Abschlussprüfung Winter 2020

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Webanwendung

Entwicklung eines Abwesenheitszeiten-Erfassungssystems mit Implementierung in Microsoft Teams

Abgabetermin 13.11.2020

Prüfungsbewerber:

Daniel Beckendorf
Arndtstr. 48
47119 Duisburg

Praktikumsbetrieb:

34digital GmbH
Solinger Str. 19
45481 Mülheim an der Ruhr

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

IBB	Institut für Berufliche Bildung
34D	34digital GmbH
AES	Abwesenheitszeiten-Erfassungsystem
MEAN	MongoDB, Express.js, Angular, Node.js
REST	Representational State Transfer
API	Application Programming Interface
JSON	JavaScript Object Notation
GUI	Graphical User Interface
DB	Datenbank
NPM	Node Package Manager
URL	Uniform Resource Locator
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
JWT	JSON Web Token

1. Einleitung

Die Folgende Projektdokumentation schildert den Ablauf des IHK-Abschlussprojekts, welches der Autor im Rahmen seiner Ausbildung zum Fachinformatiker für Anwendungsentwicklung durchgeführt hat. Der Ausbildungsbetrieb ist das Institut für Berufliche Bildung AG, im Folgenden IBB genannt. Der Praktikumsbetrieb ist die 34digital GmbH, im Folgenden 34D genannt, eine IT-Firma, welche praktikable Lösungen zur Digitalisierung in den Bereichen Kommunikation, Datenschutz und digitaler Arbeitsplatz anbietet. 34D ist unter anderem zuständig für die Vorlagenentwicklung von diversen Schreiben für Krankenkassen.

1.1 Projektbeschreibung

In diesem Projekt wurde ein Abwesenheitszeiten-Erfassungssystem, im Folgenden AES genannt, als Webanwendung entwickelt und in Microsoft Teams implementiert werden. Dieses soll zur Ermittlung der anwesenden Personen im Büro dienen. In diesem System kann jeder Mitarbeiter eintragen, ob er krank ist, abwesend ohne Grund oder sich im Home-Office befindet. Das AES dient zur Planung der Büroeinteilung, damit genügend freie Plätze und somit Abstand während der Corona Pandemie gewährleistet werden kann.

1.2 Projektziel

Das Ziel des Projekts war es, die momentane Abwesenheitsplanung aus dem Microsoft Office Kalenders zu entfernen und eine eigene Lösung dafür zu entwickeln. Mithilfe dieser Lösung soll es den Mitarbeitern einfacher und übersichtlicher gemacht werden sich abzumelden und die Anzahl der Personen im Büro im Blick zu haben. Da die meisten aktuellen Informationen in Microsoft Teams ablaufen, soll das AES dort implementiert werden.

1.3 Projektumfeld

Auftraggeber des Projekts ist 34D selbst. Seit zwei Jahren entwickelt und betreut 34D praktikable Lösungen in den Bereichen Kommunikation, Datenschutz und digitaler Arbeitsplatz. In dem Unternehmen arbeitet ein circa 15-köpfiges Team für kleine und große Geschäftskunden. 34D ist unter anderem zuständig für die Vorlagenentwicklung von diversen Schreiben für Krankenkassen.

1.4 Projektschnittstellen

Damit das AES den Mitarbeitern die Übersicht erleichtern kann, soll es sowohl in jedem gängigen Webbrowser sowie als Applikation in einem Microsoft Teams Tab funktionieren.

Das Frontend des AES wurde mittels einer App als iFrame in Microsoft Teams eingebunden. Alle nötigen Daten werden auf einem CentOS Servers über Microsoft Azure gehostet und sind als MEAN Stack entwickelt. MEAN Stack besteht aus MongoDB, Express.js, Angular und Node.js.

1.5 Projektabgrenzung

Das Projekt wird komplett eigenständig entwickelt, ohne Bezug zu anderen Projekten/Teilprojekten.

2. Projektplanung

In diesem Abschnitt werden die Zeiten sowie die Ressourcen des Projekts geplant

2.1 Projektphasen

Für das gesamte Projekt standen dem Autor insgesamt 70 Stunden zur Verfügung. Diese wurden in folgende Phasen verteilt.

Projektphase	Geplante Zeit
Analyse	8 Stunden
Entwurf	9 Stunden
Implementierung	39 Stunden
Abnahme und Deployment	3 Stunden
Dokumentation	11 Stunden
Gesamt:	70 Stunden

TABELLE 1: ERSTE ZEITPLANUNG

Die einzelnen Hauptphasen wurden in Unterpunkte aufgeteilt. Die Übersicht befindet sich im Anhang: A.1 DETAILLIERTE ZEITPLANUNG

2.2 Ressourcenplanung

Da 34D noch ein recht junges Unternehmen ist, das sich im starken Wachstum befindet, wurde trotzdem darauf geachtet die Anschaffungs- und Folgekosten des Projekts möglichst gering zu halten. Deshalb wird das Projekt auf einem kleinen CentOS Server in Microsoft Azure gehostet und für Angular wird die kostenlose Google Material Design Palette benutzt.

2.3 Entwicklungsprozess

Das Projekt soll agil entwickelt werden. Hierbei wird in kurzen Iterationszyklen, nach der SCRUM Methode, Rücksprache mit dem Projektverantwortlichen gehalten, um Feedback einzuholen. Mit diesem Feedback kann der Autor auf Änderungswünsche und Verbesserungsvorschläge sehr schnell eingehen und somit Zeit sparen. Die Umsetzung erfolgte mit der Projektmanagementsoftware „Trello“ siehe ABBILDUNG 1: TRELLO

Die SCRUM Methode ermöglicht es das Projekt agil und flexibel umzusetzen.

3. Analysephase

Nachdem die Planung des Projekts abgeschlossen ist, kann mit der Analysephase begonnen werden.

3.1 Ist-Analyse

Die Firma 34D erlaubt ihrem Personal aufgrund der Corona Pandemie das Arbeiten im Home-Office. Hierfür wird bislang die Abwesenheit im Microsoft Office Kalender eingetragen. Da Microsoft Teams einen Großteil der Projekt- und Mitarbeiterplanung im Betrieb beinhaltet, aber zu viele Einträge es schnell unübersichtlich aussehen lassen, soll eine neue Lösung gefunden werden.

3.2 Wirtschaftlichkeitsanalyse

Aufgrund der vielen Termine im Microsoft Office Kalenders, die diesen sehr schnell unübersichtlich machen und der Gewährleistung der Gesundheit und Sicherheit des Personals in dieser Pandemie, soll eine entsprechende Lösung gefunden werden.

3.3 „Make or Buy“ Entscheidung

Es wurden Vergleichbare Lösungen bei diversen Anbietern im Internet gefunden, da diese aber sehr kostspielig und nicht erweiterbar sind, hat sich 34D für das eigenständige Entwickeln entschieden, mit Möglichkeiten zu späteren Änderungen und einfacher betrieblichen Anpassungen.

