Fachhochschule für die Wirtschaft Hannover
- FHDW -

PRAXISARBEIT

Ressourcen- und Terminplanung in Office365 mit Hilfe von Microsoft Graph API

Name: **Adham Aijou**
Brinker Straße 72
30851, Langenhagen

Betreuer: **Prof. Dr. Ing. Volkhard Klinger**

Studiengruppe: **HFI421IN**

Matrikelnummer: **600142**

Ausbildungsbetrieb: **DOOH media GmbH**
Frankenring 18
30855 Langenhagen

Eingereicht am: xx.xx.xxxx

Inhaltsverzeichnis

1 Einleitung

Die Praxisarbeit befasst sich mit der Ressourcen- und Terminplanung in Office365 mithilfe von Microsoft Graph API [microsoftGraphApi]. Die Arbeit ist in drei Teile gegliedert. Im ersten Teil wird die Theorie der Ressourcen- und Terminplanung in Office365 mithilfe von Microsoft Graph API erläutert. Im zweiten Teil wird die praktische Umsetzung der Theorie beschrieben. Im dritten Teil wird die Arbeit abschließend bewertet.

1.1 Fragestellungen der Arbeit

Inwiefern und wie sinnvoll, ist der Einsatz von Microsoft Graph API für die Ressourcen- und Terminplanung in Office365 und wie lässt sich dies in der Praxis umsetzen? Dabei sollte berücksichtigt werden, dass dies auf die Anforderungen eines Kundenauftrags bezogen sind, welche im Kapitel 3.2.1 erläutert werden. Deshalb stellt sich auch die Frage, ob die Implementierung sinnvoller ist, als weiterhin die bestehende Lösung (Outlook, Teams, etc.) zu nutzen. Wie sollte sowas umgesetzt werden und welche Aspekte der Microsoft Graph API sind dafür relevant? Als Letztes stellt sich die Frage, wie die Arbeit abschließend bewertet werden sollte.

1.2 Ziele der Arbeit

Sowohl die theoretische als auch die praktische Umsetzung der Theorie, müssen in der Arbeit beschrieben werden. Abschließend bedarf es sowohl einer objektiven als auch einer subjektiven Bewertung der Arbeit, um festzustellen, ob die Implementierung dieser API, für den Kunden sinnvoll ist. Die objektive Bewertung sollte qualitative und quantitative Aspekte der Arbeit berücksichtigen. Daher sind die Ziele der Arbeit:

1. Ziel: Es soll bewertet werden, ob die Implementierung dieser API, für den Anwendungsfall sinnvoll ist.
2. Ziel: Eine funktionierende Webapplikation zu erstellen, welche die Ressourcen- und Terminplanung in Office365 mithilfe von Microsoft Graph API umsetzt und synchronisiert.
3. Ziel: Die Applikation soll die Anforderungen des Kunden erfüllen.
4. Ziel: Die Applikation soll benutzerfreundlich, intuitiv und performant sein.
5. Ziel: Es muss passende Hardware für die Applikation bereitgestellt werden.
6. Ziel: Die Software soll beim Kunden eingesetzt werden.
7. Ziel: Die Software sollte wartbar sein.
8. Ziel: Die Software sollte kostengünstig sein.

1.3 Ergebnisse der Arbeit

Die Arbeit hat alle Ziele erreicht. Die Microsoft Graph API ist erfolgreich eingesetzt und die Ressourcen- und Terminplanung in Office365, mithilfe von der Microsoft Graph API, wurde erfolgreich als **Single Page Application (SPA)** umgesetzt.
Die detaillierten Aspekte Ergebnisse der Arbeit sind in den Kapiteln 6 und 5 beschrieben. Das ausformulierte Ergebnis und die Schlussfolgerung dieser, sind in Kapitel 6 erläutert.

1.4 Zusammenfassung

Die Praxisarbeit wurde durch einen Kundenauftrag ins Leben gerufen. Es sei umständlich für gewisse Kunden für Konferenzräume Termine zu buchen. Jedes Mal muss nämlich ein Mitarbeiter die Verfügbarkeit eines Konferenzraumes auf einer Applikation mit zu vielen Zwischenschritten prüfen und dann einen Termin buchen. Dieser Prozess ist sehr zeitaufwendig und kann zu Fehlern führen.
Daher soll eine Anwendung entwickelt werden, die es ermöglichen soll, dass der Mitarbeiter selbstständig einen Termin buchen oder einsehen kann. Außerdem soll überprüft werden können, wann Termine für den Raum stattfinden sollen, wenn die Person schon im Gebäude und sich vor dem Raum befindet.
Bei diesem Kunden sind viele Mitarbeiter nicht aus der gleichen Abteilung oder dem gleichen Subunternehmen und besitzen folglich unterschiedliche Firmennamen und Logos. Wenn ein Kalender keine detaillierteren Termininformationen darstellt, müsste der Mitarbeiter immer noch nachschauen, ob der Termin für ihn gedacht sei. Die Buchungen direkt vor dem Raum sind dafür gedacht, dass ein Mitarbeiter den Raum sofort buchen kann.
Sollte dies nicht der Fall sein, müsste der Anwender, im Voraus, an einem anderen Endgerät, welches Outlook oder Teams besitzt, einen detaillierten Termin buchen, damit alle Beteiligten wissen, wann der Termin stattfindet und wer anwesend sein wird. Dort kann der Anwender dann auch angeben, welche Firma der Gastgeber und welche der Gast sein soll.

2 Grundlagen

Um die Arbeit in vollem Umfang zu verstehen, ist es wichtig, dass die Grundlagen von Webanwendungen verstanden sind.

2.1 Webanwendungen

2.1.1 Was ist eine Webanwendung?

Eine Webanwendung ist eine Anwendung, die über einen Browser aufgerufen werden. Manche Webanwendungen sind auch Offline nutzbar.

2.1.2 Vor- und Nachteile von Webanwendungen

Eine Webanwendung ermöglicht es, dass die Anwendung von jedem Endgerät, welches über eine Internetverbindung verfügt, aus aufgerufen werden kann. Webanwendungen können effizient, benötigen im Optimalfall wenig Speicherplatz, und können gut abgesichert werden. Aufgrund der Tatsache, dass Browser gut standardisiert sind, können Webanwendungen schnell entwickelt werden.

Ein Nachteil von Webanwendungen ist, dass nicht alle Offline nutzbar sind. Sie sind zudem nicht so schnell wie manche Desktopanwendungen, die auf mehr Rechenleistung zugreifen können, da sie im Regelfall Programmiersprachen nutzen, die näher an Maschinensprache sind. Webanwendungen besitzen heutzutage ausschließlich [JavaScript] als native Sprache. Um mehr Flexibilität und Funktionalität, ohne großen Aufwand zu ermöglichen, sind Webanwendungen auf **Frameworks**, angewiesen, welche oftmals umfangreich sind und Dependenzen besitzen, die weiterhin gepflegt werden müssen. Dies führt oftmals dazu, dass falls eine Abhängigkeit nicht mehr gepflegt wird oder andere geupdatet werden müssen, die Anwendung eventuell nicht mehr funktioniert, falls die Abhängigkeit nicht mehr kompatibel ist. Außerdem lässt JavaScript Laufzeitfehler zu.

2.2 Microsoft Graph API

Die Microsoft Graph API ist eine **RESTful** web **API**, die es einem erlaubt auf Daten von Microsoft 365 und Office 365 zuzugreifen.
Es gibt einige APIs, die sich auf bestimmte Microsoft 365 und Office 365 Abonnements beziehen, wie beispielsweise die Microsoft Teams API. Die Microsoft Graph vereint jedoch die meisten APIs, die Microsoft 365 und Office 365 betreffen, in einer einzigen API. Dies ermöglicht Zugang auf Daten von Microsoft 365 und Office 365, wie beispielsweise Benutzer, Kalender, E-Mails, Dateien, etc.

3 Problemanalyse

3.1 Ist-Zustand

Die Muttergesellschaft der DOOH media GmbH und einige Schwesterunternehmen nutzen Microsoft 365 und Office 365. Diese Produkte sind sehr umfangreich und bieten viele Funktionen. Die heutzutage gängige und weit verbreitete Terminplanung per Outlook oder Teams ist eines dieser Funktionen. Diese Funktion ist nicht so praktikabel wie sie es sein könnte. Vor allem nicht, wenn mehrere Unternehmen, das gleiche Gebäude und die gleiche Organisations-E-Mail besitzen. Daher besteht die Problematik, dass es nicht einfach zu unterscheiden ist, welcher Termin für welchen Mitarbeiter gedacht ist. Dies könne zwar durch die Eingabe des Firmennamens, in den Textkörper des Termins, verbessert werden, jedoch müsste der Mitarbeiter dann trotzdem jeden Termin einzeln öffnen, um zu sehen, welcher Firmennamen im Textkörper steht.
Außerdem ist es nicht möglich, Termine anzulegen, welche das Logo des Gasts oder Gastgebers und diese dann auch im Kalender, inklusive Logo, anzuzeigen.

3.1.1 User Journey

Ein anderes Kernproblem ist, dass die **User Journey**, also der Weg, den ein User durchläuft, um ein Ziel zu erreichen, meistens umständlicher ist, als notwendig.

