Meeting Quality

Bachelorarbeit

Studiengang Informatik

OST – Ostschweizer Fachhochschule

Campus Rapperswil-Jona

Herbstsemester 2022

Autor(en):	Etienne Beyeler
	Lukas Volk
Betreuer:	Prof. Frank Koch
Projektpartner:	AdaptIT GmbH Rapperswil-Jona
Experte:	Prof. Hansjörg Huser
Gegenleser:	Prof. Beat Stettler

Abstract

Ausgangslage Meetings, ob vor Ort oder online durchgeführt, sind ein essenzieller Bestandteil der modernen Arbeitswelt. Je nach Vorbereitung und Durchführung unterscheidet sich ihre Beschaffenheit jedoch merklich. Um die Qualität von Meetings zu messen und kontinuierlich zu verbessern, wurde von der AdaptIT GmbH die App «Meeting Quality» (MQ- App) entwickelt. Sie erlaubt es, strukturiertes Feedback zu Besprechungen zu sammeln und durch den/die Organisator:in einzusehen.
Zu Beginn der Bachelorarbeit liegt die MQ-App als Minimum Viable Product (MVP) vor. Das MVP basiert auf einer 3-Tier-Architektur mit Client-Server-Cuts, umgesetzt durch Technologien aus dem MERN- Stack: React im Frontend, Express und Node.js im Backend und MongoDB in der Datenbank.

Ziel der Arbeit Der bestehende Funktionsumfang der MQ-App soll um folgende Funktionen erweitert werden:
Dashboard: Es soll ein Dashboard erstellt werden, welches die Feedbackdaten aggregiert und in geeigneter Form darstellt.
Automatischer Versand des Feedbackbogens: Damit der Link zum Feedbackformular nicht manuell an die Teilnehmenden geschickt werden muss, soll er im Anschluss an ein Meeting automatisch versendet werden.
Integration in Microsoft Teams: Die MQ-App soll direkt aus Microsoft Teams (MS Teams) bedient werden können. Ausserdem sollen Besprechungen, welche in MS Teams erstellt und geändert werden, automatisch in die MQ-App übertragen werden.

Ergebnis Die MQ-App wurde um den definierten Funktionsumfang erweitert und die bestehende Architektur in diversen Punkten verbessert. Des Weiteren wurden verschiedene Refactorings zur Optimierung des Quellcodes durchgeführt.
Im neu erstellten Dashboard werden die Daten von allen Feedbacks zusammengefasst und mittels Diagrammen visualisiert. Neben der Übersicht, über Effizienz und Dauer der Besprechungen, soll eine Sterne-Bewertung einen Gesamteindruck der allgemeinen Meeting-Qualität vermitteln.
Der Link zum Feedbackformular wird automatisch via E-Mail versendet – vorausgesetzt, die E-Mail-Adressen der Teilnehmenden wurden erfasst. Dazu wurde der webbasierter E-Mail-Zustelldienst «SendGrid» an die MQ-App angebunden.
Die MS-Teams-Integration konnte auf zwei Ebenen umgesetzt werden. Einerseits wurde eine zusätzliche App für MS Teams erstellt, welche über den App-Katalog installiert werden kann. Sie beinhaltet im Wesentlichen ein iframe, der es erlaubt die MQ-App in MS Teams einzubetten. Andererseits können sich Benutzer:innen in der MQ-App mit ihrem Microsoft- Account verbinden. Dadurch wird im Backend ein Abonnement bei der Microsoft-Graph-API erstellt. Daraufhin sendet Microsoft Daten zu neu erstellten und geänderten Besprechungen an das Backend. Das Backend verarbeitet diese Daten und persistiert sie in der eigenen Datenbank.
Über eine Importfunktion ist es ausserdem möglich, bereits vorhandene MS-Teams-Meetings in die MQ-App zu importiert.

Lay Summary

Ausgangslage

Meetings sind aus der modernen Arbeitswelt nicht mehr wegzudenken. Dazu gehören besonders Online-Meetings, welche spätestens seit der globalen Covid-19-Pandemie essenzieller Teil des Berufsalltags sind, um verteilte Teams zusammenzubringen. Als Plattform für Online-Meetings hat sich – neben Zoom und weiteren Anbietern – vor allem Microsoft Teams (MS Teams) durchgesetzt.

Je nach Vorbereitung und Durchführung unterscheidet sich die Qualität von Meetings merklich: Während Informationen in einem gut organisierten Meeting ideal vermittelt werden, stellen ungenügend vorbereitete oder zwecklose Besprechungen eine Belastung für alle Teilnehmenden dar.

Um die Qualität von Meetings zu messen und diese kontinuierlich verbessern zu können, wurde vom Industriepartner, der AdaptIT GmbH, die Applikation ≪Meeting Quality≫ entwickelt. Sie erlaubt es, strukturiertes Feedback von Teilnehmenden zu sammeln. Die Feedbacks können dann anschliessend durch den/die Organisator:in eingesehen werden. Zu Beginn der Bachelorarbeit liegt die App als Minimum Viable Product (MVP) vor.

Ziel der Arbeit

Da die Anwendung nicht direkt durch den Industriepartner, sondern durch einen externen Dienstleister entwickelt wurde, soll als Erstes in einer Analyse der Ausgangszustand der App festgehalten werden.

Weiter soll der funktionale Umfang der Meeting-Quality-App nach den Vorstellungen des Industriepartners ausgebaut werden: Aus den Feedbackdaten sollen weitere aufschlussreiche Informationen über die Meeting Produktivität errechnet werden. Dabei soll der Aufwand bei Verwendung der Applikation für Benutzende weiter minimiert werden.
Im Vordergrund steht die Integration der Anwendung in MS Teams. Das Feedbackformular soll im Anschluss an eine Besprechung automatisch an die Teilnehmenden verschickt werden und über ein Dashboard können die wichtigsten Informationen aller Feedbacks zusammengefasst eingesehen werden.

Vorgehen

Das Vorgehen im Projekt ist ablauforientiert und wird durch Meilensteine in zeitliche Abschnitte unterteilt. Als Vorgehensmodell kommt eine Mischung aus Scrum und dem Unified Process (UP) zum Einsatz. Aus dem UP werden übernommen die vier Phasen (*Inception*, *Elaboration*, *Construction* und *Transition*). Innerhalb der Phasen wird in wochenbasierten Iterationen gearbeitet.

In den ersten zwei Phasen wird ein Projektplan erstellt, die Businessfälle, Use Cases und User Storys erarbeitet sowie der Ist-Zustand der Applikation analysiert. In einer Variantenstudie sollen ausserdem die verschiedenen Integrationsmöglichkeiten für MS Teams beleuchtet werden. In der Konstruktionsphase werden schliesslich die so definierten Anforderungen umgesetzt und die Architektur der Meeting-Quality-App durch Refactorings optimiert werden, um so die künftig benötigte Zeit und die damit verbundenen Kosten für

die Weiterentwicklungen der Anwendung zu verringern. In der finalen Transitionsphase wird die Anwendung in die Cloud deployed und an den Industriepartner übergeben.

Technologien

Das bestehende MVP basiert auf einer 3-Schichten-Architektur und bestehend aus einem Frontend, Backend und einer Datenbank. Die Technologien dazu stammen aus dem MERN-Stack: MongoDB, Express, React und Node.js.

Ergebnisse

Die in Zusammenarbeit mit dem Industriepartner und Betreuer definierten Mindestanforderungen konnten alle umgesetzt werden.

Einerseits wurde die Anwendung um ein Dashboard erweitert, welches die Daten der Besprechungsfeedbacks mit Diagrammen visualisiert: Ein Balkendiagramm für Effizienz und Dauer der Meetings und eine Sterne-Bewertung, die einen Gesamteindruck der von den Teilnehmenden empfundenen Qualität wiedergibt. Das Dashboard ist so aufgebaut, dass es sich zukünftig um beliebige weiter Metriken erweitern lässt.

Anderseits können beim Erstellen von Meetings nun auch die E-Mail-Adressen der Teilnehmenden und der Start- und Endzeitpunkt der Besprechung erfasst werden. Im Anschluss an ein Meeting wird so der Link zum Bewertungsbogen automatisch per Mail verschickt und der manuelle Prozess entfällt.

Des Weiteren kann die Meeting-Quality-App neu direkt aus MS Teams bedient werden. Dazu wurde eine MS-Teams-App erstellt, welche über den App-Katalog von Microsoft installiert werden kann. Die Bedienung der App via Webbrowser ist selbstverständlich weiterhin möglich.
Durch die Anbindung der Microsoft-Graph-API können Benutzende ihren Microsoft-Account mit der Meeting-Quality-App verbinden und in den Einstellungen die Option wählen, dass Besprechungen mit Microsoft synchronisiert werden sollen. So werden neue Meetings, welche der/die Benutzer:in in MS Teams erstellt und verändert, automatisch in die Meeting-Quality-App übernommen.

Die grundlegende logische Architektur der Meeting-Quality-App konnte in diversen Punkten verbessert werden. Die bestehenden Teile der Applikation wurde in eine intuitivere, Feature-basierte Projektstruktur überführt und der Quellcode von JavaScript nach TypeScript migriert. Durch die Einführung der typensicheren Programmiersprache steht den Entwickelnden nun ein besseres Tooling zur Verfügung und Fehler können bereits zur Kompilierzeit – und nicht erst zur Laufzeit – abgefangen werden. Dies erhöht die Sicherheit, Performanz und Resistenz der Applikation. Durch gezielte Refactorings konnten ausserdem viele bestehende Code Smells – Anti-Pattern im Quellcode – überarbeitet und eliminiert werden.
Als Hilfestellung für die zukünftigen Entwickler:innen wurden ausserdem ein Entwicklungsleitfaden für das Set-up und den Betrieb des Systemes erstellt.

Ausblick

Neben den Mindestanforderungen wurden in der Aufgabenstellung auch optionale Anforderungen definiert. Diese konnten während der Bachelorarbeit nicht oder nur teilweise

umgesetzt werden. Einige der Ideen sind jedoch bereits in den Use Cases dokumentiert worden und in einigen Fällen wurden auch schon User Storys dazu erstellt. Diese können als Grundlage für die Weiterentwicklung in Folgearbeiten eingesetzt werden.

Die Architektur lässt sich ausserdem weiter optimieren, zu einem späteren Zeitpunkt würde sich auch eine Aufteilung des Systemes in Microservices anbieten. Dadurch könnte das Deployment automatisiert werden, wodurch ein effizienter Entwicklungsprozess ermöglicht würde: Neue Funktionen werden automatisch per Continuous Integration (CI) getestet und mittels Continuous Delivery (CD) deployed.

Inhaltsverzeichnis

Teil I

Technischer Bericht

Kapitel 1

Einleitung und Übersicht

1.1 Aufbau der Arbeit

Die vorliegende Bachelorarbeit ist in drei Teile aufgegliedert: die Einleitung, den Hauptteil und den Schlussteil.
In der Einleitung wird die Problemstellung, Vision und das Ziel der Arbeit erläutert und das Vorgehen beschrieben. Der Hauptteil umfasst eine Untersuchung der Ausgangslage, die Anforderungsspezifikation sowie die Domänenanalyse und die Softwarearchitektur.
Im Schlussteil werden die Ergebnisse präsentiert und die Schlussfolgerungen festgehalten.

1.2 Vision

Die Vision des Industriepartners ist, dass sich Business-Kunden in der Meeting-Quality-App registrieren und diese im Geschäftsalltag einsetzen können. Die Bedienung soll einfach und intuitiv sein und ein minimaler Aufwand für die Anwendenden bestehen. Die App soll zudem in die gängigste Tools integriert werden.

Die Feedbacks zu den Besprechungen sollen dazu verwendet werden, die Qualität von Meetings kontinuierlich zu verbessern und so produktiver zu gestalten. Auch für Schulungszwecke könnte die Anwendung künftig in Einsatz kommen.
Das erfasste Feedback soll aber in keinem Fall zur Mitarbeiterbewertung o. ä. dienen, sonder ist ausschliesslich für den/die Organisator:in bestimmt.

1.3 Problemstellung

≪Pandemiebedingt wurden Online Meetings zum Normalfall. Dadurch entfallen die gemeinsamen Kaffeepausen, in denen man die Meeting-Themen oft noch informell besprach und Feedback austauschte. Die Effizienz der Meetings kann unter den neuen Bedingungen nur noch erahnt werden.
Es wurde bereit ein erstes MVP entwickelt um flankierend zu einem Online-Meeting Feedbacks einzuholen. Dieses wurde mit JavaScript, React.js und Node.js entwickelt und läuft bei Digital Ocean. Es kann unter folgendem Link getestet werden: `https://mq.adaptit.ch/login`.≫[22]

1.4 Ziele

Eine allgemeine Zielsetzung ist in der Aufgabenstellung formuliert: «Ziel der Arbeit ist ein Tool zur Auswertung von Meetings weiterzuentwickeln und dem Meeting-Verantwortlichen ein Feedback und Verbesserungsvorschläge anzubieten. Das Tool soll so in das Online-Meeting eingebunden werden, dass die Teilnehmenden die Auswertungen mit möglichst wenig Aufwand starten und abgeben können.»[22]

Ziele aufseiten des Industriepartners sind zudem, dass der Ist-Zustand des bestehenden Minimum Viable Product (MVP) analysiert wird und eine Evaluation für die Anbindung in Microsoft Teams gemacht und umgesetzt wird.

In Absprache mit dem Betreuer und Industriepartner wurden folgende Mindestanforderungen festgelegt, welche sich ebenfalls in der Aufgabenstellung in Anhang A wiederfinden lassen.

1.4.1 Funktionale Anforderungen

- Integration in MS Teams und/oder Zoom
- Dashboard für Meeting-Verantwortlichen
- Fragebogen zur Bewertung von Meetings sollen im Anschluss an das Meeting automatisch an die Teilnehmenden versendet werden.

1.4.2 Nicht-Funktionale Anforderungen

Functionality

- Für Entwicklung und Betrieb wird die cloud-basierte Umgebung genutzt
- Die Funktionalität der Applikation ist mittels Tests zu beweisen

Usability

- Die Web-Applikation soll auf Desktop- und Mobilgeräten verwendbar sein
- Die Applikation soll übersetzbar sein

1.4.3 Software-Engineering

Da es sich bei der vorliegenden Bachelorarbeit um ein Software-Engineering-Projekt handelt, darf auch dieser Aspekt nicht zu kurz kommen. Eine detailierte Analyse des Ausgangszustands zu Projektbeginn ist unabdingbar, um so im Projektverlauf fundierte Entscheidungen hinsichtlich des Software-Designs treffen zu können, sodass am Ende der Arbeit weniger Technische Schuld besteht als zu Beginn.

1.5 Vorgehen

Eine detaillierte Beschreibung des Vorgehens ist im Projektplan[4] zu finden, weshalb in diesem Abschnitt nur die wichtigsten Eckdaten erläutert werden.

Das Vorgehen im Projekt ist ablauforientiert und wird durch Meilensteine in zeitliche Abschnitte unterteilt. Als Vorgehensmodell kommt eine Mischung aus Scrum und dem Unified Process (UP) zum Einsatz. Aus dem UP werden übernommen die vier Phasen (*Inception*, *Elaboration*, *Construction* und *Transition*). Innerhalb der Phasen wird in wochenbasierten Iterationen gearbeitet.

In den ersten zwei Phasen wird der Projektplan erstellt, die Businessfälle, Use Cases und User Storys erarbeitet sowie der Ist-Zustand der Applikation analysiert. In einer Variantenstudie sollen ausserdem die verschiedenen Integrationsmöglichkeiten für MS Teams beleuchtet werden. In der Konstruktionsphase werden schliesslich die so definierten Anforderungen umgesetzt und die Architektur der Meeting-Quality-App durch Refactorings optimiert werden, um so die künftig benötigte Zeit und die damit verbundenen Kosten für die Weiterentwicklungen der Anwendung zu verringern. In der finalen Transitionsphase wird die Anwendung in die Cloud deployed und an den Industriepartner übergeben.

1.6 Abgrenzungen

Da die Arbeit in einem eng gesteckten Zeitrahmen durchgeführt wird, wird es nicht möglich sein, alle Anforderungen (Mindestanforderungen und optionale Anforderungen) zu erfüllen und gleichzeitig die bestehende Codebasis komplett zu refactoren. Es muss daher eine gute Balance zwischen den umzusetzenden Funktionen und Verbesserung des Software-Designs gefunden werden, wobei auf beiden Seiten gewisse Abstriche notwendig sind.

Auch steht eine hohe Benutzbarkeit leider oft im Widerspruch zu einer hoher Sicherheit, sodass auch hier Einschränkungen gemacht werden müssen.

Es liegt ausserdem nicht im Rahmen dieses Projektes, ein Penetration-Testing durchzuführen oder ein automatisiertes Deployment zu erstellen.

Kapitel 2

Ausgangslage

Da mit dem MVP bereits eine Applikation existiert, welche weiterentwickelt werden soll, startet das Projekt somit auf einem Brownfield[1]. In diesem Kapitel werden die technischen Gegebenheiten des MVP untersucht und der Ausgangszustand der Softwarearchitektur und der Codebasis festgehalten; der bereits bestehende Funktionsumfang ist in Abschnitt 2.1 Bestehender Funktionsumfang dokumentiert.

2.1 Bestehender Funktionsumfang

Die zu Beginn der Bachelorarbeit bestehende Meeting-Quality-Applikation umfasst bereits folgende Funktionen.

2.1.1 Benutzerverwaltung

Benutzende können sich in der App registrieren und danach mit E-Mail-Adresse und Passwort einloggen. Haben sie das Passwort vergessen, können sie dieses auf der Login-Seite zurücksetzen.

[1] https://synoptek.com/insights/it-blogs/greenfield-vs-brownfield-software-development/

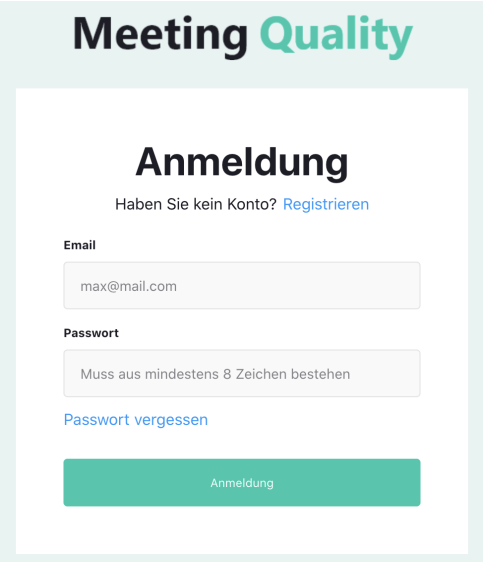


Abbildung 2.1: Login-Dialog der Meeting-Quality-App (Quelle: eigene Abbildung)

2.1.2 Meetings

In einer Listenansicht (Abbildung 2.2) werden alle Meetings aufgelistet, welche der/die Benutzende bereits erstellt hat. Auf dieser Seite können auch neue Meetings erstellt werden. Dazu wird der Name, das Datum und die Anzahl der Teilnehmenden der Besprechung erfasst.

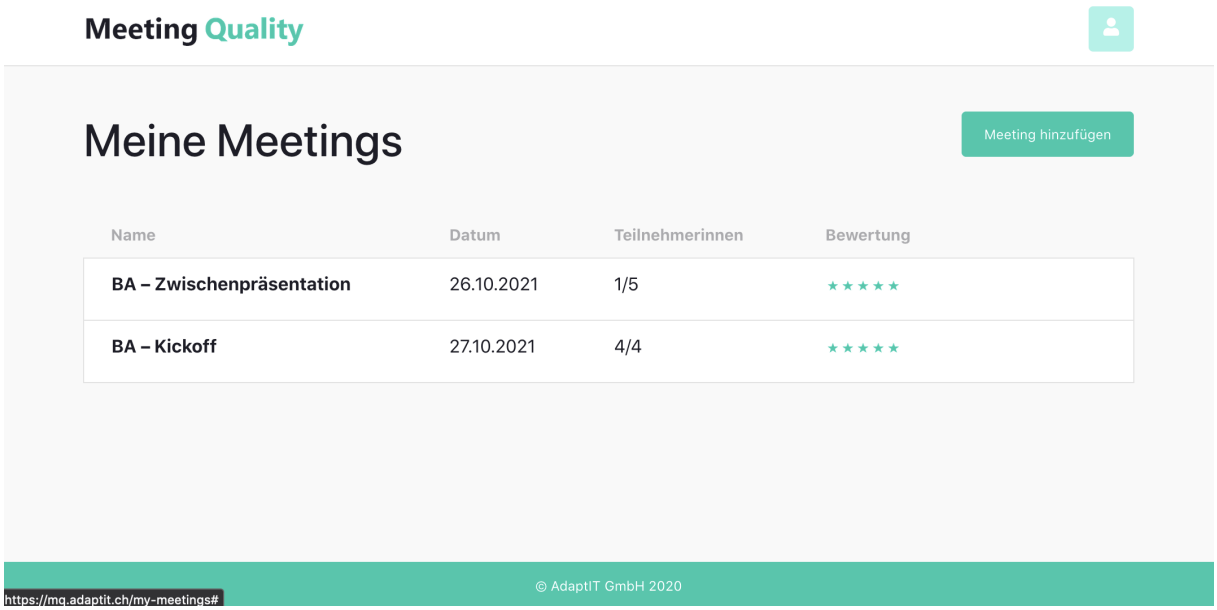


Abbildung 2.2: Listenansicht der Meetings (Quelle: eigene Abbildung)

Durch den Klick auf ein Meeting wird die Detailansicht (Abbildung 2.3) geöffnet. In der Detailansicht ist das Feedback zu sehen, welches für das ausgewählte Meeting bereits eingegangen ist. Dazu zählt je ein Diagramm zu Effizienz und Dauer der Besprechung sowie die Verbesserungsvorschläge. Hier kann auch der Link zum Feedbackformular kopiert

werden. Die Besprechungsinformationen (Name, Datum, Anzahl Teilnehmenden) können ebenfalls über diese Ansicht bearbeitet werden kann.

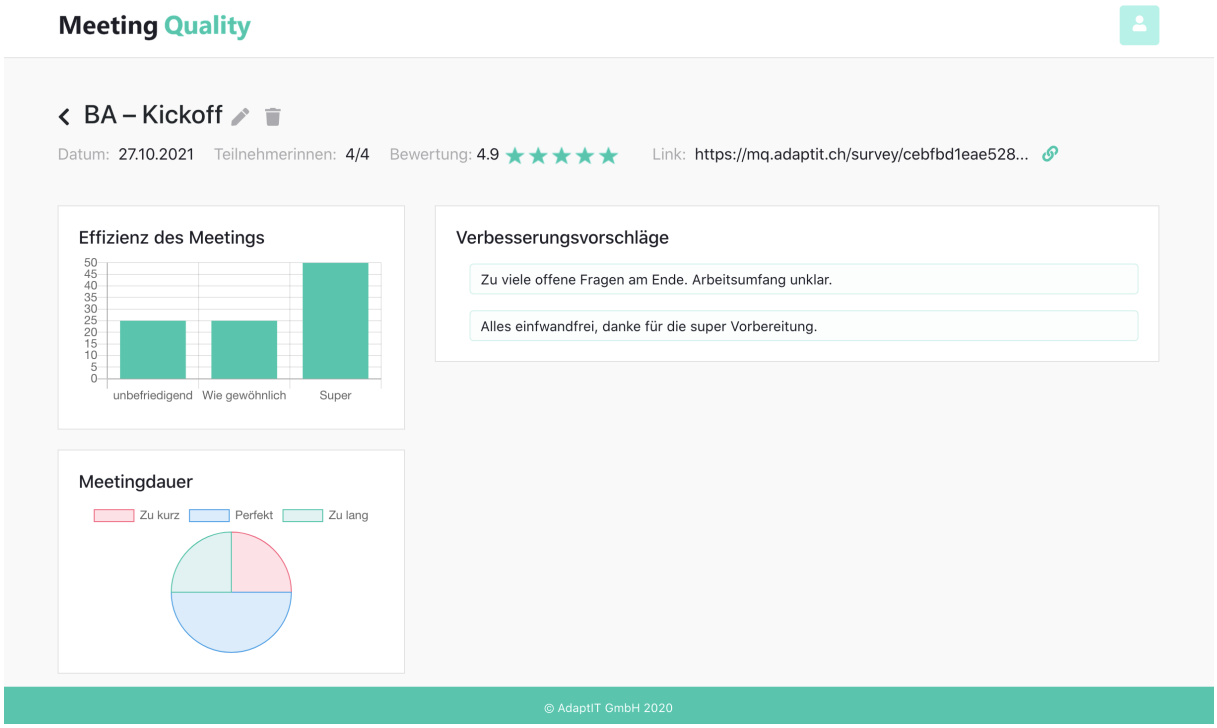


Abbildung 2.3: Detailansicht einer Besprechung (Quelle: eigene Abbildung)

2.1.3 Feedbackformular

Teilnehmende von einem Meeting gelangen über einen Link auf das Feedbackformular, mit welchem sie die Besprechung bewerten können (Abbildung 2.4). Dazu geben sie an, wie effizient das Meeting und die Dauer des Treffens gewählt war. In einem Freitextfeld können sie ausserdem beliebige weitere Verbesserungsvorschläge eingeben.

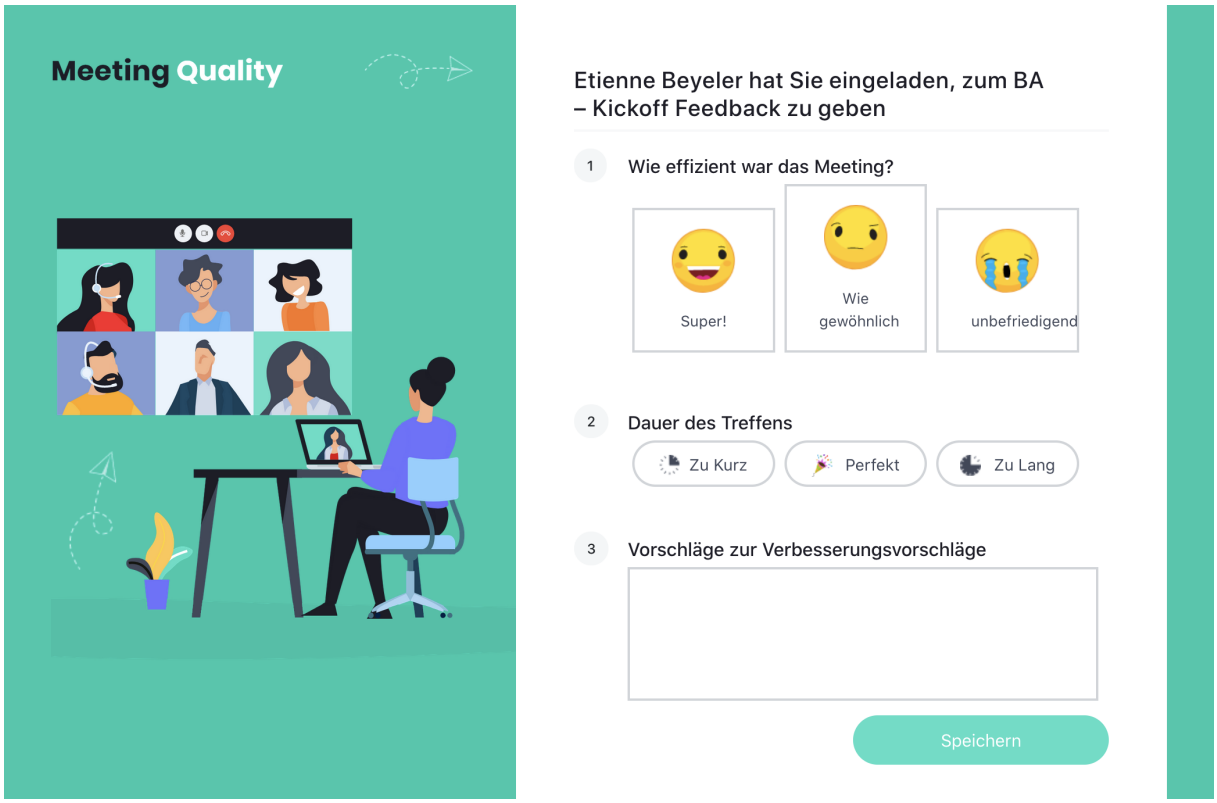


Abbildung 2.4: Feedbackformular der Meeting-Quality-App (Quelle: eigene Abbildung)

Die Umfrage wird anonym ausgefüllt, es ist keine vorgängige Registrierung notwendig; der Link zum entsprechenden Feedbackformular genügt.

2.2 Systemübersicht

Die Meeting-Quality-Applikation beruht auf einer klassischen 3-Tier-Architektur mit Client-Server-Cuts, welche in der Cloud-Umgebung von *DigitalOcean* gehostet werden. Konkret Unterteilt sich die Applikation in ein Frontend, ein Backend und eine Datenbank. Jede Komponente läuft dabei auf einem eigenen Droplet[2] mit Ubuntu 20.04 LTS als Betriebssystem, somit kann von einem verteilten Softwaresystem gesprochen werden. Abbildung 2.5 bildet diesen Sachverhalt in einem Deploymentdiagramm ab.

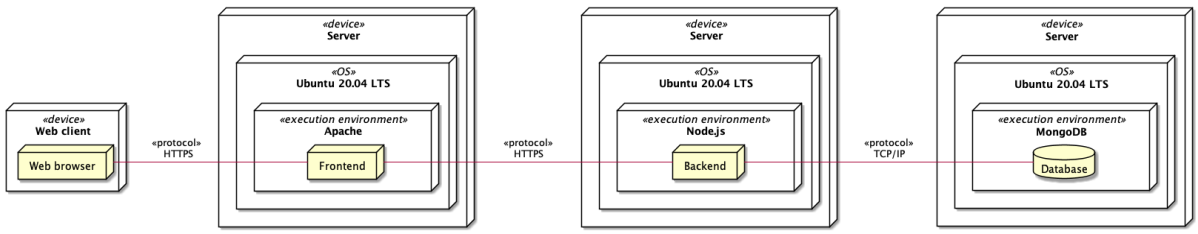


Abbildung 2.5: Deploymentdiagramm der Meeting-Quality-Applikation (Quelle: eigene Abbildung)

[2]So bezeichnet DigitalOcean ihre virtuellen Maschinen

Webbrowser Benutzer:innen greifen via Webbrowser auf die Meeting-Quality-Applikation zu. Dazu rufen sie die zugehörige Domain in ihrem Browser auf, welche das Frontend via HTTPS vom Webserver lädt.

Frontend Das Frontend, welches als zustandslose Single-page Application (SPA)[3] vorliegt, läuft auf einem Apache-HTTP-Server und ist über das Internet mit dem Backend verbunden. Dazu verfügt der Webserver über ein gültiges Let's Encrypt Zertifikat[4] um eine sichere Transport Layer Security Verbindung zu gewähren.

Backend Das Backend läuft in einer Node.js-Laufzeitumgebung und stellt Clients eine webbasierte REST-API zur Verfügung. Es kommuniziert über TCP/IP mit der Datenbank. Es verfügt ebenfalls über ein gültiges Let's Encrypt Zertifikat[5] um eine sichere Transport Layer Security Verbindung zu ermöglichen.

Datenbank Für die Persistierung der Daten wird das Datenbankmanagementsystem MongoDB eingesetzt. Die Datenbank ist dabei im Internet frei zu erreichen und deren Sicherheit wird von DigitalOcean gehandhabt.

