smino

OST Ostschweizer Fachhochschule

Microsoft Teams smino App

Bachelorarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Frühjahrssemester 2022

Autoren:	Daniel Els, Loris Keller
Betreuer:	Prof. Dr. Markus Stolze, OST, IFS
Projektpartner:	smino AG, 8645 Rapperswil-Jona
Experte:	Markus Flückiger, Zühlke Engineering AG
Gegenleser:	Prof. Dr. Daniel Patrick Politze

Abstract

smino ist ein Produkt, welches Baufirmen bei jeder Leistungsphase, von der Planung bis zur Schlüsselübergabe, unterstützt. Mit smino können unter anderem Aufgaben und Mängel schnell und einfach erfasst werden. Bilder, Dokumente und Planmarkierungen können leicht hinzugefügt und den verantwortlichen Personen zugewiesen werden. Durch die vermehrte Nutzung von Microsoft Teams wünscht sich auch smino herauszufinden, wie ihre Funktionalitäten in die Plattform integriert werden können.

Ein erstes Ziel dieser Arbeit war das Einarbeiten in die Microsoft Teams Entwicklung und das Herausarbeiten von Möglichkeiten der verschiedenen Teams Funktionalitäten in Bezug auf die Integration der smino Funktionalitäten. Das Hauptziel ist die Integration der smino Aufgaben in ein Microsoft Teams Tab eines Kanals oder eines Chats. In dieser Phase ging es um den Aufbau von Kompetenzen in der Entwicklung mit Microsoft Teams und den entsprechenden Integrationsmöglichkeiten, welche Microsoft Teams zur Verfügung stellt. Mittels Prototypen wurden die verschiedenen Ansprüche von smino validiert und auf deren Machbarkeit geprüft.

In einer zweiten Phase wurde, basierend auf den Komponenten aus der smino Web-Applikation, eine smino Teams Applikation entwickelt. Dabei wurde zuerst die Authentifizierung des smino Benutzers im Teams Kontext und die anschliessende Teams Tab-Konfiguration mit einem smino Projekt implementiert. Für die Implementation der Aufgaben mussten neue Funktionalitäten implementiert und auch komplexe, bestehende Komponenten der smino Applikation überarbeitet werden. Des Weiteren wurde ein Theming-Konzept erarbeitet und ein Dark-Theme implementiert, um das Dark-Theme der Teams Applikation selbst zu unterstützen. Für die Applikation wurde eine entsprechende Provisionierung in Teams und Hosting mit Microsoft Azure umgesetzt.

Abgeschlossen wurde die Arbeit mit einem User-Testing der smino Aufgaben in der Beta-Version von der smino Microsoft Teams Applikation.

Management Summary

Ausgangslage

Smino ist ein motiviertes IT-Startup aus Rapperswil-Jona mit grossen Ambitionen. Mit http://smino.ch ist die Firma gerade dabei, die Baubranche fundamental zu revolutionieren. Sie entwickeln moderne, digitale Werkzeuge für Planungs- und Bauprojekte, vernetzen die Bauherrschaft, Behörden, Planer und Unternehmen zu einer effizienten Einheit. Dabei stellt die Web-Applikation von smino diverse Funktionen zur Verfügung, welche die verschiedenen Parteien bei der Ausführung ihrer Arbeit unterstützen. Heutzutage ist Home-Office ein viel geläufiger Begriff, als dies noch vor einiger Zeit der Fall war. Dadurch haben auch Tools wie Microsoft Teams einen enormen Aufschwung erlebt. Auch die Nutzer von smino verwenden Teams tagtäglich als zentrales Cockpit für die Bearbeitung aller anstehenden Arbeiten. Aufgrund dessen würde sich smino wünschen herauszufinden, wie smino in Teams integriert werden kann, um die Nutzer bei Ihrer Arbeit besser zu unterstützen.