Angenommen ein fiktiver Mitarbeiter, der den Namen Max Mustermann trägt, möchte einen Termin in einem Konferenzraum buchen, der ohnehin sich in seiner Nähe befindet. Der Termin soll in ungefähr 15 Minuten stattfinden. Dazu muss er zuerst Outlook oder Teams öffnen, dann den Kalender des Raumes öffnen und schauen, wann dieser Raum verfügbar ist. Dafür muss er erstmal an einem Gerät, welches Outlook oder Teams besitzt, die Verfügbarkeit des Raumes prüfen. Er begibt sich wieder an seinen Arbeitsplatz und findet heraus, dass der Raum in 15 Minuten verfügbar ist und bucht diesen. Es sind schon 5 Minuten vergangen. Da die Organisation eine Pufferzeit von 15 Minuten einplant, hat sich der nächste verfügbare Termin um 5 Minuten verschoben.
Nun möchte eine Mitarbeiterin, die den Namen Anna Musterfrau trägt, den gleichen Raum buchen. Sie geht zufällig am Raum vorbei und möchte wissen, ob dieser Raum gleich belegt ist oder nicht. Dafür muss sie, genauso wie Max Mustermann, zuerst Outlook oder Teams öffnen, dann den Kalender des Raumes öffnen und schauen, wann dieser Raum verfügbar ist. Sie begibt sich wieder an ihren Arbeitsplatz und findet heraus, dass der Raum in 15 Minuten verfügbar sein müsste. Sie findet heraus, dass er nicht verfügbar ist, da Max Mustermann ihn bereits gebucht hat. Für Frau Musterfrau sind nun 10 Minuten vergangen, da sie in einer anderen Abteilung des Gebäudes arbeitet.
Eine schnelle Übersicht über die Verfügbarkeit des Raumes ist also nicht möglich. Zudem setzt die heutige Terminplanung voraus, dass der Benutzer die Rechte hat, den Kalender des Raumes zu öffnen.

Nun existiert eine dritte Person, die den Namen Peter Mustername trägt. Er wurde von jemandem eingeladen an einem Meeting im Konferenzraum teilzunehmen. Herr Mustername ist neu in der

einem der Schwesterfirmen und kennt sich noch nicht so gut aus. Er weiß aber noch, dass der Raum in der 3. Etage ist und dass der Betreff des Termins ”Besprechung Finanzen” lautete. Auch das Logo der Firma, bei der er eingestellt wurde, ist ihm bekannt. Herr Mustername muss am besten 15 Minuten vor dem Meeting an dem Raum sein, da er sich noch nicht auskennt und etwas präsentieren muss . Noch besitzt er keine Rechte, den Kalender des Raumes zu öffnen.
Am besten wäre es für Peter, falls er an einem digitalen Türschild erkennen könnte, dass dort ein Meeting mit dem Betreff ”Besprechung Finanzen” stattfindet. Um sicherzustellen, dass das Meeting nicht ein Finanzmeeting einer anderen Firma ist, wäre es gut, wenn das Logo der Firma, bei der er eingestellt wurde, auf dem digitalen Türschild, für das Meeting, zu sehen wäre.

3.2 Soll-Zustand

3.2.1 Anforderungen

Ein User soll am Bildschirm eines Tablets, welches vor dem Raum angebracht wird, erkennen können, ob dieser Raum belegt ist oder nicht. Welche Termine am heutigen und nächsten Tag anstehen soll einfach ersichtlich sein. Zudem möchte der Kunde einen Termin, am Gerät, vereinbaren können. Der User soll also nicht mehr auf Outlook oder Teams angewiesen sein, sondern kann direkt am Bildschirm des Tablets sehen, ob der Raum belegt ist oder nicht. Zudem soll der Raumstatus farbig dargestellt werden, sodass der User sofort erkennen kann, ob der Raum belegt ist oder nicht, sowohl an der Benutzeroberfläche, als auch an den LED Strips des Tablets. Die Benutzeroberfläche soll, so gestaltet sein, dass der User sofort erkennen kann, ob der Raum belegt ist oder nicht. Des Weiteren soll das **UserInterface** so gestaltet sein, dass der User auch Termine für den Raum vereinbaren kann.

3.2.2 User Interface

Das Userinterface (Die Benutzeroberfläche) soll also eine Übersicht über die Termine des Raumes und einen Button zum Vereinbaren eines Termins enthalten. Es wird die Hintergrundfarbe des Containers, in dem der Raumstatus angezeigt wird, in Abhängigkeit vom Raumstatus, rot, gelb oder grün sein. Falls ein Termin gerade stattfindet, soll dieser rot dargestellt werden, falls ein Termin innerhalb der nächsten 15 Minuten stattfindet, soll er gelb dargestellt werden und ansonsten grün. Sollte am heutigen Tag kein Termin stattfinden, wird der Raumstatus weiterhin grün dargestellt, jedoch wird der Text ”Keine weiteren Termine für Heute” angezeigt.

Der erste Vorschlag für die Benutzeroberfläche sieht wie folgt aus:

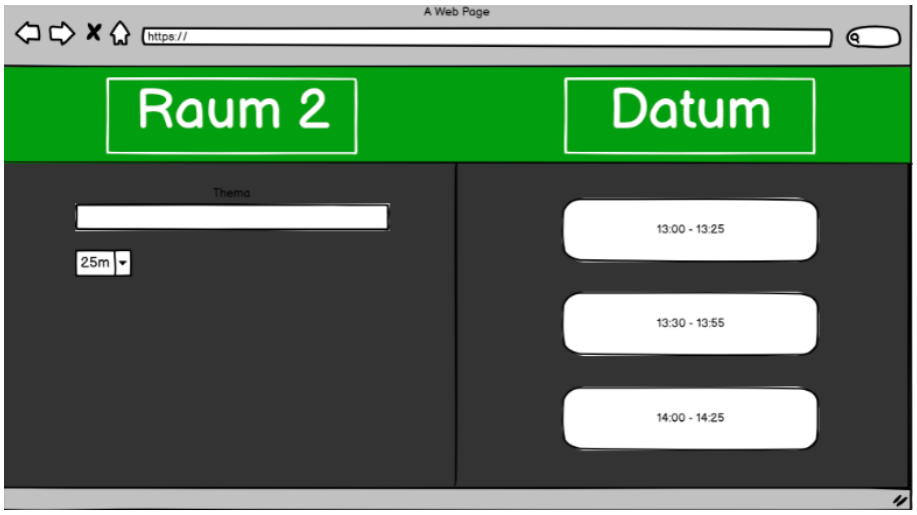

Abbildung 1: Mockup

3.2.3 User Flow

Die hauptsächliche Benutzerinteraktion mit dem System ist das Buchen eines Termins. Dafür wurde ein Workflow erstellt, der den Ablauf des Anwenders beschreibt und wie das System auf die Aktionen des Anwenders zu reagieren hat.

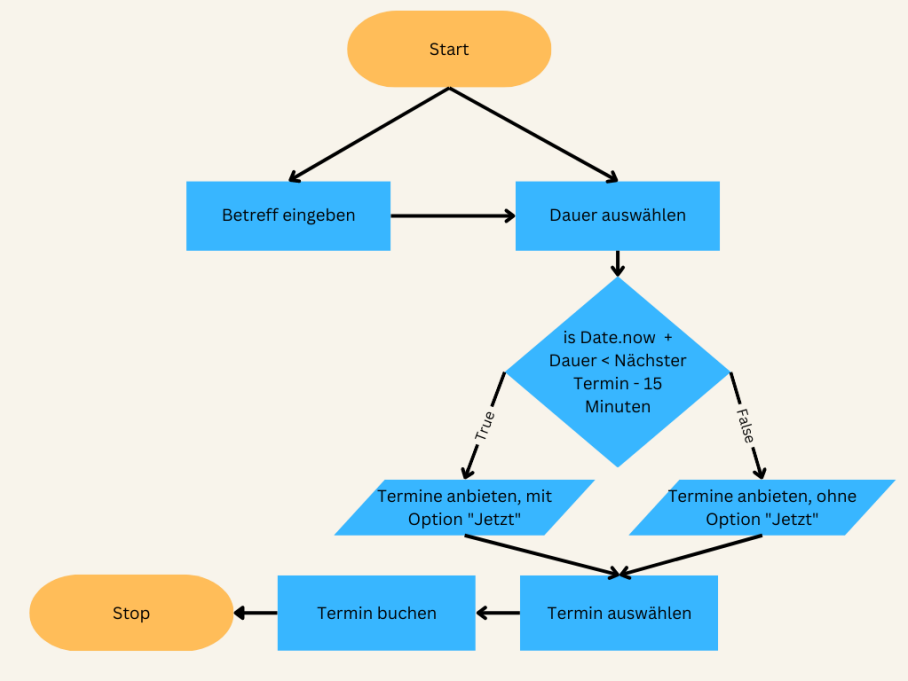

Abbildung 2: Workflow Terminbuchung

Die Rechtecke stellen die einzelnen Schritte des Anwenders dar. Pfeile zeigen den Ablauf des Anwenders. Entscheidungen der Software werden durch die Pfeile mit einem Pfeilspitze dargestellt. Falls diese Entscheidung zu einer Änderung der Daten führt, wird die Folge als Parallelogramm dargestellt.

Von den Grafikdesignerinnen unseres Unternehmens und der Muttergesellschaft wurde ein zweites Design für die Benutzeroberfläche erstellt:

Abbildung 3: GrafikdesignerMockup

Man erkennt im großen roten **Container** den jetzigen Raumstatus. In diesem Fall ist der Raum belegt, da er rot dargestellt wird. Des Weiteren ist zu erkennen, von wann bis wann der Raum belegt ist, der Name des Termins und der Firmenname der Gastfirma.
Direkt darüber sieht man den sogenannten **Header**, welcher den Namen des Raumes, die Uhrzeit und das Datum anzeigt. Außerdem ist der Gastgeber am Logo, im Header, erkennbar.
Auf der rechten Seite des Mockups, ist eine Liste an darauffolgenden Terminen zu sehen. Sie bieten ähnliche Informationen wie der aktuelle Termin, aber zeigen nicht den Gastgeber an. Jedoch zeigen sie an, ob der Termin am heutigen oder am nächsten Tag stattfindet und sind dementsprechend sortiert.

Dieses Design, ist jedoch in seiner Ausführung nicht praktikabel. Teilweise ist der Kontrast geringer als 4.5:1, was laut [WCAG] nicht ausreichend ist. Beispielsweise sind die dunkelgrauen Boxen mit einem fast schwarzen Hintergrund der Seite und einem fast weißen Text nicht gut differenzierbar. Des Weiteren wurde bei der Erstellung des Designs nicht genug auf die kleine Auflösung und Displaygröße des Tablets geachtet. Dies sind alles Eigenschaften und Variablen, die bei der Gestaltung der Benutzeroberfläche noch beachtet werden sollten.
Das Design existiert, in sechs Variationen, die sich darin unterscheiden, ob sie gerade im ”dunklen Design” oder ”hellen Design” und in welchem Belegungsstatus der Raum sich befindet.