3.4 Projektkosten

Nachfolgend befinden sich die geplanten Kosten, die während der Entwicklung des Projekts angefallen sind. Da die genauen Personalkosten nicht herausgegeben werden dürfen, wird die Kalkulation anhand von beispielhaften Kosten durchgeführt. Der Stundensatz eines Auszubildenden beträgt 10 €, der eines Mitarbeiters 25 €. Für die sonstigen Kosten wurde ein Pauschalsatz von 10 € pro Stunde verwendet. Es wird der günstigste CentOS Cloud Server verwendet mit einer Laufzeit von einem Jahr 203,52 €

Anfallende Projektkosten können der folgenden Tabelle entnommen werden.

Vorgang	Mitarbeiter	Zeit	Personal	Sonstiges	Gesamt
Analyse und Planung	1 Auszubildender	15 Std.	150,00 €	150,00 €	300 €
Entwicklung	1 Auszubildender	55 Std.	550,00 €	550,00 €	1100,00 €
Testing	1 Mitarbeiter	1,5 Std.	37,50 €	15.00 €	52,50 €
Abnahme	1 Mitarbeiter	1 Std.	25,00 €	10,00 €	35,00 €
Umgebung	1 Server	1 Jahr	-	-	203,52 €
					1691,02 €

TABELLE 2: RESSOURCENPLANUNG

3.5 Amortisationsdauer

Im Folgenden soll geprüft werden, ab welchem Zeitpunkt sich das Projekt amortisiert. Dazu werden die Anschaffungskosten und Entwicklungskosten dividiert. Da es hauptsächlich um die Mitarbeiterplanung im Büro zur Gewährleistung der Gesundheit und Sicherheit geht, ist dies Amortisation in diesem Projekt nicht unbedingt notwendig.

Vorgang	Anzahl Mitarbeiter	Zeit pro Vorgang	Zeit alt Pro Monat	Zeit neu Pro Monat	Einsparung pro Monat
Eintragen der Abwesenheiten	15	20 Minuten	300 Minuten	150 Minuten	150 Minuten
Planung der Mitarbeiter im Büro	1	60 Minuten	60 Minuten	30 Minuten	30 Minuten
					180 Minuten

TABELLE 3: AMORTISATIONSDAUER PRO MONAT

Berechnung der Amortisationsdauer:

180 Min/Monat x 12 Monate/Pro Jahr = 2.160 Min/Jahr = 36 h/Jahr

36 h/Jahr x (25€ + 20€) = 1620€/Jahr

1691,02€ / 1620€ = 1,04 Jahre = 1 Jahr und 2 Wochen

Die Projektkosten amortisieren sich somit nach der aufgeführten Rechnung, in 1 Jahr und 2 Wochen.

3.6 Nicht-monetäre Vorteile

Die nicht-monetären Vorteile gehen darin hervor, dass es eine Stabilität in der Büro Planung herrscht. Mitarbeiter haben mehr Freiheiten in der Planung ihrer Abwesenheitstage und können sich aussuchen, wann sie Home-Office machen wollen. Dies bringt mit sich das die Gesundheit aller Mitarbeiter im Vordergrund steht. Zusätzlich erleichtert es dem Personalchef die Planung.

3.7 User-Storys

Damit die Anforderungen des Kunden und der Mitarbeiter erfüllt werden können, wurden mithilfe des Projektverantwortlichen User-Storys erstellt, die die Software erfüllen soll. Unter einer User-Story versteht man eine Anforderung an eine Software. Diese Methode passt sehr gut zur agilen Entwicklung. Die User-Storys wurden ins Lastenheft übernommen.

3.8 Lastenheft

Zusammen mit dem Projektverantwortlichen wurde nach der Analysephase ein Lastenheft erstellt. Das wurde mit den voran gegangenen User-Storys umgesetzt. Ein Auszug aus dem Lastenheft befindet sich im Anhang A.4 AUSZUG AUS DEM LASTENHEFT

4. Entwurfsphase

Nach der Analysephase folgt der Entwurf des Projektes.

4.1 Zielplattform

Da das Projekt sowohl im Browser als auch in einem Microsoft Team Tab funktionieren soll, wurde die Webanwendung im MEAN-Stack entwickelt. Unter MEAN-Stack versteht man das Zusammenspiel aus MongoDB, Express.js, Angular Framework und Node.js. Für die Logik im Backend ist Node.js zuständig. Zum Übermitteln der Daten in die MongoDB Datenbank ist Express.js eingefügt. Die Benutzeroberfläche wird mit dem Angular Framework dargestellt. Die fertige Anwendung wird zum Schluss auf einem CentOS Cloud-Server in Microsoft Azure gehostet.

Bei der Auswahl der Programmiersprachen kommt es somit im Frontend mit Angular zu HTML5, CSS und TypeScript. Das Backend beruht auf Node.js, somit JavaScript.

4.2 Softwarearchitektur

Über ein lokales Intranet im Unternehmen, bzw. über einen zusätzlichen Tab im Firmen Microsoft Teams Portal, wird das Webprojekt innerhalb einer virtuellen Microsoft Azure Umgebung auf einem centOS Server mit Mongo-Datenbank bereitgestellt. Realisiert wird die Anwendung im Frontend mit dem Angular Framework (TypeScript, HTML, CSS). Das GUI wird mit Unterstützung von Material Design Komponenten dargestellt. Das Backend wird mit NodeJS und MongoDB realisiert. Wurden die Abwesenheitszeiten durch die Mitarbeiter in dem Kalender eingetragen, werden die Daten mit einer REST-API in Tabellen der Datenbank gespeichert. Die Datenbank beinhaltet Tabellen wie „User“ und „Termine“. Aufgrund der Natur von Mongo Datenbänken, wird die Verwaltung / Organisation der Datenbankstrukturen hauptsächlich innerhalb der Webanwendung durch die REST-API stattfinden. Abruf des Kalenders erfolgt durch Eingabe der E-Mail-Adresse und einem Passwort. Diese Daten sind versehen mit einer Administratorabfrage aus der Datenbank, je nach Berechtigung erhalten die

Benutzer ein Authentifizierungstoken und haben so Zugriff auf ein Administratorbereich oder nur auf ihr Profil und den Kalender. Dieses Token ist für 1 Stunde gültig. Im Administratorprofil können neue Benutzer angelegt und gelöscht werden, welches ebenfalls durch eine REST API funktioniert und werden mit der Tabelle „User“ abgeglichen. Der Kalender wird mit den Daten aus der Tabelle „Termine“ befüllt. Wurde von einem Administrator oder einem Mitarbeiter ein neuer Termin über die „neuer Termin“ Schaltfläche mit den auszufüllenden Daten getätigt, werden diese in der Tabelle „Termine“ neu angelegt.

4.3 Entwurf der Benutzeroberfläche

Der erste Entwurf für die Benutzeroberfläche wurde mit draw.io als Mockup erstellt. Die Oberfläche soll simpel und einfach zu bedienen sein. Dies ist im Anhang A.7 MOCKUP zusehen. Da das AES nur im Browser und als iFrame in Microsoft Teams benutzt werden soll, besteht keine Anforderung nach Responsivität.