2.3 Technologien

2.3.1 MERN-Stack

Die vorliegende Applikation basiert auf dem MERN-Stack. MERN steht für die vier Schlüsseltechnologien, aus denen der Stack besteht: MongoDB, Express, React, Node.js.[20]

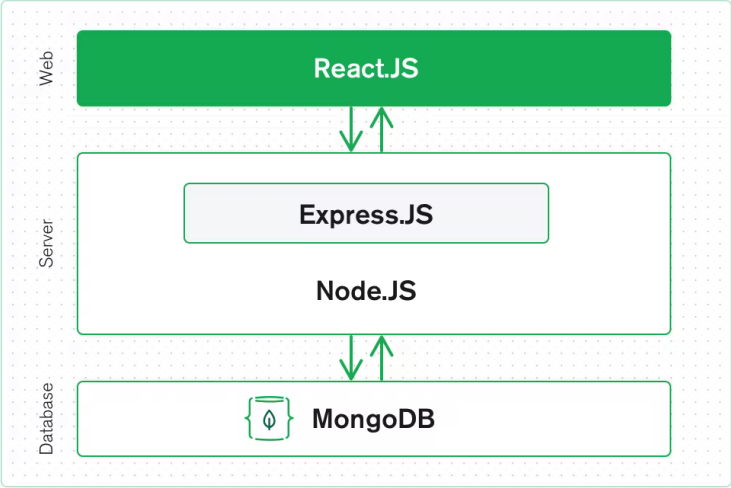


Abbildung 2.6: MERN-Stack (Quelle: MongoDB)

[3] Der Client selbst verfügt gleichwohl über Zustandsdaten. «Zustandslos» bezieht sich vielmehr darauf, dass diese Daten nicht im Client, sondern auf dem Server gespeichert sind und nur über diesen initiiert werden können.

[4] https://letsencrypt.org/de/

[5] https://letsencrypt.org/de/

React

Zur Erstellung der Benutzeroberflächen wird die deklarative, komponentenbasierte JavaScript-Bibliothek React benutzt. React, welches von Facebook entwickelt wurde, zeichnet sich durch einfache Handhabung, eine umfassende Dokumentation und eine grosse Community aus.[18]

Express und Node.js

Aufbauend auf Node.js, welches es erlaubt, serverseitiges JavaScript auszuführen, kommt das serverseitige Web-Framework Express zum Einsatz. Mit Express kann auf schnelle, unkomplizierte Weise eine funktionierende Webanwendung gebaut werden.

MongoDB

Als Datenbankmanagementsystem wird die schemafreie, dokumentorientierte NoSQL-Datenbank MongoDB verwendet. MongoDB erlaubt das direkte Abspeichern von JSON-Dokumenten, wobei die Struktur dieser Dokumente variieren kann – sie wird nicht wie bei SQL-Datenbanken erzwungen. Dies ist einer der Vorteile der Verwendung von NoSQL, da sie die Anwendungsentwicklung beschleunigt und die Komplexität von Bereitstellungen reduziert.[21]

Da in Softwaresystemen tendenziell eher SQL-Datenbanken eingesetzt werden und NoSQL-Datenbanken für lange Zeit eher tabu waren, werden die Prinzipien von MongoDB in den nachfolgenden Abschnitten etwas genauer ausgeführt.

Motivation Fraglich ist hierbei wieso man eine nicht-relationale Datenbank wählen würde. Historisch gesehen liegt das an den inzwischen stark gesunkenen Kosten von permanentem Speicher. Da heutzutage die grössten Kosten bei der Software Entwicklung die Entwickler an sich darstellen und nicht mehr die Infrastruktur selbst, hat der NoSQL-Datenbank-Ansatz viel Zulauf gewonnen.

Auch sind heutzutage die Daten die man empfangen kann weniger uniform und ändern sich schneller. Die NoSQL-Datenbanken bieten in dieser Hinsicht die grössere Flexibilität, da Felder die gespeichert werden sollen dynamisch hinzugefügt und entfernt werden können. Ebenfalls ist die Abfrage von NoSQL-Datenbanken in der Regel sehr performant und es werden verschiedenen Speicherungsformate unterstützt.

Speicherungsformat Die gängigsten Specherungsformate in nicht-relationalen Datenbanken sind Graphen-basiert, Dokumenten-basiert, basierend auf Key-Value-Stores oder auf Wide-Column-Stores.
Da MongoDB dokumentenbasiert ist, wird nachfolgend etwas genauer auf dieses Format eingegangen.

Die wichtigste Frage hierbei ist, wie die Dokumente aussehen, die abgespeichert werden. Ein Dokument stellt in diesem Kontext einen Eintrag eines Objekts mit den dazugehörigen Metadaten dar. Das Objekt wird im BSON-Format abgespeichert. Binary JSON (BSON) ist ein für die Speicherung optimiertes, auf JSON basierendes, Datenformat. Das heisst, das die Felder des Objekts als Field-Value-Paare in dem Eintrag aufgelistet werden. Weitere unterstützte Formate unter MongoDB wären JSON und XML.

Die so zustande kommenden Dokumente werden in sog. Collections unter der respektiven Datenbank gespeichert. Typischerweise sind die Dokumente in einer Collection gleichartig müssen es aber nicht sein. Es findet standardmässig also keine Schema-Validierung statt, kann aber definiert werden.

Es befinden sich also alle Informationen in dem Dokument bzw. in dem unter dem Dokument eingebetten Objekten. Das wiederum erspart die unter relationalen Daten benötigten Joins, sorgt allerdings für eine weniger effiziente Speicherung. Dafür sind die Abfragen der Datenbank schneller, da die einzelnen Tabellen nicht erst denormalisiert werden müssen.

2.3.2 Programmiersprachen

JavaScript Als Programmiersprache wird sowohl im Front- wie auch im Backend JavaScript verwendet.

2.3.3 Programmbibliotheken

Da sowohl das Frontend wie auch das Backend mit JavaScript programmiert sind, benutzen sie teilweise die selben Programmbibliotheken (en. *libraries*). Diese werden über den Node Package Manager (npm)[6] verwaltet.

Tabelle 3.1 gibt eine Übersicht über die wichtigsten Bibliotheken und zeigt auf, in welcher Komponente diese verwendet werden.

Bibliothek	Frontend	Backend
Axios	X	X
Bootstrap	X	
Formik	X	
Mongoose		X
Prettier	X	X
Redux	X	

Tabelle 2.1: Übersicht der verwendeten Programmbibliotheken

Axios

Ein populärer, Promise-basierter HTTP-Client für den Browser und Node.js.[3] Axios abstrahiert den plattformspezifischen Zugriff auf die HTTP-Schnittstelle, sodass im Frontend und Backend einheitlich darauf zugegriffen werden kann.

Bootstrap

Ein Open-Source-Toolkit mit vorgefertigten UI-Komponenten, welche sich auf flexible Weise zu einer responsiven Benutzeroberflächen zusammenbauen lassen.[5]

Formik

Eine quelloffene, leichtgewichtige Bibliothek für React, welche die Handhabung mit HTML-Formularen vereinfacht und standardisiert.[6]

[6] https://www.npmjs.com/

Mongoose

Eine Bibliothek zur Objektdatenmodellierung (ODM) für MongoDB und Node.js. Sie verwaltet Beziehungen zwischen Daten, bietet Schema-Validierungen und wird zur Übersetzung zwischen Objekten im Code und der Darstellung dieser Objekte in MongoDB verwendet.[21]

Mongoose ist das Pendant eines O/R-Mappers in einer relationalen Datenbank. Basierend auf Schemen lassen sich Modelle für die Datenstrukturen erstellen und validieren. So kann auch unter Verwendung einer NoSQL-Datenbank eine gewisse Homogenität und Sicherheit der Daten gewährleistet werden.

Prettier

Ein konfigurierbarer Code-Formatierer, der sich in die meisten Entwicklungsumgebungen integrieren lässt und die Formatierung von diversen Programmiersprachen unterstützt.[31]

Redux

Ein Zustandscontainer, der es ermöglicht, Anwendungen zu schreiben, die sich konsistent verhalten und einfach zu testen sind. Redux und React spielen ausgezeichnet zusammen und ermöglichen so eine zentrale, konsistente und vorhersagbare Verwaltung der Zustandsdaten.[1]

Redux basiert auf dem Prinzip eines unveränderlichen Zustandes (en. *Immutable State*). Die Applikation kann Daten nicht direkt, sondern nur über vordefinierte Aktionen (en. *Actions*) verändern, welche die Logik dazu beinhalten. Aktionen werden mittels *Dispatcher* verschickt. Die neuen Zustandsdaten laden dann im *Reducer*, welcher als einziger den Zustand verändern kann. React-Komponenten können über Selektoren auf die Zustandsdaten zugreifen. Abbildung 2.7 verdeutlicht diesen Ablauf.

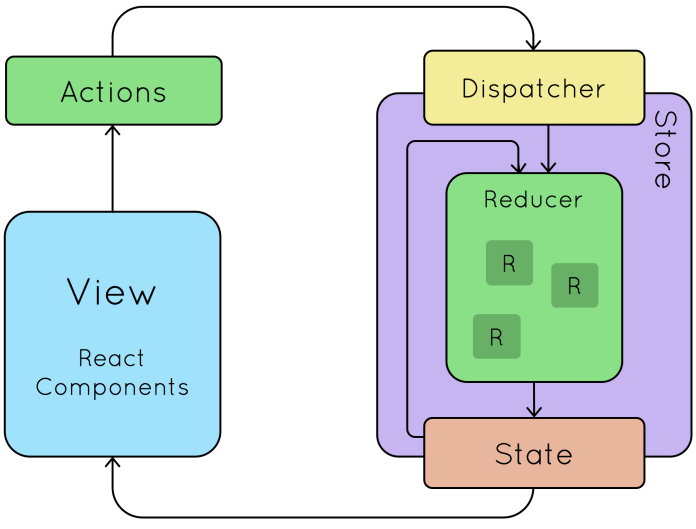


Abbildung 2.7: Redux (Quelle: Esri)

2.4 Designentscheidungen

2.4.1 Authentifizierung

Um die Meeting-Quality-App zu verwenden, ist ein Benutzerkonto notwendig. Benutzende registrieren sich dazu mit einer E-Mail-Adresse und einem Passwort in der Applikation, welche sie danach für den Login verwendet werden können.

Die technische Umsetzung des Authentifizierungsprozesses orientiert sich dabei am *OAuth 2.0 Flow*. Gemäss OAuth kommen für eine SPA zwei Abläufe in Frage: der *Authorization Code Flow with Proof Key for Code Exchange (PKCE)* und der *Implicit Flow with Form Post*.[16] In der bestehenden Anwendung wurde letzterer gewählt.

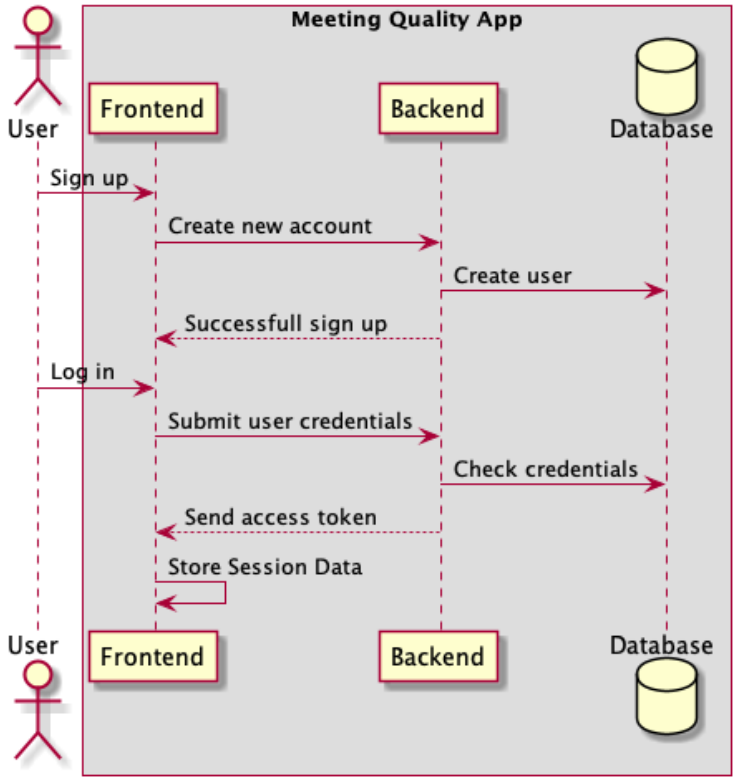


Abbildung 2.8: Benutzer-Authentifizierung in der Meeting-Quality-App

Bei der Registrierung sendet das Frontend die Informationen des/der Benutzer:in, *Credentials* genannt, an das Backend, welche dort verschlüsselt in der Datenbank gespeichert werden. Da die Verbindung zwischen Frontend und Backend über eine sichere HTTPS-Verbindung läuft muss das Passwort nicht clientseitig verschlüsselt werden.

Beim Login werden die Credentials wiederum vom Frontend an das Backend übermittelt und mit den in der Datenbank gespeicherten Credentials verglichen. Stimmen sie überein, sendet das Backend einen Zugriffstoken (en. *Access Token*) in Form eines JSON Web Token (JWT) an das Frontend zurück. Dieser Access Token wird im *LocalStorage* gespeichert und muss zukünftig in allen weiteren HTTP-Anfragen an das Backend mitgesendet werden, damit der/die Benutzer:in serverseitig identifiziert werden kann.

Abbildung 2.8 verdeutlicht diesen Ablauf in Form eines Sequenzdiagrammes.

2.4.2 Doppelte Objektvalidierung

Daten, welche von Benutzenden in der App eingegeben werden, werden sowohl im Frontend wie auch im Backend validiert. Die Überprüfung im Frontend dient in erster Linie dazu, die User Experience zu verbessern und schnelles Feedback an die Benutzer:innen zu geben, also z. B. ob ein eingegebenes Datum einem gültigen Format entspricht.

Da die serverseitige Web-API nicht nur über das Frontend, sonder auch mit anderen Clients angesprochen werden kann, ist eine erneute Validierung im Backend unerlässlich. Diese dient zwei Zwecken: der erneuten Überprüfung der Daten auf deren Korrektheit und der Unterbindung von clientseitigen Angriffen wie Code Injection, durch welche weitere Angriffe (Cross-Site-Scripting (XSS) etc.) ermöglicht werden.

2.4.3 Internationalisierung

Die Texte der Benutzeroberflächen sind in den Sprachen Deutsch und Englisch verfügbar. Basierend auf den Einstellungen des Webbrowsers, von welchem auf die App zugegriffen wird, werden die Texte in der entsprechenden Sprache angezeigt. Dieser Prozess nennt sich Internationalisierung, auch bekannt als *i18n* (kurz für en. *internationalization*).

2.4.4 Logging

Im Backend werden alle wichtigen Vorgänge in einer Log-Datei gespeichert. Dies umfasst den eingehende Datenverkehr, Verbindungsinformationen zur Datenbank und Fehlermeldungen. So kann im Nachhinein lückenlos nachvollzogen werden, auf welche Ressourcen zugegriffen wurden und wo Fehler aufgetreten sind.

2.4.5 Server Hardening

Für das in Express geschriebene Backend existiert eine Vielzahl an Hardening-Bibliotheken, welche das Ziel haben, die Sicherheit des Servers zu verbessern. Dazu gehören u. a. Bibliotheken für die Verteidigung gegen XSS-Angriffe, das gezielte Setzen von HTTP-Headern oder der angemessenen Konfiguration für Cross-Origin Resource Sharing (CORS).

2.5 Softwarearchitektur

2.5.1 Frontend

Durch das Naturell von React folgt das Frontend einer komponentenbasierten Architektur. Eine React-Komponente ist dabei ein unabhängiges und wiederverwendbares Stück Code, welches einen Teil der Benutzeroberfläche isoliert und mit der dazugehörenden Logik und den Zustandsdaten verknüpft. Komponenten lassen sich so zu komplexen Benutzeroberflächen zusammenfügen.

Übersicht der Pakete

Das Frontend lässt sich im Wesentlichen in vier Pakete unterteilen: *Pages*, *Components*, *Store* und *Utils*. Das Zusammenspiel der Pakete ist in Abbildung 2.9 ersichtlich sind.

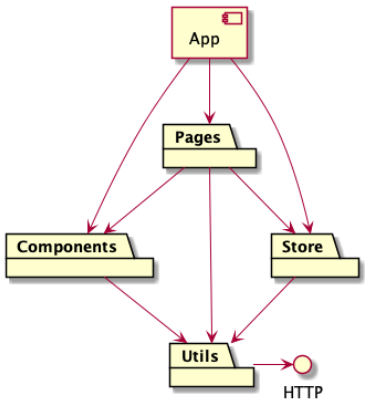


Abbildung 2.9: Paketdiagramm des Frontends (Quelle: eigene Abbildung)

Die *App*-Komponente bildet den Einstiegspunkt und fügt die restlichen Komponenten zu einer ganzheitlichen Benutzeroberfläche zusammen. Die Komponenten im Pages-Paket widerspiegeln die einzelnen Seiten der App, welche den Benutzenden im Browser dargestellt werden. Sie sind wiederum in feingranularere Komponenten unterteilt.

Um Codeduplikate zu vermeiden, werden mehrfach benutzte Komponenten als zustandslose Komponenten zur Wiederverwendung im Components-Paket abgelegt. Dasselbe gilt für mehrfach verwendete Logik, welche in Services gekapselt im Utils-Paket zu finden ist.

Das Store-Paket beinhaltet den Redux-basierten Zustandscontainer und die entsprechenden Aktionen und Reducer (vgl. Abschnitt 2.3.3 Redux).

Pages

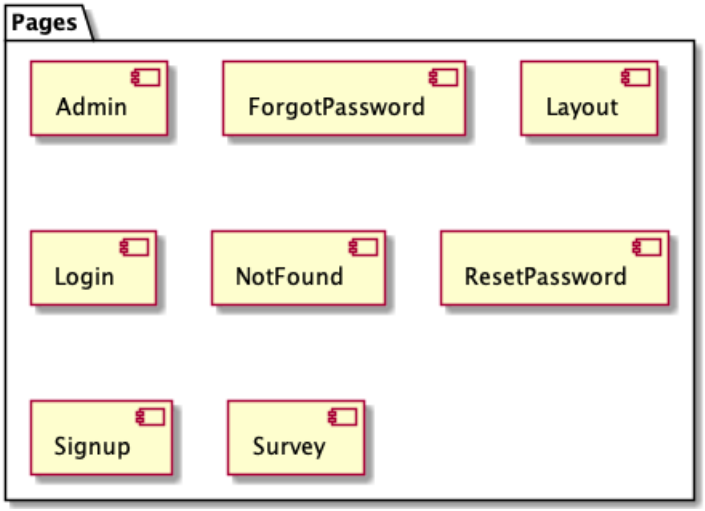


Abbildung 2.10: Komponenten im Pages-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
Admin	Beinhaltet die React-Komponenten für die Meeting-Übersicht, die Meeting-Details sowie die Dialoge zum Erstellen, Editieren und Löschen von Meetings.
ForgotPassword	Die Seite, auf welcher das Passwort zurückgesetzt werden kann. Benutzende geben ihre E-Mail-Adresse ein und erhalten einen Link um das Passwort zurückzusetzen.
Layout	Beinhaltet die React-Komponenten *Header* und *Footer*. Im Header befindet sich das Logo der App und das Benutzermenü, im Footer wird das Copyright dargestellt.
Login	Die Loginmaske, in welcher Benutzende ihre Login-Credentials eingeben.
NotFound	Die Seite die dargestellt wird, wenn eine andere Seite nicht gefunden wurde.
ResetPassword	Auf dieser Seite kann ein neues Passwort festgelegt werden.
Signup	Die Eingabemaske zum Erstellen von einem neuen Benutzerkonto.
Survey	Der Fragebogen, in welchem Besprechungsteilnehmer:innen das Feedback übermitteln können.

Tabelle 2.3: Beschreibung der Komponenten in Pages

Components

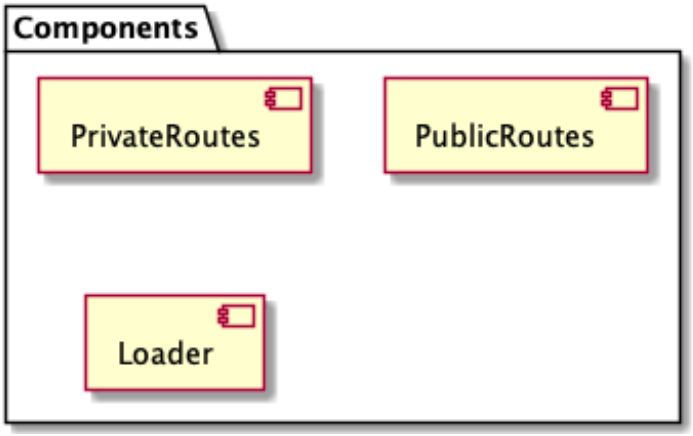


Abbildung 2.11: Komponenten im Components-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
PrivateRoutes	Routing-Komponente, welche die Seite, die ihr übergeben wird, nur darstellt wenn der/die Benutzer:in eingeloggt ist.
PublicRoutes	Routing-Komponente, welche die Seite, die ihr übergeben wird, nur darstellt wenn der/die Benutzer:in nicht eingeloggt ist. Wird für die Registrierungs- und Login-Seite verwendet.
Loader	Ein runder Ladebalken der sich im Kreis dreht.

Tabelle 2.5: Beschreibung der Komponenten in Components

Store

Jede Komponente im Store beinhaltet ihre entsprechenden Redux-Aktionen und -Reducer sowie den dazugehörigen Teil des Applikationszustandes.

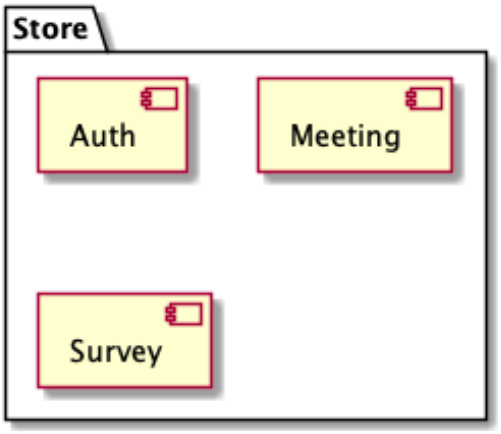


Abbildung 2.12: Komponenten im Store-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
Auth	Aktionen der Benutzerverwaltung: Registrierung, Login, Passwort zurücksetzen. Der Store beinhaltet die Benutzerdaten.
Meeting	Aktionen zum Laden, Erstellen, Bearbeiten und Löschen von Meetings. Der Store beinhaltet Daten zu allen vom Backend geladenen Meetings.
Survey	Aktion zum Übermitteln von Feedback. Der Store beinhaltet Daten zur aktuellen Umfrage.

Tabelle 2.7: Beschreibung der Komponenten in Store

Utils

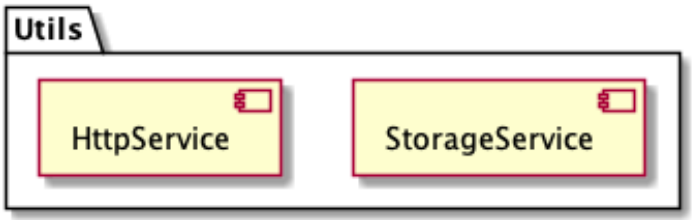


Abbildung 2.13: Komponenten im Utils-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
HttpService	Service für die Kommunkation über HTTP. Konfiguriert Axios und abstrahiert den Zugriff darauf.
StorageService	Abstrahiert den Zugriff auf den *LocalStorage*, den lokalen Speicher des Browsers, in welchem unter anderem die Sitzungsdaten der Benutzenden gespeichert sind.

Tabelle 2.9: Beschreibung der Komponenten in Utils

2.5.2 Backend

Ein oft verwendetes Pattern zur Strukturierung des Codes in Express-Applikationen ist die Unterteilung der Funktionen in *Controller*. Ein Controller nimmt HTTP-Anfragen entgegen, bearbeitet sie und sendet entsprechende HTTP-Antworten zurück. Die Zuordnung der Controller zu den API-Endpunkten geschieht mittels *Routes*.

In der Meeting-Quality-Applikation wird dieses Pattern um *Services* und *Models* erweitert. Die Businesslogik wird von den Controllern in die Services ausgelagert, die Models definieren das Datenmodell und abstrahieren den Zugriff auf die Datenbank.

Das Zusammenspiel der Komponenten wird in Abbildung 2.14 grafisch verdeutlicht.

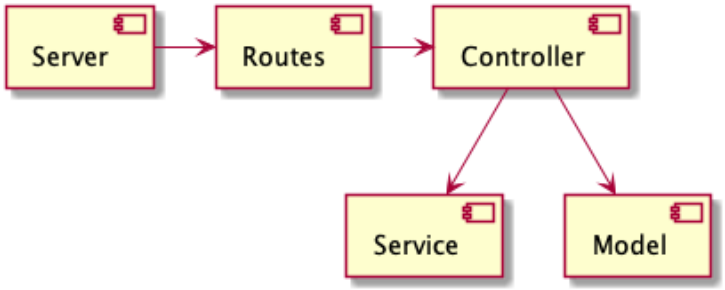


Abbildung 2.14: Zusammenspiel der Komponenten im Backend (Quelle: eigene Abbildung)

Übersicht der Pakete

Das Backend besteht aus den Paketen *API*, *Middleware*, *Utils* und *Config*. Deren Zusammenspiel ist in Abbildung 2.15 zu sehen.

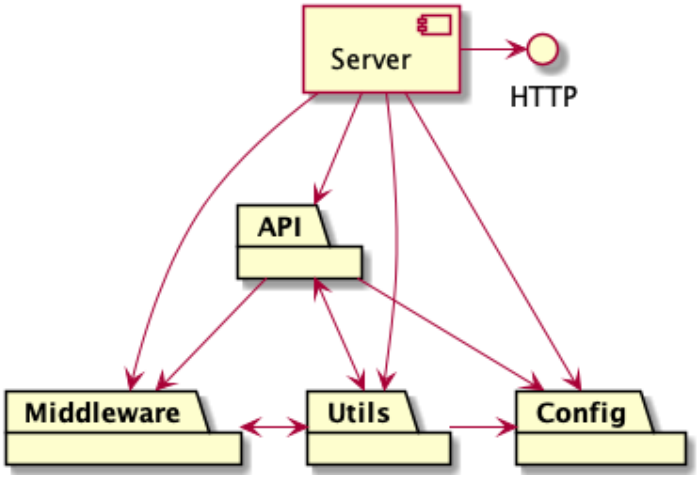


Abbildung 2.15: Paketdiagramm des Backends (Quelle: eigene Abbildung)

Als Einstiegspunkt fungiert die *Server*-Komponente, welche alle Komponenten zusammenfügt und weitere Middleware – z. B. für das Server Hardening (vgl. Unterabschnitt 2.4.5) – einbindet.

API

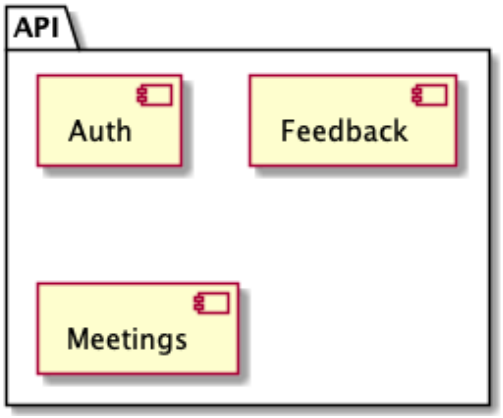


Abbildung 2.16: Komponenten im API-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
Auth	Komponente für die Benutzerverwaltung. Beinhaltet das Erstellen von neuen Accounts, den Login, Passwort zurückzusetzen und liefert Daten zu dem/der eingeloggten Benutzer:in.
Feedback	Nimmt die Daten vom ausgefüllten Fragebogen entgegen und speichert sie in der Datenbank.
Meetings	Komponente für die Verwaltung der Besprechungen. Es können Daten zu bestehenden Meetings abgerufen werden, neue Meetings erstellt und bestehende bearbeitet werden.

Tabelle 2.11: Beschreibung der Komponenten in API

Middleware

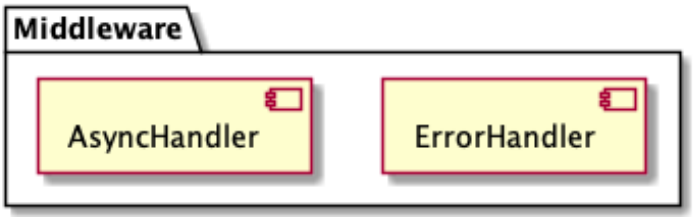


Abbildung 2.17: Komponenten im Middleware-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
AsyncHandler	Wrapper zur Verwendung in asynchronen Controllern. Fängt Fehler, damit diese nicht zum Absturz des Programmes führen.
ErrorHandler	Nimmt allgemeine Programmfehler entgegen, unterscheidet zwischen ihnen und behandelt sie entsprechend. Sendet per HTTP-Antwort Fehlermeldung zurück, wobei applikationsspezifische Informationen versteckt werden, um so keine Implementationsdetails zu leaken.

Tabelle 2.13: Beschreibung der Komponenten in Middleware

Utils

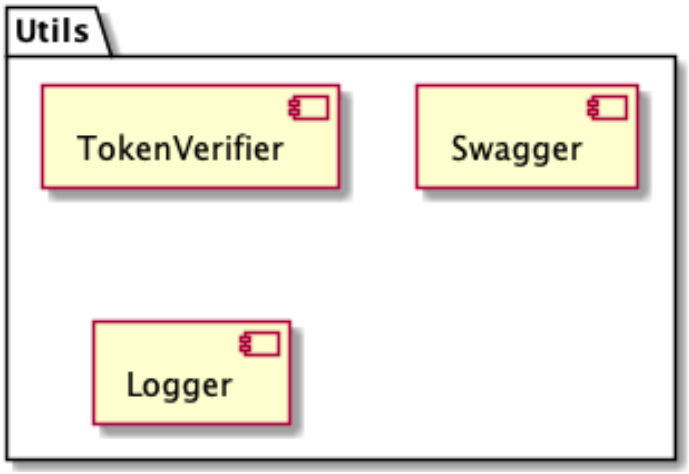


Abbildung 2.18: Komponenten im Utils-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
TokenVerifier	Verifiziert die Access Token, um gewährt/blockiert den Zugriff auf die entsprechenden Ressourcen.
Swagger	Dokumentiert die API und ermöglicht es die API-Ressourcen zu visualisieren und mit per GUI zu interagieren.
Logger	Konfiguriert die Logging-Bibliothek.

Tabelle 2.15: Beschreibung der Komponenten in Utils

Config

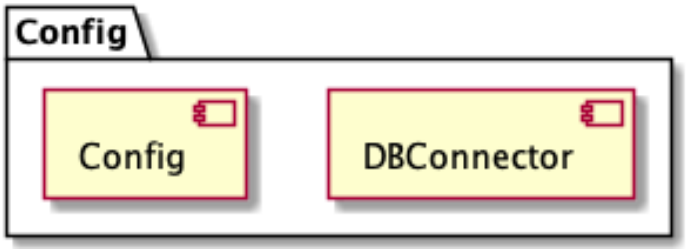


Abbildung 2.19: Komponenten im Config-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
Config	Liest die Werte der Konfigurationsdateien (`.env`-Dateien) und Umgebungsvariablen aus und bietet einen einheitlichen Zugriff darauf.
DBConnector	Stellt die Verbindung zur Datenbank her.