Responsive Design [ResponsiveWebDesign] wurde hier nur für das Format berücksichtigt, da die Anwendung nur für Tablets und Desktops in horizontaler Ausrichtung gedacht ist.

3.2.4 Hosting

Die Anwendung soll lokal gehostet werden. Durch den Verzicht auf einen Webserver werden kosten eingespart, da Updates voraussichtlich selten passieren werden. Zudem ist es für die Sicherheit der Nutzer so viel besser, da es keinen externen Datenzugriff gibt.

3.2.5 Authentifizierung

Um die Anwendung zu nutzen, muss sich der Benutzer authentifizieren. Dafür wird ein Konto benötigt, welches bei Microsoft registriert ist. Der Nutzer muss außerdem der Anwendung erlauben,

auf seine Daten zuzugreifen. Ansonsten kann die Anwendung nicht funktionieren. Diese Berechtigungen sollten delegiert sein. Da die Microsoft Graph API für das Anmelden des Nutzers, Azure AD, verwendet, ist es notwendig, dass das oAuth 2.0 Protokoll [OAuth-2.0-Simplified] verwendet wird. Die Anwendung muss daher eine absolute, voraussagbare URL haben, die bei Azure AD registriert ist. Dies wird nämlich die URL sein, auf die der Benutzer nach der Authentifizierung weitergeleitet wird.

3.2.6 ISO 8601 Konformität

Zeiten müssen, bei jeglichen API Anfragen an die Microsoft Graph API, ISO 8601 konform sein . Dies ist wichtig, da Zeitzonen des Nutzers ungleich der Zeitzonen der Server, sein können und die Microsoft Graph API nur Zeiten in UTC akzeptiert. Zudem können Sommer- und Winterzeiten unterschiedlich sein. Es muss also bei der Datenverarbeitung darauf geachtet werden, dass auch die Zeiten ISO 8601 konform sind.

3.2.7 Timer

Um zu visualisieren, wie lange der Raum noch belegt ist, soll ein Timer verwendet werden. Dieser soll einen visuellen Countdown darstellen und einfach zu lesen sein.

3.2.8 Leistung

Da die Anwendung auf schwacher Hardware laufen soll, ist es wichtig, dass die Anwendung nicht zu viele Ressourcen verbraucht. Zudem sollten so wenig Daten wie möglich übertragen als auch abgespeichert werden, da die Internetverbindung nicht immer gut sein wird. Des Weiteren sollte die Anwendung langfristig Daten wie Firmenlogos abspeichern können.

3.2.9 Fehlerbehandlung

Fehlermeldungen sollten immer so gestaltet sein, dass sie den User nicht verunsichern ([interaction-design-book1]), sondern ihn auf die Lösung des Problems hinweisen. Auch bei der Gestaltung der Fehlermeldung sollte darauf geachtet werden, dass sie nicht zu lang ist, da der User sonst nicht mehr weiß, was er tun soll. Bei Kiosk Anwendungen ist es besonders wichtig, dass die Fehlermeldungen kurz und prägnant sind, da der User ohnehin in einer Situation ist, in der er wahrscheinlich nicht viel Zeit hat. Das bedeutet, dass Fehler, die dazu führen, dass die Anwendung nicht mehr funktioniert, groß und auffällig dargestellt werden sollten, damit der User sie nicht übersehen kann und versteht, dass er die Anwendung nicht mehr nutzen kann, bis das Problem behoben wurde. Andererseits sollten Fehler, die nicht dazu führen, dass die Anwendung nicht mehr funktioniert, klein und unauffällig dargestellt werden, damit der User nicht gestört wird.

3.2.10 Spezifikationsblatt

Damit der Kunde weiß, wie die Anwendung funktionieren soll, muss ein Spezifikationsblatt erstellt werden. Dieses sollte alle Features enthalten, die die Anwendung haben soll, sowie die technischen Anforderungen, welche erfüllt werden müssen, damit die Anwendung funktioniert. Anhand des Spezifikationsblatts, sollte der Kunde also eigenständig die Anwendung in einen funktionierenden Zustand bringen können.

3.2.11 LED Strips

Die **LED Strips** sollen in der Farbe leuchten, die dem Raumstatus entspricht.

3.3 Use Case

Hierzu wurde ein Use Case erstellt, welcher die Anforderungen an das System beschreibt. Dieser Use Case wurde in einem Tabellenformat erstellt, welches in der folgenden Tabelle dargestellt wird:

Use Case	Ressourcen- und Terminplanung in Office365 mithilfe von Microsoft Graph API
Titel	Finde freie Termine in einem Kalender für eine bestimmte Ressource
Beschreibung	Der Benutzer möchte einen Termin in einem Kalender für eine bestimmte Ressource finden.
Akteure	Benutzer, Ressource, Kalender
Vorbedingungen	Der Benutzer ist angemeldet.
Nachbedingungen	Der Benutzer hat einen Termin in einem Kalender für eine bestimmte Ressource gefunden und/oder hat einen Termin gebucht.
Normaler Ablauf	1. Der Benutzer geht zu einem Tablet, welcher vor dem Raum angebracht wurde, welcher die Ressource stellvertretend repräsentieren soll. 2. Der Benutzer sieht die verfügbaren Zeiten für die Ressource und den jetzigen Status der Ressource. 3. Der Benutzer wählt einen Termin aus. 4. Der Benutzer bucht den Termin. 5. Der Benutzer erhält eine Bestätigung oder eine Fehlermeldung.

3.4 Systemschnittstellen

Die notwendige Kommunikation mit der Microsoft Graph API wurde, als Systemschnittstelle, tabellarisch erläutert:

Kurzbeschreibung:	In der UI kann der User sich mit Konto der Ressource anmelden, die für die Zukunft als Repräsentant für die reelle Ressource (z.B einem Raum) gelten soll. Daraufhin gilt dieser Raum immer als Teilnehmer der Buchungen und kann in seinen verfügbaren Zeiten gebucht werden. Basierend auf diesen Zeiten, bekommt der Benutzer verfügbare timeslots zur Buchung angezeigt. Die Dauer des Termins soll einstellbar sein. Es gibt zudem einen optionalen Betreff für den Termin. Der gebuchte Termin wird in der Terminübersicht, für den aktuellen und folgenden Tag, angezeigt. Zudem soll der Nutzer anhand farblicher Indikatoren (siehe Abschnitt 3.2.1) beurteilen können, ob die Ressource verfügbar ist. **REST API** Anfragen werden dann an die Microsoft Graph API geschickt, um diese Buchung zu kommunizieren.
Akteure:	Mitarbeiter und Software
Häufigkeit:	Abhängig von Firmengröße und Besprechungsfrequenz
Komplexität:	Mittlere bis hohe Komplexität
Vorbedingungen:	1. Der Benutzer muss eingeloggt sein und Zugriffsrechte akzeptiert haben. 2. Optionale Teilnehmer 3. Termindauer 4. Verfügbare Termine 5. Terminbetreff 6. Zeitzone 7. Access Token / Client ID zur Authentifizierung
Ausgaben:	1. Buchung erfolgreich oder nicht 2. Neue verfügbare Termine 3. Der neue Termin wird in der Liste der gebuchten Termine angezeigt
Use Case:	Siehe 3.3
Ausgeführte Aktionen:	Immer auszuführen: 1. Termin finden 2. Termin buchen Optional auszuführen: 1. Teilnehmer hinzufügen 2. Terminbetreff hinzufügen/ändern 3. Termindauer ändern

4 Abgrenzung zu ähnlichen Produkten

Um die Anwendung von anderen Anwendungen abzugrenzen, werden diese tabellarisch verglichen.

Anforderungen	**Microsoft Teams**	**Microsoft Outlook**	**Microsoft Bookings**	**Microsoft Power Automate**	**Gewichtung**
Webapplikation	✓	✓	✓	✓	2
Keine externe Registrierung	✓	✓	✓	✓	2
Gastgeber- und Teilnehmerlogos	✗	✗	✗	✗	2
Zwei-Tages-Ansicht	✓	✓	✗	✓	1
Anzeige aller verfügbaren Termine	✗	✗	✗	✓	2
Anzeige des jetzigen Raumstatus	✓	✓	✗	✓	2
Minimale Anzahl an Klicks	✗	✓	✗	✓	1
Minimale Benutzeroberfläche	✗	✗	✗	✗	1
Funktionen auf ein Minimum beschränken	✗	✗	✗	✓	1
Angemessene voreingestellte Softwaredistribution	✗	✗	✗	✗	2

Die maximal mögliche Punktzahl ist 16. Microsoft Teams erreicht 6 von 16 Punkten. Microsoft Outlook erreicht 8 von 16 Punkten. Microsoft Bookings erreicht 7 von 16 Punkten. Microsoft Power Automate erreicht 11 von 16 Punkten.
Keine der Applikationen erfüllt alle Anforderungen. Microsoft Power Automate erfüllt die meisten Anforderungen. Eine ideale Softwarelösung sollte alle Anforderungen erfüllen. Diese sind Kernanforderungen. Weitere Anforderungen, die nicht erfüllt werden, sind nicht kritisch, aber wären dennoch wünschenswert.