Die Anwendung soll einen einheitlichen Grundaufbau haben und zu den Firmenfarben passen. Der Header soll aus einem Logo und der Navigation im Firmengrün bestehen. Die Inhalte werden den jeweiligen Navigationen darunter dargestellt. Die Hauptanwendung ist der Kalender, der den größten Teil in Anspruch nehmen wird. Je nach Authentifizierung der Benutzer werden einige Elemente nur dann angezeigt, wenn die passenden Berechtigungen vorhanden sind. Der Finale Entwurf ist im Anhang zusehen: ABBILDUNG 3: FINALER ENTWURF DER BENUTZEROBERFLÄCHE

4.4 Datenmodell

Da der Entwickler sich für den MEAN-Stack entschieden hat, wird MongoDB benutzt. MongoDB benutzt NoSQL (englisch für „Not only SQL“ deutsch: „Nicht nur SQL“), also eine Datenbank, die einen nicht relationalen Ansatz annimmt und keine festgelegten Tabellenschematas folgt. Somit war es nicht nötig ein Datenmodell anzulegen. Es wurden zwei Kollektionen angelegt: „User“ und „Termine“ siehe ABBILDUNG 7: MONGODB COLLECTIONS MODELL

4.5 Pflichtenheft

Mit dem Beenden der Entwurfsphase wurde mit den gewonnenen Erkenntnissen ein Pflichtenheft erstellt. Dieses beruht auf dem Lastenheft und den vorgegebenen Anforderungen.

Ein Auszug aus dem Pflichtenheft befindet sich im Anhang A.5 AUSZUG AUS DEM PFLICHTENHEFT

5. Implementierungsphase

Nach der Entwurfsphase erfolgt die Implementierungsphase und Entwicklung

5.1 Iterationsplanung

Bei dem Entwicklungsprozess handelt es sich um die agile SCRUM Methode, in kurzen Iterationszyklen hält der Entwickler Rücksprache mit dem Projektverantwortlichen, um Feedback einzuholen. Die Iterationen teilen sich in fünf Kategorien auf: Anlegen der Datenbank, Entwicklung des Frontends, Entwicklung des Backends, Tests und Implementierung in Microsoft Teams. Anhang A.6 ITERATIONSPLAN DER ENTWICKLUNG

5.2 Implementierung der Datenbank

Wie in Abschnitt 4.4 DATENMODELL erwähnt, benutzen wir eine NoSQL Datenbank, somit müssen lediglich zwei Kollektionen angelegt werden. Da es in Node.js ein passendes Feature für die MongoDB namens „mongoose“ gibt und somit das Frontend direkt mit dem Backend verbunden ist, wird sobald eine passende Eingabe getätigt wurde automatisch eine Kollektion angelegt siehe ABBILDUNG 7: MONGODB COLLECTIONS MODELL

5.3 Einrichtung der Entwicklungsumgebung

Als Entwicklungsumgebung wird Microsoft Visual Studio Code verwendet. Um mit Node.js und dem Angular Framework entwickeln zu können, wird zunächst der Node Package Manager installiert um weitere Templates von Node.js und Angular installieren zu können. Node.js dient in diesem Projekt als das Backend und Angular als das Frontend. Es wird ein Front- und ein Backend Ordner angelegt, in diese werden die Entsprechenden Zugehörigkeiten installiert.

Für das Frontend wird ein Konsolenbefehl in dem entsprechenden Ordner benutzt „npm install -g @angular/cli“, danach kann man mit dem Befehl „ng new gewünschtername“ ein Projekt anlegen. Dieser legt automatisch ein Skelett des Angular Frameworks mit den entsprechenden Modulen an.

Für das Backend wird ebenfalls ein Konsolenbefehl in dem entsprechenden Ordner benutzt „npm init“, so wird automatisch eine erste Version der Ordner Struktur und der benötigten Module angelegt.

5.4 Implementierung der Services

Sobald die Entwicklungsumgebung eingerichtet ist, kann mit der Programmierung begonnen werden. Im ersten Schritt hat der Autor das GUI implementiert. Dies geschieht im klassischen HTML und CSS-Stil siehe A.9 HEADER UND NAVIGATION. Hier ist bereits zusehen, dass Navigationspunkte nur zugänglich sind, wenn man eingeloggt ist und andere nur wenn man als Administrator authentifiziert ist.

5.4.1 Implementierung der Funktion zum Anlegen von Benutzern und Login

Damit die Verbindung zwischen Backend und der DB hergestellt werden kann wird „mongoose“ implementiert welches über eine URL mit der API kommunizieren kann siehe A.13 MONGO DB CONNECTION.

Zum Anlegen der Benutzer hat der Autor zunächst ein UserModel angelegt siehe A.10 USER MODEL. Danach wurde eine REST API angelegt, über die das Backend angesprochen werden kann über eine URL mit Anhang kann so kommuniziert werden http://URL:3000/users/register siehe A.11 REGISTER API, URL wurde aus Sicherheitsgründen geändert. Die übermittelten Benutzerdaten werden in die Datenbank übergeben und das Passwort wird mit bcrypt verschlüsselt und abgelegt: „password:"$2b$10$W2r6JpEInqA18Do/yDllleP1dyeXduZikt2fDEYWTvmWA2.byhLNe"

Die Login Funktion läuft über die REST API http://URL:3000/users/login siehe A.12 LOGIN API . Hier vergleicht bcrypt das Passwort, das übermittelt wurde, mit dem verschlüsseltem Passwort aus der Datenbank. Wenn die Daten nicht übereinstimmen, bekommt der Benutzer eine Fehlermeldung, ansonsten wird ein Payload als JWT zur Anwendung übermittelt, um den Login auszuführen und eventuelle Adminrechte zu authentifizieren.

5.4.2 Implementierung des Authentifikations-Service

Im Frontend erstellt der Autor ein Authentifikations-Service, um auch dort den Login und eventuelle Adminrechte übergeben zu können siehe A.14 AUSZUG AUTHENTIFIKATIONS-SERVICE. Hier wird das JWT übergeben und die Funktion isLoggedIn() erstellt in der auch überprüft wird, ob der eingeloggte Benutzer ein Administrator ist.

5.4.3 Implementierung des Kalenders

Angular macht es möglich vorgefertigte Lizenzfreie Module zu importieren. Der Autor hat sich hier nach Recherche für Kalender von „mattlewis92“ https://github.com/mattlewis92/angular-calendar entschieden, ein Modul welches Kommerziell benutzt werden kann. Der Kalender wird über den Befehl „ng add angular-calender“ in das Projekt implementiert. Und im Frontend an die Vorgegebene Struktur angepasst siehe A.15 FRONTEND ANGULAR CALENDER.

Die Termine werden als JSON Object dem Backend übergeben siehe A. 16 JSON OBJECT „TERMINE“ und werden von dort in die Datenbank gespeichert und bei neuem Aufrufen geladen siehe A.17 ABRUFEN DER KALENDER TERMINE BEI AUFRUF DER ANWENDUNG

6. Abnahme und Deployment

Nach der Entwicklung kann die Anwendung nun bereitgestellt werden.

6.1 Abnahme durch den Projektverantwortlichen

Nachdem das AES fertig entwickelt war, wurde es im letzten SCRUM Meeting dem Projektverantwortlichen vorgelegt. Da die gewünschten Funktionalitäten bereits bekannt waren, gab es bei der Endabnahme keine Probleme. Der Code wurde noch einmal auf Sicherheitslücken überprüft, hier ließ sich aber nichts finden.

