerufen am 10. März 2020 von Twitter:
<https://twitter.com/sqlitebrowser/status/967968416236785665>
- N.N. (25. März 2020). *web.dev*. Abgerufen am 25. März 2020 von web.dev:
<https://web.dev/add-manifest/>
- 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-Sharp>
- 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_\(Framework\)](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. (14. März 2020). *Wikipedia*. Abgerufen am 14. März 2020 von Wikipedia:
<https://de.wikipedia.org/wiki/SQL-Injection>
- N.N. (01. April 2020). *Wikipedia*. Abgerufen am 01. April 2020 von Wikipedia:
https://de.wikipedia.org/wiki/Visual_Studio
- N.N. (01. April 2020). *Wikipedia*. Abgerufen am 01. April 2020 von Wikipedia:
https://de.wikipedia.org/wiki/Hypertext_Markup_Language

N.N. (01. April 2020). *Wikipedia*. Abgerufen am 01. April 2020 von Wikipedia:
https://de.wikipedia.org/wiki/Cascading_Style_Sheets

N.N. (01. April 2020). *Wikipedia*. Abgerufen am 01. April 2020 von Wikipedia:
<https://de.wikipedia.org/wiki/MySQL>

N.N. (1. April 2020). *Wikipedia*. Abgerufen am 1. April 2020 von Wikipedia:
<https://de.wikipedia.org/wiki/JavaScript>

6 Abbildungsverzeichnis

Abbildung 1 – Nutzer-Ansicht.....	II
Abbildung 2 – Moderator-Ansicht.....	II
Abbildung 3 – Microsoft Anmeldung.....	II
Abbildung 4 – User View	5
Abbildung 5 – Moderator View	5
Abbildung 6 – MS Login	5
Abbildung 7 - Christian Höller.....	1
Abbildung 8 - Elias Werth.....	3
Abbildung 9 - MSc BSc Johannes Egger	4
Abbildung 10 – Screenshot alte Anwendung	6
Abbildung 11 – Microsoft Login.....	8
Abbildung 12 – PWA iPhone	9
Abbildung 13 - Visual Studio	11
Abbildung 14 - SQLite DB Browser	11
Abbildung 15 – git	11
Abbildung 16 – GitHub	12
Abbildung 17 - JSON	12
Abbildung 18 – Bootstrap.....	12
Abbildung 19 – jQuery.....	13
Abbildung 20 – SQLite	14
Abbildung 21 – SignalR.....	14
Abbildung 22 – Newtonsoft	14
Abbildung 23 - Microsoft Azure	15
Abbildung 24 - C#	16
Abbildung 25 – HTML	16
Abbildung 26 – CSS.....	17
Abbildung 27 – JavaScript	17
Abbildung 28 – MySQL	17
Abbildung 29 - SQLite Tabelle General.....	20
Abbildung 30 - SQLite Tabelle State	21
Abbildung 31 - Visual Studio Paket hinzufügen.....	22

Abbildung 32 - Raumansicht	32
Abbildung 33 - Moderator Seite.....	33
Abbildung 34 - Überschneidungen	44
Abbildung 35 - Nutzeransicht	49
Abbildung 36 – Konferezeinstellungen	53
Abbildung 37 – Administratoransicht Bestätigen einer Aktion	55
Abbildung 38 – Homepage Azure Portal	56
Abbildung 39 – Registrierung einer Anwendung	57
Abbildung 40 - Anwendungsinformationen.....	57
Abbildung 41 – Prinzip der Authentifizierung	58
Abbildung 42 – Redirect URIs	59
Abbildung 43 – NuGet AzureAD.....	60
Abbildung 44 – Microsoft Anmeldung.....	63
Abbildung 45 – Logout-Button.....	63
Abbildung 46 – Login-Button	63
Abbildung 47 – Tabelle UserRights.....	64
Abbildung 48 – Nuget WebEssentials for PWA.....	66
Abbildung 49 – DevTools Service Worker.....	68
Abbildung 50 – DevTools Manifest.....	71
Abbildung 51 - Installationsbenachrichtigung	71
Abbildung 52 – Installierte Version.....	72
Abbildung 53 – Aufruf-Buttons	74
Abbildung 54 – Tabelle TeacherCall	75
Abbildung 55 – Einverständniserklärung für Aufrufe	77
Abbildung 56 - Toast	79
Abbildung 57 - Navigationsmenü	82
Abbildung 58 – Navigationsbar Toggle	83
Abbildung 59 – Navigationsbar Toggle ausgefächert.....	83
Abbildung 60 - Footer.....	84
Abbildung 61 – Nutzeransicht mit Einverständniserklärung	85
Abbildung 62 - Moderatoransicht	87
Abbildung 63 – Terminplan (ProjectLibre).....	97
Abbildung 64 - Begleitprotokoll.....	98