4.1 Spezifische Gründe für die Entwicklung einer eigenen Applikation

Wenn ein Kunde bei der DOOH media GmbH einkauft, kauft er im Regelfall ein umfangreiches Softwarepaket. Dieses Paket inkludiert die Möglichkeit die Geräte in einen sogenannten **Kiosk-Modus** zu versetzen. Dafür haben wir eine eigene Applikation, die Oxygen Player App, entwickelt. Zudem haben wir eine ”OMS”-Lösung (Oxygen Media Server), die es ermöglicht, die Inhalte auf den Geräten zu verwalten. Falls der Kunde also zu bestimmten Zeiten eigene Inhalte oder Inhalte von externen Anbietern, auf den Geräten, darstellen möchte, kann er dies über die ”OMS”-Lösung tun. Diese Inhalte werden dann auf den Geräten abgespielt.
Die Idee ist hier, dass der Kunde nicht an die Terminbuchungs-Webapplikation gebunden ist, sondern diese Applikation nur als zusätzliche Funktion angeboten bekommt. Viele der Softwarelösungen, die schon für genau diesen Anwendungsfall existieren, beinhalten Bindungen an andere Softwarelösungen, die der Kunde nicht benötigt. Sie machen es dem Kunden schwer, die Soft-

warelösung zu nutzen, da er sich mit anderen Softwarelösungen auseinandersetzen muss, die er nicht benötigt und die gegebenenfalls nicht in einem Kiosk-Modus laufen können oder parallel zu unserer restlichen Software, die der Kunde haben möchte. Außerdem erfordern einige dieser Softwarelösungen eine Installation auf dem Gerät oder erzwingen Konten-Registrierungen, die der Kunde nicht benötigt.
Die Alternativprodukte, die in den vorherigen Abschnitten beschrieben wurden, erfüllen nicht alle Anforderungen, die der Kunde an die Softwarelösung hat. Alles wird lokal auf dem Gerät abgespielt und es wird keine Installation auf dem Gerät benötigt. Es wird keine Registrierung eines Kontos benötigt und niemand kann die Daten von außerhalb, außer natürlich Microsoft, einsehen, da wirklich alles lokal gehostet wird. Die Inhalte werden dynamisch auf dem Gerät geladen und verarbeitet.

5 Technische Umsetzung

5.1 Prototyp

Basierend auf dem Mockup 1 wurde ein Prototyp erstellt, welcher testen sollte, ob Grundfunktionalitäten wie die Terminbuchung und die Darstellung der Termine funktionieren. Dies wurde mithilfe von vuejs aufgesetzt.
Der Prototyp funktionierte wie erwartet und konnte somit als Grundlage für die Weiterentwicklung dienen.

5.2 Benutzte Hardware und Software

Die Hardware ist ein Philips [10BDL4551T/00] Bildschirm. Dieser Bildschirm ist ein 10 Zoll (ca. 25 cm diagonal) Touchscreen. Es hat eine Auflösung von 1280x800 Pixeln und benutzt Android 7 als Betriebssystem. Der Bildschirm ist für Digital Signage gedacht und kann so auch als solches verwendet werden. Die eingebauten RGB LED Strips können per **SICP** angesteuert werden. Dafür wird Dory-node.js genutzt, um auf einen lokalen Server zu hören und die Anfragen an den Bildschirm weiterzuleiten. Es wird dafür keine Internetverbindung benötigt. Dokumentationen für SICP öffentlich schwierig zu finden. Befehle werden als Hexadezimal Werte geschrieben und an den Bildschirm gesendet. Die Verschlüsselung passiert, indem ein XOR aller vorausgehenden Bytes genommen und als Prüfsumme mitgegeben wird. Diese Verschlüsselung ist von Phillips definiert. Sie soll lediglich verhindern, dass jemand versehentlich falsche Befehle an den Bildschirm sendet.

Zur Entwicklung der Anwendung, wird eine **IDE** benötigt. Als JavaScript Framework wird Vue.js verwendet, da es sich um eine Single Page Application handelt und Vue.js darauf spezialisiert ist.
Um eine Kommunikation zwischen der Anwendung und dem Bildschirm zu ermöglichen, wird ein **Node.js Server** benötigt. Dieser wird per Dory-node.js lokal gehostet.
Damit die Authentifizierung funktioniert und Prozesse, wie das Einloggen, vereinfacht werden können, wird microsoft-mgt [Microsoft-mgt-npm] als **NPM** Package verwendet. Dies beinhaltet das Microsoft Graph Toolkit [Microsoft-Graph-Toolkit], welches alle Microsoft Graph API Komponenten enthält.
Für eine hohe Kompatibilitätsgewährleistung, wird Babel [Babel] und Webpack [Webpack] verwendet. Babel stellt sicher, dass der Code in allen Browsern lauffähig ist und Webpack sorgt dafür, dass die Anwendung schnell lädt.
Um eine ordentliche Versionierung zu betreiben, wird Git, welches ein dezentrales Versionskontrollsystem ist, genutzt. Es ermöglicht, Änderungen an Dateien zu verfolgen und zu verwalten. Dies wird in Zusammenhang mit GitLab verwendet, da dies der derzeitige, etablierte Standard der DooH media GmbH ist.
Zum Vermeiden von schlechter Codequalität und syntaktischen Fehlern, die der Compiler nicht erkennen kann, wird EsLint [EsLint] verwendet. EsLint ist ein **Linter**, der es ermöglicht, JavaScript Code zu analysieren und zu formatieren. Da Testfälle bei JavaScript Projekten nicht vor der Ausführung der Anwendung möglich sind, ist es wichtig, dass der Code einheitlich ist und keine syntaktischen Fehler enthält. Deshalb achtet EsLint auf die Einhaltung von Regeln, damit erst gar keine Fehler entstehen, die durch falsche Syntax entstehen. Beispielsweise wird überprüft, ob Variablen, welche deklariert werden, auch verwendet werden.

5.3 Grobe Planung

Die Software wird als eine Azure App deployt (**deployen**). Diese lässt **localhost** Verbindungen zu und gibt einem die Möglichkeit das ganze als Single Page Application zu entwickeln, mit einem lokalen **cookie cache** für den eingeloggten Account. Deshalb agiert die Anwendung erstmal nur als ein Gerüst, welches darauf wartet, dass der Benutzer sich einloggt. Sobald sich der Benutzer

einloggt, wird die Anwendung mit den Daten des Benutzers gefüllt. Somit ist keine Individualisierung der Seite für verschiedene Nutzer notwendig.
Der wichtigste Faktor für die Entscheidung, die Microsoft Graph API zu nutzen war, dass mithilfe von **Azure AD**, die Authentifizierung und Autorisierung von Benutzern erlaubt. Es wird, lokal, mithilfe von **Dory-node.js** gehostet.
Der **Player** benötigt auf der OMS den Content-Typen ”Web”,
mit der URL: ”http://localhost:3000/content”. Diese Seite wird dann lokal vom Player aufgerufen, worauf der Dory-node.js Server antwortet und die tatsächliche Seite anbietet, die lokal auf dem Bildschirm liegt. So ist die URL durchgehend, auf allen Geräten identisch, damit die OAuth 2.0 URI Bedingungen erfüllt sind. Ohne diese Bedingungen, wäre es nicht möglich, sich bei der Anwendung anzumelden.

5.4 Iterative Entwicklung der groben Planung

Es wurde eine Azure App erstellt, die die Authentifizierung der Anwendung und des Benutzers übernimmt.

Dort wird der Benutzer eingeloggt und die Anwendung leitet ihn auf die Seite weiter, die er vorher besucht hat, welche in diesem Fall, die Raumbuchungsseite ist.
Es wurde jede Woche Rücksprache mit dem Kunden gehalten, um die Anforderungen zu besprechen und zu erfüllen. Aber auch intern wurde Rücksprache gehalten, was denn sinnvoll ist und was nicht. So wurde jede Woche die Anwendung weiterentwickelt und verbessert. Teilweise wurden auch neue Features hinzugefügt, die nicht initial erwünscht, aber sinnvoll waren. Manche Features wurden auch wieder entfernt, da sie nicht sinnvoll waren oder von anderen Features abgedeckt wurden.

5.5 Rest-Anfragen

Um sicherzustellen, dass die Anfragen ISO 8601 konform sind, wird die Zeitzone bei jeglichen Anfragen mit angegeben. Um eine Terminbuchung vorzunehmen, wird per REST API eine Anfrage an die Microsoft Graph API gesendet.
Diese sieht wie folgt aus:

```
"https://graph.microsoft.com/v1.0/me/calendar/events"
```

Diese wird per POST Methode an die Microsoft Graph API gesendet. Mehrere Header sind generell bei allen Fragen notwendig. Einerseits ein Authorization Header, der den Access Token enthält, der für die Authentifizierung des Benutzers notwendig ist, und andererseits ein Content-Type Header, der auf application/json gesetzt ist.
Im Körper der Anfrage werden die Daten im JSON Format mitgegeben. Diese enthalten Daten wie den Titel, die Startzeit, die Endzeit, die Beschreibung und die Zeitzone.

```
{
    "subject": "Meeting",
    "start": {
        "dateTime": "2019-06-06T10:00:00",
        "timeZone": "Europe/Berlin"
    },
    "end": {
        "dateTime": "2019-06-06T11:00:00",
```

```
        "timeZone": "Europe/Berlin"
    },
}
```

Um herauszufinden, welche Termine frei sind und welche nicht, wird dafür ebenso eine Anfrage an die Microsoft Graph API gesendet. Diese benutzt eine andere URL und andere Daten im Körper (Siehe Kapitel 5.10.1).

5.5.1 Fehlerbehandlung

Falls mehrere Anfragen, welche Informationen aktualisieren sollen, nacheinander fehlschlagen, wird eine Fehlermeldung angezeigt, die den User darauf hinweist, dass die Verbindung zum Server derzeit nicht gewährleistet werden kann. Diese wird groß und prominent angezeigt, damit es nicht übersehen werden kann und deutlich wird, dass jegliche Daten die derzeit angezeigt werden, nicht aktuell sind. Dementsprechend wird die Fehlermeldung entfernt, sobald eine Anfrage erfolgreich war.

5.5.2 Probleme der API

Es wurden im Voraus Befehle im Microsoft Graph Explorer [Microsoft-Graph-Explorer] ausgeführt, um zu testen, ob die Anfrage funktioniert und ob die Daten in der gewünschten Form zurückgegeben werden. Jedoch hat sich herausgestellt, dass der Microsoft Graph Explorer nicht korrekt funktioniert und teilweise Befehle nicht entgegennimmt, die laut Microsoft Dokumentation funktionieren sollten. Evident ist dies, da die Anwendung korrekt funktioniert und die REST API Anfragen korrekt funktionieren und auch im Request-Response Zyklus keine Fehler auftreten.