6.2 Veröffentlichung und Deployment

Die Veröffentlichung lief reibungslos ab, die fertige Anwendung wurde auf dem bereits zur Verfügung gestellten centOS Server hochgeladen und gehostet. Die Mitarbeiter wurden in die Datenbank eingepflegt. Danach musste nur noch die passende URL integriert werden. Anschießend konnten alle Mitarbeiter problemlos über die zur Verfügung gestellte URL mit ihren angelegten Logindaten Zugriff erhalten.

6.2.1 Deployment in Microsoft Teams

Als letztes wurde das AES in Microsoft Teams als eigenständiger Tab hinzugefügt, dies ließ sich sehr schnell und problemlos über die eingebaute Registrierkartenfunktion umsetzen. Es wird eine neue Registrierkarte angelegt und die Website kann direkt mir entsprechender URL aufgerufen werden siehe ABBILDUNG 5: REGISTRIERKARTE HINZUFÜGEN. Danach muss die neu erstellte Registrierkarte im entsprechenden Team aufgerufen werden und die Anwendung kann in Teams verwendet werden. A.19 ABSCHLUSSDARSTELLUNG IN MICROSOFT TEAMS

6.3 Dokumentation

Die Dokumentation des AES setzt sich aus dem Anwenderhandbuch und der Projektdokumentation zusammen, diese wurden beide vom Autor erstellt.

Das Anwenderhandbuch dient als Bedienungsanleitung für das AES und soll es den Mitarbeitern schnell und unkompliziert machen die Anwendung zu benutzen. Ein Auszug der Anwenderdokumentation befindet sich im Anhang:

7. Fazit

Zum Abschluss des Projektes wird ein entsprechendes Fazit gezogen.

7.1 Soll/Ist-Vergleich

Wenn man auf das Projekt zurückblickt, wurden alle Anforderungen erfüllt. Der am Anfang erstellte Projektplan konnte eingehalten werden. In der Tabelle ist der benötigte Zeitaufwand dargestellt. Es ist zu erkennen, dass es in der Entwurfsphase eine Differenz gegeben hat, diese konnte aber bei anderen Projektphasen wieder rausgeholt werden.

Projektphase	Soll	Ist	Differenz
Analyse	8 Std.	8 Std.	0 Std.
Entwurf	9 Std.	10 Std.	+2 Std.
Implementierung	39 Std.	39 Std.	0 Std.
Abnahme und Deployment	3 Std.	2 Std.	-1 Std.
Dokumentation	11 Std.	10 Std.	-1 Std
Gesamt:	70 Std.	70 Std.	70 Std.

TABELLE 4: SOLL/IST-VERGLEICH

7.2 Gewonnene Erkenntnisse

Im Laufe des Projektes konnte der Autor Wissen in den Bereichen Projektmanagement, agiler Softwareentwicklung, Single-Page-Applikation, MEAN-Stack erlernen, aber auch neue Erfahrung in den Programmiersprachen TypeScript und Javascript sammeln. Besonders die gewonnenen Erfahrungen mit dem MEAN-Stack werden weiterhin sehr nützlich sein und in diese Richtung legt der Autor auch weiterhin sein Interesse. Somit war das Projekt nicht nur für 34D ein Gewinn, sondern auch für den Autor und seinen weiteren Werdegang.

7.3 Ausblick

Aufgrund der vielen Möglichkeiten, die noch aus diesem Projekt herauszuholen sind, plant die 34D schon neue Addons die im firmenbetrieblichen Alltag weitere Terminplanung oder Rechnungsplanung abnehmen kann. Somit war und ist das Projekt für alle Beteiligten ein voller Erfolg!

A. Anhang

A.1 Detaillierte Zeitplanung

Analyse		**8 Std.**
1. Ist-Analyse	2 Std.	
2. Soll-Konzeptionierung	2 Std.	
3. Erstellen des Lastenheftes	2 Std.	
4. Wirtschaftlichkeitsprüfung	2 Std.	
Planung		**9 Std.**
1. Entwurf der Oberflächen und Masken	3 Std.	
2. Backendstruktur planen	3 Std.	
3. Pflichtenheft erstellen	3 Std.	
Implementierung und Tests		**39 Std.**
1. Entwicklung der GUI	10 Std.	
2. Entwicklung Backend	18 Std.	
- Implementierung Login	9 Std.	
- Implementierung der APIs	9 Std.	
3. Anlegen des Hostservers	6 Std.	
4. Bereitstellung der Software auf den Server	2 Std.	
5. Testvorgänge der Software	3 Std.	
- Test der Login Funktion	0,5 Std.	
- Test mit Beispiel-Daten	1 Std.	
- Test der Benutzerfreundlichkeit	1,5 Std.	
Deployment		**3 Std.**
1. Einpflegen der Mitarbeiter in die DB	1 Std.	
2. Als Iframe in Teams einbinden	2 Std.	
Erstellen der Dokumentation		**11 Std.**
1. Erstellen der Projektdokumentation	9 Std.	
2. Erstellen der Entwicklerdokumentation	1 Std.	
3. Erstellen der Anwenderdokumentation	1 Std.	

TABELLE 5: DETAILLIERTE ZEITPLANUNG

A.2 Verwendete Ressourcen

Hardware

- Büroarbeitsplatz mit Laptop und 2 Monitoren

Software

- Windows 10
- Visual Studio Code
- Postman – Software zum Testen der APIs
- MongoDB Compass
- Microsoft Azure Cloud
- Trello – Projektmanagementsoftware
- Microsoft Word
- Draw.io – Online Mockup Tool
- Adobe XD

Personal

- Entwickler – Projektumsetzung
- Mitarbeiter – Test der Benutzerfreundlichkeit
- Projektverantwortlicher