Tabelle 2.17: Beschreibung der Komponenten in Config

2.5.3 Datenmodell

Das Datenmodell bildet das Schema der Daten, welches für die Speicherung in der Datenbank verwendet wird. Dieses ist relativ einfach gehalten und wird in Abbildung 2.20 veranschaulicht.

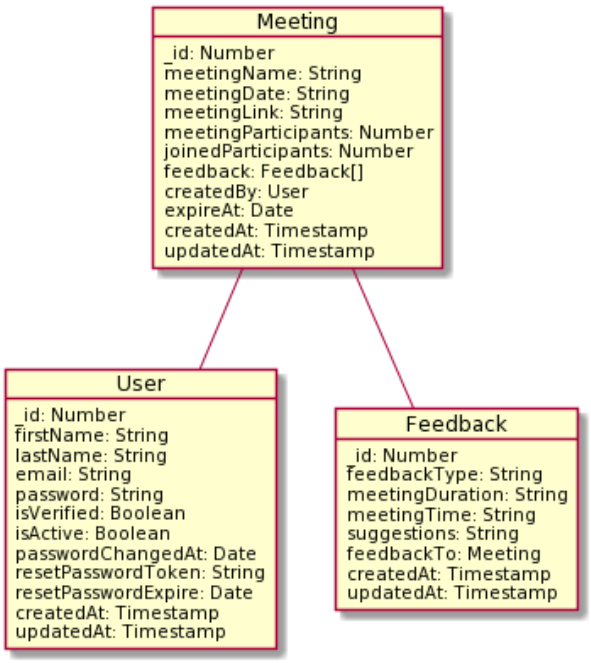


Abbildung 2.20: Datenmodell des MVP (Quelle: eigene Abbildung)

2.6 Testing

Das grösste Manko der bestehenden Applikation ist der vollständige Verzicht auf Testing. Es sind weder Unit- noch Integrationstests implementiert. Besonders der Verzicht von Testen der Business-Logik ist schmerzlich, da hier die Funktionalität zu einem relativ hohen Grad nachgewiesen werden sollte.

2.7 Beurteilung des Software-Engineerings

Das Software-Engineering der Meeting-Quality-App folgt den für die Technologien gängigen Praktiken und scheint bei Betrachtung auf oberster Ebene in Ordnung zu sein. So verfügt das Frontend etwa über eine Internationalisierung und verwendet Redux für die Zustandsverwaltung und im Backend kommen etliche Sicherheits-Bibliotheken zum Einsatz. Beim Betrachten auf einer tieferen Ebene fallen jedoch hinsichtlich Softwarearchitektur und Quellcode verschiedenste Mängel ins Auge.

Im Frontend betrifft dies vor allem die React-Komponenten: React ist eigentlich so konzipiert, um viele kleine, wiederverwendbare Komponenten zu erstellen, welche dann zu einem grossen Ganzen zusammengefügt werden. Stattdessen sind im Frontend aber vor allem wenige, dafür extrem umfassende Komponenten zu finden. So weist die `Survey`-Komponente bspw. eine Länge von 634 Codezeilen auf und fällt somit in den als *Bloater*

bekannten Code Smells *Long Method*.
Für die Definition der Redux-Aktionen (vgl. Abschnitt 2.5.1 Store) fällt zudem enorm viel Boilerplate-Code an. Redux bietet dafür geeignetere Patterns an, welche in Kapitel 5 Softwarearchitektur und -design noch genauer erläutert werden.

Im Backend sind vor allem die Komponenten stark aneinander gekoppelt und die Daten nur sehr schlecht gekapselt. Dieser Sachverhalt ist auch bereits an den bidirektionalen Assoziationen in Abbildung 2.15 zu erkennen; eine Unterteilung in Layer exisitiert de facto nicht. Beim Betrachten des Quellcodes werden weitere Makel sichtbar: Die Controller und Services im API-Paket (vgl. Abschnitt 2.5.2 API) weisen eine sehr geringe Kohäsion auf und enthalten diverse Code Smells der Kategorien *Bloaters* (*Long Method*, *Data Clumps*, *Primitive Obsession*) und *Dispensables* (*Comments*, *Duplicate Code*).[32]

Ausserdem befinden sich Teile der Logik, wie z. B. die Generierung der Meeting-Links, im Frontend, obwohl diese in die Business-Logik im Backend gehörten.

Kapitel 3

Anforderungsanalyse

Dieses Kapitel befasst sich mit der Anforderungsanalyse und -spezifikation. Das Vorgehen dazu ist im Projektplan[4] beschrieben.

3.1 Stakeholder

Stakeholder sind Personen, deren Interesse für ein Projekt von Belangen sind und die daher befriedigt werden sollten.[12] Dieser Abschnitt befasst sich hierbei mit den Stakeholdern der Meeting-Quality-Applikation selbst, im Gegensatz zu den Projekt-Stakeholdern, welche separat im Projektplan behandelt werden.[4] Die verschiedene Stakeholder werden nachfolgend erörtert und ihre Interessen und Absichten an der Anwendung beschrieben.

Organisierende

Der/die Organisator:in erstellt und plant Meetings und möchte deren reibungsloser Ablauf sicherstellen. Dazu sollen einerseits die richtigen Teilnehmenden ausgewählt und über den Ort und Zeitpunkt der Besprechung verständigt werden, und andererseits die Traktanden des Meetings im voraus veröffentlichen werden, damit sich die Teilnehmenden darauf vorbereiten können.

Nach einer Besprechung will der/die Organisator:in rasch Feedback zu der Veranstaltung bekommen, um so beispielsweise zu sehen, ob das Meeting insgesamt als zu lange oder zu kurz empfunden wurde und ob die Wahl der Traktanden sinnvoll war.

Teilnehmende

Teilnehmende ihrerseits möchten, dass Meetings effizient und unkompliziert vonstatten gehen. Sie wollen im Voraus über Zeit, Ort, Dauer und Inhalt einer Besprechung informiert werden, um den Besuch zu planen und Überschneidungen in ihrem Terminkalender zu vermeiden. Ausserdem wollen sie nur an für sie relevanten Besprechungen teilnehmen. Da oft mehrere Meetings nacheinander anstehen, wollen Teilnehmende dazwischen genügend Zeit für einen allfälligen Wechsel des Besprechungsraums und gegebenenfalls für eine kurze Pause (um Auszutreten, das Getränk aufzufüllen etc.) haben.

Administrierende

Administrator:innen verwaltet das System einer Organisation und haben den Anspruch, dass dessen Komplexität möglichst gering ist. Ihnen soll die Möglichkeit gegeben werden, das Unternehmen, das sie verwalten, möglichst simpel in die Meeting-Quality-App zu überführen.

Unternehmen

Aus Sicht einer Unternehmung sollen Besprechungen effizient vonstatten gehen und auf ein Minimum reduziert werden. Denn während in Meetings zwar essenzielle Angelegenheiten besprochen werden und Wissen ausgetauscht wird, steht währenddessen die Produktion still.

In der heutigen Industrie sind zudem Daten eines der wertvollsten Güter für ein Unternehmen. Kennzahlen, mit welchen sich die Qualität von Meetings messen, bewerten und analysieren lassen, können einen wichtige Grundlage für eine Steigerung der Arbeitsproduktivität darbieten.

Unternehmen profitieren daher von wirksamen Geschäftsprozessen, welche Fragen wie «Hat der Zeitpunkt einer Besprechung Auswirkungen auf dessen Qualität?» und «War der Inhalt einer Besprechung relevant?» quantitativ beantworten lassen. So kann ein Wandel von den heute gebräuchlichen Faustregeln (bspw. die «Zwei-Pizza-Regel» von Jeff Bezos[17]) zu empirischen, konkreten Datensätzen stattfinden.

3.2 Anwendungsfälle

Anwendungsfälle (en. *Use Cases*) sind ein einfaches Modell zur Dokumentation der Funktionalität eines (Software-)Systems. Ein Anwendungsfall beschreibt das Verhalten des Systems aus Sicht der Anwendenden (*Akteure* genannt) und offenbart ihre Ziele und Absichten.

In Abbildung 3.1 sind die Akteure, die Anwendungsfälle und das Zielsystem – die Meeting-Quality-App – in Form eines Use-Case-Diagramms ersichtlich. Die gelben Use Cases markieren dabei die Mindestanforderungen aus der Aufgabenstellung, während die grauen Anwendungsfälle die optionalen Anforderungen kennzeichnen.

3.2.1 Akteure

Wie dem Use-Case-Diagramm (Abbildung 3.1) entnommen werden kann, gibt es insgesamt drei Akteure, welche durch die beschriebenen Anwendungsfälle mit der Meeting-Quality-App interagieren: Organisator:innen, Teilnehmer:innen und Administrator:innen. Da Akteure automatisch auch Stakeholder sind (sie haben ein zu befriedigendes Interesse) und bereits im Abschnitt 3.1 Stakeholder vorgestellt wurden, werden sie hier nicht erneut behandelt.

3.2.2 Beschreibung der Anwendungsfälle

Das Format und die Terminologie, welche für die nachfolgende Beschreibung der Anwendungsfälle verwendet wird, basiert auf den von Craig Larman vorgestellten Techniken

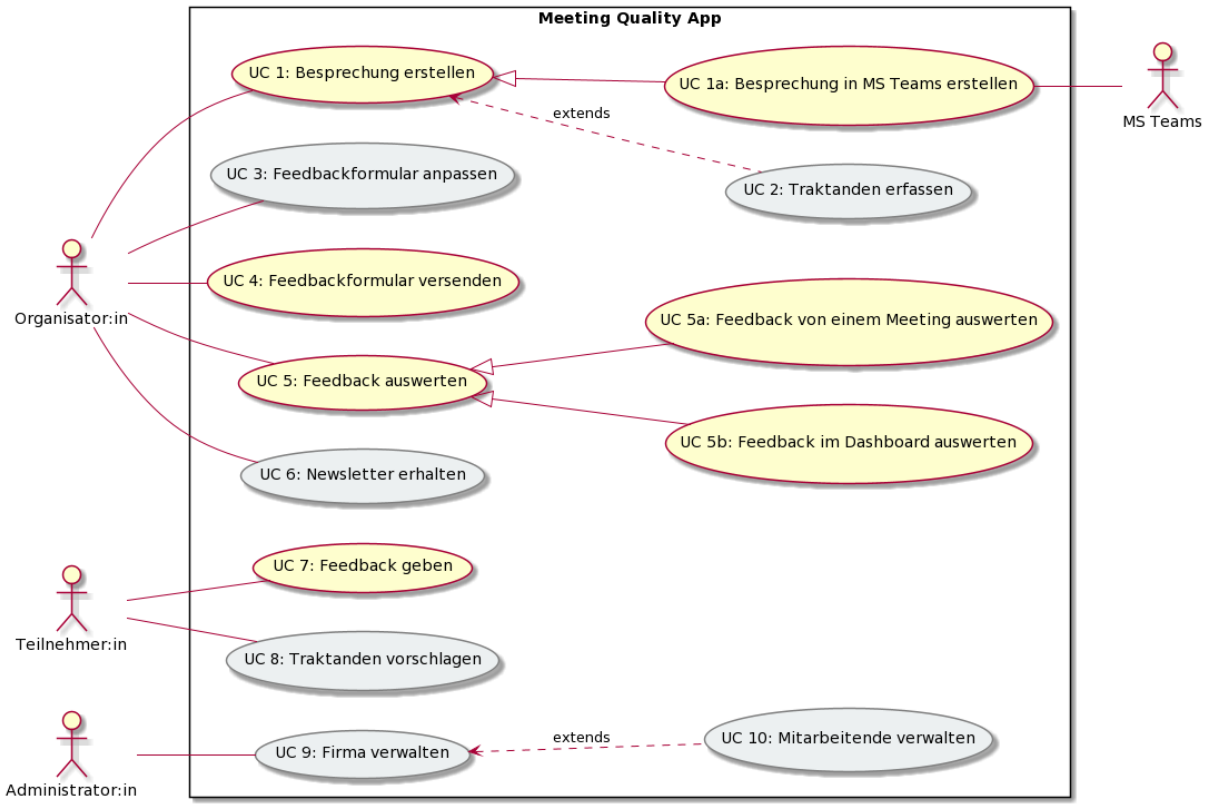


Abbildung 3.1: Use-Case-Diagramm

im Buch «Applying UML and Patterns»[23], welche wiederum auf dem Standardwerk «Writing Effective Use Cases»[7] von Alistair Cockburn basieren.

Die Anwendungsfälle, welche im Use-Case-Diagramm (Abbildung 3.1) ersichtlich sind, sollen an diesem Punkt genauer spezifiziert werden. Dabei bieten sich zwei Formate an: *brief* und *fully dressed*. Das Brief-Format liefert eine kurze Zusammenfassung eines Anwendungsfalls in einem Absatz, in der Regel über das wichtigste Erfolgsszenario. Im Fully-Dressed-Format werden hingegen alle Schritte und Varianten detailliert beschrieben, und es gibt unterstützende Abschnitte, wie Vorbedingungen und Erfolgsgarantien.

Anwendungsfälle, welche zu den Mindestanforderungen gehören (im Diagramm gelb), werden nachfolgend im Fully-Dressed-Format beschrieben; alle anderen im Brief-Format.

Die Anwendungsfälle werden ausserdem in zwei Level gegliedert: Benutzerziel (en. *User Goal*) und Unterfunktion (en. *Subfunction*). Ein Anwendungsfall auf Benutzerziel-Ebene ist die gängige Art, die Szenarien zur Erfüllung der Ziele eines Hauptakteurs zu beschrieben. Demgegenüber erläutert ein Anwendungsfall auf Unterfunktions-Ebene die Teilschritte, die zur Unterstützung eines Benutzerzieles erforderlich sind.

UC 1: Besprechung erstellen

UC 1: Besprechung erstellen

Der/die Organisator:in erstellt ein neues Meeting, zu welchem Feedback gegeben werden soll und erfasst dazu die notwendigen Daten (Besprechungsname, Datum, Anzahl der Teilnehmenden etc.).

Scope Meeting Quality App

Level Benutzerziel

Primärer Akteur Organisator:in

Vorbedingungen Organisator:in ist authentifiziert.

Basisablauf

1. Organisator:in öffnet die Meeting-Übersicht.
2. Organisator:in erstellt eine neues Meeting.
3. Organisator:in gibt die notwendigen Daten des Meetings in der Benutzeroberfläche ein.
4. Organisator:in kontrolliert die eingegebenen Daten und speichert diese.

Variationen

2a. Anstatt ein neues Meeting zu erstellen, wird ein bestehendes bearbeitet.
   1. Organisator:in wählt das zu bearbeitende Meeting aus.
   2. Organisator:in fährt mit Schritt 3 fort.

3a. Die Daten werden automatisch von einem externen System (Kalender, Skype, Zoom etc.) ausgelesen und ausgefüllt.

3b. Die eingegebenen Daten sind fehlerhaft. Schritt 4 kann nicht ausgeführt werden.
   1. System hebt das Feld, welches fehlerhafte Daten enthält, hervor.
   2. Organisator:in korrigiert die fehlerhaften Daten.
   3. Organisator:in fährt mit Schritt 3 fort.

*a. Organisator:in bricht den Vorgang ab. Es werden keine Daten gespeichert und kein neues Meeting im System angelegt.

Erfolgsgarantie Das Meeting ist im System mit den eingegeben Daten erstellt worden und in der Übersicht für den/die Organisator:in sichtbar.

UC 1a: Besprechung in MS Teams erstellen

UC 1a: Besprechung in MS Teams erstellen

Dieser Anwendungsfall ist eine Spezialisierung von UC 1: Besprechung erstellen.

Der/die Organisator:in erstellt analog zu UC 1 ein neues Meeting, jedoch direkt in MS Teams, und nicht in der Meeting-Quality-App.

Scope Microsoft-Teams-Applikation

Level Benutzerziel

Primärer Akteur Organisator:in

Vorbedingungen

1. Organisator:in ist in MS Teams authentifiziert.
2. Meeting-Quality-App ist mit MS Teams verbunden.

Basisablauf

1. Organisator:in erstellt eine neue Besprechung in MS Teams.
2. Organisator:in gibt die notwendigen Daten des Meetings in der Benutzeroberfläche ein.
3. Organisator:in kontrolliert die eingegebenen Daten und speichert diese.
4. Die Daten werden an die Meeting-Quality-App übermittelt.

Variationen

1a. Anstatt eine neue Besprechung zu erstellen, wird eine bestehende bearbeitet.
   1. Organisator:in wählt die zu bearbeitende Besprechung in MS Teams aus.
   2. Organisator:in fährt mit Schritt 2 fort.

*a. Organisator:in bricht den Vorgang ab. Es werden keine Daten an die Meeting-Quality-App übermittelt.

Erfolgsgarantie Das Meeting ist sowohl in MS Teams als auch in der Meeting-Quality-App erstellt worden und in der Übersicht für den/die Organisator:in sichtbar.

UC 2: Traktanden erfassen

UC 2: Traktanden erfassen

Der/die Organisator:in kann vor der Besprechung Traktanden erfassen, damit Teilnehmende diese einsehen und sich vorbereiten können. Zu jedem Traktandum kann ein geplanter Zeitaufwand angegeben werden. Basierend auf diesen Zeitangaben wird die Gesamtdauer des Meetings berechnet und dem/der Organisator:in vorgeschlagen.

UC 3: Feedbackformular anpassen

UC 3: Feedbackformular anpassen

Der/die Organisator:in kann das Formular, über welches Teilnehmende ihr Feedback abgegeben, individuell anpassen. Standardmässig besteht das Formular aus den zwei Fragen «Wie effizient war das Meeting?», «Wie war die Dauer der Besprechung gewählt?» und einem Freitextfeld für Verbesserungsvorschläge. Diese können mit weiteren Feldern (Single Choice, Multiple Choice, Freitext) ergänzt werden.

UC 4: Feedbackformular versenden

UC 4: Feedbackformular versenden

Der/die Organisator:in versendet den Link zum Feedbackformular an die Teilnehmenden, damit diese ihre Rückmeldungen abgeben können.

Scope Meeting Quality App

Level Benutzerziel

Primärer Akteur Organisator:in

Vorbedingungen

1. Organisator:in ist authentifiziert.
2. Das zu bewertende Meeting ist im System erstellt worden (UC 1: Besprechung erstellen).

Basisablauf

1. Organisator:in öffnet die Meeting-Übersicht.
2. Organisator:in wählt das zu bewertenden Meeting aus.
3. Organisator:in kopiert den generierten Link.
4. Organisator:in verschickt den Link über das bevorzugte Medium (E-Mail, Slack, MS Teams etc.) and die Teilnehmenden.

Variationen

3a. Die E-Mail-Adressen der Teilnehmenden wurden im Voraus erfasst.
   1. Organisator:in hat die Option, den Link automatisch an die hinterlegten E-Mail-Adressen zu versenden.

*a. Die Option zum automatischen versenden des Feedbackformulars wurde ausgewählt.
   1. System verschickt das Formular automatisch im Anschluss an das Meeting an die Teilnehmenden.

Erfolgsgarantie Teilnehmende haben einen funktionierenden Link zum Fragebogen erhalten und können diesen ausfüllen.

UC 5: Feedback auswerten

Bei diesem Anwendungsfall handelt es sich um einen abstrakten Use Case, mit welchem die Akteure nicht direkt interagieren; sie interagieren ausschliesslich mit seinen Spezialisierungen (UC 5a: Feedback von einem Meeting auswerten und UC 5b: Dashboard ansehen). Der Anwendungsfall dient somit der Verallgemeinerung und wird hier nicht weiter beschrieben.

UC 5a: Feedback von einem Meeting auswerten

UC 5a: Feedback von einem Meeting auswerten

Der/die Organisator:in sieht sich das erhaltene Feedback zu einer einzelnen Besprechung an. Das Feedback wird dabei vom System aufbereitet und in geeigneter Form dargestellt. Dazu gehört das Aggregieren der Daten (falls möglich, bspw. bei Single-Choice-Optionen) und das Darstellen mittels Diagrammen.

Scope Meeting Quality App

Level Benutzerziel

Primärer Akteur Organisator:in

Vorbedingungen

1. Organisator:in ist authentifiziert.
2. Das zu bewertende Meeting ist im System erstellt worden (UC 1: Besprechung erstellen).

Basisablauf

1. Organisator:in öffnet die Meeting-Übersicht. In dieser ist für jedes Meeting die Anzahl an eingetroffenem Feedback und eine Gesamtbewertung (1–5 Sterne) ersichtlich.
2. Organisator:in öffnet das Meetings, die von Interesse ist.
3. Organisator:in sieht in der Detailansicht die Auswertung des Feedbacks.

Variationen

3a. Es ist noch kein Feedback eingetroffen.
    1. System zeigt an, dass noch kein Feedback eingetroffen ist.

Erfolgsgarantie Das Feedback wird dem/der Organisator:in in geeigneter Form dargestellt.

UC 5b: Dashboard ansehen

UC 5b: Dashboard einsehen

Der/die Organisator:in sieht eine Zusammenfassung des von mehreren Besprechungen erhaltenen Feedbacks übersichtlich in einem Dashboard dargestellt. Im Dashboard dargestellte Informationen (z. B. durchschnittliche Effizienz, längstes und kürzestes Meeting etc.) dient dem Zweck, künftige Besprechungen zu optimieren.

Scope Meeting Quality App

Level Benutzerziel

Primärer Akteur Organisator:in

Vorbedingungen Organisator:in ist authentifiziert.

Basisablauf

1. Organisator:in öffnet das Dashboard.
2. System berechnet aus dem Feedback Masszahlen, welche übersichtlich dargestellt werden.

Variationen

2a. Es wurde noch kein Meeting erstellt.
    1. System zeigt an, dass noch kein Meeting existiert.

2b. Es ist noch kein Feedback eingetroffen.
    1. System zeigt an, dass noch kein Feedback eingetroffen ist.

Erfolgsgarantie Das Dashboard wird dargestellt und das aggregierte Feedback passend visualisiert.

UC 6: Newsletter erhalten

UC 6: Newsletter erhalten

Der/die Organisator:in erhält per E-Mail regelmässig einen Newsletter mit Tipps, wie Besprechungen verbessert werden können. Der Inhalt des Newsletter basiert auf dem Feedback, welches der/die Organisator:in für seine/ihre Besprechungen in einem festgelegten Zeitraum erhalten hat. Der Newsletter kann in den Einstellungen der Meeting-Quality-App an- und abgemeldet werden.

Scope Meeting Quality App

Level Benutzerziel

Primärer Akteur System

Vorbedingungen Organisator:in hat den Newsletter aktiviert.

Basisablauf

1. System berechnet aus dem Feedback des/der Organisator:in Masszahlen.
2. System stellt basierend auf den Masszahlen einen personalisierten Newsletter zusammen.
3. System schickt den Newsletter an den/die Organisator:in.

Erfolgsgarantie Der/die Organisator:in erhält einen persönlichen, auf ihn/sie zugeschnittenen Newsletter mit Tipps, wie künftige Meetings verbessert werden können.

UC 7: Feedback geben

UC 7: Feedback geben

Der/die Teilnehmer:in bewertet ein Meeting, an welchem er/sie teilgenommen hat, indem das Feedbackformular geöffnet und ausgefüllt wird. Zum Übermitteln der Rückmeldung ist keine Authentifizierung am System erforderlich, das Feedback wird somit anonym übergeben. Nachdem das Feedback abgegeben wurde, kann es nicht mehr verändert werden.

Scope Feedbackformular der Meeting-Quality-App

Level Benutzerziel

Primärer Akteur Teilnehmer:in

Vorbedingungen Teilnehmer:in hat den Link auf das zu bewertende Meeting erhalten (UC 4: Feedbackformular versenden).

Basisablauf

1. Teilnehmer:in öffnet den Fragebogen per Link.
2. Teilnehmer:in füllt die Felder des Formulars aus.
3. Teilnehmer:in übermittelt den Fragebogen.
4. System speichert die Daten und zeigt dem/der Teilnehmer:in eine Bestätigung an.

Variationen

1a. Das Meeting wurde in MS Teams abgehalten.
   1. Das Feedbackformular öffnet sich automatisch, sobald das Meeting beendet wird.

1b. Der Bewertungszeitraum ist abgelaufen.
   1. System zeigt an, dass kein Feedback mehr abgegeben werden kann.

1c. Der Fragebogen ist bereits von der festgelegten Anzahl Teilnehmer:innen ausgefüllt worden.
   1. System zeigt an, dass kein Feedback mehr abgegeben werden kann.

2a. Es sind nicht alle erforderlichen Felder ausgefüllt. Schritt 3 kann nicht ausgeführt werden.
   1. System hebt die erforderlichen Felder hervor.
   2. Teilnehmer:in füllt die benötigten Felder aus.
   3. Teilnehmer:in fährt mit Schritt 3 fort.

*a. Teilnehmer:in bricht den Vorgang ab. Es werden keine Daten gespeichert oder übermittelt.

Erfolgsgarantie Das Feedback wurde vom System verarbeitet und ist für den/die Organisator:in unmittelbar ersichtlich, ohne dass aus den Metadaten auf den/die Teilnehmer:in rückgeschlossen werden kann.

UC 8: Traktanden vorschlagen

UC 8: Traktanden vorschlagen

Der/die Teilnehmer:in kann Traktanden zu einer angesetzten Besprechung vorschlagen, welche er/sie gerne behandeln würde.

UC 9: Firma verwalten

UC 9: Firma verwalten

Der/die Administrator:in kann eine Firma anlegen, bearbeiten und löschen. Dies ist nötig, um später Mitarbeitende verwalten zu können (siehe UC 10: Mitarbeitende verwalten).

UC 10: Mitarbeitende verwalten

UC 10: Mitarbeitende verwalten

Der/die Administrator:in kann Benutzerkonten für die Mitarbeitenden der Firma anlegen, damit diese sich nicht selbst registrieren müssen. Benutzerkonten können deaktiviert werden, falls bspw. ein:e Mitarbeiter:in die Firma verlässt.

3.3 User Storys

Die User Storys dienen als Grundlage für die Entwicklung und beschreiben die zu implementierenden Funktionen aus Benutzersicht. Die verwendete Terminologie in diesem Abschnittes sowie das Vorgehen zum Herleiten der User Storys aus den Use Cases ist detailliert im Projektplan[4] festgehalten.

Vorlage für die User Storys Damit die User Storys einheitlich gestaltet werden, wird mit folgender Vorlage gearbeitet:

User-Story-Vorlage

Als [Akteur] von [Anwendung] möchte ich [Zielfunktion], damit [Nutzen].

Akzeptanzkriterien

- Falls [Kontext], wenn [eine bestimmte Aktion durchgeführt wird], dann [sollte eine Reihe von Konsequenzen eintreten].

Technische Hinweise

- ...

3.3.1 Epics

Im Backlog des Projektes sind drei Epics erstellt worden, welche sich gemäss nachfolgender Tabelle auf die Use Cases abbilden lassen.

Epic	Use Case
MS Teams Integration	UC1a: Besprechungen in MS Teams erstellen
Fragebogen automatisch versenden	UC 4: Feedbackformular versenden
Dashboard	UC 5b: Feedback im Dashboard auswerten

Tabelle 3.1: Epics mit der Zuordnung der Use Cases

3.3.2 Spikes

Spike: Teams-Integration

Die Integration der Meeting-Quality-App in MS Teams lässt sich auf diverse Arten umsetzen. In diesem Spike sollen erste Informationen über die Integrationsmöglichkeiten gesammelt und anhand von einem Prototypen ausprobiert werden.

Zeitrahmen 5 Stunden

Jira-Link: MQ-21

Spike: Meetings in MS Teams via API erstellen/auslesen

Die Microsoft Graph API ist sehr umfassend. Momentan ist unklar, wie die Authentifizierung bei der API abläuft und welche Berechtigungen eine App haben muss, um auf die API zugreifen zu können. Dieser Spike soll darüber klarheit schaffen, indem ein Prototyp erstellt wird, welcher mittels Single Sign-on (SSO) auf die Graph API zugreift und versucht, Meetings auszulesen udn zu erstellen.

Zeitrahmen 8 Stunden

Jira-Link: MQ-60

3.3.3 User Storys

Während der Anforderungserhebung sind folgende User Storys erstellt worden:

- Meeting Quality in MS Teams nutzen
- Link zum Feedbackformular in die Zwischenablage kopieren
- Meeting automatisch von MS Teams aus erstellen
- Meetings importieren
- Fragebogen automatisch versenden

- Meetingzeit erfassen
- Mail der Teilnehmenden erfassen
- Meetingdauer im Dashboard anzeigen
- Effizienz der Meetings im Dashboard anzeigen
- Durchschnittliches Rating pro Woche

Aus Darstellungsgründen wird an dieser Stelle nur der Titel angegeben. Die kompletten User Storys sind in Anhang C zu finden.

Kapitel 4

Domänenanalyse

In der Domänenanalyse werden die Konzepte, welche in Kapitel 3 erörtert wurden, in Business-Entitäten überführt und in einem Domänenmodell miteinander in Verbindung gebracht.

4.1 Domänenmodell

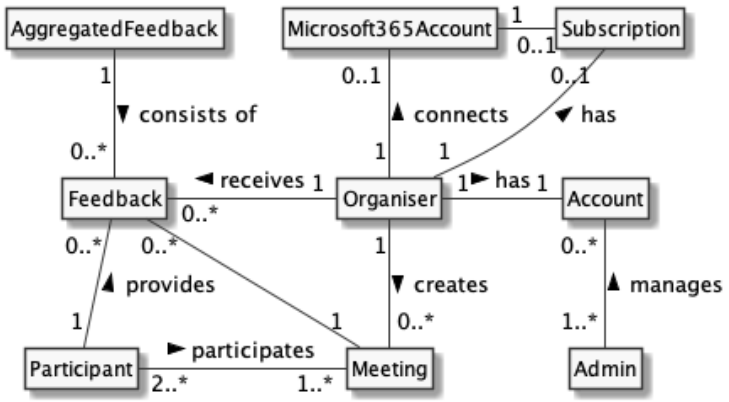


Abbildung 4.1: Domänenmodell der Meeting-Quality-Domäne

Die grundlegenden Konzepte drehen sich dabei um das Meeting. Ein/eine Organisator:in (*Organiser*) erstellt Meetings und erhält Feedback dazu. Ein Feedback gehört immer zu genau einem Meeting und wird von einem einzelnen Teilnehmenden (*Participant*) übermittelt.

Zu jedem Meeting können beliebig viele Feedbacks eingegeben werden, um so Feedback von allen Teilnehmenden zu erhalten. Teilnehmende können Feedbacks für mehrere Meetings abgeben, pro Meeting jedoch nur eines.
Da eine Besprechung per Definition ein «ausführliches Gespräch über eine bestimmte Sache»[11] ist, besteht ein Meeting immer aus mindestens zwei Teilnehmenden.