Ziel und Vorgehen

Ziel ist eine erste Integration der smino Funktionalitäten in Microsoft Teams. Vorgängig soll eine Analyse der Teams-Integration vorgenommen werden. In Zusammenarbeit mit smino sollen anschliessend die ersten Funktionalitäten definiert und entsprechend umgesetzt werden. Dies bedeutete, dass zum Beginn der Arbeit vor allem der Aufbau von Kompetenzen für die Entwicklung mit Teams und die Einarbeitung in die bestehende smino Web-Applikation im Vordergrund standen. In einem zweiten Schritt wurden zwei Prototypen, je einer für React und Angular, erstellt, welche die geplante Integration validieren, sowie die Lösungsfindung zur Definition der Technologien und Architektur unterstützen. Dies bildete die Basis für die Implementation der effektiven Funktionalitäten von smino. Das finale Ziel war es, die smino Aufgaben in einer produktiven Beta-Version in Teams bereitzustellen.

Ergebnis

In der Arbeit wurden Funktionalitäten für die Ansicht und Bearbeitung von smino-Aufgaben als Tab in Teams intergiert. Die Anwendung verwendet dabei das Teams Toolkit für die Verwaltung der Teams-Anwendung selbst, wobei die Tabs Web-Applikation mit Angular entwickelt wurde. Um das Tab hinzuzufügen und die Verwendung der Aufgaben zu ermöglichen, wurde die Authentifizierung mit smino integriert und die Konfiguration eines Aufgaben-Tabs umgesetzt. Ebenso wurde eine entsprechende Architektur und automatisierte Bereitstellung der Anwendung auf Microsoft Azure entwickelt. Die Teams-Applikation wird automatisiert in eine Organisation publiziert. Abgeschlossen wurde die Arbeit mit einem User-Testing der Funktionalitäten.

Ausblick

In einem nächsten Schritt können weitere Teams-Funktionalitäten, oder auch «Capabilites» genannt, integriert werden. Message Extension für Link Unfurling könnten benutzt werden, um Rich-Content für eine smino Aufgabe anzuzeigen, wenn ein Link zu dieser im Chat verwendet wird. Dabei wird es den Nutzern auch ermöglicht zu Kollaborieren und im Chat mit dem Content zu interagieren. Ebenso könnte ein Bot eingebunden werden, welcher eine Hilfestellung zur Verwendung der Applikation gibt oder dem Nutzer eine weitere Möglichkeit für die An- und Abmeldung des smino Kontos in Teams bietet. Die Möglichkeiten sind nahezu endlos und es kann noch viel Potential ausgeschöpft werden. Die Erfahrungen und Konzepte könnten schlussendlich auch dazu verwendet werden, weitere Funktionalitäten von smino in Teams abzubilden.

Inhaltsverzeichnis

1. Einführung

1.1 Motivation

Immer mehr Menschen arbeiten von zuhause, was dazu geführt hat, dass Anwendungen wie Microsoft Teams enorm in den Vordergrund gerückt sind, um remote kommunizieren zu können. Um einem Nutzer ein angenehmes Arbeiten mit Teams zu ermöglichen, möchten wir gerne herausfinden, wie smino in Microsoft Teams integriert werden kann, um den Workflow mit smino für die Nutzer zu verbessern.

Die Nutzer von smino nutzen Microsoft Teams als zentrales Cockpit, wobei alle möglichen Anwendungen zur Erledigung Ihrer Arbeiten in den entsprechenden Teams integriert werden. Aus diesem Grund, soll auch smino für Teams verfügbar gemacht werden.

Durch eine Benutzerbefragung wurde zuerst evaluiert, welche Funktionalitäten von smino in Teams integriert werden sollen oder auch, welche zusätzliche Funktionen sich die Benutzer in der Microsoft Teams smino App wünschen würden.

1.2 Ziel der Arbeit

Die Arbeit umfasst eine Recherche, welche die Integrationsmöglichkeiten von Microsoft Teams aufzeigt. Dabei sollen die Entwickler Kompetenzen im Bereich der Teams Entwicklung aufbauen und smino bei der Integration ihrer Funktionalitäten in Teams unterstützen.

Eine mögliche Integration kann verschiedene Funktionalitäten beinhalten wie z.B. eigene Tabs für smino Projekte, ein Chatbot, welcher Aufgaben erstellen kann oder eine Kalenderintegration, welche Ereignisse von smino abbildet. Auch muss berücksichtigt werden, wie ein Benutzer sich auf Microsoft Teams mit smino verbindet.

Ziel der Arbeit ist die Umsetzung der ersten Funktionalitäten von smino in Microsoft Teams.