A.3 Trello Board

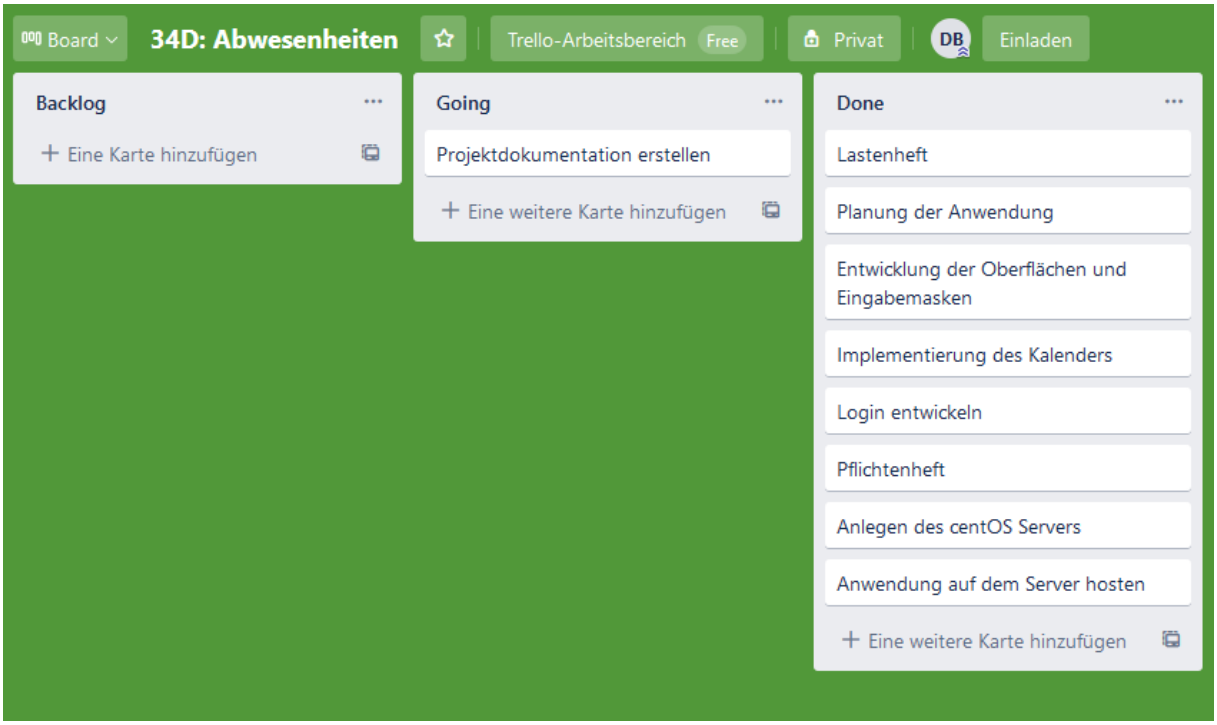


ABBILDUNG 1: TRELLO

A.4 Auszug aus dem Lastenheft

User Story 1

Nr.	1	**Titel**	Login Funktion
Quelle	Mitarbeiter	**Priorität**	Hoch

Beschreibung

Als Benutzer oder Administrator möchte ich mich mit meiner E-Mail-Adresse und meinem Passwort anmelden können, damit ich sicherstellen kann, dass niemand meine Daten einsehen kann.

Voraussetzung

- Es muss eine Loginmaske vorhanden sein
- Es muss eine E-Mail und Passwort Abfrage gesendet werden
- Benutzer muss als Mitarbeiter oder Administrator authentifiziert werden

Auslöser

- Benutzer gibt in der Loginmaske seine E-Mail-Adresse, sein Passwort ein und klickt auf Login

Schätzung des Aufwands

- Masken entwickeln
- Backendlogik entwickeln
- Aufwand ca. 10 Stunden

User Story 2

Nr.	2	**Titel**	Administrator Bereich
Quelle	Administrator	**Priorität**	Hoch

Beschreibung

Als Administrator möchte ich einen Bereich haben, um Benutzer verwalten zu können.

Voraussetzung

- Es muss einen Administrator Bereich geben
- Es müssen Masken vorhanden sein, um neue Benutzer anzulegen
- Es muss eine Übersicht aller angelegten Benutzer geben

Auslöser

- Beim Login wird der Administrator authentifiziert

Schätzung des Aufwands

- Masken und Auflistung entwickeln
- Backendlogik entwickeln
- Aufwand ca. 3 Stunden

A.5 Auszug aus dem Pflichtenheft

Benutzeroberfläche

- **Layout**
  - Besteht aus einem Header mit einer Navigationsleiste (Profil, Adminbereich, Kalenderansicht und Logout Funktion). Darunter ist Platz für den Inhalt
- **Login**
  - Besteht aus einem Feld mit Eingabefunktion von E-Mail-Adresse und Passwort und Login Button
- **Administratorbereich**
  - Besteht aus einer Übersichtstabelle aller angelegten Benutzer mit Namen und E-Mail-Adressen und einer Löschfunktion
  - Eingabe Maske für neue Benutzer (Vorname, Nachname, E-Mail-Adresse, Einmal-Passwort)
- **Benutzerprofil**
  - Besteht aus einer Übersicht der persönlichen Daten (Vorname, Nachname und E-Mail-Adresse)
  - Funktion zur Änderung des Passworts

- **Kalenderansicht**
    - Besteht aus dem Kalender mit Schaltflächen zum Ändern des Monats/Wochen/Tage
    - Button zum Hinzufügen neuer Termine
    - Einsicht in bereits erstellte Termine

Technische Produktumgebung

- **Software**
    - Server-Betriebssystem: auf Linux basiertes CentOS
    - Client-Betriebssystem: Windows mit gängigem Browser
- **Hardware**
    - Server: wird in der Cloud gehostet
    - Client: PC oder Laptop und browserfähiges Gerät mit Grafikbildschirm

A.6 Iterationsplan der Entwicklung

- Anlegen der Datenbank Collections mit MongoDB
- Implementierung des GUIs
- Implementierung der Backend-Logik
- Testen der Software mit Mitarbeitern
- Implementierung in Microsoft Teams