Zudem war es initial nicht möglich, alle verfügbaren Termine, für den angegebenen Zeitraum, abzufragen. Dies wurde gelöst, indem die maximale Anzahl an Teilnehmern, für einen Termin, welche im Körper der Anfrage übergeben werden, auf 99 gesetzt wurde. Diese Lösung wurde durch einen Nutzer, in einem Forum, herausgefunden. Zwar ist dies nicht von Microsoft so vorgesehen, aber die Lösungs beeinträchtigt die Anwendung nicht, da es ohnehin nicht mehr möglich ist,Teilnehmer hinzu zu fügen.

5.6 Timer

Der Timer, welcher die übrig bleibende Zeit bis zum Ende des jetzigen Termins anzeigt funktioniert, wie folgt:

Es wird die Zeit bis zum Ende den jetzigen Termin berechnet, ausgehend von der Zeit, die zum Anfang des jetzigen Termins vergangen ist, und in Millisekunden umgerechnet. Dann wird eine sich drehende Animation erstellt, die für diese Dauer abläuft.
Da die Dauer angepasst werden kann, muss bei Datenänderungen geprüft werden, ob sich die Dauer verändert hat. Die Daten dafür werden im **Local Storage** gespeichert. Falls eine Änderung stattgefunden hat, wird die Animation neu berechnet. Bei der Änderung werden, damit die Animation visuell nicht von vorne anfängt, die Animationsdauer und die vergangene Zeit als neue Animationsdauer aufaddiert und die vergangene Zeit als negative Animationsverzögerung gesetzt, damit die Animation visuell betrachtet dort ist, wo sie relativ zur neuen Dauer sein sollte.

Die Formel lautet, wie folgt:

$$\begin{aligned} \text{Vergangene Zeit} &= \text{Jetzt} - \text{Start} \\ \text{Animationsdauer} &= \text{Ende} - \text{Start} \\ \text{Animationsverzögerung} &= \text{-Vergangene Zeit} \end{aligned} \tag{1}$$

Die Berechnung wurde wie folgt implementiert:

```
let currentEventBeginningTime = localStorage.getItem('
    currentEventBeginningTime');
let timePassed = (new Date() - new Date(currentEventBeginningTime + 'Z')) /
     1000;
let animationDuration = ((new Date(currentEventEndTime + 'Z') - new Date(
    currentEventBeginningTime + 'Z')) / 1000);
firstHandSpan.style.animationDuration = animationDuration  + 's';
secondHandSpan.style.animationDuration = animationDuration + 's';
firstHandSpan.style.animationDelay = -timePassed + 's';
secondHandSpan.style.animationDelay = -timePassed  + 's';
```

Um die Animation während ihrer Laufzeit zu ändern, wird ein sogenannter Reflow[1] erzwungen, indem die offsetWidth Eigenschaft abgefragt, die Animationsdauer und -verzögerung neu gesetzt und der Animationsname erst entfernt und dann wieder hinzugefügt wird.

5.7 Persistente Datenspeicherung

Um die Daten der Anwendung zu speichern, wurde für kleinere Datensätze der Local Storage verwendet. Dieser ist in jedem Browser verfügbar und darf bis zu 5 MB groß sein.

Für größere Datensätze wurde IndexedDB [IndexedDB] verwendet. Dies ist eine NoSQL Datenbank, die in jedem gängigen Browser [caniuse-indexedDB] (**caniuse.com**) verfügbar ist. Die maximale Größe bei Chrome beträgt 80% des verfügbaren Speichers. Da es im Internet viele, sich widersprechende Angaben zu dieser Größe gibt, wurde diese manuell, mit folgendem Code, in der Konsole des Browsers, getestet:

```
    if (navigator.storage && navigator.storage.estimate) {
    const quota = await navigator.storage.estimate();
    // quota.usage -> Number of bytes used.
    // quota.quota -> Maximum number of bytes available.
    const percentageUsed = (quota.usage / quota.quota) * 100;
    console.log("You've used ${percentageUsed}% of the available storage.")
        ;
    const remaining = quota.quota - quota.usage;
    console.log("You can write up to ${remaining} more bytes.");
    }
```

Die Festplatte, auf der dies getestet wurde, hatte ca. 828 GB freien Speicher. Das Ergebnis lautete, wie folgt:

[1] Reflow.

```
You've used 0.000002393592594674544% of the available storage.
You can write up to 599726105785 more bytes.
```

$$\text{verbleibender Speicher} = 599726105785 \text{ Bytes}$$
$$599726105785 \text{ Bytes} * 10^{-9} = 599{,}726105785 \text{ Gigabytes} \quad (2)$$

Das sind etwas über 70 % der verfügbaren Speicherkapazität. Hier ist davon auszugehen, dass die restlichen 10 % für andere Daten schon verwendet werden. Zudem ist zu betrachten, dass die ¨.¨ Schreibweise für Dezimalzahlen in JavaScript verwendet wurde, während wiederum die ¨,¨ Schreibweise für die mathematische Gleichung verwendet wurde. Dies hat den Hintergrund, dass die ¨,¨ Schreibweise in JavaScript nicht grundsätzlich unterstützt wird, beziehungsweise, die Konvention beim Programmieren verlangt, dass man auf Englisch programmiert und im Englischen die Dezimalzahlen mit einem Punkt geschrieben werden.

Die Datenbank befindet sich aufgrund ihrer geringen Komplexität in der 5. Normalform. Der Schlüssel besteht aus dem Namen der Firma, zu der, das Logo gehört und gibt das nicht Primärattribut zurück, welches das Bild, in Base64, enthält. Schlüssel müssen eindeutig sein. IndexedDB unterstützt die Eigenschaft ¨unique¨, welche dafür sorgt, dass der Schlüssel eindeutig ist. Jedoch gehen wir auch manuell noch einmal sicher, dass der Schlüssel eindeutig ist, also dass sich dieser, noch nicht in der Datenbank befindet. Eine weitere Aufteilung würde zu Informationsverlust führen, da der Firmenname eindeutig ist.

Zwar gibt es die Möglichkeit die Bilder für Firmen, zu updaten, welcher der User auch so wahrnimmt, jedoch wird der Schlüssel des alten Bildes entfernt und ein neuer Schlüssel mit dem neuen Bild hinzugefügt. Dies ist notwendig, da der Schlüssel eindeutig sein muss. Somit löschen wir den alten Eintrag und fügen einen neuen hinzu. Unabhängig von der Interaktion mit der Datenbank, wird sichergestellt, dass sinnfreie Anfragen an die Datenbank nicht gesendet werden. Falls der User beispielsweise ein Bild hochlädt, welches bereits in der Datenbank, für genau den gleichen Schlüssel, vorhanden ist, wird die Anfrage an die Datenbank nicht gesendet oder falls der User versucht ein Bild aus der Datenbank zu entfernen, welches gar nicht existiert.

5.8 Azure Authentifizierung

Um die Authentifizierung zu realisieren, wurden folgende Berechtigungen benötigt:

Calendars.Read	**Lesezugriff auf Benutzerkalender**
Calendars.Read.Shared	Benutzer und freigegebene Kalender lesen
Calendars.ReadWrite	Verfügt über Vollzugriff auf Benutzerkalender.
Calendars.ReadWrite.Shared	Benutzerdefinierte und freigegebene Kalender lesen und schreiben
email	E-Mail-Adresse von Benutzern anzeigen
IMAP.AccessAsUser.All	Read and write access to mailboxes via IMAP.
Mail.Read	Lesezugriff auf Benutzer-E-Mails
Mail.Send	E-Mails unter einem anderen Benutzernamen senden
Mail.Send.Shared	E-Mails im Namen anderer Benutzer senden
offline_access	Zugriff auf Daten beibehalten, für die Sie Zugriff erteilt haben
OnlineMeetings.Read	Read user's online meetings
OnlineMeetings.ReadWrite	Read and create user's online meetings
openid	Benutzer anmelden
profile	Grundlegendes Profil von Benutzern anzeigen
User.Read	Anmelden und Benutzerprofil lesen

Diese werden beim erstmaligen Login, vom User, seitens Microsoft, angefordert. Ohne die Zustimmung des Users, wird die Anwendung nicht entsprechend den Anforderungen funktionieren und weiterhin darauf hinweisen, dass der User sich anmelden soll. Alle Berechtigungen werden als delegierte Berechtigungen angefordert. Das bedeutet, dass die Anwendung, für den Benutzer, die Handlung durchführen darf, die durch die Berechtigung angefordert wurde. Mit diesen Berechtigungen kann sichergestellt werden, dass auch Exchange-Konten, welche eine eigene **IMAP**-Adresse haben, alle Funktionalitäten der Anwendung nutzen können.

5.9 Performanztests

Die Performance einer **SPA** zu messen ist trotz der geringen Komplexität der Anwendung, nicht simpel. Auch die gängige Total-Blocking-Time Messung ist nicht zwingend sinnvoll, da die Anwendung nicht für den Nutzer, während der Nutzung, blockiert ist, sondern lediglich die Daten vom Server geladen werden, um anschließend die Nutzung der Anwendung sinnvoll zu ermöglichen. Daher wurden manuell einige Tests durchgeführt, um die Performance zu messen und in Zusammenhang mit dem RAIL Modell [RAILModell] bewertet. Es sollte dafür berücksichtigt werden, dass der Prozessor des Tablets, welches die Anwendung verwendet, ein 32 Bit Prozessor ist, welcher im Jahr 2013 auf den Markt kam.

5.9.1 Initiales Laden

Die **Total-Blocking-Time (TBT)** der Applikation liegt, aufgrund von hoher Komplexität, bei ca. 500ms. Somit ist das Laden der Webseite akzeptabel [TotalBlockingTime], da die Total-Blocking-Time nicht die Grenze von 600ms überschreitet. Das RAIL Modell erlaubt sogar Ladezeiten von bis zu 5 Sekunden beim initialen Laden der Webseite, falls innerhalb dieser 5 Sekunden die Interaktivität mit der Webseite gegeben ist. Die Hardware, die hier verwendet wurde, besitzt einen 32 Bit Prozessor, welcher im Jahr 2013 auf den Markt kam.

