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

Mentor: **Prof. Dr. Ing. 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 mit Hilfe von Microsoft Graph API. Die Arbeit ist in drei Teile gegliedert. Im ersten Teil wird die Theorie der Ressourcen- und Terminplanung in Office365 mit Hilfe 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

Die Fragestellungen der Arbeit lauten:

- Wie funktioniert die Ressourcen- und Terminplanung in Office365 mit Hilfe von Microsoft Graph API?
- Wie kann die Ressourcen- und Terminplanung in Office365 mit Hilfe von Microsoft Graph API praktisch umgesetzt werden?
- Wie kann die Arbeit abschließend bewertet werden?

1.2 Ziele der Arbeit

Die Ziele der Arbeit sind deshalb wie folgt:

- Die Theorie der Ressourcen- und Terminplanung in Office365 mit Hilfe von Microsoft Graph API zu erläutern.
- Die praktische Umsetzung der Theorie zu beschreiben.
- Die Arbeit abschließend zu bewerten.

1.3 Use Case

Use Case Titel	Ressourcen- und Terminplanung in Office365 mit Hilfe von Microsoft Graph API 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.

1.4 Systemschnittstellen

Kurzbeschreibung:	Im UI kann der User sich mit der Ressource einloggen, die für die Zukunft als Repräsentant für die reelle Ressource (z.B einen Raum) gelten soll. Dann gilt dieser User immer als Teilnehmer der Buchungen und kann in seinen verfügbaren Zeiten gebucht werden. Der User kann zudem andere Mitarbeiter hinzufügen, um sie einerseits über das Meeting zu informieren und andererseits, um automatisch überprüfen zu können, wann alle am besten Zeit haben. Darauf basierend werden dem User verfügbare timeslots zur Buchung angezeigt. Auch die Dauer des Termins soll einstellbar sein. Es gibt zudem einen optionalen Betreff für den Termin. Der gebuchte Termin reflektiert sich dann im UI in der Liste der für den Tag schon gebuchten Termine, für die Ressource. Zudem soll der Nutzer sehen können ob innerhalb der nächsten 30 Minuten ein Termin für die Ressource ansteht oder ob einer schon am Passieren ist, anhand einer farblichen UI . REST API Anfragen werden dann an die Microsoft Graph API geschickt, um diese Wünsche zu kommunizieren.
Akteure:	Mitarbeiter
Häufigkeit:	Variablel pro Ressource, Firma und Anzahl Ressourcen
Komplexität:	Mittlere bis hohe Komplexität
Vorbedingungen:	1. User muss eingeloggt sein (Nutzerdaten können nicht gesehen oder eingespeichert werden) und Zugriffsrechte akzeptieren. 2. Optionale Teilnehmer 3. Termindauer 4. Verfügbare Termine 5. Terminbetreff 6. Zeitzone 7. Für die Microsoft Graph API kommt noch der Access Token / Client ID von unserer Azure App zur Authentifizierung dazu
Ausgaben:	1. Buchung erfolgreich oder nicht 2. Neue verfügbare Termine 3. Der neue Termin wird in der Liste der gebuchten Termine angezeigt 4. Falls die Buchung nicht geklappt haben sollte, wird der User darüber informiert
Use Case:	1.3
Schnittstellen:	1. Microsoft Graph API 2. User Interface 3. Backend
Aufgerufene Aktionen:	Notwendige: 1. Termin buchen 2. Termin finden Optionale: 1. Teilnehmer hinzufügen 2. Terminbetreff hinzufügen/ändern 3. Termindauer ändern

2 Theorie

2.1 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. Mit Hilfe dieser API wurde das Projekt letztendlich umgesetzt. Weitere standen jedoch zur Verfügung:

- Microsoft Outlook API
- Microsoft Exchange API
- Microsoft SharePoint API
- Microsoft OneDrive API
- Microsoft Teams API
- Microsoft Power Automate

Einige dieser APIs sind nur für bestimmte Microsoft 365 und Office 365 Abonnements verfügbar. Die Microsoft Graph API ist jedoch für alle Abonnements verfügbar. Zudem ist die Microsoft Graph API die einzige API, die es einem erlaubt auf alle Daten von Microsoft 365 und Office 365 zuzugreifen, da sie die meisten anderen APIs integriert. Der wichtigste Faktor bei der Entscheidung war es jedoch, dass die Microsoft Graph API, mit Hilfe von Azure AD, die Authentifizierung und Autorisierung von Benutzern erlaubt. Dies ist für die Anwendung von großer Bedeutung, da es dem Benutzer ermöglicht sich mit seinem Microsoft 365 oder Office 365 Account anzumelden und somit auf seine Daten zuzugreifen.

3 Ist-Zustand

Unsere Muttergesellschaft 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 jedoch nicht immer so praktikabel wie sie es vielleicht sein sollte, vor allem nicht, wenn mehrere Unternehmen, das gleiche Gebäude und die gleiche Organisations-E-Mail besitzen.
//UserJourney Beispiel einfügen
Falls ein User zufällig an einem Raum vorbeigeht oder vor einem Raum steht und sich fragt, ob dieser Raum belegt ist oder nicht, muss er erstmal Outlook oder Teams auf einem Gerät öffnen, zum Kalender des jeweiligen Raumes, falls er darauf überhaupt Zugriff hat und dann schauen, ob dieser Raum belegt ist oder nicht. Dies ist sehr umständlich und kann sehr viel Zeit in Anspruch nehmen.