A.7 Mockups

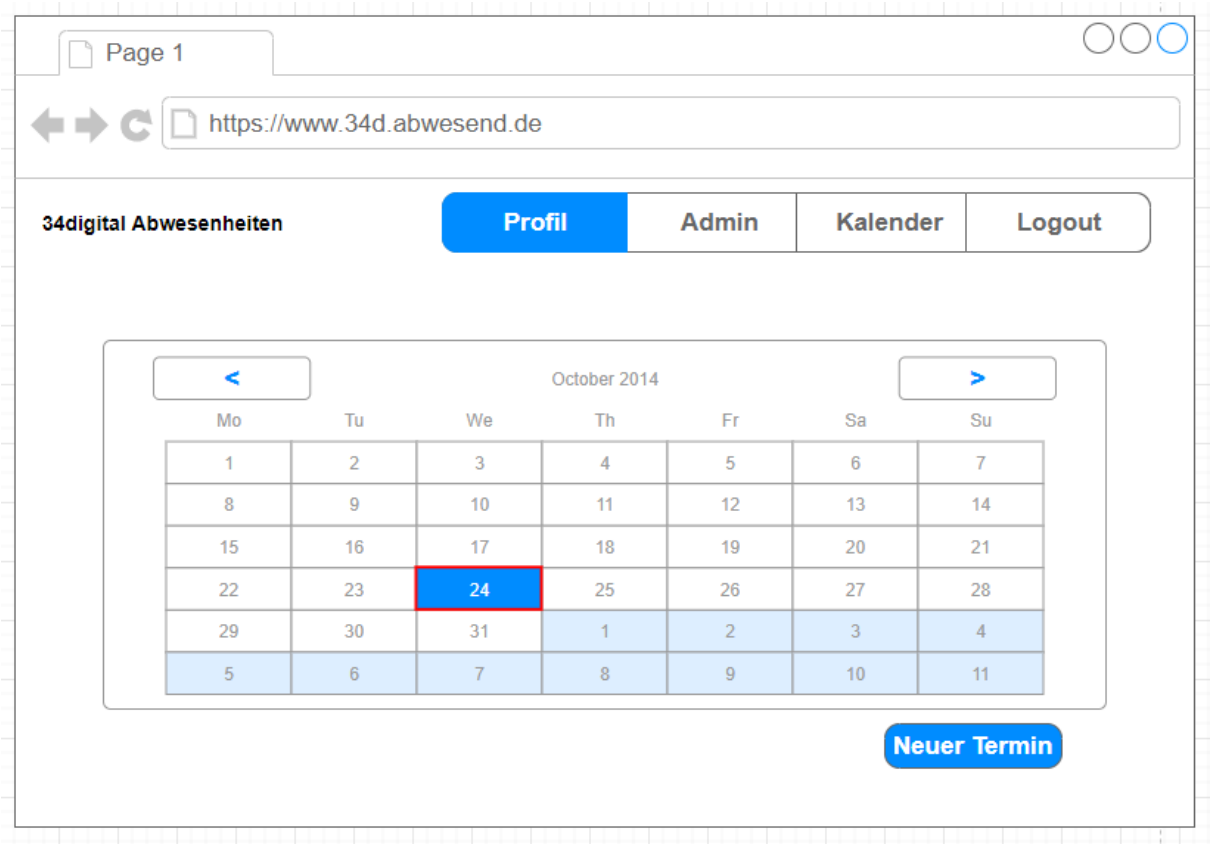


ABBILDUNG 2: ERSTER ENTWURF DER BENUTZEROBERFLÄCHE

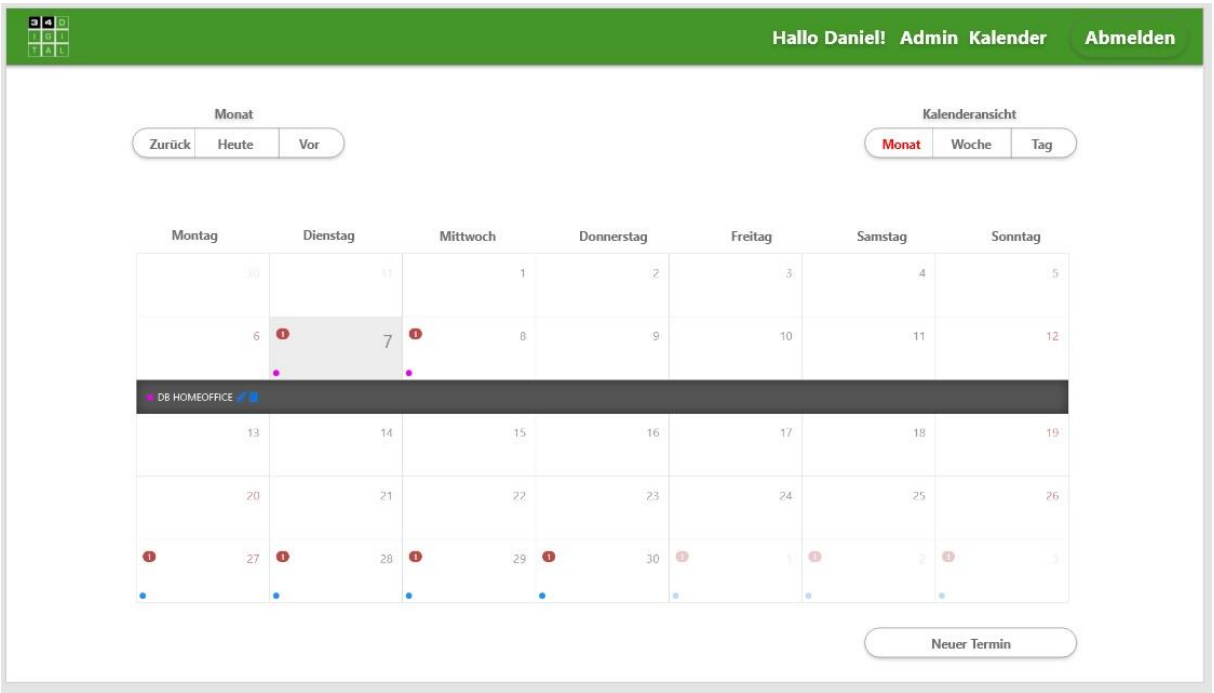


ABBILDUNG 3: FINALER ENTWURF DER BENUTZEROBERFLÄCHE

A.8 Auszug Anwenderdokumentation

4. Der Kalender

4.1 Die Kalenderansicht

Nachdem Login oder durch das Klicken auf **Kalender** in der Navigationsleiste befinden Sie sich direkt in der Firmeninternen Kalenderansicht:

Mit den 3 Buttons oben rechts 

können Sie die Ansichten ändern.

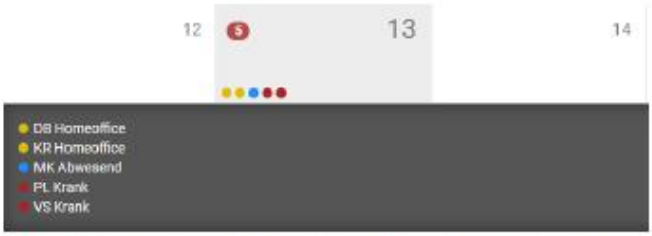
4.1.1 Die Monatsansicht

Die standard Ansicht ist die Monatsansicht wie in „4.1 Die Kalenderansicht" schon abgebildet. Hier sehen Sie als erstes immer den aktuellen Monat **OKTOBER 2020** .

Mit den 3 Button oben links 

kann man die Monate wechseln oder auf den Aktuellen Monat zurückspringen.

Wenn Sie einen Tag mit Abwesenheiten sehen und den Tag anklicken sehen Sie genauere Informationen:



An diesem Tag gibt es 5 Abwesenheiten dafür steht die „5 im roten Feld" und unten werden diese aufgelistet.

ABBILDUNG 4: AUSZUG ANWENDERDOKUMENTATION

A.9 Header und Navigation

```
<header class="mat-elevation-z4">
  <div class="container">
    <a class="logo">Abwesenheiten</a>
    <nav>
      <ul>
        <li *ngIf="auth.isLoggedIn()"><a routerLinkActive="headeractive" route
rLink="/profil">Hallo {{auth.getUserDetails()?.first_name}}</a></li>
        <li *ngIf="auth.isLoggedIn()"><a routerLinkActive="headeractive" route
rLink="/kalender">Kalender</a></li>
        <li *ngIf="auth.isLoggedIn() && auth.AuthIsAdmin"><a routerLinkActive=
"headeractive" routerLink="/admin">Admin</a></li>
        <li *ngIf="auth.isLoggedIn()"><a  routerLink="" (click)="auth.logout()
">Logout</a></li>
      </ul>
    </nav>
  </div>
</header>

<div class="container">
  <router-outlet></router-outlet>
</div>
```

A.10 User Model

```
const UserSchema = new Schema({
    first_name: {
        type: String
    },
    last_name: {
        type: String
    },
    email: {
        type: String,
        require: true
    },
    password: {
        type: String,
        require: true
    },
    date: {
        type: Date,
        default: Date.now
    },
    admin: {
        type: Boolean,
        require: true
    }})
```