Jede/jeder Organisator:in hat ausserdem einen Account, mit welchem er/sie sich in der Meeting-Quality-App anmeldet. Diese Accounts sollen künftig durch Administrator:innen verwaltet werden können (vgl. UC 10: Mitarbeitende verwalten)

4.1.1 Integration von Microsoft Teams

Für die Anbindung von MS Teams verbindet sich ein/eine Organisator:in mit seinem/ihrem persönlichen Microsoft-Account und löst ein Abonnement (*Subscription*) bei Microsoft, um über neue und geänderte Besprechungen informiert zu werden. Das Abonnement ist immer einem Microsoft-Account zugeordnet und Account ist maximal ein Abonnement erlaubt.

4.1.2 Dashboard

Für die Umsetzung von UC 5b: Dashboard ansehen ist es nötig, die Daten von mehreren Feedbacks miteinander zu verknüpfen. Das geschieht über das AggregatedFeedback, welches aus beliebig vielen Feedbacks besteht. Jedes Feedback kommt dabei nur in einer Aggregation vor.

Kapitel 5

Softwarearchitektur und -design

Um die Systemkomponenten sinnvoll zu strukturieren und somit die vorgängig erhobenen Anforderungen an das System sauber und effizient umzusetzen, ist eine adäquate Softwarearchitektur notwendig. Diese wird in nachfolgenden Kapitel erläutert und die grundlegenden Designentscheidungen aufgezeigt.

5.1 Systemkontext

Für eine sinnvolle Umsetzung der in Kapitel 3 Anforderungsanalyse definierten Anforderung ist eine Anbindung der Meeting-Quality-App an weitere externe Systeme notwendig. Bei diesen handelt es sich um *Microsoft Teams*, die *Microsoft Graph API* sowie den webbasierten E-Mail-Service *SendGrid*. Die Interaktion der Systeme ist in Abbildung 5.1 zu sehen.

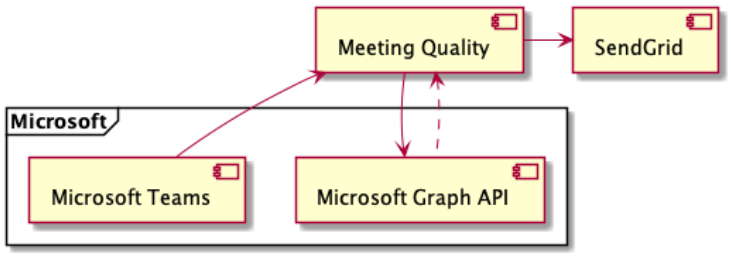


Abbildung 5.1: Systemkontext (Quelle: eigene Abbildung)

Für die Anbindung von Microsoft haben sich mehrere Optionen angeboten. Eine vertiefte Analyse dieser Optionen findet sich in der Variantenstudie in Anhang B; an dieser Stelle wird nur die ausgewählte Option erläutert.

5.1.1 Microsoft Teams

Die Anbindung zu MS Teams erfolgt über eine Microsoft Teams App und ist denkbar einfach: Die Meeting-Quality-App wird per `iframe` in MS Teams eingebettet und kann so direkt in der Plattform bedient werden. Eine genauere Abhandlung der Umsetzung ist Abschnitt 6.2 Microsoft Teams Integration festgehalten.

5.1.2 Microsoft Graph API

Die zentrale Schnittstelle, um Informationen aller Art von Microsoft abzurufen. Für die Meeting-Quality-App sind besonders Daten zu Besprechungen relevant.

Für die Interaktion mit der Graph API bietet Microsoft verschiedene Tools und Softwarebibliotheken an, z. B. die *Microsoft Graph JavaScript Client Library*[1] oder den *Graph Explorer*[2]. Diese Programme müssen nicht zwingend verwendet werden, erleichtern die Entwicklung aber ungemein.

Datenformat

Für den Austausch der Daten über die Graph API verwendet Microsoft eine hauseigene Implementation der HTTP-basierte Open Data Protocol (OData) Spezifikation. OData baut auf Level 3 des Richardson Maturity Model auf, wodurch sich eine zustandslose Webapplikationen nach den Prinzipien Hypermedia as the Engine of Application State (HATEOAS) umsetzen lässt.[15]

Autorisierung

Um auf die Graph API zugreifen zu können, ist eine vorgängige Autorisierung mittels OAuth notwendig. Wie diese Abläuft ist in Unterabschnitt 5.3.1 Authentifizierung bei der Microsoft Identitätsplattform beschrieben.

API-Übersicht

Hier wird ein Überblick über die relevantesten Endpunkte der Graph API gegeben. Ausführlichere Informationen sind in der offiziellen Dokumentation der Graph API[3] zu finden.

Die Basis-URL ist für alle Endpunkten gleich: `https://graph.microsoft.com/v1.0/`

Endpunkt	Beschreibung
`/me`	Liefert Informationen über den/die aktuell eingeloggte:n Nutzer:in.[27]
`/events/<eventID>`	Liefert Daten zu den Besprechungen im Kalender eines/einer Benutzer:in. Durch die Angabe einer `eventId` kann ein spezifischer Event angesprochen werden.[26]

Tabelle 5.2: Übersicht der relevanten Endpunkte der Graph API

Für die Endpunkte existiert auch eine alternative URL, welche unter Verwendung der `homeId` benutzt werden kann (z. B. `/Users/<homeId>/events`).

Abbildung 5.2 zeigt exemplarische eine Abfrage von `https://graph.microsoft.com/v1.0/me` über den Graph Explorer.

[1] https://www.npmjs.com/package/@microsoft/microsoft-graph-client
[2] https://developer.microsoft.com/en-us/graph/graph-explorer
[3] https://docs.microsoft.com/en-us/graph/

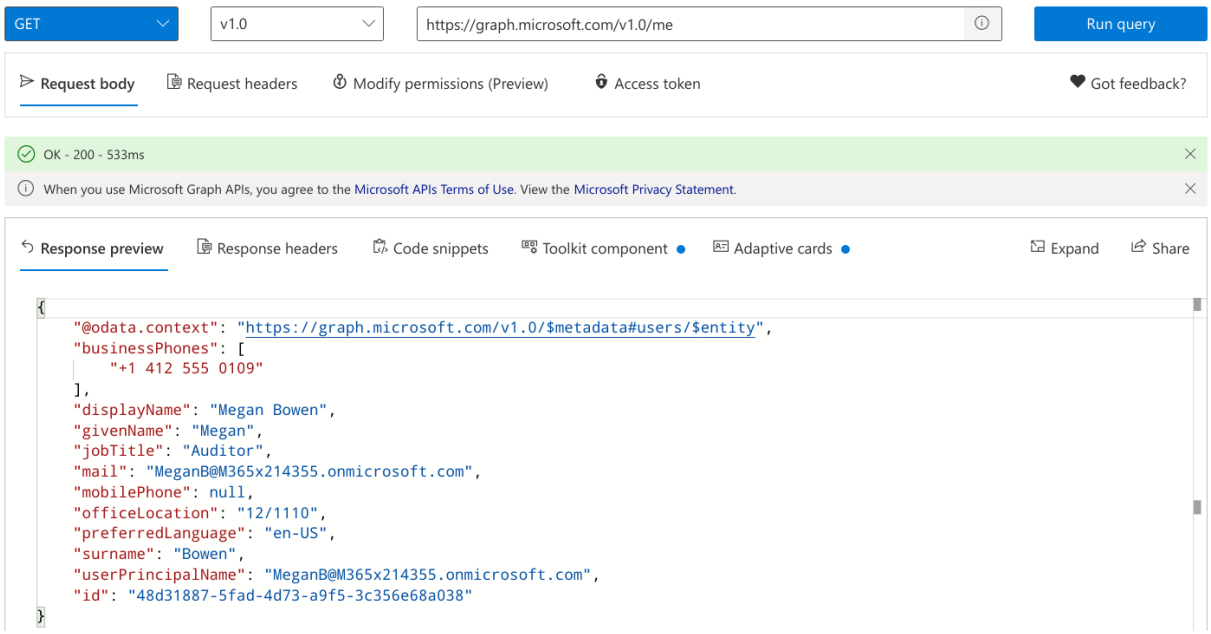


Abbildung 5.2: Beispiel einer Abfrage des `/me`-Endpunktes mittels Graph Explorer (Quelle: Microsoft)

Abonnemente

Für diverse Ressourcen der Graph API können ausserdem Änderungsbenachrichtigungen abonniert werden. Dazu gehören auch die im vorherigen Abschnitt erwähnten Events. Bei einer Änderungen an der abonnierten Ressource sendet die Graph API eine Benachrichtigung an einen vordefinierten Listener-Endpunkt (sog. *Webhook*, auch bekannt als *Callback-URL*) im Zielsystem. Der Endpunkt muss dabei per HTTPS erreichbar sein.[25]

5.1.3 SendGrid

Um auf einfache Weise E-Mails aus der Meeting-Quality-App versenden zu können erfolgt eine Anbindung an den cloudbasierten E-Mail-Service SendGrid. SendGrid wird seit 2009 vom US-amerikanischen Unternehmen Twilio betrieben, welches sich auf Cloud-Kommunkationsplattformen spezialisiert hat, und wird von Unternehmen wie Uber, Spotify und Airbnb verwendet.[33]

SendGrid besteht aus einer Web-App für die Konfiguration und das Controlling und wird via API an ein System angebunden. Die Web-App zeichnet sich u.a. durch ein intuitives Dashboard aus, in dem diverse Buisness-Metriken verfügbar sind, sowie einer Funktion zum Verwalten von E-Mail-Vorlagen. Weiter können Postfächer und Domains verwaltet werden.

Autorisierung und Konfiguration

Für die Verwendung von SendGrid wird ein Benutzerkonto benötigt. Nach Erstellung des Accounts wird die Absenderadresse definiert, von welcher die E-Mails versendet werden sollen.

API

Die SendGrid-API ist sehr einfach gehalten und intuitiv zu verwenden. Mit *@sendgrid/mail*[4] existiert zusätzlich eine Programmbibliothek, welche den Zugriff auf die API abstrahiert und eine typisierte Schnittstelle (für TypeScript) anbietet.

5.2 Designentscheidungen

Dieser Abschnitt hält die essenziellen Entscheidungen fest, welche bezüglich Systemdesign getroffen wurden.

5.2.1 Microsoft als Single Source of Truth

Für die Integration von MS Teams sollen Meeting-Daten von Microsoft mit der Meeting-Quality-App synchronisiert werden. Fraglich ist dabei, wie sich das System verhalten soll, wenn Daten sowohl in der Meeting-Quality-App als auch bei Microsoft verändert wurden. In Absprache mit dem Industriepartner wurde festgelegt, dass Microsoft als Single Source of Truth verwendet wird: Wird zum Beispiel der Zeitpunkt einer Besprechung sowohl in der Meeting-Quality-App wie auch im Microsoft-Kalender geändert, gelten die Änderungen von Microsoft.

5.2.2 Objektorientierung

Der bestehende Code folgt vor allem den Paradigmen der funktionalen Programmierung. Durch die Einführung von objektorientierten Ansätzen unter Einhaltung der SOLID-Prinzipien sollen jedoch deren diverse Vorteile (Datenkapselung, hohe Kohäsion bei gleichzeitig geringer Koppelung usw.) nutzbar gemacht werden.[10]

5.2.3 Schichtenarchitektur

Um die Komplexität im Backend zu reduzieren und eine geringe Koppelung bei gleichzeitig hoher Kohäsion zu erreichen, wird die Software in drei logische Layer strukturiert: *Presentation*, *Business Logic* und *Data Access*.

Presentation Layer Der Presentation-Layer nimmt HTTP-Anfragen entgegen, greift auf die Business-Logik zu und reicht die angefragten Ressourcen in einem für das Frontend geeigneten Format dar.

Business Logic Layer Beinhaltet die Business-Logik, das Herzstück der Applikation. Dieser Layer ist z. B. für Authentifizierung der Benutzenden oder die Aggregation der Feedbacks zuständig und kommuniziert dazu mit dem Data Access Layer.

Data Access Layer In diesem Layer findet der Zugriff auf die Daten der Applikation, also v. a. die Datenbank, statt.

5.2.4 Business-Modelle und DAOs

Für die konkrete Umsetzung der oben genannten Entscheidungen werden Business-Modelle und Data Access Objects (DAOs) verwendet. Die Business-Logik wird nach dem Single-responsiblity-Prinzip (vgl. Unterabschnitt 5.2.2 Objektorientierung) in Business-Modelle gekapselt. Diese beziehen ihre Daten von den DAOs, welche im Data Access Layer leben. Das ermöglicht bspw. eine einfache Anpassung der Logik in einem der Data Access Objects, falls ein externes System seine Datenstruktur verändert, ohne dass das Business-Modell angepasst werden muss.

5.2.5 Refactorings

Um die in Abschnitt 2.7 Beurteilung des Software-Engineerings festgestellten Mängel zu adressieren, werden auf Code-Ebene im Frontend und Backend zusätzlich Refactorings durchgeführt.

App-Struktur

Der Quellcode ist sowohl im Frontend wie auch im Backend *by type* organisiert, das heisst, dass z. B. alle Controller zusammen im selben Ordner abgelegt sind. Durch den steigen ausbau der Codebasis wird diese Struktur aber immer unübersichtlicher. Der Code soll daher neu *by feature* organisiert werden, wobei alle zu einem Feature gehörenden Codestücke (z. B. Controller, Modelle und DAOs von Meetings) im selben Verzeichnis abgelegt sind.

Slices für Redux

Um den Boilerplate-Code, welcher durch Redux anfällt, zu reduzieren, werden die Aktionen und Reducer zusammengelegt in sogenannte *Slices* reorganisiert, was auch den Best Practices von Redux enstpricht.[2]

Vereinfachen der React-Komponenten

Die grossen React-Komponenten im Frontend werden in kleinere, überschaubarere Komponenten heruntergebrochen. Es wird ausserdem zwischen zustandsbehafteten und zustandslosen Komponenten unterschieden werden. Die zustandsbehafteten Komponenten übergeben dabei via Parameter genau die Portion des Zustands an die zustandslosen Komponenten, welchen diese benötigen. So soll die Codebasis übersichtlicher und der Datenfluss transparenter werden.

5.3 Prozesse und Abläufe

5.3.1 Authentifizierung bei der Microsoft Identitätsplattform

Um die Microsoft Graph API verwenden zu können, muss sich ein:e Benutzer:in vorgängig über die Microsoft Identitätsplattform authentifizieren. Diese stellt dann einen Access Token in Form eines JSON Web Token zur Verfügung, welcher den Zugriff auf die Graph API ermöglicht. Der Access Token muss für jede Abfrage der Graph API im `Authorization`-Header der HTTP-Anfrage mitgeschickt werden.

Dieses Vorgehen basiert im Grunde auf dem OAuth 2.0-Standard, wobei einige Microsoft-spezifische Abweichungen auftreten können.[8] In Abbildung 5.3 ist dieser Ablauf in Form eines Sequenzdiagrammes ersichtlich.

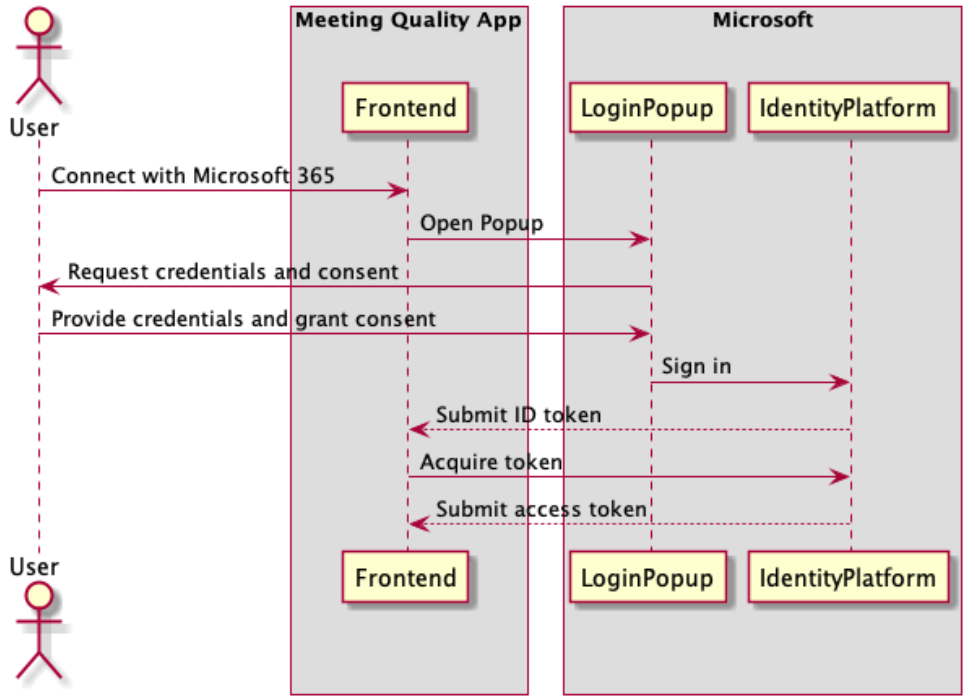


Abbildung 5.3: Authentifizierung bei der Microsoft Identitätsplattform (Quelle: eigene Abbildung)

Ein:e Benutzer:in der Meeting-Quality-App wählt dabei im Frontend die Option, sich mit Microsoft 365 zu verbinden. Das Frontend öffnet ein Popup-Dialog mit der Login-Seite für die Microsoft Identitätsplattform. Darin gibt der/die Benutzer:in seine/ihre Credentials (E-Mail-Adresse und Passwort) ein. Beim ersten Login erscheint dann ein Dialog, in welchem nach der Einverständnis (en. *Consent*) gefragt wird, damit die Meeting-Quality-App auf Daten aus dem Benutzerkonto zugreifen darf.

On-Behalf-Of Flow

Das Backend kann mit dem Access Token, welches es vom Frontend bekommt, nicht direkt auf die Graph API zugreifen, sondern muss unter Benutzung dieses Tokens selbst ein neues, sogenanntes On Behalf Of (OBO) Token, lösen. Der OBO-Flow ist ebenfalls Teil der OAuth-2.0-Spezifikation und wird im Sequenzdiagramm in Abbildung 5.4 abgebildet.[28]

Der Ablauf wird durch die Übermittlung des Access Token (vgl. Unterabschnitt 5.3.1 Authentifizierung bei der Microsoft Identitätsplattform) durch das Frontend ausgelöst. Nach erfolgreichem Erhalt des OBO-Tokens wird dieses für die zukünftige Kommunikation mit der Graph API in der Datenbank abgelegt.

5.3.2 Abonnement auf Änderungen von Events

Für die erfolgreiche Integration von Microsoft Teams müssen Änderungen an Meeting-Daten mit der Meeting-Quality-App synchronisiert werden. Wie in Abschnitt 5.1.2 Abonnemente

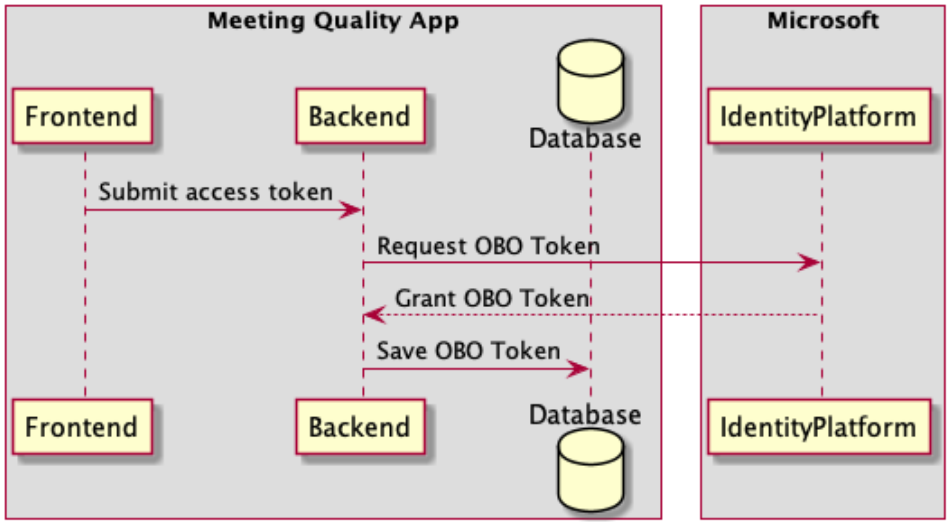


Abbildung 5.4: Ablauf des OAuth 2.0 On-Behalf-Of Flow

erläutert wurde, bietet die Graph API dazu die Möglichkeit, Abonnemente auf Änderungen an Ressourcen zu erstellen. Auf die ressourcenintensive Alternative von Polling, der zyklische Abfrage der Graph API, kann daher verzichtet werden. Der Ablauf zum Erstellen eines Abonnements (en. *Subscription*) bei der Graph API ist im Sequenzdiagramm in Abbildung 5.5 ersichtlich.

Der Ablauf wird durch den/die Benutzer:in ausgelöst, welche:r im Frontend die Einstellung zum Synchronisieren der Meetings mit Microsoft aktiviert. Voraussetzung dafür ist natürlich eine bestehende Verbindung mit Microsoft (vgl. Unterabschnitt 5.3.1 Authentifizierung bei der Microsoft Identitätsplattform), welche durch das Frontend sichergestellt wird.

Zum Erstellen des Abonnements wird dazu eine Anfrage an die Graph API gesendet. In der Anfrage wird die zu abonnierende Ressource und die Art von Änderung spezifiziert, über welche benachrichtigt werden soll. Auch die Dauer des Abonnements und die Callback-URL kann festgelegt werden.
Microsoft empfängt diese Anfrage und überprüft zuerst die Callback-URL. Diese muss innerhalb von wenigen Sekunden mit einem `HTTP 200 OK` anworten, ansonsten wird der Prozess abgebrochen.

Danach wird das Abonnement aufseiten von Microsoft erstellt und an das Backend übermittelt. Das Backend speichert dieses zusammen mit der ID des/der Benutzer:in in der Datenbank ab, um die eintreffenden Benachrichtigung dem/der Benutzer:in zuordnen zu können.

E-Mails versenden

Ein externer Job führt ein Skript aus, welches Abfragen für die Datenbank enthält. In diesen Abfragen werden die E-Mails zu den Meetings, die in der nächsten halben Stunde ihre Endzeitpunkt haben, herausgesucht. Diese E-Mails werden dann mit einer Personalisierung für die Teilnehmer ausgestattet und an die SendGrid-API verschickt. SendGrid verschickt die E-Mails dann zum Endzeitpunkt der Meetings an die angegebenen Empfänger:innen.

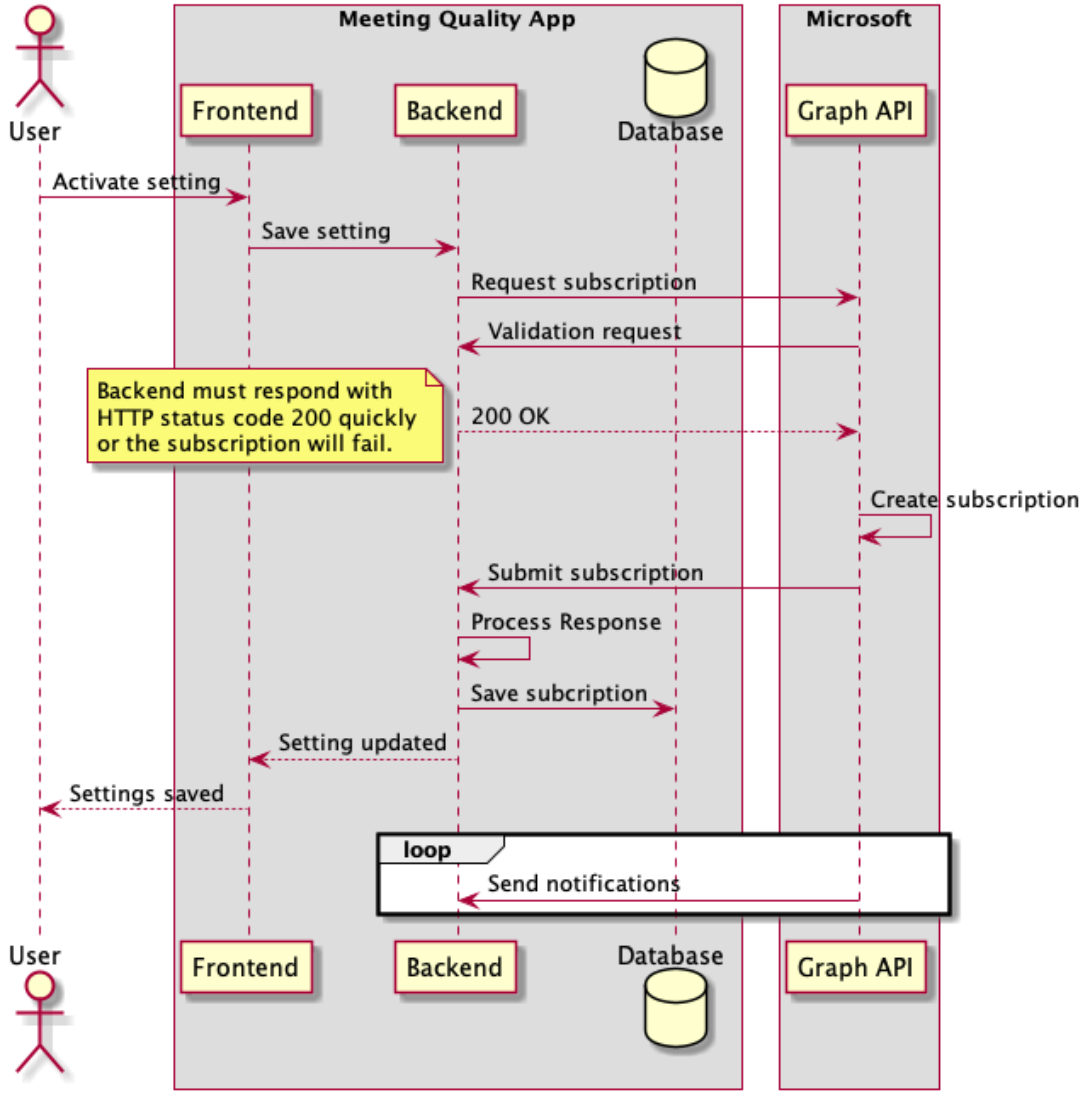


Abbildung 5.5: Sequenzdiagramm

Graph-Abonnemente erneuern

Die Abonnements bei der Graph-API müssen routinemässig erneuert werden, da sie nur für einen bestimmten Zeitraum gelöst werden können. Da Microsoft keine Benachrichtung zum Auslaufen eines Abonnements bietet, muss dies ebenfalls mit einem externen Prozess gehandhabt werden.

Dazu wird ein mal pro Tag durch ein Skript die Datenbank abgefragt. In der Abfrage werden die innerhalb des kommenden Tages auslaufenden Abonnemente aggregiert. Über diese Liste wird dann iteriert und das auslaufende Abonnement bei Microsoft storniert, neu gelöst und wieder in die Datenbank gespeichert.

5.4 Erweiterung des Technologie-Stacks

Der in Abschnitt 2.3 Technologien beschriebene Technologie-Stack wird um die in den nachfolgenden Abschnitten aufgelisteten Technologien erweiter.

5.4.1 Programmiersprachen

TypeScript Der bestehende JavaScript-Code im Front- sowie auch im Backend wird durch TypeScript-Code ersetzt. TypeScript ist eine Obermenge (en. *superset*) von JavaScript und ergänzt dieses unter anderem mit Typen. Dies hilft nicht nur Laufzeitfehler vorzubeugen, sondern vereinfacht auch die Entwicklung, z. B. durch Auto-Vervollständigung in der Entwicklungsumgebung. Insgesamt soll so die Zuverlässigkeit und Wartungsfreundlichkeit der Applikation verbessert werden.

5.4.2 Programmbibliotheken

ESLint Ein statisches Codeanalyseprogramm, mit welchem problematische Stellen im Quellcode aufgedeckt werden können.

Jest Für das Schreiben und Ausführen von Tests im Front- und Backend wird das Testing-Framework Jest eingesetzt. Jest ist einfach zu konfigurieren ist und sich hervorragend mit React und Node.js verwenden.[14]

Microsoft Authentication Library (MSAL) Da der Autorisierungsprozess bei der Microsoft-Identitätsplattform ziemlich umfassend ist (vgl. Unterabschnitt 5.3.1 Authentifizierung bei der Microsoft Identitätsplattform), bietet Microsoft mit der Microsoft Authentication Library (MSAL) einen vereinfachte Schnittstelle zum Erwerben von Zugriffstoken an. Diese werden benötigt, um Benutzer:innen zu authentifizieren und um sicher auf die Microsoft Graph API zuzugreifen.[29]

5.5 Frontend

Da im Frontend die grundlegende Architektur und der Datenfluss bereits in der ursprünglichen Applikation ganz akzeptabel waren, muss diese nicht massgeblich verändert werden.

Es erfolgt einzig eine Reorganisation der App-Struktur, wie in Unterabschnitt 5.2.5 Refactorings beschrieben. Dadurch lassen sich die beiden Paket *Pages* und *Store* in ein neues Paket, *Features*, kombinieren. Das Paket *Services* wird in *Core* umbenennt und *Components* nennt sich neu *Common*.

In den nachfolgenden Abschnitten werden nur noch die Komponenten beschrieben, welche neu hinzugekommen sind oder relevante Änderungen erfahren haben.

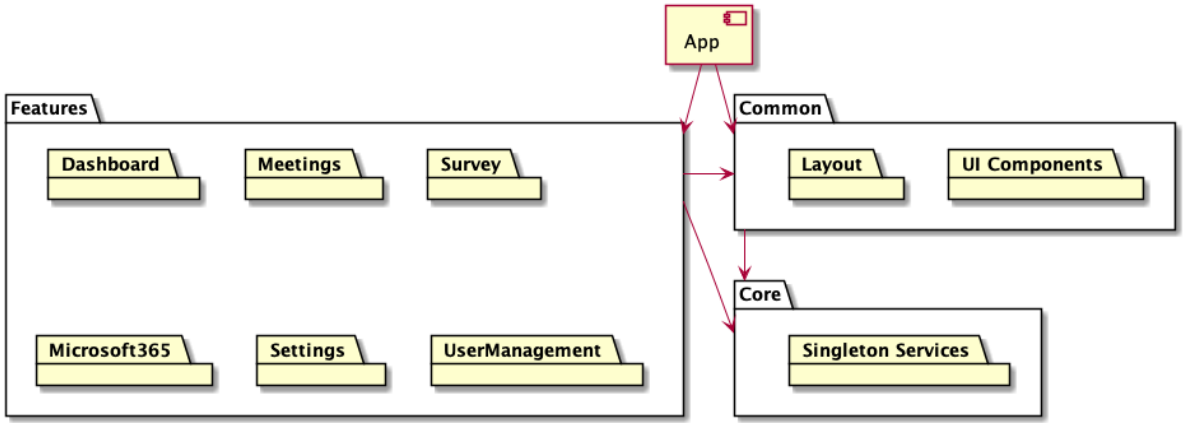


Abbildung 5.6: Frontend-Softwarepakete nach der Reorganisation (Quelle: eigene Abbildung)

5.5.1 Core

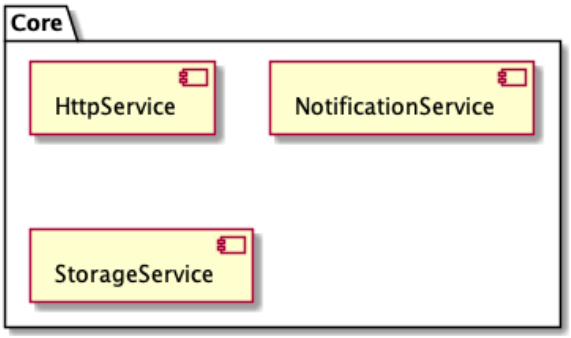


Abbildung 5.7: Komponenten im Core-Paket (Quelle: eigene Abbildung)

Komponente	Beschreibung
NotificationService	Schnittstellen um Benachrichtigungen in Form von Toast-Nachrichten an einem zentralen Ort zu konfigurieren und im UI darzustellen. Reduziert Code-Duplikate.