3.1 Definitionen

- RESTful: RESTful ist ein Synonym für Representational State Transfer. RESTful ist ein Architekturstil für die Entwicklung von Webdiensten.
- : API steht für Application Programming Interface. API ist eine Schnittstelle, die es einem erlaubt auf Daten zuzugreifen.
- : REST steht für Representational State Transfer. REST ist ein Architekturstil für die Entwicklung von Webdiensten.
- : HTTP steht für Hypertext Transfer Protocol. HTTP ist ein Protokoll, das es einem erlaubt auf Daten zuzugreifen.
- : JSON steht für JavaScript Object Notation. JSON ist ein Datenformat, das es einem erlaubt Daten zu speichern und zu übertragen.
- : OAuth ist ein Protokoll, das es einem erlaubt sich mit einem Account anzumelden und somit auf Daten zuzug
- UserJourney: UserJourney ist ein Begriff aus der User Experience Design. UserJourney ist ein Weg, den ein User durchläuft, um ein Ziel zu erreichen.
- UserInterface: UserInterface ist ein Begriff aus der User Experience Design. UserInterface ist die grafische Oberfläche, die ein User sieht und mit der er interagiert.

4 Soll-Zustand

4.1 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 heute noch anstehen und spontan auch einen Termin 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 am User Interface, als auch an den LED Strips des Tablets.

4.2 User Interface

Das User Interface soll so gestaltet sein, dass der User sofort erkennen kann, ob der Raum belegt ist oder nicht. Zudem soll das User Interface so gestaltet sein, dass der User auch Termine für den Raum vereinbaren kann. Das User Interface soll also eine Übersicht über die Termine des Raumes und einen Button zum vereinbaren eines Termins enthalten.

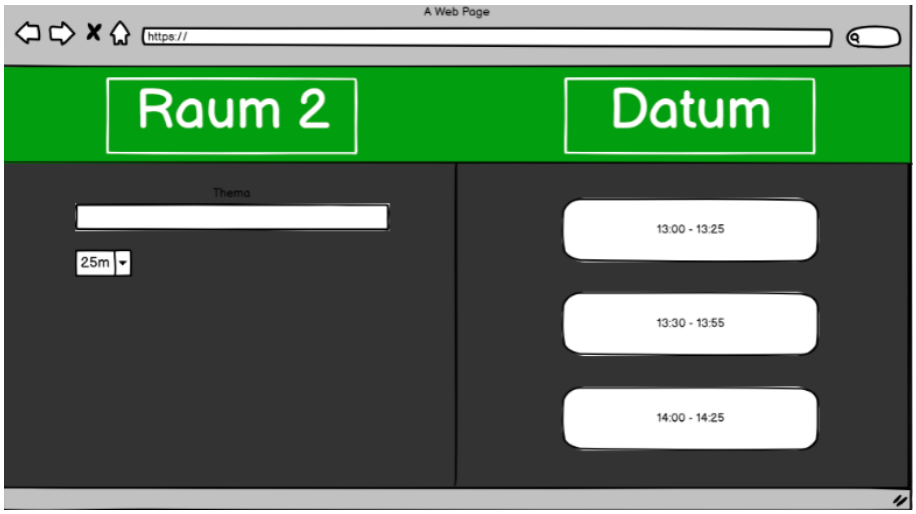
4.3 LED Strips

Die LED Strips sollen die Farbe des Raumes anzeigen. Wenn der Raum belegt ist, sollen die LED Strips rot leuchten. Wenn der Raum frei ist, sollen die LED Strips grün leuchten. Wenn der Raum nicht verfügbar ist, sollen die LED Strips rot leuchten. Falls jedoch der Raum in den nächsten 15 Minuten belegt sein wird, sollen die LED Strips gelb leuchten.

5 Vorgehensweise

5.1 Prototyp

Erst wurde ein Prototyp mithilfe von vuejs entwickelt, basierend auf folgendem Mockup:



6 Technische Umsetzung

Das ganze wird als eine Azure App deployed. 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 Acccount. Somit ist keine Individualisierung notwendig.
Es wird lokal mithilfe von Dory-node.js gehostet.Das ganze wird als eine Azure App deployed. Diese lässt localhost Verbindungen zu und gibt einem die Möglichkeit die Web-Applikation wird als Single-Page-Application,mit einem lokalen cookie cache für den eingeloggten Acccount, entwickelt. Somit ist keine Individualisierung notwendig. Zudem kann dann mit dem Phillips Display die Web-Applikation aufgerufen werden und als eine Art Model eines Model-Ciew-Controller's verwendet werden.
Der Player braucht 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-nodejs Server antwortet und die tatsächliche Seite anbietet, die lokal auf dem Display liegt. So ist die URL immer die Gleiche, auf allen Geräten, damit Microsoft's redirect URI Bedingungen erfüllt werden können und Webserver kosten eingespart werden können, da Updates eher selten passieren sollten.
Zudem ist es auch für die Sicherheit der Nutzer so viel besser, da wir deshalb keinen Zugriff auf ihre Daten haben. Zeiten müssen ISO 8601 Konform sein bei jeglichen API Anfragen an die Microsoft Graph API.