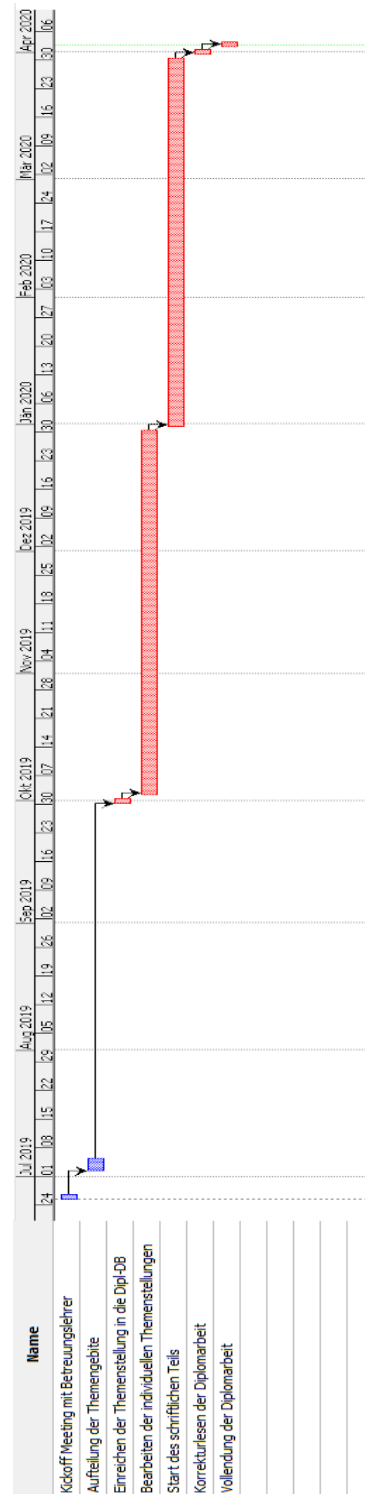
7 Codeverzeichnis

Code 1 – JSON Lehrer.....	19
Code 2 - JSON Klassen	19
Code 3 - JSON Reihenfolge.....	19
Code 4 - Configure Hub Route.....	23
Code 5 - HTML moderator_view.js script tag	24
Code 6 - HTML user_view.js script tag	24
Code 7 - Modellklasse Teacher.....	25
Code 8 - Modellklasse MyClasses	25
Code 9 - Modellklasse Order.....	25
Code 10 –General Datenbank Pfadangabe	26
Code 11 – General Pfad zur JSON-Datei	27
Code 12 – General Tabellenname Eigenschaft	27
Code 13 – General JSON Daten als String.....	28
Code 14 - Codebeispiel Entgegenwirken von SQL-Injections.....	29
Code 15 – Methode Query.....	30
Code 16 – Methode Reader.....	31
Code 17 - Raumausgabe in HTML.....	32
Code 18 - Weiterleitung auf Moderator-Seite	32
Code 19 - Verbindungsaufbau zum MainHub	34
Code 20 - Stehende Verbindung zum MainHub.....	34
Code 21 – JavaScript Methode GetCurrentRoom.....	34
Code 22 - Methode LoadModeratorPage.....	35
Code 23 – MainHub Methode LoadModeratorContent.....	36
Code 24 – Methode GetTeacher.....	37
Code 25 – Methode GetButtonText.....	37
Code 26 – JavaScript ReceiveModeratorContent	38
Code 27 – JavaScript WriteTeachersInTable.....	38
Code 28 – Lehrer title-attribut.....	39
Code 29 – JavaScript Methode getShorthandForTeacher	39