Tabelle 5.4: Beschreibung der Komponenten in Core

5.5.2 Common

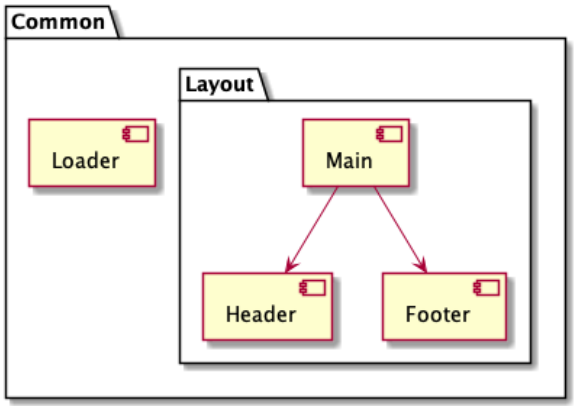


Abbildung 5.8: Komponenten im Core (Quelle: eigene Abbildung)

Komponente	Beschreibung
Main	Die Hauptkomponente für die Darstellung der Seiten in der App. Nimmt eine Seite zur Darstellung entgegen und rendert diese zwischen dem Header und Footer. So können bestehende Code-Duplikate reduziert werden.

Tabelle 5.6: Beschreibung der Komponenten in Common

5.5.3 Features

Jedes Feature beinhaltet mehrere React-Komponenten (*UI Components*) und verfügt über einen eigenen Data-Access-Layer.

Komponente	Beschreibung
*Feature*State	Beinhalten den für das Feature relevanten Teil des Applikationzustandes. Entspricht dem Redux-Slice, der die Aktionen und den Reducer beinhalten. Der HTTP-Service wird aus den Aktionen aufgerufen.
*Feature*HttpService	Enkapsuliert den Zugriff auf HTTP und die dazu notwendigen Einstellungen (URL, Datenstruktur etc.). Liefert die Datenobjekte zurück, welche vom State verarbeitet werden.

Tabelle 5.8: Beschreibung der Komponenten Features

Dashboard

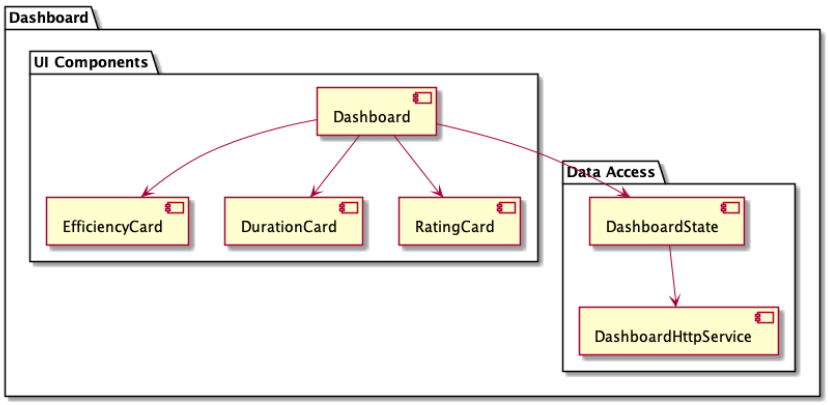


Abbildung 5.9: Komponenten im Dashboard-Paket (Quelle: eigene Abbildung)

Meetings

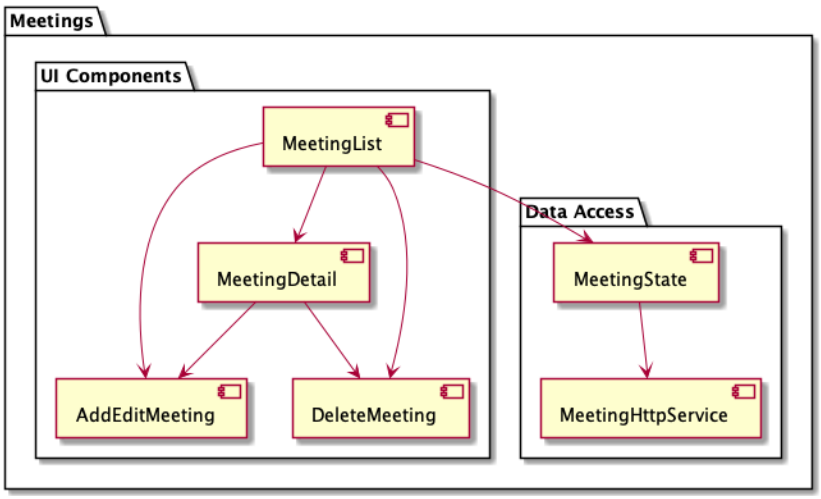


Abbildung 5.10: Komponenten im Meetings-Paket (Quelle: eigene Abbildung)

Survey

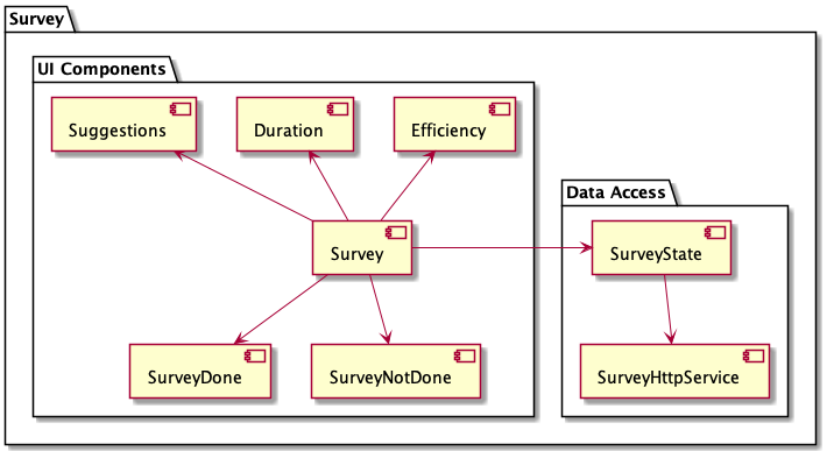


Abbildung 5.11: Komponenten im Survey-Paket (Quelle: eigene Abbildung)

Microsoft365

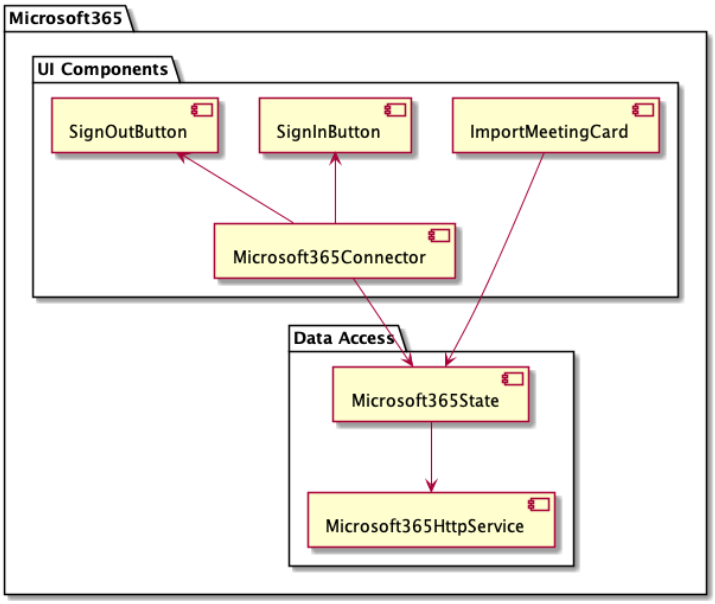


Abbildung 5.12: Komponenten im Microsoft365-Paket (Quelle: eigene Abbildung)

Settings

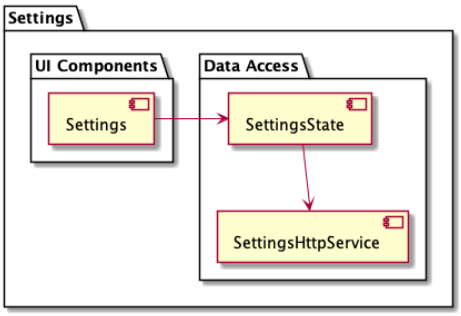


Abbildung 5.13: Komponenten im Settings-Paket (Quelle: eigene Abbildung)

UserManagement

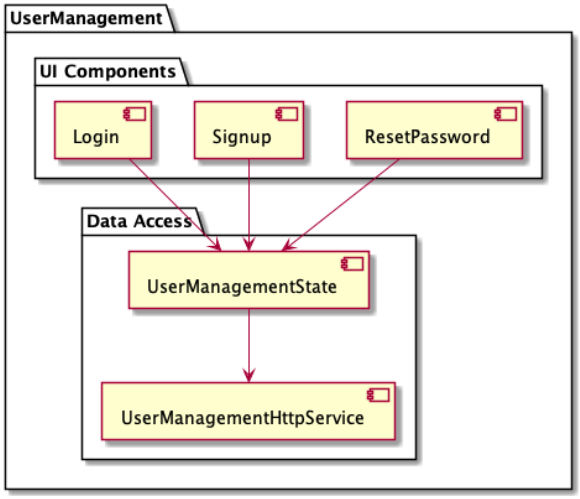


Abbildung 5.14: Komponenten im UserManagement-Paket (Quelle: eigene Abbildung)

5.6 Backend

Der bestehende Code im Backend wird komplett neu organisiert und in die drei Layer *Presentation*, *Business Logic* und *Data Access* aufgeteilt.

5.6.1 Presentation Layer

Der Presentation Layer enthält die bereits bestehenden Routes und Controller. Die Logik wird jedoch aus den Controllern extrahiert und in die Business-Logik ausgelagert.

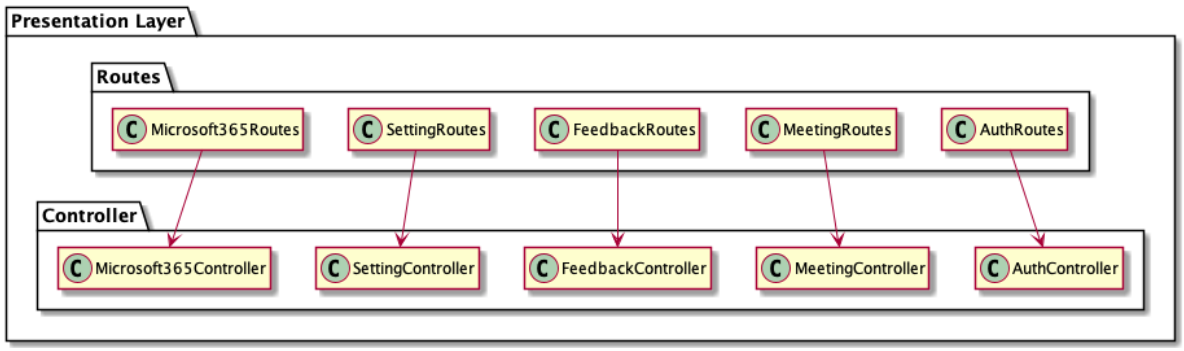


Abbildung 5.15: Klassen im Presentation Layer

5.6.2 Business Logic Layer

Der Business Logic Layer enthält die nach Funktionen organisierte Business-Logik. Diese inkludiert die vorhin im API-Paket ansässigen Komponenten *Auth*, *Meetings* und *Feedback* (vgl. Unterabschnitt 2.5.2 Backend), welche hier nicht erneut beschrieben werden.

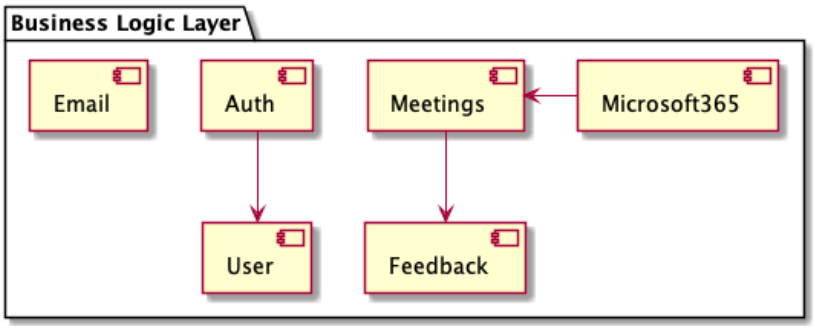


Abbildung 5.16: Komponenten im Business Logic Layer

Komponente	Beschreibung
Email	Komponente für den Versand von E-Mails.
User	Stellt Informationen zum/zur aktuell eingeloggten Benutzer:in bereit.
Microsoft365	Verwaltet die Verbindung und Abonnemente zur Graph API verarbeitet die eingehenden Benachrichtigungen.

Tabelle 5.10: Beschreibung der Komponenten in der Business-Logik

Komponenten, die substanziell verändert wurden oder für das System besonders relevant sind, werden nachfolgend noch um Klassendiagramme ergänzt.

Feedback

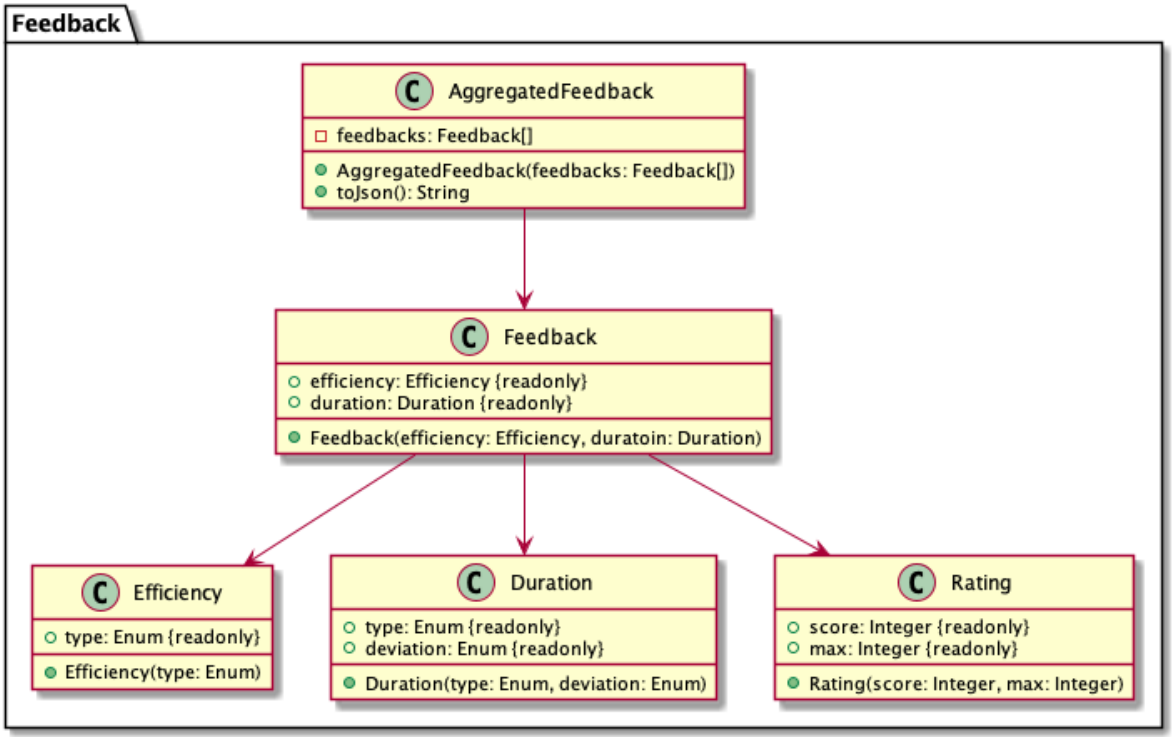


Abbildung 5.17: Klassendiagramm der Feedback-Komponente

Microsoft365

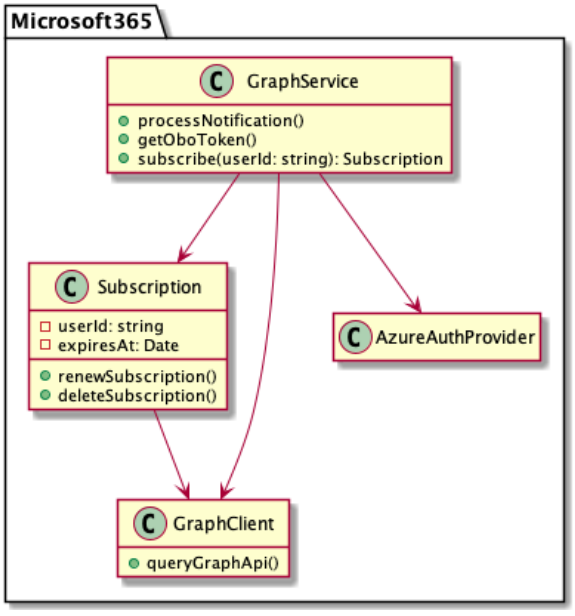


Abbildung 5.18: Klassendiagramm der Microsoft365-Komponente

User

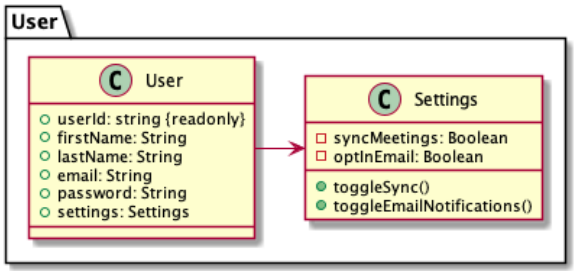


Abbildung 5.19: Klassendiagramm der User-Komponente

5.6.3 Data Access Layer

Das Data Access Layer beinhaltet die Data Access Objects, auf die vom Business Logic Layer aus zugegriffen wird. Sie abstrahieren den Zugriff auf die Mongoose-Modelle, welche die Operationen zum auf der Datenbank vornehmen.

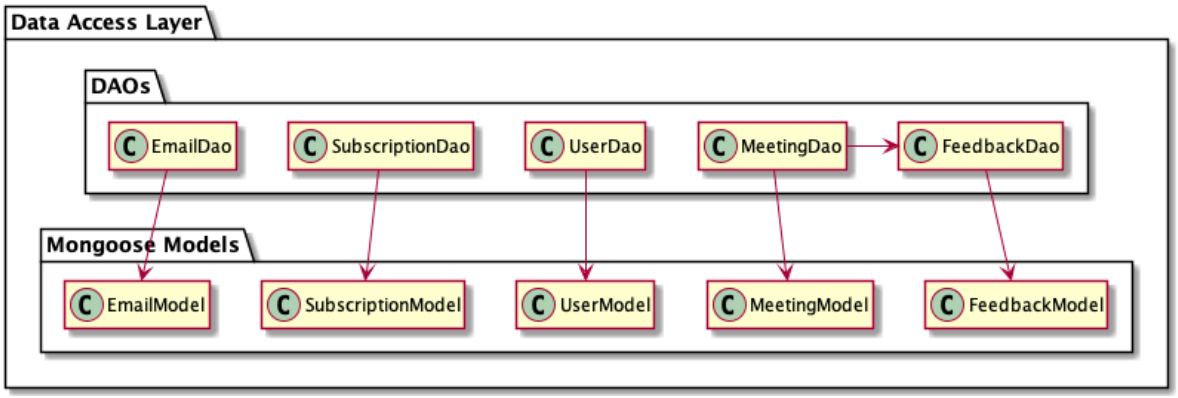


Abbildung 5.20: Klassen im Data Access Layer

5.7 Erweiterung des Datenmodells

Da mit den neuen Anforderungen an die Applikation und der Weiterentwicklung der Architektur auch neue Daten anfallen, muss das Datenmodell entsprechend erweitert werden. Das bisherige Schema (Abbildung 2.20) bleibt bestehen und es werden nur Ergänzungen gemacht.

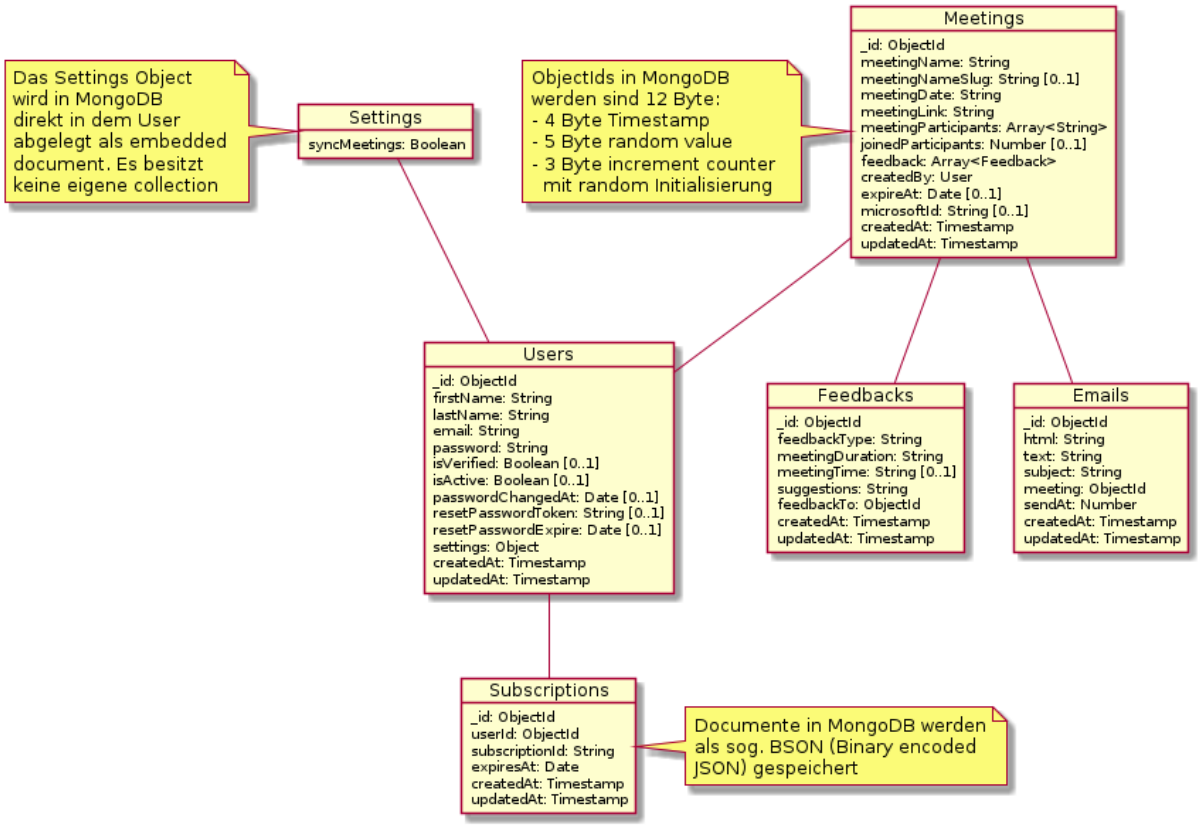


Abbildung 5.21: Erweitertes Datenmodell (Quelle: eigene Abbildung)

Kapitel 6

Ergebnisse

Im nun folgenden Kapitel werden die Ergebnisse der Arbeit aufgezeigt. Dazu wird die zu lösende Problemstellung nochmals aufgegriffen und mit der ausgearbeiteten Architektur verbunden, Hintergründe dargestellt und die Funktionalität an konkreten Beispielen erläutert.

6.1 Dashboard

Das Dashboard, welches in UC 5b: Dashboard ansehen beschrieben ist, konnte erfolgreich implementiert werden. In Zusammenarbeit mit dem Industriepartner wurde der Inhalt festgelegt, welcher im Dashboard dargestellt wird: Eine Zusammenfassung der Effizienz und Dauer aller Meetings, sowie eine Bewertung in Form von Sternen.

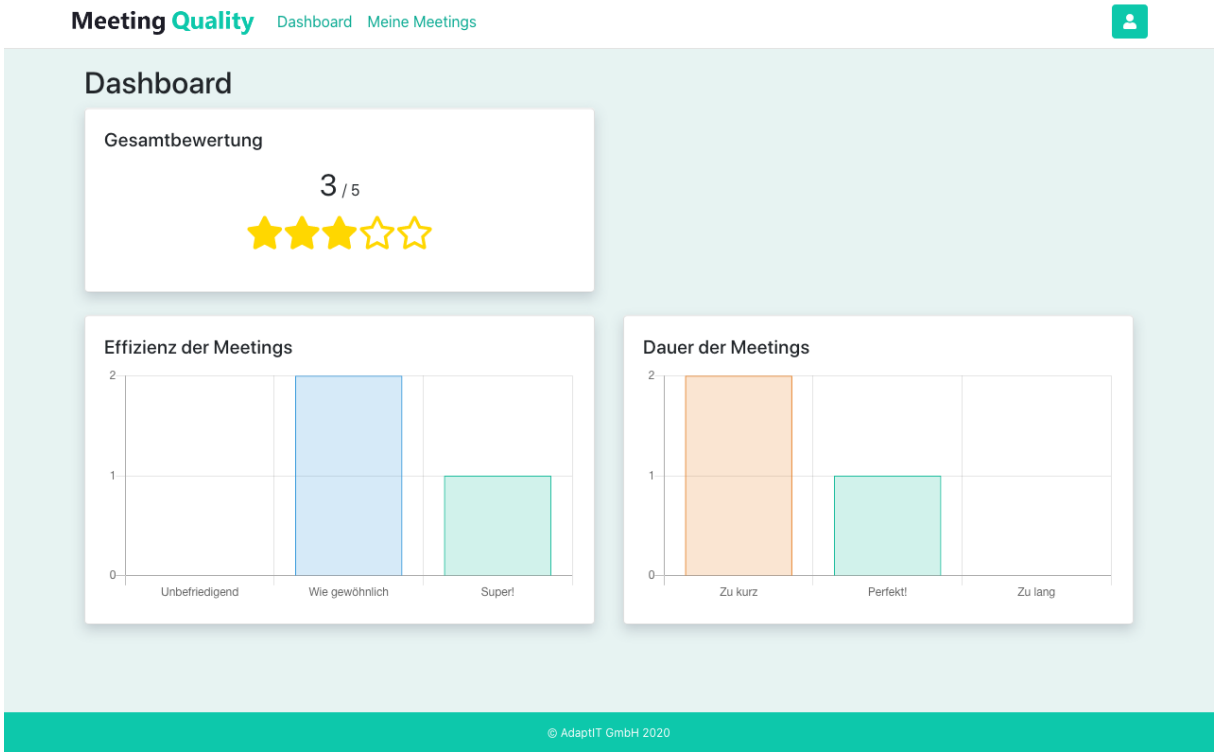


Abbildung 6.1: Dashboard der Meeting-Quality-App (Quelle: eigene Abbildung)

Die Daten dazu werden im Backend aufbereitet, wo eine Aggregation der Feedbacks gemacht wird. Jedem Feedback wird dazu eine Bewertung zugeordnet, das aus der Punktzahl und der maximal zu erreichenden Punktzahl besteht.[1] Durch Verrechnung all dieser Bewertungen mit dem arithmetischen Mittel lässt sich die Gesamtpunktzahl berechnen. Wird diese Zahl anschliessend in einen Prozentwert umgewandelt und mit 5 multipliziert, erhält man so die 5-Sterne-Bewertung. Da diese Zahl beliebig viele Nachkommastellen aufweisen kann, wird sie auf zwei Nachkommastellen gerundet.

Die Umsetzung im Fronten war dagegen vergleichsweise simpel. Die App wurde um eine neue Seite erweitert, in der die Diagramme dargestellt werden. Da die Anwendung nun über mehrere Seiten verfügt, musste sie zusätzlich mit einer Navigation bestückt werden, damit das Dashboard überhaupt aufgerufen werden kann. Die Navigation wurde im Header angesiedelt.

Für die Visualisierung wurde die Diagrammbibliothek *Chart.js*[2] eingesetzt, welche bereits an anderen Stellen im Frontend verwendet wird. Dabei wurde mit verschiedene Diagrammtypen ausprobiert. Durch Inputs des Projektbetreuers und des Industriepartners wurde sich im Endeffekt auf Balkendiagramme verständigt. Aus ihnen können an der Y-Achse sowohl die absoluten Zahlen abgelesen werden und sie machen auch das relative Verhältnis zwischen den Balken sichtbar.

Da es für die Sterne-Bewertung keine vordefinierte Komponente gab, musste diese selbst entwickelt werden. Für die Sterne wurden Icons aus der bereits installierten Icon-Bibliothek *Font Awesome*[3] verwendet. Dies führt aber dazu, dass nur ganze Sterne angezeigt werden können. Im Frontend wird die Punktzahl daher zusätzlich auf die nächste Ganzzahl gerundet.

6.2 Microsoft Teams Integration

Für die Integration mit Microsoft Teams wurde einerseits eine Teams-App erstellt, durch welche die Meeting-Quality-Anwendung in MS Teams verwendet werden kann. Anderseits wurde eine Anbindung an die Graph API umgesetzt, um über neu erstellte und geänderte Besprechungen informiert zu werden.

Zu Beginn der Bachelorarbeit war aber noch unklar, wie die Integration mit MS Teams umgesetzt werden sollte. Denn neben den gewählten Möglichkeiten existiert noch eine Vielzahl an weitere Optionen, um eine App in Microsoft Teams zu integrieren: Bots, Webhooks und Cards, um nur einige davon zu nennen. Um eine Übersicht über all diese Möglichkeiten zu bekommen und so eine Basis für eine fundierte Entscheidung zu schaffen, wurde eine ausführliche Variantenanalyse erstellt, welche die verschiedenen Optionen beleuchtet und miteinander vergleicht.

Die Variantenanalyse ist in Anhang B zu finden.

[1] Die Vergabe der Punkte wurde vom Industriepartner festgelegt und ist in der User Story «Durchschnittliches Rating pro Woche» in Anhang C dokumentiert.

[2] https://www.chartjs.org/

[3] https://fontawesome.com/

6.2.1 Microsoft Teams App

Es wurde mit dem Industriepartner festgelegt, dass für die Integration möglichst wenig am bestehenden Code und der Infrastruktur geändert werden soll. Dies legt eine Integration auf Basis eines persönlichen Tabs nahe: In diesem können via `iframe`[4] beliebige externe Webinhalte dargestellt werden. Die Meeting-Quality-App könnte so also direkt in MS Teams eingebettet werden, ohne dass am Quellcode eine Änderung vorgenommen werden muss. Wichtig ist dabei, dass die einzubettende Seite die Sicherheitsstandards von Microsoft befolgt, wie z. B. eine Transport Layer Security Verbindung mit gültigem Zertifikat, was bei der Meeting-Quality-App bereits der Fall war.

Sowohl die Diplomanden wie auch der Industriepartner haben sich für diese Lösung ausgesprochen. Das Ergebnis ist in Abbildung 6.2 zu sehen.