A.11 Register API

```
// registrieren
users.post('/register', (req, res) => {
    const today = new Date()
    const userData = {
        first_name: req.body.first_name,
        last_name: req.body.last_name,
        email: req.body.email,
        password: req.body.password,
        admin: req.body.admin,
        created: today
    }

    User.findOne({
        email: req.body.email
    })
    .then(user => {
        if(!user) {
            bcrypt.hash(req.body.password, 10, (err, hash) => {
                userData.password = hash
                User.create(userData)
                .then(user => {
                    res.json({status: user.email + ' wurde registriert!'})
                })
                .catch(err =>{
                    res.send('error ' + err)
                })
            })
        } else {
            res.json({ error: 'Benutzer existiert bereits!'})
        }
    })
    .catch(err => {
        res.send('error: ' + err)
    })
})
```

A.12 Login API

```
//login
users.post('/login', (req,res) => {
    User.findOne({
        email: req.body.email
    })
    .then(user =>{
        if(user) {
            if(bcrypt.compareSync(req.body.password, user.password)){
                const payload = {
```

```
                    _id: user._id,
                    first_name: user.first_name,
                    last_name: user.last_name,
                    email: user.email,
                    admin: user.admin,
                }
                let token = jwt.sign(payload, process.env.SECRET_KEY, {
                    expiresIn: '1d'
                })
                res.json({token : token})
            }else {
                res.json({error: "Benutzer nicht gefunden!"})
            }
        }else{
            res.json({error: "Benutzer nicht gefunden!"})
        }
    })
    .catch(err => {
        res.send('error ' + err)
    })
})
```

A.13 Mongo DB Connection

```
const mongoURI = 'mongodb://URL:27017/ProjektDB'

mongoose
    .connect(mongoURI, {useNewUrlParser: true})
    .then(() => console.log("MongoDB ist Verbunden :-)"))
    .catch(err => console.log(err))

var Users = require('./routes/Users')

app.use('/users', Users)


app.listen(port, function() {
    console.log("Server laeuft auf port: " + port)
})
```

A.14 Auszug Authentifikations-Service

```
export interface TokenResponse {
    token: any
}

export interface TokenPayload {
    _id: string
```

```
    first_name: string
    last_name: string
    email: string
    admin: boolean
}

@Injectable()
export class AuthenticationService {
  private token: any
  public AuthIsAdmin: boolean

  constructor(private http: HttpClient, private router: Router) {}

  private saveToken(token: any): void {
    localStorage.setItem('usertoken', token)
    this.token = token
  }

  private getToken(): any {
    if(!this.token){
        this.token = localStorage.getItem('usertoken')
    }
    return this.token
  }

  public getUserDetails(): UserDetails {
    const token = this.getToken()
    let payload
    if(token) {
        payload = token.split('.')[1]
        payload = window.atob(payload)
        return JSON.parse(payload)
    }else{
        return null
    }
  }

  public isLoggedIn(): boolean {
      const user = this.getUserDetails()
      if(user) {
          this.AuthIsAdmin=user.admin
          return user.exp > Date.now() / 1000
      }else{
        return false
      }
  }
```

A.15 Frontend Angular Calender

```
<div class="kalendercontainer">
  <div class=buttoncontainer>
    <button
      mat-raised-button
      mwlCalendarPreviousView
      [view]="view"
      [(viewDate)]="viewDate"
    >
    <mat-icon>undo</mat-icon>
    </button>
    <button
      mat-raised-button color="warn"
      mwlCalendarToday
      [(viewDate)]="viewDate"
    >
      Aktuell
  </button>
    <button
    mat-raised-button
      mwlCalendarNextView
      [view]="view"
      [(viewDate)]="viewDate"
    >
    <mat-icon>redo</mat-icon>
</button>
<button
class="monthbutton"
mat-raised-button
(click)="setView(CalendarView.Month)"
[class.active]="view === CalendarView.Month"
>
Monat
</button>
<button
mat-raised-button
(click)="setView(CalendarView.Week)"
[class.active]="view === CalendarView.Week"
>
Woche
</button>
<button
mat-raised-button
(click)="setView(CalendarView.Day)"
[class.active]="view === CalendarView.Day"
>
Tag
</button>
</div>
```

```
  <h3>
    {{ viewDate | calendarDate:(view + 'ViewTitle'):locale:weekStartsOn }}
  </h3>

  <div [ngSwitch]="view">
  <mwl-calendar-month-view
  *ngSwitchCase="'month'"
  [viewDate]="viewDate"
  [events]="events"
  [locale]="locale"
  [weekStartsOn]="weekStartsOn"
  [weekendDays]="weekendDays"
  [activeDayIsOpen]="true"
  (dayClicked)="dayClicked($event.day)"
  (beforeViewRender)="beforeMonthViewRender($event)"

>
</mwl-calendar-month-view>
<mwl-calendar-week-view
    *ngSwitchCase="'week'"
    [viewDate]="viewDate"
    [events]="events"
    [locale]="locale"
    [weekStartsOn]="weekStartsOn"
    [weekendDays]="weekendDays"
    [dayStartHour]="6"
    [dayEndHour]="18"
  >
  </mwl-calendar-week-view>
  <mwl-calendar-day-view
    *ngSwitchCase="'day'"
    [viewDate]="viewDate"
    [events]="events"
    [locale]="locale"
    [dayStartHour]="6"
    [dayEndHour]="18"
  >
  </mwl-calendar-day-view>
</div>
</div>
```

A. 16 JSON Object „Termine“

```
events: CalendarEvent[] = [
    {
      start: new Date('2020-10-13'),
      end: new Date('2020-10-14'),
      title: 'DB Homeoffice',
      allDay: true,
      color: colors.yellow,
    },
    {
      start: new Date('2020-09-24'),
      end: new Date('2020-09-24'),
      title: 'DB Krank',
      allDay: true,
      color: colors.blue,
    },
```

A.17 Abrufen der Kalender Termine bei Aufruf der Anwendung

```
  beforeMonthViewRender({
    body,
  }: {
    body: CalendarMonthViewDay<EventGroupMeta>[];
  }): void {
    body.forEach((cell) => {
      const groups = {};
      cell.events.forEach((event: CalendarEvent<EventGroupMeta>) => {
        groups[event.meta.type] = groups[event.meta.type] || [];
        groups[event.meta.type].push(event);
      });
      cell['eventGroups'] = Object.entries(groups);
    });
  }

}
```