Code 30 - Hub Methode LoadGeneralContent Switch	40
Code 31 - Hub Methode LoadGeneralContent Klassen	40
Code 32 – Methode GetClassesCompleted	41
Code 33 – Methode GetClassesNotEdited	42
Code 34 – JavaScript ReceiveGeneralContent.....	43
Code 35 – JavaScript Methode WriteDataInTable	43
Code 36 – Überschneidungen Konferenzstatusüberprüfung	44
Code 37 – Überschneidungen gemeinsame Lehrer.....	45
Code 38 – JavaScript ReceiveIntersections.....	45
Code 39 – JavaScript Methode WriteIntersectionsInTable	46
Code 40 – JavaScript Button-event.....	46
Code 41 – MainHub ConferenceAction	47
Code 42 – MainHub WriteTimeInDatabase.....	47
Code 43 – MainHub SetStateOfConference	48
Code 44 – MainHub NextClass.....	48
Code 45 – JavaScript Verbindungsaufbau zum MainHub.....	49
Code 46 – MainHub LoadRooms	50
Code 47 – JavaScript ReceiveRooms.....	50
Code 48 – JavaScript ReceiveGeneralContent.....	51
Code 49 – JavaScript WriteUserViewInformation	52
Code 50 – Admin_Settings OnPostReset	53
Code 51 – Administratoransicht Methode OnPostSetJSONData	54
Code 52 – AzureAD appsettings	59
Code 53 – Implementation in der Startup-Klasse.....	61
Code 54 – Implementation in die Navigationsleiste.....	62
Code 55 – Abfrage der Rechte.....	64
Code 56 – Hinzufügen der PWA in die Startup-Klasse	67
Code 57 – PWA-Einstellungen.....	67
Code 58 – Inhalt des Manifests.....	69
Code 59 – Verlinkung des Manifests.....	70
Code 60 – Methode WriteTeachersWithButtons	73
Code 61 – Methode callTeacher	74
Code 62 – MainHub SendTeacherCall.....	75

Code 63 – SQL-Befehl für Tabelle TeacherCall	76
Code 64 – HTML-Code Einverständniserklärung	76
Code 65 – Akzeptieren/Verweigern von Benachrichtigungen	77
Code 66 – Erlangen der UserID	77
Code 67 – Empfangen des Aufrufs	78
Code 68 – Verlinkung der CSS-Datei.....	78
Code 69 – Verlinkung der JavaScript-Datei	79
Code 70 – Ausgabe des Toasts	79
Code 71 – Text-to-Speech	80
Code 72 – HTML Navigationsmenü	81
Code 73 – CSS-Klasse Farbenverlauf	82
Code 74 – HTML Navigationsmenü Toggle	82
Code 75 – HTML Footer	83
Code 76 – CSS-Klasse Footer	83
Code 77 – HTML Nutzeransicht	84
Code 78 – Css-Klasse Nutzeransicht.....	84
Code 79 – HTML Einverständniserklärung	85
Code 80 – CSS-Klasse Panel	85
Code 81 – HTML Moderatoransicht	86
Code 82 – HTML Moderator-Button	86
Code 83 – CSS-Klasse Moderator-Button	86

ANHANG A – Terminplan



ANHANG B - Begleitprotokolle

Begleitprotokoll

Thema des übergeordneten komplexen Aufgabenbereichs oder Projekts:

Web-basiertes Managementsystem für Klassenkonferenzen

Individuelle Themenstellung:

Betreuer/in: Johannes Egger

E-Mail-Adresse: eggj@htlvb.at

Telefonnummer:

Name der Diplomandin/des Diplomanden und Klasse: Christian Höller, Elias Werth; 5AHWII

E-Mail-Adresse: christian.hoeller@htlvb.at / elias.werth@htlvb.at

Telefonnummer: 0660/4923639; 0660/5704711

Name der Kooperationspartnerin/des Kooperationspartners und Ansprechperson:

E-Mail-Adresse:

Telefonnummer:

Teammitglieder: Christian Höller & Elias Werth

Datum der Besprechung	Teilnehmer/innen der Besprechung	Vereinbarungen	Termin zur Erledigung	Paraphe	
				Betreuer/in	Schüler/innen
04.06.2019	Christian Höller; Elias Werth; Johannes Egger	Bekanntmachung mit der Anforderungsbeschreibung der Konferenzenanwendung	2h		
25.06.2019	Christian Höller; Elias Werth; Johannes Egger	Überblick über die notwendigen Technologien, Sprachen und Programme; Kickoff	2h 30min		
29.10.2019	Christian Höller; Elias Werth; Johannes Egger	Überblick über AzureAD; Besprechung der Authentifizierung	1h 30min		
11.02.2020	Christian Höller; Elias Werth; Johannes Egger	Besprechung der Progressive Web-App; Beheben einiger Fehler; Vorführung Text-to-Speech	1h		

Datum:

Unterschrift der Schülerin/des Schülers:

Abbildung 64 - Begleitprotokoll

ANHANG C Tätigkeitsprotokoll

Höller Christian

Tag	Datum	Uhrzeit		Beschreibung der Tätigkeit	Stunden
		von	bis		
Do	25.07.2019	15:40	17:00	Beschäftigen mit Razor Pages	1:20
Do	25.07.2019	16:00	19:00	Testprogramme mit .Net Core erstellt	3:00
Sa	27.07.2019	21:05	23:50	Beschäftigen mit SQLite	2:45
So	28.07.2019	13:00	16:22	Datenbankstruktur erstellt	3:22
Mo	29.07.2019	08:30	11:50	Beschäftigen mit Newtonsoft Library	3:20
Mo	29.07.2019	13:30	14:00	JSON Datei converence-info.json erstellt	0:30
Di	30.07.2019	16:00	18:00	Klassen und Methoden erstellt zum schreiben der Lehrer in den HTML Table	2:00
Di	30.07.2019	18:00	19:30	Klasse starten in SQLite Datenbank schreiben	1:30
Mi	31.07.2019	15:30	18:55	Knopf zum stoppen der Konferenz implementiert	3:25
Do	01.08.2019	20:00	23:00	Lesen der aktuellen Klasse, aktuellen Status lesen und Starten + Stoppen der Konferenz implementiert	3:00
Mo	05.08.2019	15:00	15:20	Index Datei umbenannt	0:20
Di	06.08.2019	16:00	18:50	Seite zum Weiterleiten an die entsprechende Moderatorseite	2:50
Mi	07.08.2019	20:45	00:00	Implementierung von Aktionen, wenn Klasse abgeschlossen wurde	3:15
Do	15.08.2019	18:00	22:00	SignalR einstellungen implementiert (Hub, JavaScript, USW)	4:00
So	18.08.2019	14:15	16:25	Hub Metghoden integriert	2:10
Di	03.09.2019	12:20	15:20	abgeschlossene und nächste Klassen table in HTML hinzugefügt	3:00
Di	03.09.2019	15:00	15:40	Klasse page content erstellt	0:40
Fr	06.09.2019	17:43	18:00	Mehrere Testlehrer in JSON eingefügt	0:17
So	08.09.2019	12:23	15:40	Implementierung nächste Klasse starten, wenn die alte abgeschlossen wurde	3:17
Di	10.09.2019	11:15	15:00	Erstellung einer Admin Seite zum Hinzufügen von Lehrern zur JSON-Datei(wurde wieder gelöscht)	3:45
Mi	11.09.2019	20:30	00:00	Erstellung einer Admin Seite zum Löschen von Lehrern zur JSON-Datei(wurde wieder gelöscht)	3:30