5.9.2 Interaktivität/Leerlauf

Wie lange braucht die Anwendung, um nach einem Klick auf den Button "+" das Terminerstellungs-Menü zu öffnen? name=Event-Listener,description=Ein Event-Listener ist ein

Objekt, welches auf bestimmte Ereignisse wartet und dann eine Funktion ausführt. Dieses Objekt braucht ein HTML Element, welches es überwachen soll. Die Antwort auf diese Fragestellung wurde gemessen, indem der Event-Listener für den Button ”+” registriert ausgab, wann die Taste betätigt und damit Zeit gemessen wurde, die vergeht, bis der das Menü darstellt. Es dauert im Durchschnitt 2ms, bis das Menü angezeigt wird. Damit sind wir weit unter der 100ms Grenze, welche das RAIL Modell für Interaktivität mindestens empfiehlt und weit unter den 50ms, welche das RAIL Modell anstrebt. Dies bedeutet also, dass der Leerlauf der Anwendung sich nicht beeinträchtigend auf die Echtzeit-Wahrnehmung des Nutzers auswirkt.

5.9.3 Animationen

Wie viele Bilder pro Sekunde zeigt die Anwendung durchschnittlich an? Dies wurde mithilfe von LightHouse über 60 Sekunden hinweg gemessen.
Um ein **Worst-Case** Szenario zu simulieren, wurden 30 Termine erstellt, welche unterschiedliche Gastlogos, inklusive GIFs, enthielten. Ein aktueller Termin wurde initialisiert, um die Animation des Timers zu starten. Hinzufügend wurde die Anwendung benutzt, um Termine zu buchen und zwischen dem dunklen und hellen Design zu wechseln. Die Anwendung zeigt durchschnittlich 60 Bilder pro Sekunde an. Da die maximale Wiederholrate des Bildschirms und des Browsers bei 60Hz liegen, besteht nicht die Möglichkeit, für diesen Anwendungsfall zu testen, ob noch mehr Bilder pro Sekunde angezeigt werden können. Nichtsdestotrotz sind damit die Anforderungen des RAIL Modells für Bilder pro Sekunde erfüllt.

5.9.4 Bottleneck

Wir wollen betrachten wie oft die Anwendung theoretisch und praktisch pro Sekunde aktualisiert werden kann. Für jede Kombination aus **Azure APP ID** und E-Mail-Adresse, dürfen 10000 Anfragen pro 10 Minuten gemacht werden. Dies sind 16,6 Anfragen pro Sekunde. Praktisch wurden eine bis drei Anfragen, je nach Bedarf, pro drei Sekunden durchgeführt. Dieser Bedarf hängt davon ab, ob sich an den Daten, die bei Microsoft gespeichert sind, etwas geändert hat und daran, ob Bilder-Anhänge für Gäste vorhanden sind, die nochmal separat per REST API abgefragt werden müssen. Pro Sekunde wären das also 0,33 bis 1 Anfragen. Einerseits ist das schnell genug für die Anwendung und gibt den langsamen Geräten, genug Zeit, die Anwendung zu aktualisieren, andererseits hat man so genug Anfragen übrig, falls jemand sein Konto mit mehreren Geräten benutzt oder dies in Kombinationen mit anderen Applikationen verwendet. Sollte dieses Limit überschritten werden, verlangsamt Microsoft die Anzahl an neuen Antworten auf die Anfragen. Da Microsoft auch Zeit benötigt, um die Anfragen zu bearbeiten und die neuen Daten bei sich zu verarbeiten, würde der Anwender den Unterschied selten bemerken. Der **Bottleneck** der Anwendung ist also in diesem Fall die Anzahl an Anfragen, die Microsoft, in ihren Servern, verarbeiten kann.

5.10 Optimierung

5.10.1 Abfrageoptimierung

Mithilfe der oDatav4 ([oData]) API wurden die Daten aus der jeweiligen Microsoft Datenbank im Voraus gefiltert, um so nicht notwendige Daten zu vermeiden und die Performance drastisch zu erhöhen, als auch Datenvolumen zu sparen. Die oData v4 API verhält sich dabei wie eine SQL Datenbank, wobei die Daten in Tabellen gespeichert sind. Es wurden die Klauseln ”$filter” und

”$top” verwendet, um nur Termine für den aktuellen und nächsten Tag zu erhalten und dies einzuschränken auf die top 300 Termine, falls jemand versucht das System zu überlasten. Eine Sortierung ist nicht notwendig, da die Termine bereits nach Startzeit sortiert sind. Hier sieht man die oData v4 API Anfrage, die an die Microsoft Graph API gesendet wird:

```
let url =  "https://graph.microsoft.com/v1.0/me/findMeetingTimes/?
    $filter=start/dateTime" +  "ge"  + "${todayDate} and end/dateTime
    le ${tomorrowDate}&$top=300";
```

”https://graph.microsoft.com/v1.0/me/findMeetingTimes/” ist die URL der Microsoft Graph API. Des Weiteren kommuniziert sie, welche API, spezifisch angesprochen werden soll. Da die Termine für den aktuellen und nächsten Tag benötigt werden, wird die ”FindMeetingTimes” verwendet, welche die Kalender API verwendet. Mit dem ”me” wird der aktuelle Benutzer mitgegeben. Dieser wird automatisch vom Microsoft Graph erkannt. Ab dem Fragezeichen werden die Query Parameter übergeben. Die Antwort der Anfrage, im Teil der URL, vor dem Fragezeichen, werden durch die Query Parameter auf die benötigten Daten gefiltert.
Mit ”start/dateTime” + ”ge” + ”todayDate” wird der Startzeitpunkt des Termins mit dem heutigen Datum verglichen. Es wird geprüft, ob der Startzeitpunkt des Termins nach dem heutigen Datum liegt. Das ”ge” steht für ”greater than or equal to” und bedeutet, dass der Startzeitpunkt des Termins nach oder gleich dem heutigen Datum liegt. Dies wird eigentlich für mathematische Vergleiche verwendet, aber es funktioniert auch für Strings, da jeder Charakter einen ASCII Wert hat.
Somit erhalten wir nur Daten zurück, die heute oder in der Zukunft liegen.
Mit ”end/dateTime” + ”le” + ” tomorrowDate” wird der Endzeitpunkt des Termins mit dem morgigen Datum verglichen. Letzten Endes erhalten wir nur Termine zurück, die heute oder morgen stattfinden. Die Verarbeitung dieser Daten erfolgt ISO 8601 konform. Sobald diese jedoch angezeigt werden, wird darauf geachtet, dass es die lokale Zeitzone ist.

6 Ergebnis

Abgleich der Ergebnisse mit den Soll-Anforderungen.

6.1 Visuelle Darstellung

Abbildung 4: Ergebnis mit nächstem anstehenden Termin

Abbildung 5: Ergebnis mit laufendem Termin

Oben links im Bild, ist das Logo des Gastgebers des nächsten, beziehungsweise jetzigen, Termins zu sehen. In diesem Fall ist dies das Logo der DOOH media GmbH. Solch ein Logo kann dargestellt werden, indem beim Erstellen des Termins, außerhalb des Tablets, bei Outlook beispielsweise, ein Bild an den Termin angehangen wird, welches im Dateinamen ”Termin_Logo” enthält.
Die anderen Logos sind alle Logos vom Gast des Termins. Sie werden angezeigt, indem der Firmenname im Textkörper des Termins vorkommt. Um diese Logos initial den Firmennamen zuzuordnen, muss ein spezieller Termin erstellt wird, der nur für die Logos gedacht ist und eine einzigarte ID, sowie einen Befehl enthält, die dann das Bild, inklusive Firmennamen, in einer lokalen Datenbank abspeichert. Diese Logos können hinzugefügt, gelöscht oder aktualisiert werden.
Die Uhrzeit wird immer in der Zeitzone des Tablets angezeigt.

Hier sieht man das Menü, in welchem ein Termin gebucht werden kann, welches durch das Drücken des Plus-Symbols aufgerufen wird:

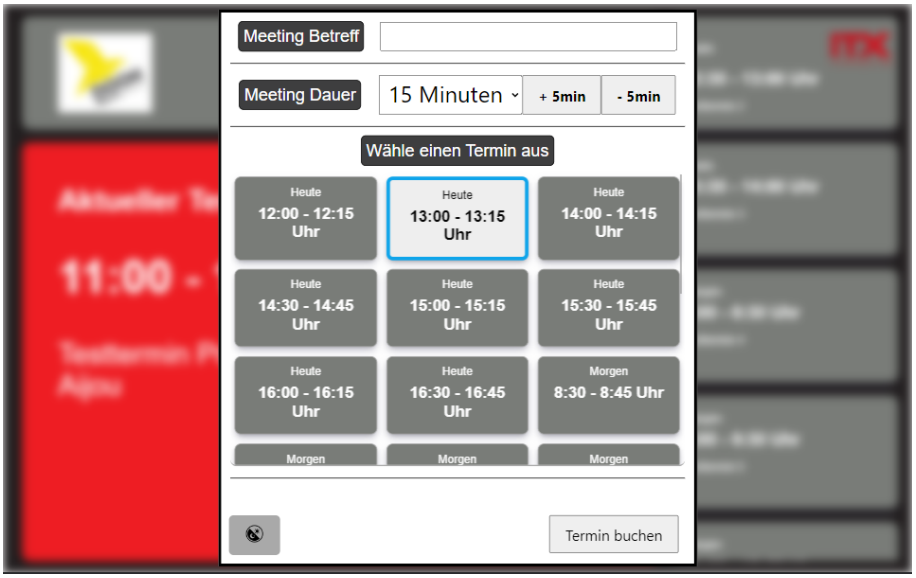

Abbildung 6: Terminbuchungsmenü

Die ”Jetzt” Option wird im Terminbuchungsmenü nicht angezeigt, weil bereits ein Termin stattfindet. Falls ein kein Termin stattfindet, wird eine ”Jetzt” Option angezeigt. Wenn das Ende des ”Jetzt” Termins innerhalb von 15 Minuten vom nächsten anstehenden Termin beginnt, wird der ”Jetzt” Termin rot markiert, sodass der Nutzer weiß, dass er den Termin nicht buchen kann, weil er zu lange dauert. Diese Option ist dann auch deaktiviert. So wird dem Nutzer übermittelt, dass die Option generell schon verfügbar ist, aber nur falls die Dauer verkürzt wird. Es werden alle möglichen Termine, innerhalb der Arbeitszeiten, für den Jetzigen und nächsten Tag angezeigt. Falls z.B. der nächste Tag ein Feiertag ist, werden für den nächsten Tag keine Termine angezeigt. Auf die anderen Optionen, wie z.B. Pufferzeiten, hat ein Anwender hier wenig Einfluss, da sie von den Einstellungen des Kalenders, des jeweiligen Nutzers, abhängen und es somit in der Verantwortung des Administrators liegt, diese zu ändern.