A.18 Registrierkarte in Microsoft Teams hinzufügen

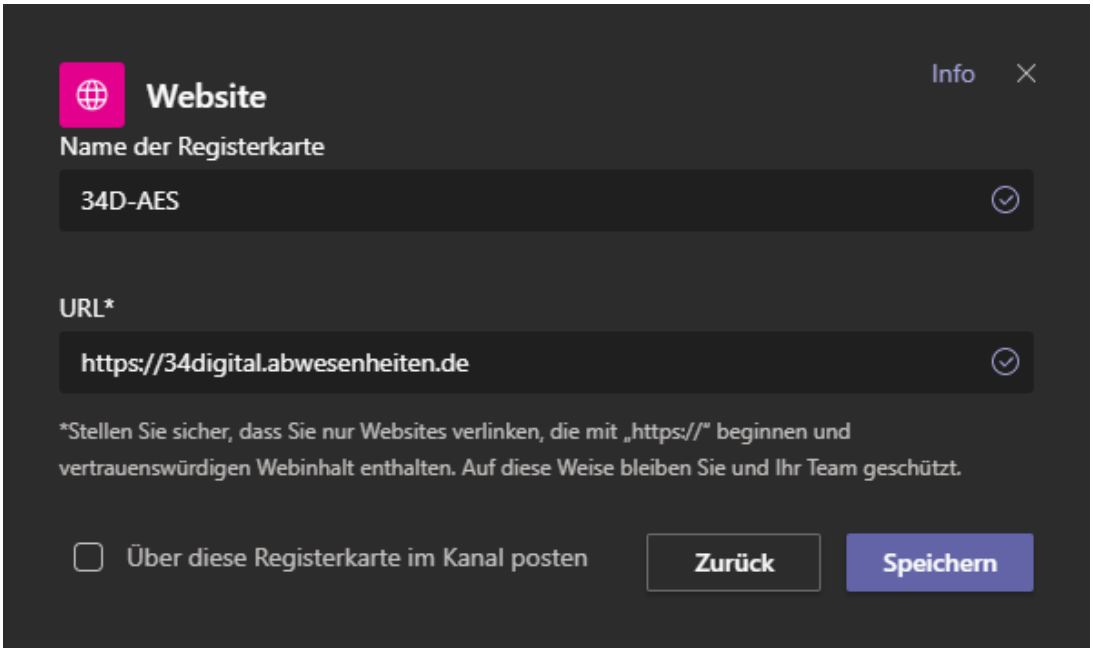


ABBILDUNG 5: REGISTRIERKARTE HINZUFÜGEN

A.19 Abschlussdarstellung in Microsoft Teams

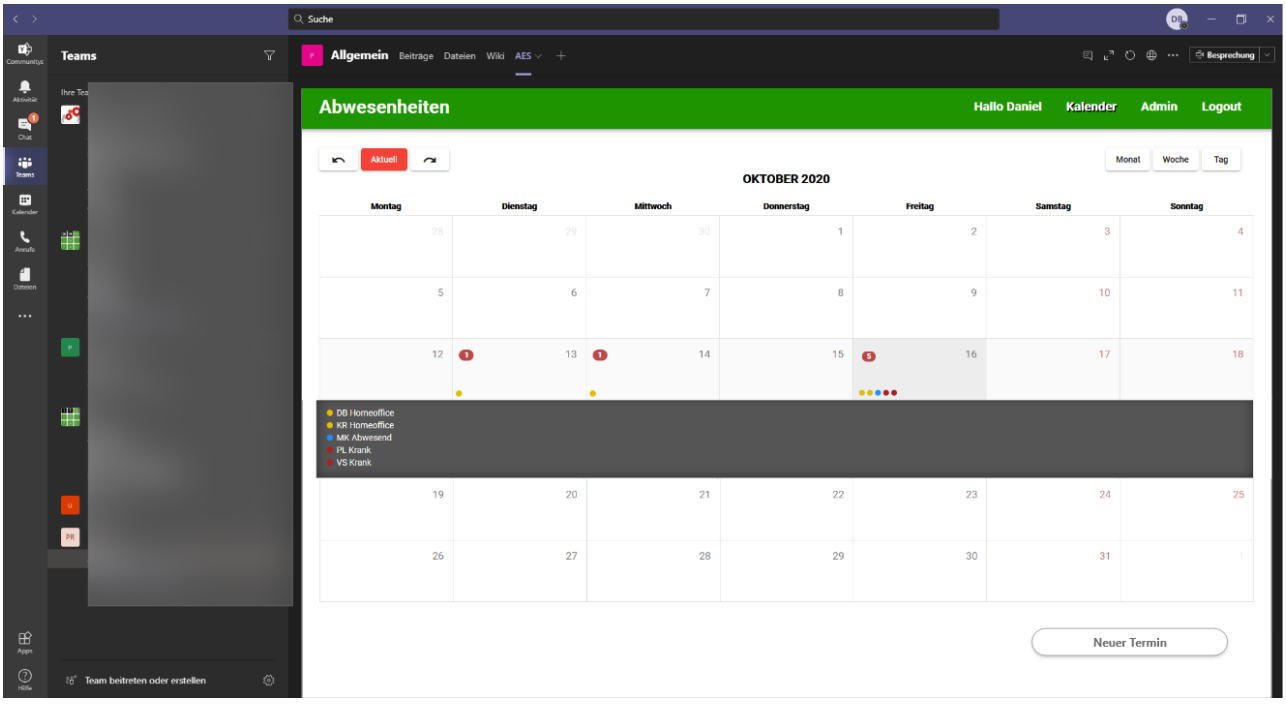


ABBILDUNG 6: FERTIGE DARSTELLUNG DES AES IN MICROSOFT TEAMS

A. 20 MongoDB Collections Modell

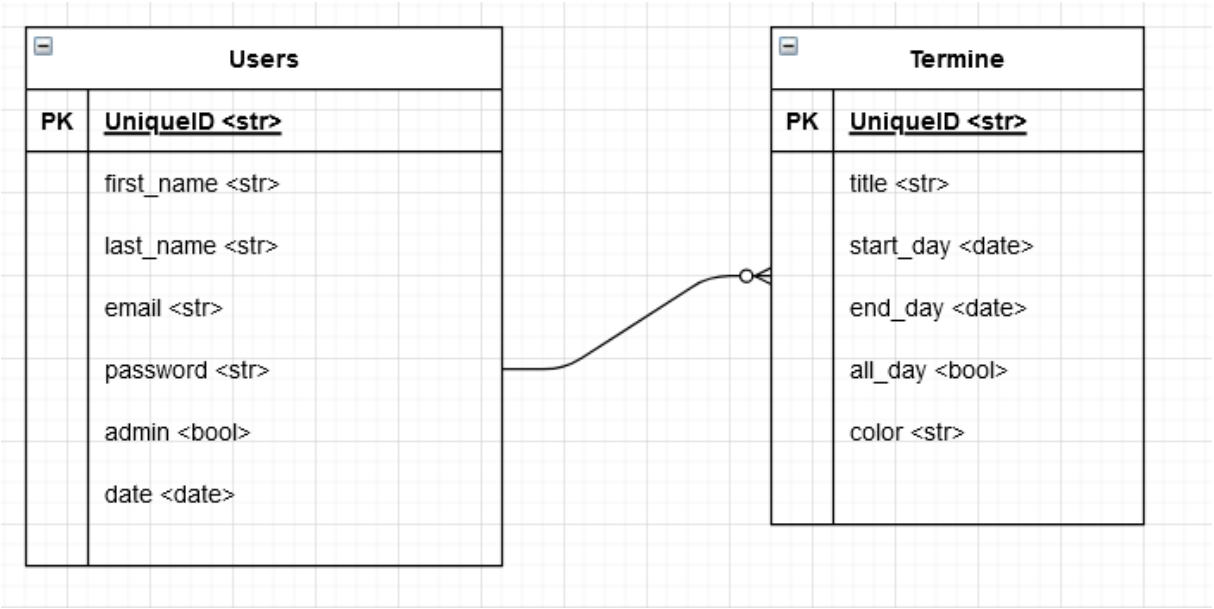


ABBILDUNG 7: MONGODB COLLECTIONS MODELL