Do	12.09.2019	16:00	17:30	Erstellung einer Admin Seite zum Löschen von Lehrern zur JSON-Datei(wurde wieder gelöscht)	1:30
Do	26.09.2019	17:04	19:40	Erstellung einer Admin Seite zum Bearbeiten der Klassen der JSON-Datei(wurde wieder gelöscht)	2:36
Sa	02.11.2019	12:54	16:30	Administrator Seite zum Bearbeiten der Datenbankeinstellung erstellt	3:36
Mo	04.11.2019	19:30	22:00	Einstellungen zum Zurücksetzen der Datenbankeinstellungen implementiert	2:30
Di	05.11.2019	15:40	19:30	Konferenzablauf überarbeitet	3:50
Di	05.11.2019	17:00	18:00	Fehler beim Start der Anwendung behoben	1:00
Mi	06.11.2019	16:43	21:55	Konferenzablauf optimiert	5:12
Mi	06.11.2019	11:40	14:00	Nutzeransicht für Normale Nutzer erstellt	2:20
Mi	06.11.2019	19:32	00:00	Nutzeransicht für Normale Nutzer erstellt	4:28
Di	12.11.2019	20:09	22:15	JavaScript Seite mit Methoden erstellt und Hub-Methoden erstellt	2:06
Di	03.12.2019	19:10	21:30	Für jeden Lehrere einen Knopf zum Ausrufen eingefügt	2:20
So	08.12.2019	20:00	22:00	DbConnection Klasse erstellt	2:00
Do	19.12.2019	15:10	17:00	Aktuelle Klasse von den abgeschlossenen Klassen entfernt	1:50
Do	19.12.2019	18:15	19:30	Alle Klassen anzeigen, wenn Konferenz abgeschlossen wurde	1:15
Sa	21.12.2019	10:50	14:00	Refactoring von Methoden der Hub-Klasse	3:10
So	22.12.2019	11:00	13:55	Refactoring von Load_information	2:55
Mo	23.12.2019	13:00	14:55	Diplomarbeit Buch erstellt	1:55
Di	24.12.2019	09:00	12:20	DA-Buch Moderator Dokumentation	3:20
Di	24.12.2019	15:00	19:00	DA-Buch Moderator Dokumentation	4:00
Mi	25.12.2019	19:21	20:00	DA-Buch SignalR und JSON Dokumentation	0:39
Di	11.02.2020	10:40	15:00	Beginn des Designs der Startseite; Arbeiten am DA-Buch	4:20
Di	11.02.2020	13:50	14:17	DA-Buch Persönlicher Werdegang	0:27
Mi	12.02.2020	14:10	15:00	DA-Buch verwendete Technologien	0:50
Mi	12.02.2020	16:00	17:00	DA-Buch Individuelle Zielsetzung	1:00
Do	13.02.2020	08:30	09:30	DA-Buch NuGet Dokumentation	1:00
Do	13.02.2020	11:00	11:40	Hub Klasse Rückgabewerte an JavaScript als Jarray deklariert	0:40
Fr	14.02.2020	11:00	12:15	DA-Buch Technologien dokumentation	1:15
Sa	15.12.2020	10:20	12:00	Refactoring von mainHub und DA Doku	1:40