Hier die normale Ansicht nochmal, im hellen Design:

Abbildung 7: Ergebnis im hellen Design

6.2 Ausblick

6.2.1 kurzfristig

Die Software wird vom Kunden eingesetzt. Da dies ein Pilotprojekt ist, wird die Software nur von einem Kunden genutzt. Dafür wurden acht Tablets angeschafft, die an den Räumen des Kunden installiert werden.

6.2.2 mittelfristig

Der Kunde wird die Anwendung erstmal benutzen müssen. Rückmeldungen werden gesammelt und kritische Fehler werden behoben. Trotz ausgiebigen Tests, kann es immer noch zu Fehlern kommen, die erst im Einsatz auffallen.

6.2.3 langfristig

Die Schnittstelle wurde nicht vollständig genutzt, da die Anwendung nicht alle Funktionen benötigt, aber sie bietet für die Zukunft viele Erweiterungsmöglichkeiten. Mit der gesammelten Erfahrung und den gewonnenen Erkenntnissen kann die Anwendung in Zukunft erweitert werden. Zudem ist die API neu und wird ständig weiterentwickelt [microsoft-graph-api-version]. Insbesondere mit den jüngsten Entwicklungen von künstlichen Intelligenzen, wie z.B. der Spracherkennung, könnte die Anwendung erweitert werden.

6.3 Fazit

Von den Zielen wurden alle erreicht (Siehe 1.2). Die Implementierung, mithilfe der Microsoft Graph API, wurde als sinnvoll eingeschätzt (Siehe 4) und wurde dementsprechend auch implementiert refsec:technische-umsetzung. Diese ermöglicht synchronisierte Ressourcen- und Terminplanung mithilfe der selbst entwickelten Anwendung. Die Anwendung erfüllte mithilfe der Microsoft Graph API alle Anforderungen des Kunden(Siehe 3.2). Mithilfe der minimalen Benutzerführung, die die Anwendung bietet, kann der Anwender schnell und einfach Termine erstellen, löschen und aktualisieren. Die Leistung der Anwendung wurde mithilfe des RAIL-Modells bewertet(Siehe **??**) und erfüllte alle Anforderungen im vollen Umfang. Die Software ist durch den Einsatz von Babel

rückwärtskompatibel und kann somit auch von älteren Browsern genutzt werden. Aufgrund der sauberen Auftrennung von Funktionen und Komponenten, ist die Anwendung gut wartbar und erweiterbar. Die Anwendung ist auch für die Zukunft gut gerüstet, da die Microsoft Graph API ständig weiterentwickelt wird. Abschließend ist die Webapplikation kostengünstig, da sie lokal gehostet werden kann(Siehe 3.2.4)

7 Literaturverzeichnis

Literatur

[10BDL4551T/00] Phillips10BDL4551T/00. *Phillips10BDL4551T/00*. 1. Sep. 2022. URL: https://www.download.p4c.philips.com/files/1/10bdl4551t_00/10bdl4551t_00_pss_enggb.pdf?_ga=2.87916248.193099575.1678716502-1541604121.1678716502.

[Babel] Babel. *Babel*. 2023. URL: https://babeljs.io/.

[caniuse-indexedDB] caniuse-indexedDB. *caniuse-indexedDB*. 2023. URL: https://caniuse.com/indexeddb.

[EsLint] EsLint. *EsLint*. 2023. URL: https://eslint.org/.

[FirstInputDelay] FirstInputDelay. *First Input Delay*. 17. Aug. 2022. URL: https://web.dev/fid/.

[IndexedDB] IndexedDB. *IndexedDB*. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.

[interaction-design-book1] David Cronin Alan Cooper Robert Reimann. *About Face: The Essentials of Interaction Design*. 15. Mai 2007. ISBN: 978-0470084113.

[JavaScript] Mozilla. *JavaScript*. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[microsoft-graph-api-termin] microsoft-graph-api-termin. *microsoft-graph-api-termin*. 7. Feb. 2022. URL: https://docs.microsoft.com/en-us/graph/api/resources/event?view=graph-rest-1.0.

[microsoft-graph-api-termin-aktualisieren] microsoft-graph-api-termin-aktualisieren. *microsoft-graph-api-termin-aktualisieren*. 7. Feb. 2022. URL: https://docs.microsoft.com/en-us/graph/api/event-update?view=graph-rest-1.0&tabs=http.

[microsoft-graph-api-termin-bearbeiten] microsoft-graph-api-termin-bearbeiten. *microsoft-graph-api-termin-bearbeiten*. 7. Feb. 2022. URL: https://docs.microsoft.com/en-us/graph/api/event-update?view=graph-rest-1.0&tabs=http.

[microsoft-graph-api-termin-erstellen] microsoft-graph-api-termin-erstellen. *microsoft-graph-api-termin-erstellen*. 7. Feb. 2022. URL: https://docs.microsoft.com/en-us/graph/api/event-post-events?view=graph-rest-1.0&tabs=http.

[microsoft-graph-api-termin-loeschen] microsoft-graph-api-termin-loeschen. *microsoft-graph-api-termin-loeschen*. 7. Feb. 2022. URL: https://docs.microsoft.com/en-us/graph/api/event-delete?view=graph-rest-1.0&tabs=http.

[microsoft-graph-api-version] microsoft-graph-api-version. *microsoft-graph-api-version*. 7. Feb. 2022. URL: https://docs.microsoft.com/en-us/graph/overview?view=graph-rest-1.0.

[Microsoft-Graph-Explorer] Microsoft-Graph-Explorer. *Microsoft-Graph-Explorer*. 2023. URL: https://developer.microsoft.com/en-us/graph/graph-explorer.

[Microsoft-Graph-Toolkit] Microsoft-Graph-Toolkit. *Microsoft-Graph-Toolkit*. 2023. URL: https://docs.microsoft.com/en-us/graph/toolkit/overview.

[Microsoft-mgt-npm] Microsoft-mgt-npm. *Microsoft-mgt-npm*. 2023. URL: https://www.npmjs.com/package/@microsoft/mgt.

[microsoftGraphApi] microsoftGraphApi. *microsoftGraphApi*. 2023. URL: https://docs.microsoft.com/en-us/graph/overview.

[NodeJS] NodeJS. *NodeJS*. 2023. URL: https://nodejs.org/en/.

[NPM] NPM. *NPM*. 2023. URL: https://www.npmjs.com/.

[oAuth] oAuth. *oAuth*. 2023. URL: https://oauth.net/2/.

[OAuth-2.0-Simplified] Aaron Parecki. *OAuth 2.0 Simplified*. 4. Mai 2018.

[OAuth2.0$_R$edirect$_U$ri] OAuth2.0$_R$edirect$_U$ri. *OAuth2.0$_R$edirect$_U$ri*. 2023. URL: https://www.oauth.com/oauth2-servers/redirect-uris/.

[oData] oData. *oData*. 2023. URL: https://www.odata.org/.

[Ofe10] Stefan Szeider Ofer Strichman. *Theory and Applications of Satisfiability Testing - SAT 2010*. Springer-Verlag Berlin Heidelberg, 30. Okt. 2010. ISBN: 978-3642141850.

[RAILModell] RAILModell. *RAIL Modell*. 6. Jan. 2023. URL: https://web.dev/rail/#focus-on-the-user.

[Reflow] JavaScriptAnimationsReflow. *JavaScriptAnimations-Reflow*. 2023. URL: https://www.html5rocks.com/en/tutorials/speed/animations/.

[ResponsiveWebDesign] Ethan Marcotte. *Responsive Web Design*. 2011. ISBN: 978-0-9844425-7-7. URL: https://abookapart.com/products/responsive-web-design.

[REST] REST. *REST*. 2023. URL: https://restfulapi.net/.

[TotalBlockingTime] TotalBlockingTime. *Total Blocking Time*. 4. Juni 2023. URL: https://web.dev/lighthouse-total-blocking-time/.

[Vue.js] vuejs. *vuejs*. 2023. URL: https://vuejs.org/.

[WCAG] WCAG. *Web Content Accessibility Guidelines (WCAG) 2.1*. 25. Jan. 2023. URL: https://www.w3.org/TR/WCAG21/.

[Webpack] Webpack. *Webpack*. 2023. URL: https://webpack.js.org/.