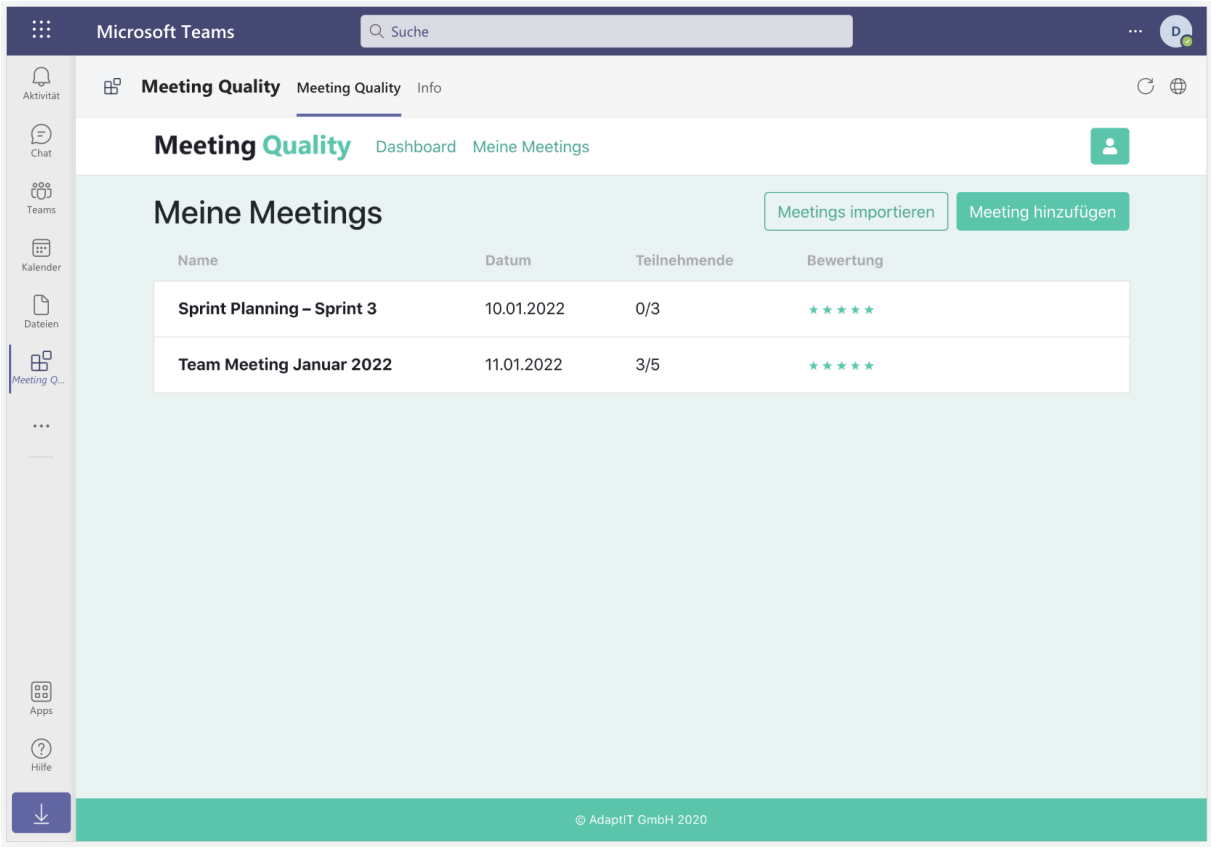


Abbildung 6.2: Meeting-Quality-App in Microsoft Teams (Quelle: eigene Abbildung)

Für die Erstellung der Teams-App wurde das offizielle *Microsoft Teams Framework (TeamsFx)*[5] verwendet. Durch den bereitgestellten Codegenerator konnte innert kürzester Zeit eine funktionierende App gebaut werden. Es musste einzig die Konfiguration in der Manifest-Datei so angepasst werden, dass die Meeting-Quality-App via `https://mq.adaptit.ch` eingebunden wird.

Um die erstellte App lokal testen zu können, musste in MS Teams die Option für *Sideloading* aktiviert werden, ein Prozess für den man zwingend Administrationsberechtigun-

[4] https://developer.mozilla.org/de/docs/Web/HTML/Element/iframe
[5] https://github.com/OfficeDev/TeamsFx

gen benötigt.[30] Dies stellte anfänglich ein Problem dar, da die von der Ostschweizer Fachhochschule (OST) bereitgestellten Microsoft-Accounts diese Voraussetzungen nicht erfüllten. Der Industriepartner konnte jedoch Abhilfe schaffen, indem er den Diplomanden einen Account zur Verfügung stellte.

6.2.2 Graph API

Damit eine App die für API-Abfragen benötigten Access Token lösen können, ist eine vorgängige Registration der App im Acitve Directory von Azure notwendig. Die Registrierung kann über das Azure-Portal[6] vorgenommen werden, bringt aber einige Tücken mit sich. Die Dokumentation von Microsoft ist in diesem Punkt ziemlich zerstreut und es fällt schwer, die richtigen Konfigurationswerte zu finden. Um diesen Prozess für die zukünftige Entwicklung einfacher zu gestalten wurden die wichtigsten Erkentnisse im Entwicklerleitfaden festgehalten, welcher in Anhang D zu finden ist.

Die Implementation des Authentifizierungsprozesses (vgl. Unterabschnitt 5.3.1 Authentifizierung bei der Microsoft Identitätsplattform) im Frontend war unter Benutzung der MSAL-Bibliothek denkbar einfach; die Umsetzung des OBO-Flows im Backend, für welche die Bibliothek keine grosse Hilfestellung bietet, war um einiges kniffliger.

Benutzende können sich nun über das Benutzermenü mit Microsoft verbinden. Beim Klicken auf den Button im Frontend öffnet sich Popup, wo sie sich mit ihrem Microsoft-Account einloggen können.

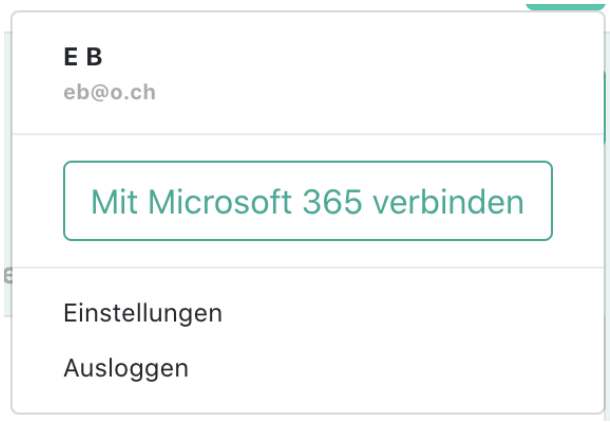


Abbildung 6.3: Button für die Verbindung mit Microsoft 365 (Quelle: eigene Abbildung)

Dadurch wird im Backend aber noch kein Abonnement erstellt. Dieser Prozess wird erst ausgelöst, nachdem auch die Option zum Synchronisieren der Besprechungen in den Einstellungen auswählt wurde.

[6] https://portal.azure.com/

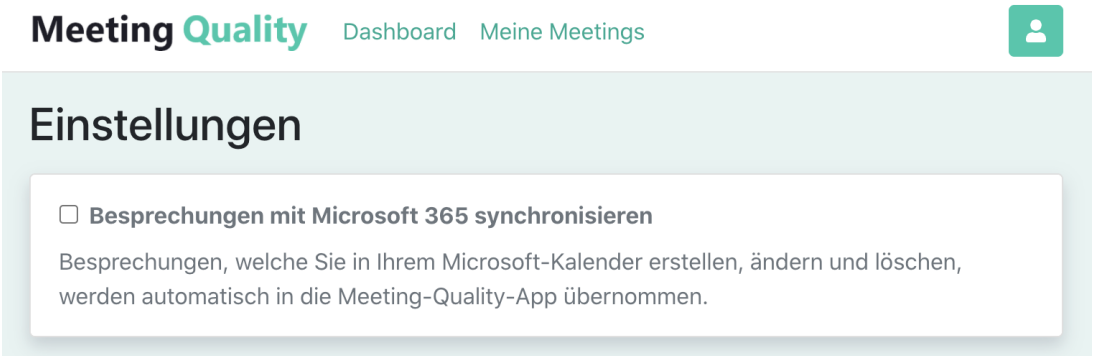


Abbildung 6.4: Einstellungen der Meeting-Quality-App (Quelle: eigene Abbildung)

Ist das Abonnement erfolgreich erstellt, werden dann, sofern Änderungen auf der zugrunde liegenden Ressource passieren, Benachrichtigungen an die konfigurierte Callback-URL gesendet.

6.2.3 Meetings importieren

Da das Abonnement nur Informationen zu neu erstellten Meetings liefert, wurde zusätzlich eine Funktion zum Importieren von älteren Meetings geschaffen. Dazu werden dem/der Benutzer:in in einem Dialog die Besprechungen aus dem Kalender angezeigt, und es können einzelne, oder aber alle Meetings auf einmal in die Meeting-Quality-App importiert werden.

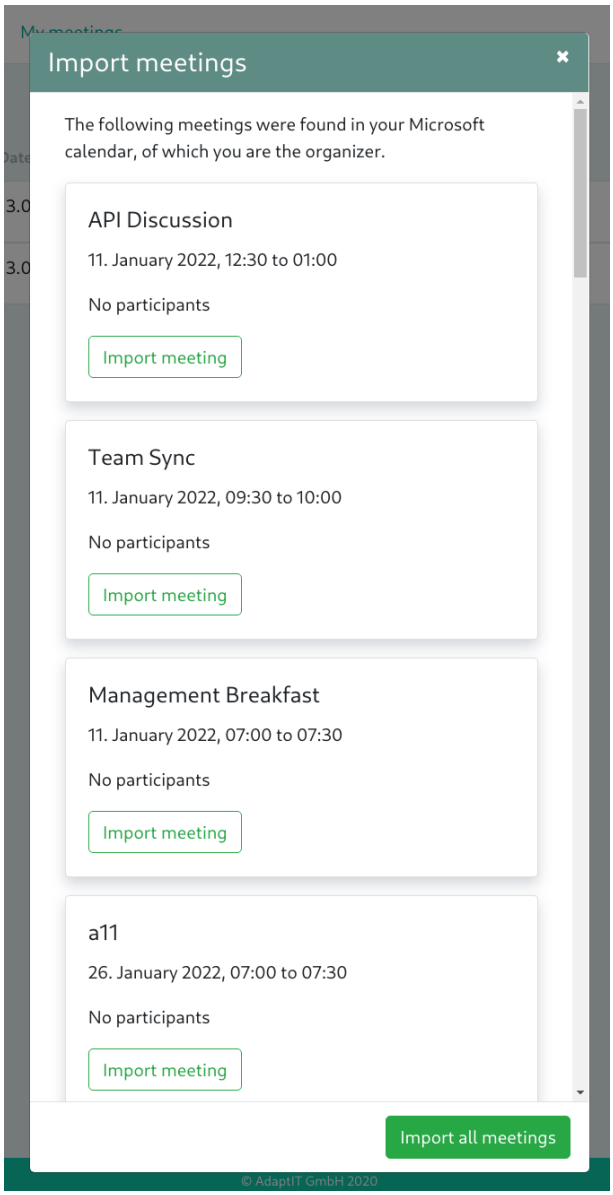


Abbildung 6.5: Dialog zum importieren von Besprechungen (Quelle: eigene Abbildung)

6.3 Automatischer Versand des Meeting-Links

An dieser Stelle sollen die Ergebnisse des automatischen E-Mail Versands diskutiert werden. Zuerst soll der Erfolg der Implementation gezeigt werden. Ein passender Screenshot befindet sich dazu in Abbildung 6.6.

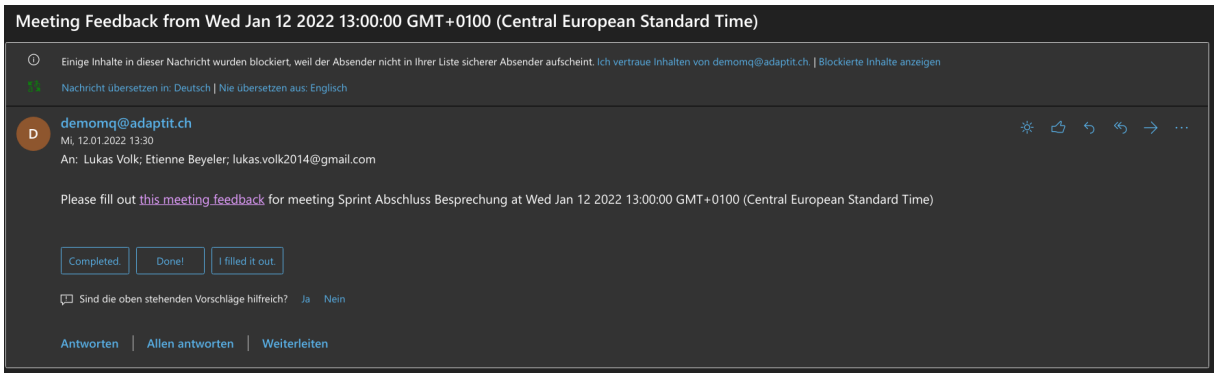


Abbildung 6.6: Beispiel einer versendeten Mail (Quelle: eigene Abbildung)

Beispiel der Email-Entität in der Datenbank (Abbildung 6.7).

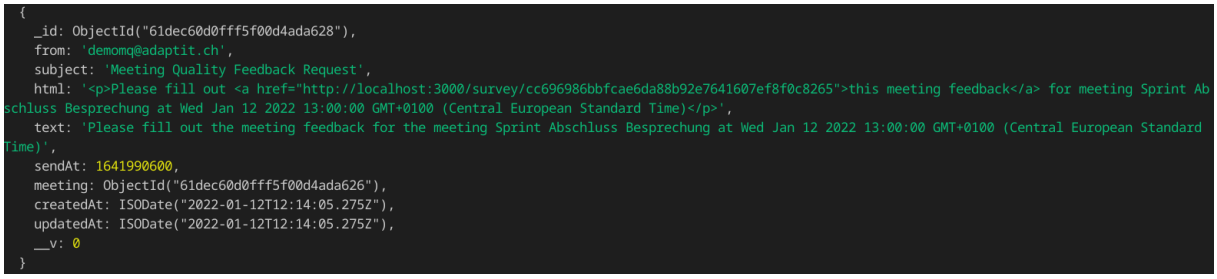


Abbildung 6.7: Erstellte E-Mail-Entität in der Datenbank (Quelle: eigene Abbildung)

Es ist zu erkennen, dass der in der Entität das Subjekt der Mail, der Absender, der Text, sowie das zu sendende HTML abespeichert ist. Weiter verfügt die E-Mail-Entität eine Referenz auf das behandelte Meeting.

Wichtig ist ebenfalls das `sendAt`-Feld, welches den Absendungszeitpunkt bei SendGrid als Unix-Timestamp codiert. So wird erreicht, wie auch in Abbildung 6.6 erkenntlich, dass die Feedback-E-Mail zu dem eingetragenen Endzeitpunkt des Meetings an die Teilnehmenden verschickt wird. Zusätzlich sind in der E-Mail-Entität noch Metadaten enthalten, welche durch MongoDB hinzugefügt werden.

Zum Senden der E-Mail wird die Referenz auf das Meeting aufgelöst und die darin enthaltenen `meetingParticipants` werden als Adressaten hinzugefügt. Um einen näheren Einblick in das Beispiel zu bekommen wird die dazugehörige Meeting-Entität in Abbildung 6.8 dargestellt.

```
{
  _id: ObjectId("61dec60d0fff5f00d4ada626"),
  meetingName: 'Sprint Abschluss Besprechung',
  meetingDate: 'Wed Jan 12 2022 13:00:00 GMT+0100 (Central European Standard Time)',
  meetingEndDate: 'Wed Jan 12 2022 13:30:00 GMT+0100 (Central European Standard Time)',
  meetingLink: 'cc696986bbfcae6da88b92e7641607ef8f0c8265',
  meetingParticipants: [
    'lukas.volk@ost.ch',
    'etienne.beyeler@ost.ch',
    'lukas.volk2014@gmail.com'
  ],
  joinedParticipants: 0,
  feedback: [],
  createdBy: ObjectId("61debc040fff5f00d4ada5f5"),
  createdAt: ISODate("2022-01-12T12:14:05.273Z"),
  updatedAt: ISODate("2022-01-12T12:14:05.273Z"),
  slug: 'sprint-abschluss-besprechung',
  __v: 0
}
```


Abbildung 6.8: Zum Email Beispiel gehörige Meeting-Entität (Quelle: eigene Abbildung)

Man sieht, dass sich der Inhalt der Mail aus den Feldern `meetingNamen` und `meetingDate` ergibt. So soll der Nutzer rasch erkennen, um welches Meeting es sich handelt, zu dem er Feedback geben soll. Der Betreff der Mail ist dabei generisch gewählt, aber aussagekräftig genug bezüglich des Inhaltes.

Schlussendlich kann der/die Nutzer:in auf den in der Mail beinhalteten Link klicken und erreicht das zu dem Meeting gehörende Feedbackformular.

Automatisierung Die weiter benötigte Automatisierung des in Abschnitt 5.3.2 E-Mails versenden beschriebenen Prozesses läuft über eines der bereitgestellten Management-Skripte. Diese werden folgend behandelt.

6.4 Management-Skripte

In der Meeting-Quality-Applikation sind, aufgrund der monolithischen Architektur des Backends Management-Skripte erforderlich. Diese speisen Jobs in den Server ein, um gewisse Funktionalitäten zu erreichen. Das diese erforderlich sind, wurde erst bei der Umsetzung der entsprechenden Anwendungsfälle deutlich. Es benötigte dabei einen externen Mechanismus um die Lebenszyklen bestimmter Entitäten zu verwalten.

Dazu wurde, in Abklärung mit dem Produkteigner definiert, dass auf den mit Ubuntu laufenden Servern der Meeting-Quality-App Cronjobs[7] verwendet werden sollen, welche periodisch aufgerufen werden.

Die Definition eines Cronjobs findet innerhalb des Crontabs statt. Dabei wird eine auf dem Server befindliche Binär-Datei mit einem Schema ausgestattet, welches den Zeitabstand bestimmt nach dem das hinterlegte Programm aufgerufen werden soll. Die Zeitabstände

[7] https://www.ionos.de/digitalguide/hosting/hosting-technik/cronjob/

werden durch die interne Uhr des Servers gemessen. Der zu startende Prozess wird dann nach Auslösen des Jobs vom Prozess Scheduler des Betriebssystems ausgeführt.

6.4.1 Abonnemente

Bei den Abonnementen ist das Problem der Lebenszyklen sehr prägnant. Microsoft bietet keine Benachrichtungen bezüglich des Auslaufens eines Abonnementes an. Nur in ausserordentlichen Fällen, wie der Sperrung eines Accounts, werden die API-Endpunkte von einem Abonnement über sein erlöschen informiert.

Da Microsoft diese Aufgabe nicht übernimmt, muss ein Prozess geschaffen werden, in welchem überprüft wird ob ein Abonnement demnächst ausläuft. Falls ja muss dieses erneuert und neu in die Datenbank eingespiesen werden, wenn nicht kann das Skript die Entität ignorieren.

Dieser Ablauf wird durch einen serverseitigen Job durchgeführt. Ungefähr einmal am Tag wird das Skript, ebenfalls durch einen Cronjob, aktiviert. Der genaue Ablauf des Prozesses wird in Abschnitt 5.3.2 Graph-Abonnemente erneuern beschrieben.

Ein Beispielbild eines erfolgreichen Ablaufs des Skripts findet sich in Abbildung 6.9.

```
> meetingquality-management@1.0.0 renew
> node subscription-lifecyclemgmt.js

1642040990950 subscription renewal job started
Found #subscriptions to renew: 1
1642040994429 subscription renewal job finished
```

Abbildung 6.9: Beispielbild eines erfolgreichen Laufs des Abonnement Management-Skripts

6.4.2 E-Mail

Bei der E-Mail-Verwaltung ist das Problem ebenso nicht ohne einen zusätzlichen Job lösbar. Das liegt daran, dass die E-Mails zwar bei Meetingerstellung in die SendGrid-API geschoben werden könnten, aber danach ausserhalb der Kontrolle der Meeting-Quality-App ist. Das würde bedeuten, dass wenn ein Meeting verschoben oder gelöscht wird, SendGrid den Teilnehmenden den Fragebogen trotzdem zum ursprünglichen Zeitpunkt zukommen lässt. Das ist mit den Businessaspekt der Anwendungen nicht vereinbar und muss daher gehandhabt werden.

Es wurde somit in Übereinstimmung mit dem Kunden entschieden, dass die E-Mails ebenfalls per Cronjob versendet werden. Die entsprechend ablaufende Sequenz ist in Abschnitt 5.3.2 E-Mails versenden beschrieben.

Daraus ergibt sich die Limitation, dass die E-Mails möglicherweise nicht exakt zu dem Endzeitpunkt eines Meetings versendet werden, sondern verschoben in einem Intervall von bis zu 15 Minuten.

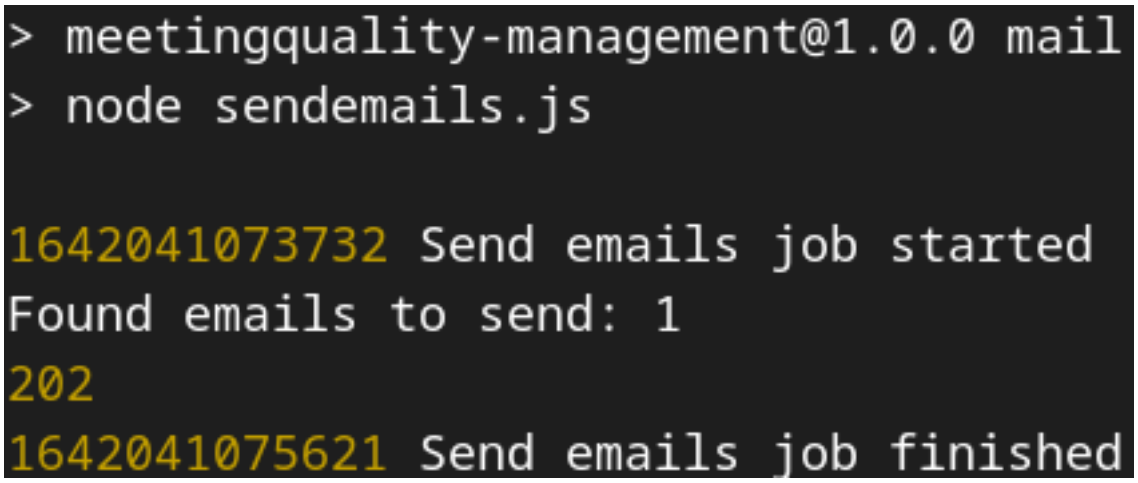


Abbildung 6.10: Erfolgreiches durchlaufen des E-Mail versenden Jobs (Quelle: eigene Abbildung)

In Abbildung 6.10 befindet sich die Konsolen-Ausgabe eines erfolgreiche durchgelaufenen Jobs zum Versenden der E-Mails. Dabei wird der Start- und Endzeitpunkt des Jobs geloggt. Die Zahl bzw. die Zahlen zwischen der Start- und Endnachricht sind die HTTP-Status-Codes der Sendgrid-API. Der Status Code 202 stellt dabei `Accepted` dar. Das heisst, dass die Anfrage mit den beinhalteten Formdaten von SendGrid angenommen wurde und zum Versendungszeitpunkt an die Adressaten verschickt werden wird. Eine solche Beispiel-E-Mail ist in Abbildung 6.6 zu sehen. Die Kommandos zum Starten des Skriptes sind im Entwicklerleitfaden (Anhang D) festgehalten.

6.5 Verbesserungen und Refactorings

Ein wesentlicher Teil der Arbeit waren umfangreichen Refactorings. Es wurde ein erheblicher Aufwand investiert, um die Software besser erweiterbar und eleganter zu gestalten. Die Elimination von Technische Schuld und Verbesserung der Projektstruktur erhöhen die Erfolgsaussichten zukünftiger Erweiterungen.

Qualitativ hochwertigen Code zu schreiben, der sich gut liest und somit einfach verständlich ist, war ein Anliegen der Diplomanden. Um diesen Zweck zu erreichen und den Software-Engineering-Kriterien gerechter zu werden, wurden folgende Massnahmen unternommen.

6.5.1 Umstellung auf Typescript

Der gesamte Applikationcode wurde schrittweise in Typescript umgeschrieben. Das führt zu einer höheren Resistenz und Lesbarkeit der Applikation. Weiter verbessert sich dadurch die Entwicklungsgeschwindigkeit für die Applikation durch statische Typenüberprüfung und mehr verfügbaren Entwicklungswerkzeugen.

6.5.2 ESLint mit Airbnb Style Guide

Im Front- und Backend wurde ESLint für die statische Codeanalyse hinzugefügt. Dadurch besteht die Möglichkeit, automatisch Fehler im Projekt zu erkennen und zu beheben. Als

Konfigurationsgrundlage dazu dient der populäre *Airbnb JavaScript Style Guide*[8].

6.5.3 Frontend

Es wurden etliche Mängel der Applikation behoben. In der Regel handelt es sich um Dinge die den Diplomanden direkt aufgefallen sind. Diese betrafen vor allem die React-Komponenten und waren Technische Schuld, die früher oder später sowieso hätten behoben werden müssen.

React-Komponenten

Die Page-Strukturen des Frontends waren teilweise komplett entgegen der React-Philosophie entwickelt. Die Verfehlung dabei war vor allem, das keine Komponentenhierarchie aufgebaut wurde[19]. So mussten zahlreiche Komponenten aufgeteilt und wieder zusammengefügt werden. Dies dient dazu die Komponenten besser Wiederzuverwenden oder Austauschen zu können. Weiter war die Logik vollkommen in die Komponenten integriert. Diese beiden Aspekte führen zu einer hohen Kopplung und geringen Kohäsion der Software. Das bringt die Applikation irgendwann an den Punkt, dass keine neuen Features implementiert werden können und jedwede Änderungen zu Fehlern innerhalb der Applikation führen[24].

Meeting-Aktionen aus der Listenansicht

Neu besteht die Möglichkeit, den Link für Meetings auch direkt aus der Listenansicht in die Zwischenablage zu kopieren, ohne dass dazu erst die Detailansicht der Besprechung geöffnet werden muss. Hier kann das Meeting auch gleich gelöscht werden. Die Summe dieser vermeintliche kleinen Anpassung macht die Meeting-Quality-App deutlich benutzerfreundlicher.

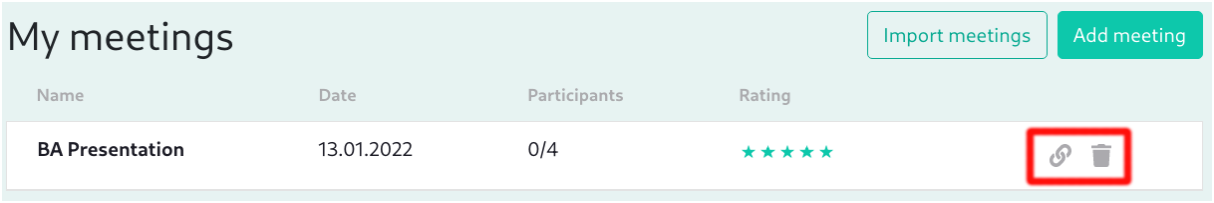


Abbildung 6.11: Beispiel Bild der Aktionsbuttons (Quelle: eigene Abbildung)

Dialog zum Erstellen von Meetings

Der Dialog zum Erstellen von neuen Meetings wurde komplett refactored und erweitert. So wird neu nicht nur die Anzahl der Meetingteilnehmenden gespeichert werden, sondern es können die E-Mail-Adressen erfasst werden. Dies ist auch Voraussetzung für den automatischen Versand des Fragebogens per Mail (vgl. Abschnitt 5.3.2 E-Mails versenden).

[8] https://airbnb.io/javascript/react/

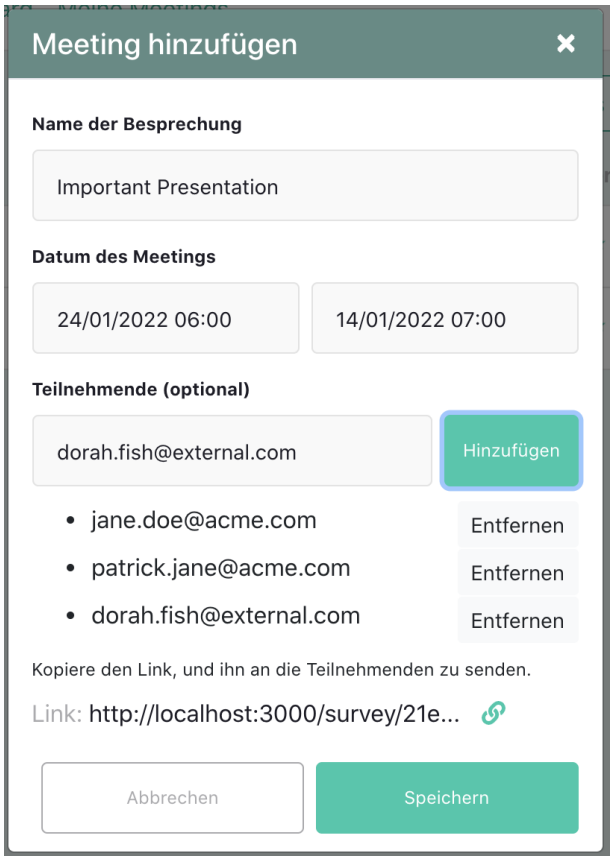


Abbildung 6.12: Beispiel Bild des Meeting Erstellungsdialog (Quelle: eigene Abbildung)

6.5.4 Backend

Die diversen Probleme im Backend, auf welche bereits in anderen Kapitel Bezug genommen wurde, konnten teilweise beseitigt werden. Mit Einführung der dreischichtige Architektur wurde ein wichtiger Schritt Richtung lockere Kopplung gemacht. Es konnten aber nicht alle Code Smells bezüglich *Bloaters* eliminiert werden, und auch das Überführen vm funktionalen in einen mehr objektorientierten Ansatz dauert noch an. Dennoch konnten einige der Technische Schuld adressiert werden und die Code-Basis wurde signifikant aufgeräumt.

Daneben wurden etliche unbenützte Codestellen und auskommentierter Code entfernt.

6.5.5 Continous Integration

Unter Benutzung der *GitLab-CI*[9] wurde im Frontend und im Backend eine Continous Integration (CI) eingerichtet, welche aus den zwei Phasen *Build* und *Test* besteht.

In der Build-Phase wird dabei überprüft, ob sich der Code erfolgreich kompilieren lässt. In dieser Phase wird auch der Linter ausgeführt. In der Test-Phase werden dann, wie der Name bereits impliziert, die Software-Tests ausgeführt.

[9] https://docs.gitlab.com/ee/ci/

Kapitel 7

Schlussfolgerungen

7.1 Bewertung der Ergebnisse

Die Mindestanforderungen, welche in der Aufgabenstellung definiert wurden, konnten vollumfänglich umgesetzt werden. Die restlich Zeit wurde dann vor allem dafür aufgewendet, die bestehende Anwendung zu verbessern. Die Investitionen in gutes Software Engineering haben sich gelohnt, die Architektur und Codebasis konnte erheblich verbessert werden.

7.1.1 Microsoft Integration

Die Abonnemente stellen eine gute Lösung dar, sind allerdings auch mit Problemen behaftet. Es werden nur limitiert Notifikationen bezüglich der Lebensdauer der bestehenden Abonnemente kommuniziert. Teile der Graph API, des Azure Active Directory und das Teams Toolkit befinden sich ausserdem noch in Beta-Version. Es besteht durchaus die Möglichkeit, dass hier aufseiten von Microsoft noch weitere Verbesserungen folgen.