Sa	15.02.2020	20:00	23:40	SQL Query und Reader Klasse verändert, DA-Buch Dokumentation und Refactoring von mainHub Klassen	3:40
Mo	17.02.2020	09:55	10:31	Alle SQL Befehle parameterisiert	0:36
Do	20.02.2020	17:30	18:14	Überarbeiten von User Ansicht Dokumentation	0:44
Fr	21.02.2020	09:41	12:20	DA-Buch Klassen und Modellklassen Dokumentation	2:39
Fr	21.02.2020	12:50	14:31	Ausruf button für jeden Lehrer einer Klasse hinzugefügt	1:41
Sa	22.02.2020	10:35	11:22	hinzufügen einer Klasse JavaScript Datei für allgemeine Funktionen	0:47
Sa	22.02.2020	11:28	12:03	Exportieren der Funktionen in javascript	0:35
Sa	22.02.2020	12:30	13:29	Änderungen der Informationsübermittlung an JavaScript	0:59
Sa	22.02.2020	17:19	18:37	mainHub methoden geändert	1:18
So	23.02.2020	10:27	10:45	mainHub cleanup	0:18
So	23.02.2020	19:45	20:40	Dokumentation der Änderungen	0:55
Mo	24.02.2020	16:15	17:08	Dokumentation der Änderungen	0:53
So	01.03.2020	08:40	09:25	mainHub refactoring	0:45
Mo	02.03.2020	15:14	16:07	Lehrer button einfügen	0:53
Mi	04.03.2020	12:30	14:55	Lehrer ausrufen button und functions.js gelöscht	2:25
Do	05.03.2020	07:45	09:20	neue Tabelle TeacherCall + Implementierung bei Lehrer Aufruf. DA-Buch Doku	1:35
Mo	09.03.2020	13:15	14:30	Da Buch Quellverzeichnis	1:15
Di	10.03.2020	07:45	13:10	Da Buch Quellverzeichnis + Code Verzeichnis	5:25
Di	10.03.2020	18:40	20:56	DA Buch Überarbeitung der Realisierung	2:16
Mi	11.03.2020	19:20	20:29	DA Buch Überarbeitung der Realisierung	1:09
Do	12.03.2020	08:40	09:17	DA Buch Code Verzeichnis	0:37
Sa	28.03.2020	18:00	22:30	Da-Buch Fehler Überarbeitet	4:30
Mi	01.04.2020	17:00	21:00	DA-Buch Korrigiert	4:00
Do	02.04.2020	20:00	22:00	Fertigstellen der DA	2:00
Stunden Gesamt					156:55

Werth Elias

Tag	Datum	Uhrzeit		Beschreibung der Tätigkeit	Stunden
		von	bis		
Di	02.07.2019	16:30	19:20	Testen der notwendigen Features/Commands von Git und GitHub	2:50
Mi	10.07.2019	14:00	16:20	Über ASP.Net 3.0 informiert	2:20
Mo	29.07.2019	14:30	16:30	Projekt Start/Aufgabenaufteilung	2:00
So	11.08.2019	16:50	19:00	Recherche über ASP.NET Core	2:10
Do	22.08.2019	14:30	19:00	Im Test-Projekt programmieren	4:30
Di	27.08.2019	18:00	21:20	Bekanntmachung mit Razorpages	3:20
Do	29.08.2019	18:00	19:45	JSON ausprobieren	1:45
Mo	02.09.2019	17:30	19:30	Programmieren	2:00
Di	03.09.2019	17:45	20:10	Vorbereitung und lernen	2:25
Di	07.09.2019	14:00	16:20	Beschäftigung mit SignalR	2:20
Di	29.10.2019	14:30	15:30	Besprechung mit EGGJ	1:00
Fr	08.11.2019	13:10	14:15	Kurzer Check was noch wie gemacht werden muss	1:05
Di	19.11.2019	18:15	20:15	Implementierung von Microsoft Identity (Azure AD)	2:00
Mi	20.11.2019	18:10	20:00	Implementierung von Microsoft Identity (Azure AD)	1:50
So	24.11.2019	15:00	17:30	Implementierung von Microsoft Identity (Azure AD)	2:30
Fr	13.12.2019	12:20	13:20	Implementierung von Microsoft Identity (Azure AD)	1:00
Fr	17.01.2020	15:45	18:00	Authentifizierung mit Azure AD	2:15
Di	21.01.2020	16:15	19:30	Getting started with Progressive Web-App	3:15
Do	23.01.2020	17:00	19:30	Einzelne Tests mit PWA	2:30
Fr	24.01.2020	09:40	11:00	Problembesehung beim Login mit EGGJ	1:20
Di	28.01.2020	18:00	20:00	Programmierung von Feinheiten am Login	2:00
Di	04.02.2020	17:00	18:35	Hinzufügen von Push-Notifications	1:35
So	09.02.2020	10:40	15:00	Gemeinsam mit Höller am Design der Webapp arbeiten; anschließend am DA-Buch geschrieben	4:20
Mo	10.02.2020	09:00	10:00	Diplomarbeitsbuch:Allgemeiner Teil	1:00
Di	11.02.2020	07:50	10:30	Design der Seite verbessert; Push-Notifications der PWA	2:40
Di	11.02.2020	12:15	13:35	Besprechung mit EGGJ	1:20
Mo	17.02.2020	16:00	18:15	Anpassung des Designs + Weiterleitung nach Login	2:15