8 Glossar

API Eine Schnittstelle, die es ermöglicht, auf Daten zuzugreifen.. 3

Azure AD Azure Active Directory ist ein Verzeichnisdienst, der von Microsoft entwickelt wurde.. 14

Azure APP ID Azure APP ID ist eine eindeutige Kennung, welche von Microsoft vergeben wird, um eine Applikation zu identifizieren.. 19

Bottleneck Der Bottleneck ist, in der Softwareentwicklung, die Stelle, an der die Leistung der gesamten Anwendung am meisten eingeschränkt wird.. 20

caniuse.com Caniuse is a website that shows you browser support for various features and includes references to the relevant specifications.. 17

Container Ein Container ist ein Element, welches andere Elemente enthält.. 7

cookie cache Ein Cookie ist eine kleine Textdatei, die von einem Webserver auf dem Computer des Benutzers gespeichert wird. Cookies werden verwendet, um Informationen über den Benutzer zu speichern.. 13

deployen Eine Anwendung zu deployen bedeutet, sie auf einem Server zu installieren und zu konfigurieren, sodass sie für den Benutzer verfügbar ist.. 13

Dory-node.js Dory-node.js ist eine Android App, die es ermöglicht, node.js Server auf einem Android Gerät zu hosten.. 14

Frameworks Eine Sammlung von Bibliotheken, die es ermöglichen, schneller und einfacher eine Anwendung zu entwickeln.. 2

Header Ein Header ist ein Element, welches bei HTML Seiten, die in einem Browser angezeigt werden, oben angezeigt wird.. 7

IDE Eine IDE ist eine Entwicklungsumgebung, die es ermöglicht, Software zu entwickeln.. 13

IMAP Internet Message Access Protocol. Ein Protokoll, welches es ermöglicht, E-Mails von einem Mailserver abzurufen.. 18

Kiosk-Modus Der Kiosk-Modus ist ein Modus, in dem ein Gerät, wie beispielsweise ein Tablet, nur eine Applikation ausführen kann.. 11

LED Strips LED Strips sind eine Art von LED Leuchtmittel. LED Strips sind in der Regel lang und dünn und werden in Streifen angebracht.. 8

Linter Ein Linter ist ein Programm, das es ermöglicht, Code auf Fehler zu prüfen.. 13

Local Storage Local Storage ist ein Speicherort, in Browsern, welcher Daten lokal speichert.. 16

localhost Die lokale Adresse des Computers, auf dem die Anwendung läuft.. 13

Node.js Server Ein Node.js Server ist ein Server, der auf Node.js basiert und JavaScript Code ausführt.. 13

NPM NPM steht für Node Package Manager.. 13

Player Ein Player ist ein Digital Signage Gerät, welches Inhalte anzeigt.. 14

Responsive Design Responsive Design ist ein Begriff aus der User Experience Design. Responsive Design ist ein Design, das sich an die Größe des Bildschirms anpasst.. 7

REST API REST API ist eine Abkürzung für Representational State Transfer Application Programming Interface. REST ist ein Architekturstil, der die Kommunikation zwischen verschiedenen Systemen ermöglicht. REST ist ein Architekturstil, der die Kommunikation zwischen verschiedenen Systemen ermöglicht.. 10

RESTful REST ist ein Synonym für Representational State Transfer. RESTful ist ein Architekturstil für die Entwicklung von Webdiensten.. 3

SICP SICP steht für Serial (Ethernet) Interface Communication Protocol. SICP ist ein Protokoll, welches es ermöglicht, mit dem Bildschirm direkt zu kommunizieren, ohne mit dem Android Betriebssystem zu interagieren.. 13

Single Page Application (SPA) Eine Single Page Application (SPA) ist eine Webanwendung, die nur eine HTML-Seite besitzt. Diese Seite wird beim Laden der Anwendung geladen und bleibt während der gesamten Nutzung der Anwendung bestehen.. 1, 18

Total-Blocking-Time (TBT) Die Total-Blocking-Time (TBT) ist eine Metrik, die die Gesamtzeit misst, die eine Seite blockiert wird, bis sie interaktiv ist.. 19

User Journey User Journey ist ein Begriff aus der User Experience Design. User Journey ist ein Weg, den ein User durchläuft, um ein Ziel zu erreichen.. 3

UserInterface UserInterface ist ein Begriff aus der User Experience Design. UserInterface ist die grafische Oberfläche, die ein User sieht und mit der er interagiert.. 5

Worst-Case Der Worst-Case ist der Fall, in dem die Leistung der Anwendung am schlechtesten ist.. 19

9 Anhang

9.1 Spezifikationsblatt

Spezifikationsblatt - Raumbuchungsanzeige

Contents

1. Übersicht: Lokal gehostete Single-Page-Anwendung für Raumbuchungen. Mit Azure-Integration für die Anmeldung von Benutzern.
2. Benutzeranmeldung: Integration von Azure zur Anmeldung von Benutzern.
3. Funktionalität:

- Raumbuchungsanzeige für angemeldete Benutzer.
- Nach Anmeldung muss maximal 3-6 Sekunden gewartet werden, ohne mit dem Gerät zu interagieren oder die Seite muss manuell neu geladen werden
- Anzeige von Terminen für den aktuellen Tag und nächsten Tag
- Buchung von Terminen.
- Informationen aktualisieren alle 3 Sekunden
- Farbliche Kennzeichnung vom nächsten Termin bzw. dem aktuell laufenden:
    - Grün für Termine, die 15 Minuten oder mehr entfernt sind.

    - Gelb für Termine, die weniger als 15 Minuten entfernt sind.
    - Rot für laufende Termine.
- LED-Strip-Anzeige für farbliche Kennzeichnung von Terminen.
- Für Querformat vorgesehen
- Bei langem Klicken auf einen „kleinen“ Termin, werden die Details des Termins dargestellt. Der Terminkörpertext wird jedoch aus Sicherheitsgründen nicht dargestellt

4. Technische Anforderungen:

- Unterstützte Browser: modernste Versionen von Chrome, Firefox, Edge und Safari.
- Lokale Hosting-Umgebung erforderlich.
- Internetverbindung
- Microsoft Arbeits- oder Schulkonto erforderlich (Ressourcen-Konten sind derzeit nicht unterstützt)
- Automatisches Akzeptieren von Meetings unterstützt Microsoft nur für Exchange/IMAP/POP3. Falls das Konto lediglich IMAP/POP3 ist, ist diese Funktionalität nicht gegeben. Buchungen lokal auf dem Gerät sind weiterhin möglich. Die Einrichtung hierfür erfolgt Seitens der Nutzerorganisation.
- Für die LED Strip Kontrolle ist ein 10BDL4551T vorgesehen. Andere Modelle können funktionieren, falls sie das gleiche SICP Protokoll unterstützen.
- Die Navigationskontrolle des Displays sollte ausgeschaltet werden, damit diese bei Texteingaben nicht permanent danach die Seite überdecken. Es sollte vorher Teamviewer o.ä. eingerichtet werden, um die Displays weiterhin kontrollieren zu können, falls notwendig.
- SICP muss bei dem Display angeschaltet sein und auf localhost:5000 hören
- Es benötigt zudem eine Applikation, die nodeJs code lokal hosten kann, wie z.B. „Dory-node.js“. Folgende Version wird benötigt: https://dorynode.firebaseapp.com/v10.15.1_arm_release/node

- Bei dory-node.js auf download file -> url eingeben -> appfiles anklicken -> executable auch und dann ok.

- Das Dory-node.js Script sollte auf automatisches Starten eingestellt werden.
- Der nodeJs Ordner mit dem nodeJs code für die hostende Applikation (in diesem Fall, Dory-node.js) muss in einem Verzeichnis abgespeichert sein, wo auch der Ordner „bookingPage“ existiert. In bookingPage muss die Buchungsseite hinterlegt sein und muss „index.html“ heißen. Es ist egal wo diese zwei Ordner abgespeichert werden, solange sie relativ zu einander, sich im gleichen Verzeichnis befinden.

5. Logos darstellen:

- Unterstützt werden PNGs, JPEGs (JPGs sind identisch), WebPs GIFs und SVGs (svg+xml)
- Um ein Gastgeber-Logo darzustellen, sollte ein Bild angehangen werden, welches „Termin_Logo“ im Namen beinhaltet. Groß- und Kleinschreibung sind nicht wichtig. Es kann nur ein Gastgeber Logo angezeigt werden und nur das erste welches auftaucht. Andere Fotos sind beeinflussen diesen Vorgang nicht. Diese können weiterhin normal angehangen werden.
  Dieses Gastgeber-Logo taucht nur auf, falls der Termin, den es betrifft, der Nächste bzw. Jetzige ist.
- Gästelogos werden durch den Termintextkörper identifiziert (damit ist nicht der Betreff gemeint). Eine Firma muss lediglich namentlich erwähnt werden und dann taucht die Firma als Gast auf. Es wird nur der erste Gast angezeigt.
  Um den Geräten mitzuteilen, wie das Logo eines Gasts aussieht, wird ein Termin erstellt und die Geräte zum Termin hinzugefügt, die diese Information erfahren sollen.
  Dieser Termin benötigt ein angehangenes Bild, welches das Logo darstellen soll und einen spezifischen Betreff. Der Betreff setzt sich wie folgt zusammen:
  uniqueKey + ADD/REMOVE/UPDATE + “Firmenname zum

Identifizieren für die Logodarstellung“
Ein Beispiel könnte wie folgt aussehen:
uniqueKey *ADD “Dooh“*
Die Groß- und Kleinschreibung von ADD/REMOVE/UPDATE und dem Firmennamen sind nicht wichtig. Der Schlüssel jedoch ist aus Sicherheitsgründen genau so festgelegt. Dieser Schlüssel sollte nur vertraulichen Personen weitergegeben werden.

6. Wartung und Support: Regelmäßige Überprüfung und Wartung, um die reibungslose Funktion der Anwendung zu gewährleisten. Kundensupport steht zur Verfügung, um bei Problemen oder Fragen behilflich zu sein.

9.2 Ablauf der Entwicklung

Nachdem der erste Prototyp fertig war und Rücksprache gehalten wurde, wurde angefangen die Anwendung zu entwickeln. Design und Funktionalität wurden dabei partiell parallel entwickelt, wobei die Funktionalität immer Priorität hatte. In über 95 Commits wurden die Änderungen an der Anwendung festgehalten.

Es wurde ein Testgerät benötigt, um die Anwendung zu testen. Ein Phillips 10BDL4551T/00 Bildschirm wurde dafür an die Wand gehängt und mit dem Internet verbunden. Die Internetverbindung ist notwendig, um die Rest-Anfragen zwischen der Anwendung und der Microsoft Graph API zu ermöglichen. Es wurde so eingerichtet, wie es auch beim Kunden laufen soll.

Die Anwendung wurde täglich aktiv genutzt, um so Fehler zu finden und Feedback zu geben. Das Feedback wurde dann evaluiert. Solche explorativen Tests sind sehr wichtig, um Intuitivität und Benutzerfreundlichkeit zu gewährleisten. Es war einerseits hilfreich, um vorgesehene Abläufe zu testen, andererseits aber auch um Fehler zu finden, die nicht vorgesehen waren, indem Monkey Testing [Ofe10] betrieben wurde.