Es ist ausserdem fraglich, ob das Abspeichern der OBO-Token in der Datenbank die beste Lösung ist, um den langfristigen Zugriff auf Abonnemente zu gewährleisten. Unter Umständen emergieren hier noch geeignetere Vorgehensweisen, vor allem durch eine bessere Unterstützung des Verfahrens seitens der Microsoft Bibliotheken.

7.1.2 Dashboard

Das Dashboard bietet eine gute Übersicht über die wichtigsten Informationen, die bis jetzt über den Fragebogen erhoben werden. Das Dashboard ist modular gestaltet und lässt sich beliebig erweitern.

Die Berechnung und Gewichtung für die Gesamtbewertung ist aktuell noch weitestgehend trivial. Für eine noch aussagekräftigere Bewertung könnten hierzu vertiefte Studien angefertigt werden.

7.1.3 E-Mail-Versand

Die bestehende Lösung zum Versand der E-Mails ist funktional in Ordnung und logistisch vertretbar. Um künftig auf die Management-Skripte verzichten zu können müssten grundlegende Architekturänderungen getätigt werden.

Hierbei besteht zudem die Möglichkeit das ein anderer Email-Service-Provider einen grösseren Zugang zu seinen Interna bietet und damit für Änderungen einer Mail nach verschieben in die respektive API zulässt.

Die E-Mail mit dem Feedback-Link enthält zwar alle Informationen, ist aber optisch ziemlich unspektakulär. Eine Personalisierung der E-Mail-Vorlage würde aus Marketingsicht sicherlich Sinn machen.

7.1.4 Management-Skripte

Diese Lösung bringt ihre eigenen Limitationen mit sich, wobei sich vor allem bei der Skalierung der Meeting-Quality-Applikation Probleme ergeben können. Die Skripte zielen direkt auf die Datenbank ab, das heisst, das bei grossen Datensätzen die Query auf Seite der Datenbank möglicherweise lange dauert. Auch ein Rate-Limiting aufseiten SendGrids oder Microsofts könnte zu einem Problem werden, wenn gleichzeitig eine grosse Anzahl an Anfragen an deren APIs gesendet werden.

7.1.5 Software Engineering

Das Software Engineering der Meeting-Quality-App sollte nun auf einem guten Stand sein, um eine zügige Weiterentwicklung zu ermöglichen. Wie im vorherigen Abschnitten schon erwähnt wurde sind jedoch noch nicht alle angedachten Refactorings komplett abgeschlossen. Das liegt auch in der Sache der Natur: Wenn Applikation, die weiterentwickelt wird, unterliegt ständigen Veränderungen. Demnach muss auch das darunterliegende Software-Engineering stetig weiterentwickelt werden.

7.1.6 Testing

Aufgrund des limitierten Zeitrahmens und den zahlreichen Feature-Requests mussten Abstriche in puncto Testing gemacht werden. Obwohl den Diplomanden bewusst ist, dass ein umfangreiches Testing das A und O für eine solide Applikation ist, haben sie die Buisness-Aspekte höher priorisiert als ein umfassendes Testing, wie auch von dem Projektpartner verlangt.

7.2 Ausblick

7.2.1 Mögliche Funktionserweiterungen

Mögliche Erweiterungen im Funktionsumfang sind bereits in der Aufgabenstellung[22] enthalten und teilweise auch schon in den Use Cases behandelt worden (vgl.Abschnitt 3.2 Anwendungsfälle). Sie werden daher an dieser Stelle nicht nochmals wiederholt.

Microsoft Teams App

Die Teams-App lässt sich durch diverse weitere, sinnvolle Features ergänzen. Eine mögliche Auflistung davon findet sich in der Variantenanalyse in Anhang B.
So könnte bspw. ein Chat-Bot hinzugefügt werden, um eine sprachbasierte Interaktionen mit den Benutzer:innen anzubieten, oder die Meeting-Quality-App via Meeting-Erweiterungen direkt in den Online-Meetings verfügbar gemacht werden.

Integration in weitere Plattformen

Auch Zoom bietet die Möglichkeit, eigene Apps via Marketplace einzubinden.[34], was die Meeting-Quality-App einem noch breiteren Publikum zugänglich machen würde.

Newsletter

Viel Potenzial hat der personalisierte Newsletter: Bei der Verarbeitung der zu sendenden Tipps und Tricks könnte z. B. ein KI-System zur Anwendung kommen, welches die Meetings und Feedbacks in das Trainingsmodell miteinbezieht.

7.2.2 Automatisiertes Testing

Die Applikation wurde im Laufe der Arbeit grösstenteils manuell getestet. Es existiert nun zwar eine CI-Pipeline, die Testsabdeckung der Applikation ist aber noch unzureichend, um eine stabile Funktionalität der Applikation garantieren zu können.

Spätestens wenn die MQ-App eine signifikante Anzahl an Nutzer:innen besitzt, ist ein umfangreiches Testing unabdingbar, um Beeinträchtigungen am System zu einem gewissen Grad ausschliessen zu können.

7.2.3 Automatisiertes Deployment

Auch ein Continuous Delivery (CD) kann in Betracht gezogen werden, um neue Funktionen zukünftig automatisch zu deployen und so den manuellen Aufwand weiter zu minimieren. Es ist jedoch selbstredend, dass dafür zuerst die Bedingungen eines flächendeckenden Testings erfüllt werden müssen, damit sich keine Regressionen einschleichen.

7.2.4 Microservices

Eine weiter Entkopplung des Systems könnte mit einer Microservice-basierten Architektur[1] erreicht werden. Die Microservices könnten dann mittels Message-Passing effektiv miteinander kommunizieren. Mittels einem Tool wie Kubernetes[2] wäre so eine automatisch Skalierung der Applikation sehr einfach umzusetzen nach Bedarf.

7.3 Schlusswort

Obwohl es in der Meeting-Quality-App noch etliche offene Baustellen gibt, – sei dies auf Funktions- oder auf Code-Ebene – trifft die Idee den Zahn der Zeit wie keine andere. Online-Meetings waren nicht nur für diese Bachelorarbeit das primäre Kommunikationstool, sondern sind auch fixer Bestandteil der modernen Geschäftswelt, sei dies aufgrund der immer weiter verbreiteten globalisierten Arbeitsweise oder durch andere, wie die aktuellen pandemischen Umständen.

All dies verdeutlicht die Not für ein Tool zur Verbesserung der Besprechungsqualität. Selbst wenn viele Chef:innen der Meinung sind, dass ihre Mitarbeiter:innen im Homeoffice trotzdem weiter produktiv bleiben[13], stehen sie einer Lösung wie der Meeting-Quality-App bestimmt offen gegenüber.

[1] https://martinfowler.com/articles/microservices.html
[2] https://kubernetes.io/

Glossar

API Application Programming Interface. *siehe* Application Programming Interface

Application Programming Interface Ein Programmierschnittstelle (von englisch *Application Programming Interface*) ist eine digitale Schnittstelle zwischen zwei Softwaresystemen und erlaubt deren Anbindung zueinander. 72

BSON Binary JSON. 10

CD Continuous Delivery. 71

CI Continous Integration. 68, 71

CORS Cross-Origin Resource Sharing. 14

DAOs Data Access Objects. 43, 55

Droplet Eine Linux-basierte virtuelle Maschinen (VMs), die auf virtualisierter Hardware laufen. Jedes neu erstellte Droplet ist ein neuer Server, der entweder eigenständig oder als Teil einer grösseren, cloudbasierten Infrastruktur genutzt werden kann[9]. 8

Graph Explorer Von Microsoft bereitgestellte Online-Konsole um auf die Microsoft Graph API zuzugreifen. 40, 41, 76

HATEOAS Hypermedia as the Engine of Application State. 40

JSON JavaScript Object Notation. 10, 72

JWT JSON Web Token. 13, 43

KI Künstliche Intelligenz. 71

MSAL Microsoft Authentication Library. 47

MVP Minimum Viable Product. 3, 5

NoSQL-Datenbank Als NoSQL-Datenbank werden in der Regel jegliche nicht relationale Datenbank bezeichnet. 10, 12

npm Node Package Manager. 11

OBO On Behalf Of. 44, 60, 69

OData Open Data Protocol. 40

ODM Objektdatenmodellierung. 12

OST Ostschweizer Fachhochschule. 60

SPA Single-page Application. 9, 13

Technische Schuld Beschreibt die technischen Mängel in einer bestehender Software, welche sich durch kontinuierlicher Weiterentwicklung ansammelt. 3, 66–68

Transport Layer Security Ein im OSI-Modell auf Transport Layer ansässiges Protokoll, um eine kryptografisch schichere Webkommunikation zu gewährleisten. 9, 59

XSS Cross-Site-Scripting. 14

Literaturverzeichnis

[1] Dan Abramov und die Autoren der Redux-Dokumentation. *Redux*. URL: https://redux.js.org/. (besucht am 05.10.2021).

[2] Dan Abramov und die Autoren der Redux-Dokumentation. *Redux Style Guide*. URL: https://redux.js.org/style-guide/style-guide. (besucht am 13.01.2022).

[3] Axios. *Axios*. URL: https://axios-http.com/. (besucht am 10.01.2022).

[4] Etienne Beyeler und Lukas Volk. *Meeting Quality - Projektplan*.

[5] Bootstrap. *Bootstrap*. URL: https://getbootstrap.com/. (besucht am 10.01.2022).

[6] Ronald Caldwell. *What is Formik?* URL: https://www.liquidweb.com/kb/formik-react/. (besucht am 10.01.2022).

[7] Alistair Cockburn. *Writing Effective Use Cases*. 1. Aufl. Addison-Wesley Professional, 2000.

[8] OAuth 2.0 Community. *OAuth 2.0*. URL: https://oauth.net/2/. (besucht am 13.01.2022).

[9] DigitalOcean. *Droplets*. URL: https://docs.digitalocean.com/products/droplets/. (besucht am 05.10.2021).

[10] LLC DigitalOcean. *SOLID: The First 5 Principles of Object Oriented Design*. URL: https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design. (besucht am 13.01.2022).

[11] Duden. *Besprechung, die*. URL: https://www.duden.de/rechtschreibung/Besprechung. (besucht am 12.01.2022).

[12] Duden. *Stakeholder, der*. URL: https://www.duden.de/rechtschreibung/Stakeholder. (besucht am 11.10.2021).

[13] Philippe Erath. *Chefinnen und Chefs glauben an Produktivität zu Hause*. URL: https://www.srf.ch/kultur/gesellschaft-religion/neue-homeoffice-studie-chefinnen-und-chefs-glauben-an-produktivitaet-zu-hause. (besucht am 12.01.2022).

[14] Inc Facebook. *Jest*. URL: https://jestjs.io/. (besucht am 12.01.2022).

[15] Martin Fowler. *Richardson Maturity Model*. URL: https://martinfowler.com/articles/richardsonMaturityModel.html. (besucht am 12.01.2022).

[16] Auth0 Inc. *Which OAuth 2.0 Flow Should I Use?* URL: https://auth0.com/docs/get-started/authentication-and-authorization-flow/which-oauth-2-0-flow-should-i-use. (besucht am 11.01.2022).

[17] AWS Inc. *Two Pizza Teams*. URL: https://docs.aws.amazon.com/whitepapers/latest/introduction-devops-aws/two-pizza-teams.html. (besucht am 10.01.2022).

[18] Facebook Inc. *React*. URL: https://reactjs.org/. (besucht am 05.10.2021).

[19] Facebook Inc. *Thinking in React*. URL: https://reactjs.org/docs/thinking-in-react.html. (besucht am 12.01.2022).
[20] MongoDB Inc. *MERN Stack Explained*. URL: https://www.mongodb.com/mern-stack. (besucht am 10.01.2022).
[21] Nick Karnik. *Introduction to Mongoose for MongoDB*. URL: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/. (besucht am 06.10.2021).
[22] Frank Koch. *Aufgabenstellung Bachelor Thesis «Meeting Quality»*.
[23] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3. Aufl. Prentice Hall PTR, 2004.
[24] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship 1st Edition*. 1. Aufl. Pearson Education, 2008.
[25] Microsoft. *Create subscription*. URL: https://docs.microsoft.com/en-gb/graph/api/subscription-post-subscriptions. (besucht am 13.01.2022).
[26] Microsoft. *event resource type*. URL: https://docs.microsoft.com/en-us/graph/api/resources/event. (besucht am 13.01.2022).
[27] Microsoft. *Get a user*. URL: https://docs.microsoft.com/en-us/graph/api/user-get. (besucht am 13.01.2022).
[28] Microsoft. *Microsoft identity platform and OAuth 2.0 On-Behalf-Of flow*. URL: https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-on-behalf-of-flow. (besucht am 13.01.2022).
[29] Microsoft. *Overview of the Microsoft Authentication Library (MSAL)*. URL: https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-overview. (besucht am 13.01.2022).
[30] Microsoft. *Upload your app in Microsoft Teams*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/apps-upload. (besucht am 12.01.2022).
[31] Prettier. *Prettier*. URL: https://prettier.io/. (besucht am 10.01.2022).
[32] Refactoring.Guru. *Code Smells*. URL: https://refactoring.guru/refactoring/smells. (besucht am 13.01.2022).
[33] SendGrid. *Why SendGrid*. URL: https://sendgrid.com/why-sendgrid/. (besucht am 12.01.2022).
[34] Inc Zoom Video Communications. *Build an App*. URL: https://marketplace.zoom.us/docs/guides/build. (besucht am 14.01.2022).

Abbildungsverzeichnis

Tabellenverzeichnis

Teil II

Anhang

Anhang A

Aufgabenstellung

OST
Ostschweizer
Fachhochschule

Aufgabenstellung Bachelor Thesis „Meeting Quality“

Autor: Frank Koch

Version: 3.0

Anpasst am: 07. Oktober 2021

1. Beteiligte Personen

Diplomanden: Lukas Volk, Etienne Beyeler

Partner: AdaptIT GmbH, Michael Güntensperger

Betreuer: Prof. Frank Koch

Experte: Prof. Hansjörg Huser

2. Problembeschrieb

Pandemiebedingt wurden Online Meetings zum Normalfall. Dadurch entfallen die gemeinsamen Kaffeepausen, in denen man die Meeting-Themen oft noch informell besprach und Feedback austauschte. Die Effizienz der Meetings kann unter den neuen Bedingungen nur noch erahnt werden.

Es wurde bereit ein erstes MVP entwickelt um flankierend zu einem Online-Meeting Feedbacks einzuholen. Dieses wurde mit JavaScript, React.js und Node.js entwickelt und läuft bei Digital Ocean. Es kann unter folgendem Link getestet werden: https://mq.adaptit.ch/login.

3. Aufgabenstellung

Grundlage dieser Aufgabenstellung ist der für das HS21 gültige Leitfaden für Bachelor- und Studienarbeiten[1].

Ziel der Arbeit ist ein Tool zur Auswertung von Meetings weiterzuentwickeln und dem Meeting-Verantwortlichen ein Feedback und Verbesserungsvorschläge anzubieten. Das Tool soll so in das Online-Meeting eingebunden werden, dass die Teilnehmenden die Auswertungen mit möglichst wenig Aufwand starten und abgeben können.

Funktionale Anforderungen

- Integration in MS Teams und/oder Zoom
- Dashboard für Meeting-Verantwortlichen
- Fragebogen zur Bewertung von Meetings sollen im Anschluss an das Meeting automatisch an die Teilnehmenden versendet werden

Optionale Anforderungen

Die Applikation kann um eine Vielzahl an Funktionen erweitert werden. Soweit genügend Zeit vorhanden ist, können zum Beispiel eine Auswahl der folgenden Features umgesetzt werden:

- Managen von Mitarbeitenden in einer Firma als User
- Bereitstellung von kollaborativen Traktanden vor dem Meeting
- Generieren von Terminvorschlägen in Abhängigkeit von den Kalendern der Teilnehmenden
- Bereitstellung von Informationen der Meeting-Teilnehmenden (z.B. von LinkedIn)

[1] https://teams.microsoft.com/l/file/A7522E3F-6931-46F7-A965-B2A9081E71EF?tenantId=a6e70fa3-1c7a-4aa2-a25e-836eea52ca22&fileType=pdf&objectUrl=https%3A%2F%2Fostch.sharepoint.com%2Fteams%2FTS-StudiengangInformatik%2FFreigegebene%20Dokumente%2FStudieninformationen%2FLeitfaden%20BA%20SA%20v1.0.pdf&baseUrl=https%3A%2F%2Fostch.sharepoint.com%2Fteams%2FTS-StudiengangInformatik&serviceName=teams&threadId=19:88ac24bbee8e4682a73e81839cd2103c@thread.tacv2&groupId=4ad37fbb-4c36-4982-8bb5-971b8a539d2d

- Möglichkeit während des Meetings ein Protokoll kollaborativ zu editieren, evtl auch als strukturiertes Ergebnis-Protokoll (was, wer, wann)
- Knowledge Base um Meeting-Verbesserungsvorschläge zu erstellen

Nicht-Funktionale Anforderungen

Functionality

- Für Entwicklung und Betrieb wird die cloud-basierte Umgebung genutzt
- Die Funktionalität der Applikation ist mittels Tests zu beweisen

Usability

- Die Web-Applikation soll auf Desktop- und Mobilgeräten verwendbar sein
- Die Applikation soll übersetzbar sein

4. Zur Durchführung

Mit dem Betreuer finden Besprechungen gemäss Absprache statt. Die Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben.

5. Dokumentation und Abgabe

Siehe Leitfaden für Bachelor- und Studienarbeiten Abschnitt 5.5.

6. Termine

Es gelten folgende Termine:

- 20.09.2021 Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch den Betreuer/die Betreuerin.
- bis 10.01.2022 Die Studierenden erfassen den Abstract im Online Tool http://abstract.rj.ost.ch/ und geben den Abstract zur Kontrolle an ihren Betreuer/ihre Betreuerin frei. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen auf Teams zur Verfügung.
- bis 12.01.2022 Der Betreuer/die Betreuerin gibt das Dokument mit dem korrekten und vollständigen Abstract der Broschüre zur Weiterverarbeitung an das Studiengangsekretariat frei.
- **14.01.2022 Abgabe des Berichts an den Betreuer/die Betreuerin bis 17.00 Uhr**
- 14.01.2022 Hochladen aller Dokumente auf https://archiv-i.hsr.ch/Overview/All bis 17 Uhr
- bis 11.02.2022 Mündliche BA-Prüfung
- bis 13.02.2022 Abgabe BA-Note

7. Bewertung

- Organisation und Durchführung: 10%
- Formale Qualität des Berichts: 10%
- Analyse, Entwurf und Auswertung: 20%
- Technische Umsetzung: 40%
- Bachelorprüfung: 20%

Siehe auch Leitfaden für Bachelor- und Studienarbeiten Abschnitt 6.5.

Rapperswil, den 28.09.21

Frank Koch

Anhang B

Variantenanalyse

Microsoft Teams Integration – Variantenstudie

Etienne Beyeler, Lukas Volk

7. November 2021

Inhaltsverzeichnis

1 Einführung

Im Zuge der Bachelorarbeit «Meeting Quality – Improve Online Meetings» soll die bestehende Anwendung «Meeting Quality» in die Kommunikations- und Kollaborationsplattform *Microsoft Teams* integriert werden. Dazu bietet Microsoft (MS) die Möglichkeit, MS Teams mittels einer *Microsoft Teams Apps* zu erweitern.

Teams-Apps bestehen aus einer Kombination von verschiedenen Funktionen und können auf vielfältige Arten und Weisen in MS Teams eingebunden werden. Die verschiedenen

Möglichkeiten werden im vorliegenden Dokument genauer erörtert und ihre Anwendbarkeit im Bezug auf die Meeting-Quality-App diskutiert, um im Anschluss mit dem Industriepartner einen geeigneten Ansatz für die Umsetzung auszuwählen.

2 Microsoft Teams Apps

In diesem Abschnitt werden die Grundprinzipien einer Microsoft Teams App aufgezeigt. Die Inhalte dazu basieren auf dem Lernmodul «Introduction to building apps for Microsoft Teams»[6] von Microsoft.

Eine auf der Microsoft Teams-Plattform entwickelte App erweitert den Teams-Client (Web, Mobile und Desktop) um selbst gehostete Webservices. Die Teams-Plattform bietet dafür ein reichhaltiges und flexibles Angebot an Erweiterungsmöglichkeiten, UI-Konstruktionen und APIs an, die bei der Entwicklung der App genutzt werden können.

Teams-Apps können dabei beliebig einfach oder komplex sein: Von der Einbettung einer bestehenden Website in eine Registerkarte bis hin zu einer voll funktionsfähigen, facettenreichen App mit mehreren Oberflächen, die Chatbots, natürliche Sprachverarbeitung und eingebettete Web-Erlebnisse beinhalten, ist alles denkbar. Apps können für eine Einzelperson, ein Team oder eine Organisation erstellt, oder aber im App Store veröffentlichen werden, damit sie von allen genutzt werden kann.

2.1 Bestandteile einer Microsoft Teams App

Eine Teams-App lässt sich in drei Hauptbestandteilen zerlegen:

- Den **Microsoft Teams-Client**, welcher die Erweiterungspunkte und UI-Elemente bietet, die durch die App verwendet werden.
- Das **App-Paket**, welches in MS Teams installiert wird. Es enthält neben den Icons ein Manifest im JSON-Format. Dieses beinhaltet die Metadaten der App, die verwendeten Erweiterungspunkte sowie Verweise auf die Webservices, mit welcher die Anwendung kommuniziert.
- Die eigenen **Webservices**, welche die APIs und Logik für die Anwendung bereitstellen.

Es ist zu beachten, dass die Microsoft Teams-Plattform kein Hosting-Dienst ist; die Webservices müssen selbst gehostet und per HTTPS über das Internet zugänglich sein.

3 Funktionen

Teams-Apps haben verschiedene Möglichkeiten, um den Teams-Client zu erweitern, welche von Microsoft zusammenfassend Funktionen (en. *capabilities)* genannt werden. Eine App benutzt eine oder mehrere dieser Funktionen:[11]

- Tabs
- Bots
- Messaging-Erweiterungen (en. *Messaging extensions*)

- Webhooks und Konnektoren (en. *Webhooks and connectors*)
- Meeting-Erweiterungen (en. *Meeting extensions)*

Nebst diesen Kernfunktionen kann eine App auch erweiterte Funktionen, wie die Microsoft Graph-API für Teams, nutzen.

Die folgende Abbildung gibt eine Vorstellung davon, welche Möglichkeiten die Funktionen einer Teams-App bieten:[11]

Abbildung 1: Übersicht der Kernfunktionen für MS Teams-App (Quelle: Microsoft)

3.1 Tabs

Tabs sind in MS Teams eingebettete, für Teams geeignete Webseiten. Technisch werden die Websiten dabei mittels HTML-`<iframe>`-Tag eingebunden, welche auf im App-Manifest deklarierte Domänen verweisen. Tabs können in Kanälen innerhalb eines Teams, in (Gruppen-)Chats oder in persönlichen Apps hinzugefügt werden.[9]

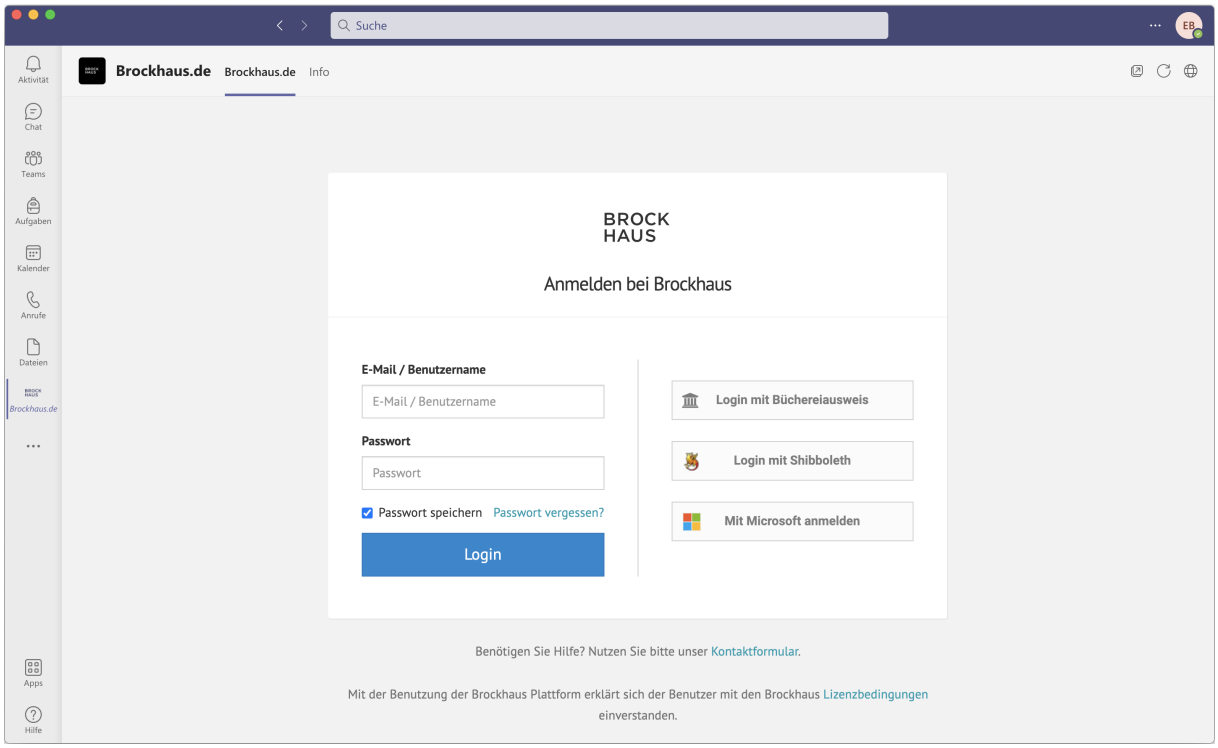


Abbildung 2: Tab der Anwendung «Brockhaus.de» (Quelle: eigene Abbildung)

3.2 Bots

Ein Bot, kurz für Chatbot, führt vordefinierte und sich wiederholende Aufgaben für Nutzer:innen aus. Die Interaktion mit dem Bot findet dabei mittels Chat statt: Der/die Benutzer:in schreibt eine Nachricht in den Chat, welche vom Bot mittels natürliche Sprachverarbeitung (en. *natural language processing*) prozessiert wird. Der Bot kann ebenfalls Nachricht verschicken, welche mit interaktive Karten und Task-Modulen angereichert sein können.[2]

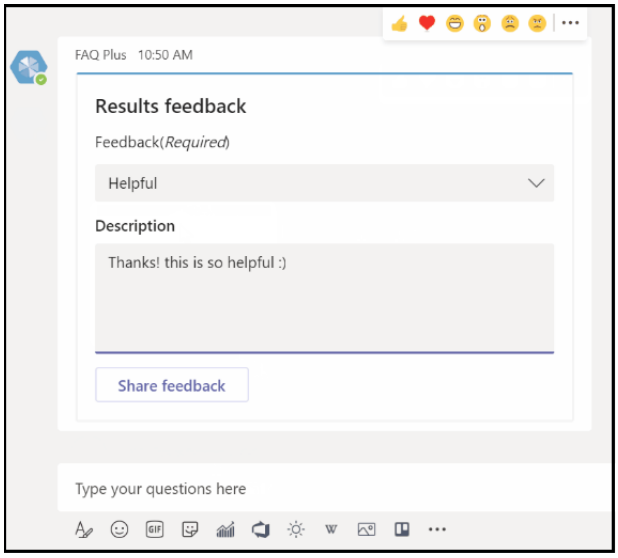


Abbildung 3: Eine Bot-Nachricht in Form einer Karte zum Senden von Feedback (Quelle: Microsoft)

In Kombination mit Webhooks und Konnektoren kann ein Bot auch «von aussen» von Webservices gesteuert werden. Ausserdem kann er auf Ereignisse reagieren, wie z. B. das Hinzufügen eines neuen Mitglieds in einem Team oder das Umbenennen eines Kanals.[2]

3.3 Messaging-Erweiterungen

Messaging-Erweiterungen ermöglichen es den Benutzenden, im Bereich zum Verfassen von Nachrichten, im Befehlsfeld oder direkt aus einer Nachricht heraus (vgl. Abbildung 4) mit Webservices zu interagieren, um so Suchabfragen oder Aktionen in einem externen System durchzuführen. Die Ergebnisse können mittels Karten dargestellt werden.[7]

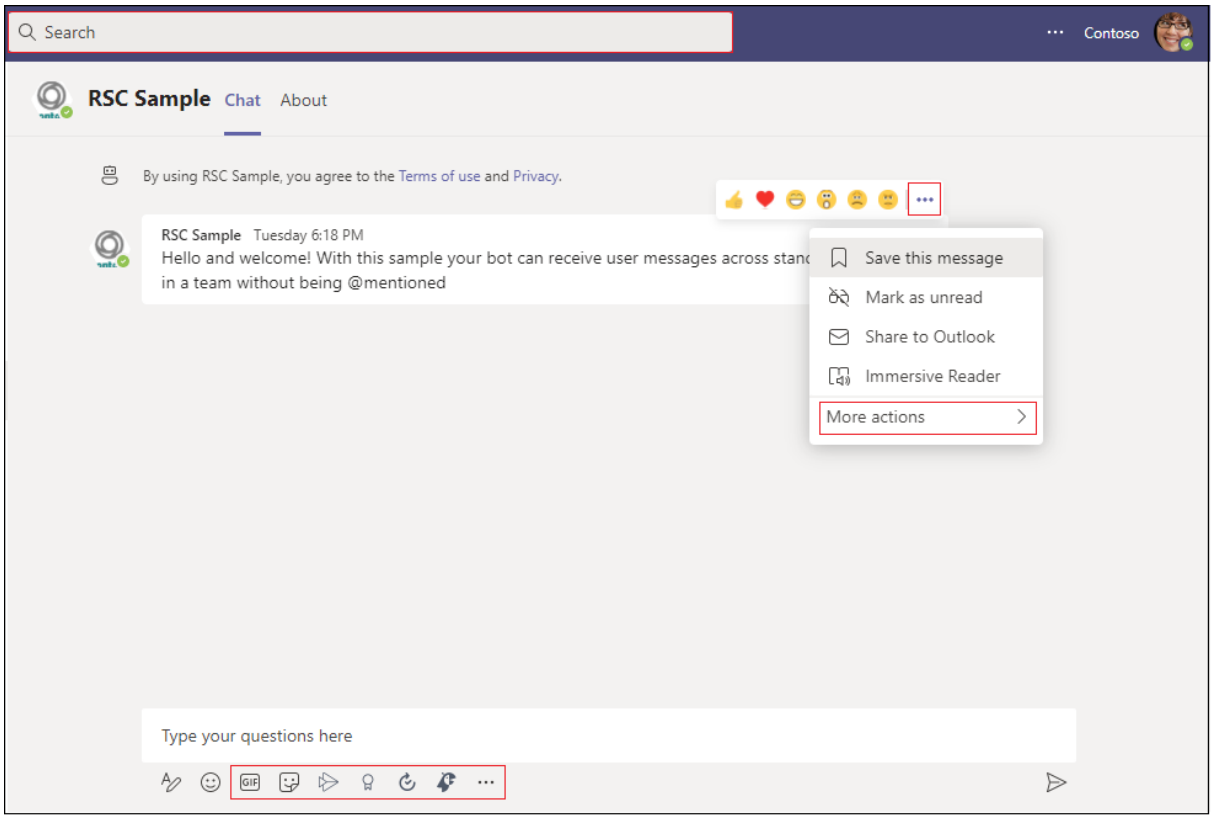


Abbildung 4: Orte, von denen aus Messaging-Erweiterungen aufgerufen werden können (Quelle: Microsoft)