Di	18.02.2020	10:00	12:00	Design angepasst + Push-Notifications mit SignalR	2:00
Mi	19.02.2020	09:15	10:30	Diplomarbeitsbuch: Zielsetzung Authentifizierung	1:15
Mi	19.02.2020	16:20	19:10	Diplomarbeitsbuch: Zielsetzung PWA	2:50
Do	20.02.2020	08:15	10:00	Diplomarbeitsbuch: Realisierung Authentifizierung	1:45
Do	20.02.2020	14:30	18:00	Diplomarbeitsbuch: Realisierung Authentifizierung	3:30
Fr	21.02.2020	11:15	13:30	Hinzufügen von Push-Notifications	2:15
So	23.02.2020	10:00	14:30	Hinzufügen von Push-Notifications + Design der Page	4:30
So	23.02.2020	15:00	19:00	Weiterleitung der User nach Login	4:00
Mi	04.03.2020	12:20	14:50	Aufruf von Lehrkräfte + Text to Speech Notifications	2:30
Do	05.03.2020	07:45	09:15	Anpassung Weiterleitung Admin und hinzufügen von TeacherCall-Tabelle	1:30
Di	11.03.2020	10:15	15:30	Diplomarbeitsbuch	5:15
Do	12.03.2020	08:00	11:10	Design der Meldungen angepasst + Benachrichtungen	3:10
Mo	16.03.2020	10:30	13:35	Verbesserungen an der PWA	3:05
Mi	18.03.2020	18:20	20:30	Diplomarbeitsbuch: NuGet-Paket + Feinheiten am Design und Manifest	2:10
Mo	23.03.2020	10:00	14:50	Diplomarbeitsbuch: Realisierung PWA	4:50
Mi	25.03.2020	10:40	15:00	Diplomarbeitsbuch: Realisierung PWA; kleine Änderungen an der PWA	4:20
So	29.03.2020	11:30	15:40	Überarbeitung des Designs der einzelnen Pages	4:10
Mo	30.03.2020	13:30	18:00	Diplomarbeitsbuch: Realisierung Aufrufe von Lehrkräften	4:30
Di	31.03.2020	14:50	17:10	Diplomarbeitsbuch: Realisierung Aufrufe von Lehrkräften	2:20
Di	31.03.2020	18:15	20:00	DA-Buch-Besprechung mit Höller	1:45
Mi	01.04.2020	14:30	17:30	Diplomarbeitsbuch: Realisierung Design und Layout	3:00
Mi	01.04.2020	18:00	21:20	Korrekturlesen des Buchs gemeinsam mit Höller	3:20
Do	02.04.2020	14:00	15:45	Verbesserungen am Diplomarbeitsbuch	1:45
Do	02.04.2020	20:00	22:00	Korrekturlesen des Buchs	2:00
Fr	03.04.2020	08:00	10:30	Fertigstellen der Diplomarbeit	2:30
Stunden Gesamt					133:50