Im App-Manifest der App wird dazu eine Messaging-Erweiterung mit bis zu zehn verschiedenen Befehlen definiert. Es gibt zwei Arten von Messaging-Erweiterungsbefehlen: Aktionsbefehle und Suchbefehle.[7]

Mit Suchbefehle können externe Systeme nach Informationen durchsucht werden, entweder manuell über ein Suchfeld oder durch das Einfügen von einem Link im Bereich zum Verfassen von Nachrichten. Dabei werden die Ergebnisse der Suche automatisch in die Nachricht eingefügt.[7]

Aktionsbefehle werden verwendet, um den Benutzenden einen modalen Popup-Dialog zu präsentieren, um Informationen zu sammeln oder anzuzeigen. Wird das Formular abschickt, antwortet der Webservice, indem er eine Nachricht direkt in den Chat schickt oder im Bereich zum Verfassen von Nachrichten einfügt. Für komplexere Arbeitsabläufe können mehrere Formulare miteinander verbunden werden.[7]

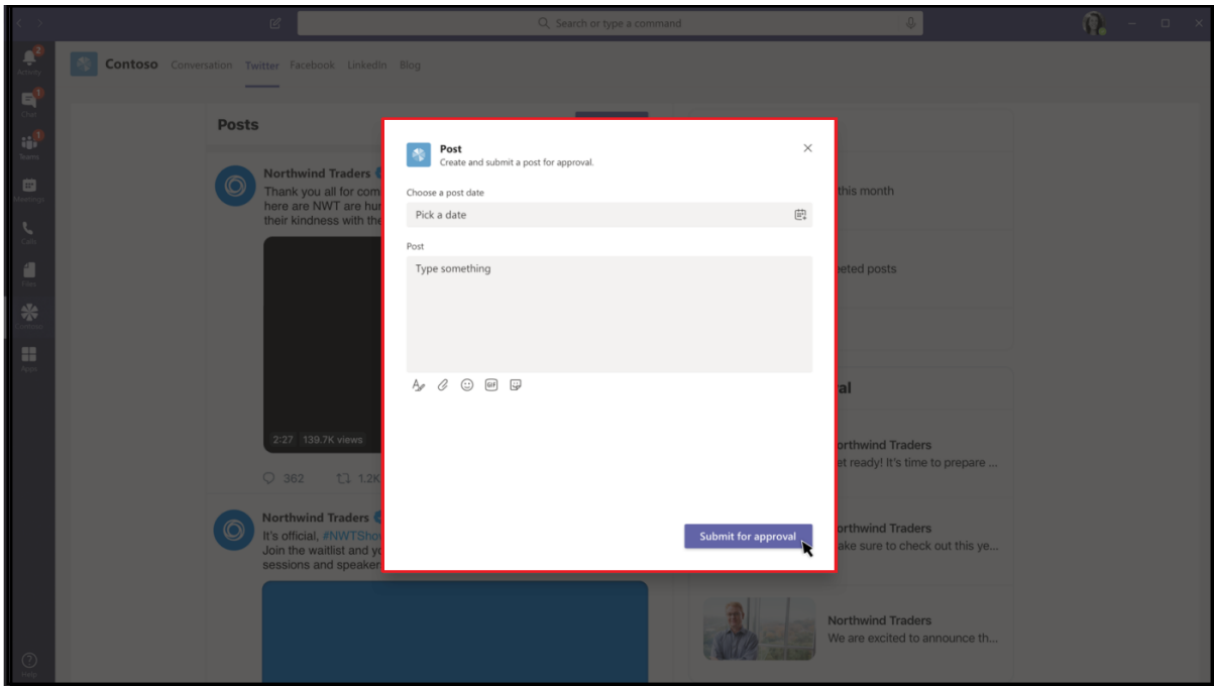


Abbildung 5: Messaging-Erweiterung mit einem Aktionsbefehl, dargestellt als Task-Modul (Quelle: Microsoft)

3.4 Webhooks und Konnektoren

Webhooks und Konnektoren dienen dazu, Teams und Kanälen mit Webservices zu verbinden. Webhooks sind benutzerdefinierte HTTP-Callbacks, welche von MS Teams aufgerufen werden, um den Service über Aktionen zu benachrichtigen, die in einem MS Teams-Kanal stattgefunden haben. Konnektoren ermöglichen es, Benachrichtigungen von Webservice zu subskribieren. Webhooks stellen einen HTTPS-Endpunkt für den Webdienst bereit, um so bspw. Nachrichten in Form von Karten zu versenden.[12]

Webhooks werden für jedes Team einzeln konfiguriert und können nicht als Teil einer normalen Teams-App eingebunden werden. Konnektoren werden auf Kanal-Ebene konfiguriert, aber auf Team-Ebene installiert.[12]

3.5 Meeting-Erweiterungen

Teams-Apps können auch zu Besprechungen hinzugefügt werden. Dieses Konzept wird dann Meeting-Erweiterung genannt. Apps können entweder vor oder während dem Meeting hinzugefügt werden.[4]

Vor einer Besprechung ist die App in einer Registerkarte verfügbar. Das folgende Beispiel zeigt einen Umfrageentwurf, den die Teilnehmenden während des Meetings beantworten können:[4]

Während der Besprechung kann die App direkt im Meeting angezeigt werden oder Benachrichtigung senden, welche dann jeweils für alle Teilnehmenden sichtbar sind. Nach einer Besprechung kann im App-Tab der Besprechung weiterhin auf die App und ihre Daten zugegriffen werden, um so bspw. die Auswertungen von Umfragen, welche während dem Meeting stattgefunden haben, anzusehen.[4]

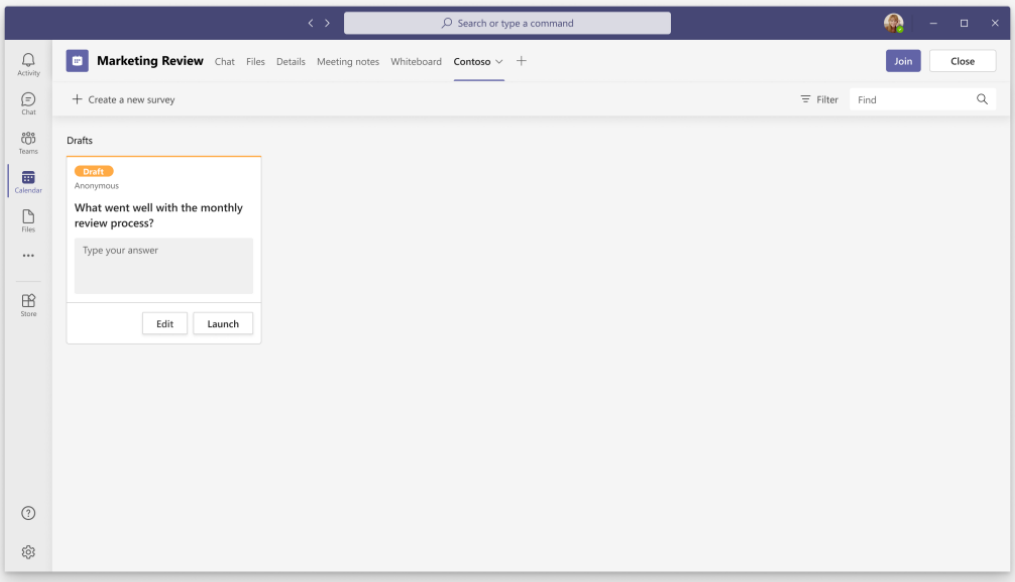


Abbildung 6: Meeting-Erweiterung vor einer Besprechung (Quelle: Microsoft)

3.6 Microsoft Graph API für Teams

Die Microsoft Graph API für Teams bietet umfangreichen Zugriff auf Informationen über Teams, Kanäle, Benutzer:innen und Chats.[3] Die REST-API beschränkt sich dabei nicht nur auf MS Teams, sondern ist das Gateway zu Daten und Informationen aus dem gesamten Microsoft-365-Paket.[10]

Damit eine App die Microsoft Graph nutzen kann, müssen ihr zuvor die richtigen Berechtigungen erteilt werden, entweder von einem/einer Benutzer:in oder Administrator:in.[8]

4 Erweiterungspunkte

Eine App kann an mehreren Stellen in MS Teams eingebunden werden. Diese Stellen werden Erweiterungspunkte (en. *extension points*) gennant. Eine App fällt, je nach dem welche Erweiterungspunkte sie verwendet, in die Kategorie persönliche und/oder geteilte App.[13]

4.1 Geteilte Apps

Geteilte Apps sind auf die Zusammenarbeit von mehreren Benutzenden ausgelegt und werden in den kollaborativen Bereichen von MS Teams eingesetzt: Teams, Kanäle, Besprechungen und Chats. Apps in diesem Kontext sind für alle Mitgliedern der Gruppe oder Unterhaltung verfügbar. Eine App hat dabei Zugriff auf APIs, mit der Informationen über die Mitglieder, die Kanäle und andere Metadaten über das Team oder die Konversation abgefragt werden können.[13]

Die folgende Auflistung zeigt, welche App-Funktionen (vgl. Abschnitt 3 Funktionen) in diesen Bereichen typischerweise genutzt werden.[5]

- Tabs: im Kanal oder (Gruppen-)Chat. Alle Mitglieder interagieren mit demselben webbasierten Inhalt, so dass üblicherweise eine zustandslose Single-Page-App eingebettet wird.
- Bots: im Kanal oder (Gruppen-)Chat
- Messaging-Erweiterungen
- Webhooks und Konnektoren
- Meeting-Erweiterungen
- Die Microsoft Graph REST API

4.2 Persönliche Apps

Persönliche Apps fokussieren sich auf die Interaktion mit einem/einer einzelnen Nutzer:in und werden individuell zu MS Teams hinzugefügt. Die folgende Liste zeigt, wie Teams-Funktionen (vgl. Abschnitt 3 Funktionen) häufig im persönlichen Kontext verwendet werden:[5]

- Bots
- Tabs

5 Umsetzungsmöglichkeiten

Mit den vorhanden Erweiterungspunkten und Funktionen sind diverse Umsetzungen denkbar, um die Meeting-Quality-Anwendung in MS Teams zu integrieren. Aus unserer Sicht macht es Sinn, die App in zwei Teile zu zerlegen: einen Teil für die Organisation der Meetings und einen Teil für das Geben von Feedback.

Der Organisationsteil kann in eine persönliche App verpackt werden, welche Meeting-Organisator:innen auf individueller Ebene zu MS Teams hinzufügen können. Dort wird in einem Tab die Webanwendung, wie sie heute bereits besteht, eingebunden. Alternativ dazu kann auch eine vereinfachte oder auf MS Teams angepasste Version der Meeting-Quality-App angezeigt werden. Auch ein Chatbot wäre denkbar, über welchen Besprechungen verwaltet werden können.

Der andere, gemeinsame Teil fokussiert sich auf das Einholen der Feedbacks. Hier sind mehrere Implementationen denkbar: ein Chatbot, welcher nach einer Besprechung automatisch das Formular in den entsprechenden Chat schickt (entweder mittels Karte oder als Link auf das Formular) oder als Tab, der selbstständig von den Mitgliedern aufgerufen wird.

Die notwendigen Informationen dazu werden via Graph API abgefragt und über Webhooks und Konnektoren mit der Meeting-Quality-App ausgetauscht.

Eine Einbindung via Meeting-Erweiterungen klingt auf den ersten Blick vielversprechend: Das Feedbackformular könnte so direkt nach dem Meeting für alle Teilnehmenden verfügbar gemacht werden. Bei genauerem Hinblick stellt sich diese Lösung aber als nicht sehr praktikabel heraus. Einerseits müsste die App so für jede Besprechung aufs Neue hinzugefügt werden, andererseits müssten Teilnehmende, um das Feedbackformular auszufüllen, erst

ihren Kalender öffnen, dort das entsprechende Meeting auswählen und dann zum App-Tab navigieren. Die Meeting-Quality-App wird so hinter diversen Schritten versteckt. Etwas, was psychologisch wohl als *Cognitive Friction* – kognitive Reibung – bezeichnet werden würde.

6 Siehe auch

- What are Microsoft Teams apps?
- Entry points for Teams apps
- Understand the Microsoft Teams app structure
- Designing your Microsoft Teams meeting extension

Glossar

Chatbot «Elektronisches Dialogsystem, das einen natürlichen Chatteilnehmer imitiert»[1]. 2, 4

Literaturverzeichnis

[1] Duden. *Chatbot, der*. URL: https://www.duden.de/rechtschreibung/Chatbot (besucht am 31. 10. 2021).

[2] Microsoft. *Bots in Microsoft Teams*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/bots/what-are-bots (besucht am 06. 11. 2021).

[3] Microsoft. *Build apps for Microsoft Teams*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/overview (besucht am 07. 11. 2021).

[4] Microsoft. *Designing your Microsoft Teams meeting extension*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/apps-in-teams-meetings/design/designing-apps-in-meetings#after-a-meeting (besucht am 07. 11. 2021).

[5] Microsoft. *Entry points for Teams apps*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/extensibility-points (besucht am 02. 11. 2021).

[6] Microsoft. *Introduction to building apps for Microsoft Teams*. URL: https://docs.microsoft.com/en-us/learn/modules/intro-microsoft-teams-apps/ (besucht am 31. 10. 2021).

[7] Microsoft. *Messaging extensions*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/messaging-extensions/what-are-messaging-extensions (besucht am 06. 11. 2021).

[8] Microsoft. *Microsoft Graph permissions references*. URL: https://docs.microsoft.com/en-us/graph/permissions-reference (besucht am 07. 11. 2021).

[9] Microsoft. *Microsoft Teams tabs*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/tabs/what-are-tabs (besucht am 01. 11. 2021).

[10] Microsoft. *Overview of Microsoft Graph*. URL: https://docs.microsoft.com/en-us/graph/overview (besucht am 07. 11. 2021).

[11] Microsoft. *Understand Microsoft Teams app capabilities*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/capabilities-overview (besucht am 02. 11. 2021).
[12] Microsoft. *Webhooks and connectors*. URL: https://docs.microsoft.com/en-us/microsoftteams/platform/webhooks-and-connectors/what-are-webhooks-and-connectors (besucht am 06. 11. 2021).
[13] Microsoft. *What are Microsoft Teams apps?* URL: https://docs.microsoft.com/en-us/learn/modules/intro-microsoft-teams-apps/2-microsoft-teams-apps (besucht am 02. 11. 2021).

Abbildungsverzeichnis

Anhang C

User Stories

Meeting Quality in MS Teams nutzen

Als Organisator:in von Meetings möchte ich die Meeting-Quality-App (MQ-App) direkt in MS Teams nutzen können, damit ich zum Organisieren meiner Meetings die Plattform nicht verlassen muss. Dazu installiere ich die MQ-Teams-App und kann sie danach in gewohntem Funktionsumfang verwenden, so wie wenn ich sie über den Browser aufrufen würde.

Akzeptanzkriterien

- Ich kann die MQ-Teams-App zu MS Teams hinzufügen.
- Wenn ich noch nicht eingeloggt bin, erscheint die Login-Seite, auf welcher ich mich anmelden kann und danach zur Meeting-Übersicht weitergeleitet werde.
- Wenn ich noch keinen MQ-Account habe, kann ich anstelle des Logins auch einen Account erstellen.
- Das Ausfüllen vom Feedbackformular ist nicht Teil dieser User Story.

Technische Hinweise

- Die MQ-App wird in einem Tab eingebunden.
- Die MQ-Teams-App wird vorerst noch nicht im App Store veröffentlicht, sondern via Sideloading zu MS Teams hinzugefügt.

Jira-Link: MQ-57

Link zum Feedbackformular in die Zwischenablage kopieren

Als Organisator:in von Meetings in der Meeting-Quality-App möchte ich den Link zum Feedbackformular direkt in der Meetingübersicht kopieren können, damit ich nicht zuerst in die Detailansicht des Meetings wechseln muss.

Akzeptanzkriterien

- Wenn schon Meetings vorhanden sind, kann ich den Link zum dazugehörigen Feedbackformular durch einen Klick auf den Link-Kopieren-Knopf in die Zwischenablage kopieren.

Jira-Link: MQ-58

Meeting automatisch von MS Teams erstellen

Als Organisator:in von Meetings in der Meeting-Quality-App (MQ-App) möchte ich meine Meetings nicht doppelt erfassen müssen (in der MQ-App und in MS Teams). Stattdessen möchte ich, dass ein Meeting, welches ich in MS Teams erstelle, automatisch auch in der MQ-App erstellt wird.

Akzeptanzkriterien

- Ich will folgende Informationen erfassen können (* = optional):
  - Titel
  - Datum und Uhrzeit
  - Anzahl der Teilnehmenden*

Jira-Link: MQ-62

Meetings importieren

Als Organisator:in von Meetings in der Meeting-Quality-App (MQ-App) möchte ich Meetings, welche ich in MS Teams erstellt habe und die nicht automatisch in die MQ-App übernommen werden, importieren können.

Akzeptanzkriterien

- Wenn ich nicht eingeloggt bin erscheint beim Klick auf den Meetings importierenKnopf die Aufforderung, dass ich mich mit meinem Microsoft-Account verbinden muss.
- Wenn ich eingeloggt bin, werden durch einen Klick auf den Meetings importierenKnopf alle Meetings der letzten zwei Tage sowie zukünftigen Meetings aus MS Teams angezeigt, welche ich erstellt habe.
- Ich habe die Möglichkeit, Meetings einzeln zu importieren oder aber alle Meetings auf einmal zu importieren.
- Meetings, welche ich bereits importiert habe, werden als solche gekennzeichnet. Ich habe aber die Möglichkeit, diese erneut zu importieren.

Technische Hinweise

- Der/die Benutzer:in muss mit Microsoft authentifiziert sein.
- Um bereits importierte Meetings zu identifizieren wird die UUID der Meetings mitabgespeichert.

Jira-Link: MQ-103

Fragebogen automatisch versenden

Als Organisator:in von Meetings in der Meeting-Quality-App möchte ich, dass zum Endzeitpunkt des Meetings automatisch der Link zum Feedbackformular an die Teilnehmenden versendet wird, damit ich das nicht manuell erledigen muss.

Akzeptanzkriterien

- Falls zu einem Meeting die E-Mail-Adressen der Teilnehmenden hinterlegt sind, versendet die Applikation zum Endzeitpunkt des Meetings automatisch den Link an die Beteiligten per E-Mail.
- Ich kann zu jedem Meeting bestimmen, ob der Link zum Fragebogen automatisch versendet werden soll. Diese Option wird nur angezeigt, wenn E-Mail-Adressen erfasst worden sind und ist standardmässig gesetzt.

Technische Hinweise

- Es muss ein Endzeitpunkt bei den Meetings gegeben werden, damit die Applikation weiss, wann die Mails versendet werden sollen

Jira-Link: MQ-63

Meetingzeit erfassen

Als Organisator:in von Meetings in der Meeting-Quality-App (MQ-App) möchte ich zu jedem Meeting auch die Zeit (Start und Ende) erfassen können.

Akzeptanzkriterien

- Zu jedem Meeting, das ich erstelle, kann ich eine Start- und Endzeit festlegen.
- Wenn ein Meeting erstellt wurde, kann ich die Start- und Endzeit beliebig bearbeiten.
- Die Endzeit kann nicht vor der Startzeit liegen. Eine Zeitangabe ist eine Kombination aus Datum und Uhrzeit.
- In der Meetingübersicht sowie in der Detailansicht wird mir, zusätzlich zum Datum, auch die Zeit und die Dauer des Meetings angezeigt.
- Ganztägige Besprechungen sollen noch nicht möglich sein.

Jira-Link: MQ-64

Mail der Teilnehmenden erfassen

Als Organisator:in von Meetings in der Meeting-Quality-App kann ich zu jedem Meeting die E-Mail-Adressen der Teilnehmenden erfassen.

Jira-Link: MQ-68

Meetingdauer im Dashboard anzeigen

Als Organisator:in von Meetings in der Meeting-Quality-App möchte ich sehen, ob ich die Dauer meiner Meetings ideal wähle. Dazu wird mir im Dashboard ein Diagramm dargestellt, welches aufzeigt wie viele meiner Meetings *zu kurz*, *perfekt* oder *zu lange* waren.

Jira-Link: MQ-78

Effizienz der Meetings im Dashboard anzeigen

Als Organisator:in von Meetings in der Meeting-Quality-App möchte ich sehen, wie die Effizienz meiner Meetings empfunden wird. Dazu wird mir im Dashboard ein Diagramm dargestellt, welches aufzeigt wie viele meiner Meetings *unbefriedigend*, *wie gewöhnlich* oder *super* waren.

Jira-Link: MQ-89

Durchschnittliches Rating pro Woche

Als Organisator:in von Meetings in der Meeting-Quality-App möchte ich auf einen Blick sehen, wie meine Meetings abschneiden. Dazu wird mir im Dashboard eine Sterne-Bewertung angezeigt, welche die Feedbacks von allen Meeting berücksichtigt: kein Stern ist der schlechteste Wert, fünf Sterne der Beste.

Technische Hinweise Für die Berechnung der Sterne werden die Kriterien pro Feedback mit Punkte bewertet und miteinander verrechnet. Die Punkte für die Kriterien werden folgendermassen vergeben:
Effizienz:

- Unsatisfying: 0 Punkte
- As usual: 5 Punkte
- Awesome: 10 Punkte

Zeit:

- Perfect: 10 Punkte
- 5 min: 8 Punkte
- 15 min: 3 Punkte
- 30 min: 2 Punkte
- 45 min: 1 Punkte
- more: 0 Punkte

Die Punkte aus den beiden Kriterien werden addiert und daraus die Sterne-Bewertung pro Meeting generiert: erreichte Punkte / maximale Punkte * 5. Das Ergebnis wird auf ganze Zahlen gerundet (min = 0, max = 5). Für die Gesamtbewertung wird das nach der Anzahl Teilnehmenden gewichtete arithmetische Mittel verwendet und ebenfalls gerundet.

Jira-Link: MQ-92

Anhang D

Entwicklungsleitfäden

MeetingQuality-Backend Deployment and Development Guide

Prerequisites

- Node.js (16.13.0 LTS)
- Yarn (Facebooks node package manager)
- MongoDB (container or native installation)

1. Install the dependencies: in your console, simply run `yarn`.
2. Create a `.env` file in root and add the environment variables. (see sample env file)
3. (optional) Create a DB in MongoDB, e. g. `missingquality` if not exists.

Usage

`yarn start`

Builds and watches the project (using TypeScript and nodemon) and starts the server in developement mode.

`yarn start:prod`

Builds the application and starts the server in production mode.

`yarn build`

Compiles the current project using the settings defined in `tsconfig.json`. Emits the build output into the *build/* folder.

Use the `--noEmit` option to verify if the build is still working.

Option	Description	Type	Default	Required?
`--noEmit`	Compiles the project without emitting files from the compilation.	`bool`	`false`	No

`yarn lint`

Lint the project using ESLint.

Option	Description	Type	Default	Required?
`--fix`	Fixes some of the lint errors	`bool`	`false`	No

`yarn format`

Formats the project using Prettier. This can conflict with the ESLint, so it's recommended to run `yarn lint --fix` afterward.

API Docs

The API docs, generated with Swagger, can be found at http://localhost:5000/api-docs/.

This, however, requires starting the dev server in advance (see `yarn start`).

Developer Guide

Additional Tooling

- Ngrok
- Postman

To locally develop all of this APIs features you will need these additional programms

Working with Ngrok

To use Ngrok you just have to have the ngrok binary installed and launch it from your command shell with `ngrok http 5000`

After this you will have connected to the socks network which allows you to tunnel your localhost to an ephemeral https URL.

This is necessary to develop the MS Graph Subscriptions because microsoft only accepts responses from Https Endpoints for the subscriptions.

Debugging with Postman

You can download the MS Graph Collection for Postman to see if your Backend is capable of accepting the required format from Microsoft

Postman can send HTTP requests using different Authorizations and payloads making it indispenseable for debugging.

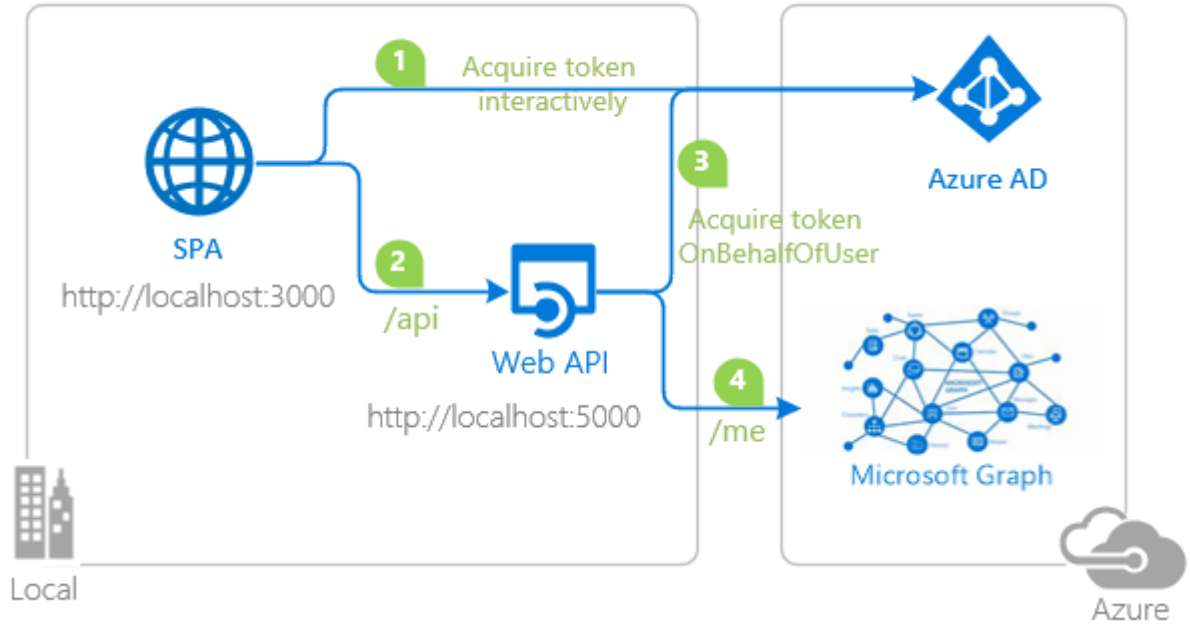
Env Var

In the sample env file there is a `NGROK_PROXY=<Ngrok_Tunnel_URL>` environment Variable which will need to be set to enable Microsoft to message the application. This is because Microsoft does not allow messages to non HTTPS endpoints.

Microsoft Integration

The used authentication method to access the graph API is the Obo flow

Passport middleware is required to acquire the microsoft on behalf of token used for the Backend in the frontend



SendGrid Integration

To use SendGrid the developer needs access to a SendGrid account and has to authenticate the a email sender with SendGrid.

Microsoft Active Directory App Configuration

See Microsoft Identity Tutorial for how to configure the required app registrations for usage of the Obo Authentication Flow.

Deployment Guide

1. Verify that the Microsoft App Registrations exist in the companies Azure Active Directory
2. Replace the notification_url field used with the `NGROK_PROXY` env var with the production domain URL
3. Check other env vars for reachability of Frontend and DB
4. Start production build

MeetingQuality-Frontend

Requirements

- Node.js (16.13.0 LTS)
- Yarn (Facebooks node package manager)

1. Install the dependencies: in your console, simply run `yarn`.
2. Create a `.env` file in root and add the environment variables. (see sample env file)

Usage

- `yarn`
- Create .env.development file in root and REACT_APP_API_URL variables and pass dev server url.
- Create .env.production file in root and REACT_APP_API_URL variables and pass prod server url.

Available Scripts

In the project directory, you can run:

`yarn start`

Runs the app in the development mode.
Open http://localhost:3000 to view it in the browser.

The page will reload if you make edits.
You will also see any lint errors in the console.

`yarn test`

Launches the test runner in the interactive watch mode.
See the section about running tests for more information.

`yarn build`

Builds the app for production to the `build` folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.
Your app is ready to be deployed!

See the section about deployment for more information.

`yarn eject`

Note: this is a one-way operation. Once you `eject`, you can't go back!

If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own.

You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.

`yarn format`

Formats the project using Prettier. This can conflict with the ESLint, so it's recommended to run `yarn lint --fix` afterward.

Developer Guide

Additional Tooling

- React Developer Tools
- Redux DevTools

Microsoft Active Directory App Configuration

See Microsoft Identity Tutorial for how to configure the required app registrations for usage of the Obo Authentication Flow.

Deployment Guide

1. Verify that the Microsoft App Registrations exist in the companies Azure Active Directory
2. Check authConfig.ts file for correct Microsoft configuration
3. Check other env vars for reachability of Backend
4. Create production build
5. Use Webserver like Nginx to serve production build

MeetingQuality-Teams-App Development & Deployment Guidey

Prerequisites

- Node.js (14.18.1 LTS)
- teamsfx (Either VSC Extension or Binary)

1. Install the dependencies: in your console, simply run `npm i`.
2. Configure .fx folder files for useage with local system and Microsoft user

Usage

`teamsfx preview`

Used to run the local development build. To install the package sideloading needs to be enabled for the Microsoft user in Azure Active Directory.

This is also invoked pressing F5 using the Teams Toolkit VSC Extension

`teamsfx validate`

Validate the current application

`teamsfx package`

Build your Teams app into a package for publishing

`teamsfx publish`

Publish the app to Teams

`yarn format`

Formats the project using Prettier. This can conflict with the ESLint, so it's recommended to run `yarn lint --fix` afterward.

Developer Guide

Additional Tooling

- Ngrok

Ngrok is required to Tunnel your localhost application to an external HTTPS endpoint, otherwise Microsoft Teams does not accept non HTTPS endpoints.

Deployment Guide

1. Validate your App package through the Teams Toolkit tab in Visual Studio Code.

2. To publish your app, navigate to the Teams Toolkit tab in Visual Studio Code.
3. Select Publish to Teams in the Project section.

This hosts the App on Azure, which is a requirement for Microsoft Teams Extensions.

MeetingQuality-Management Scripts

Prerequisites

- Node.js (16.13.0 LTS)

1. Install the dependencies: in your console, simply run `npm i`.
2. Create a `.env` file in root and add the environment variables. (see sample env file)

Usage

`npm run mail`

Invokes the sendemails.js script to query and send the emails of Meetings with a meeting enddate which has passend at the moment of invocation.

!!!Warning!!! If the scripts runs successfully the emails saved in the database associated to the meeting are deleted

`npm run renew`

Invokes subscription-lifecyclemgmt.js script which queries the database for Subscriptions which will end the next day.

!!!Warning!!! If the scripts runs successfully the subscriptions saved in the database associated to the user are deleted and new subscriptions will be associated with the owner of the previous subscriptions ensureing continueing functionality, as long as the Backend Endpoint responsible for handling the validation from Microsoft is active.