Konkret wurden folgende Ziele für die Arbeit definiert:

1. Architektur Prototyp zur Entwicklung von Microsoft Teams App (inklusive Authentifizierung)
2. Testkonzept für Microsoft Teams App erstellen
3. Provisionierung der Applikation (Hosting/Store/O365 Plattform)
4. Erarbeitung einer Dokumentation für die Entwicklung mit Microsoft Teams
5. Integration von Funktionalitäten der smino Applikation in Teams als Tabs (MVP)
6. Architektur für eine Production-Ready Applikation mit der Integration von smino Komponenten
7. (Optional) Erarbeitung weiterer Integrationsmöglichkeiten (Bot, Meeting Extensions, Chat Extensions)

1.3 Vorgehen

Zu Beginn der Arbeit ging es um den initialen Aufbau von Kompetenzen in der Teams-Entwicklung und die Einarbeitung in die smino-Domain. Dabei wurden die einzelnen Teams Funktionen evaluiert und ein erstes Konzept für die Authentifizierung sowie für die Einbindung von Tabs erstellt.

In einem nächsten Schritt wurden Prototypen für Angular und React entwickelt, um das Konzept zu validieren und die zu verwenden Technologien, wie auch die Architektur der Anwendung zu validieren.

Der Hauptteil, die Construction Phase, ist die effektive Implementierung der smino Aufgabenfunktionalitäten in Microsoft Teams. Als Vorlage dazu dienen die Screen-Designs, welche durch smino bereitgestellt wurden. Dabei wurden auch die Komponenten der smino Web-Applikation in die Anwendung integriert, teils ergänzt und teils komplett überarbeitet.

Zum Schluss wurden die Bereitstellung und Publizierung der Teams Applikation erarbeitet. Dabei wurde die Architektur für die automatisierte Bereitstellung und notwendige Infrastruktur definiert.

2. Kontextabgrenzung

Dieser Abschnitt soll einen Einblick in den Gesamtkontext der smino Teams Applikation geben. Einerseits auf einer fachlichen und andererseits eben auch auf einer technischen Ebene. Es sollte hervorgehen, in welchem Umfeld sich die Teams App bewegt und wo sich diese positioniert.

2.1 Fachlicher Kontext

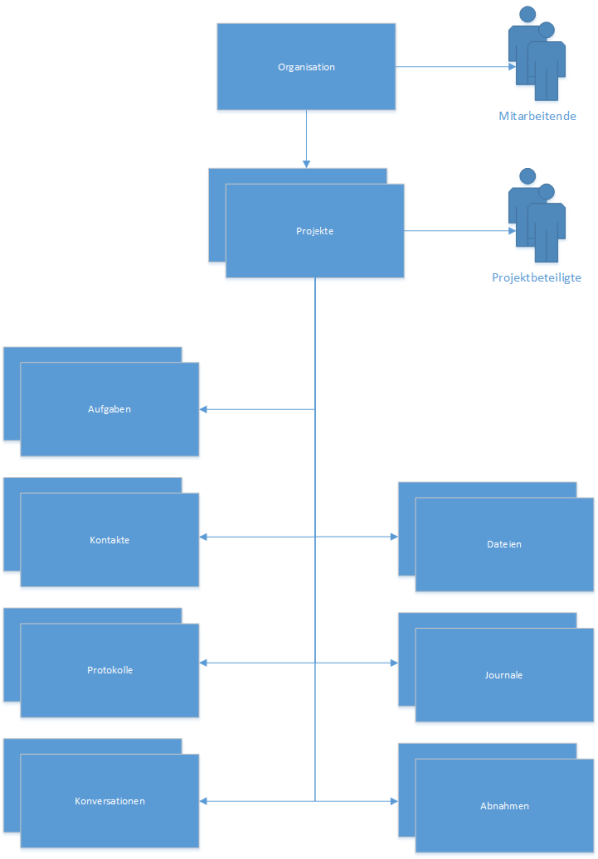


Abbildung 1: Übersicht smino Funktionalitäten (eigene Darstellung)

Smino bietet diverse Funktionalitäten an, welche die Vernetzung der verschiedenen Parteien in einem Bauprojekt zum Ziel hat. Der Grundbaustein bildet dabei die Organisation als «oberstes» Element. Für eine Organisation werden die Mitarbeitenden mit ihren entsprechenden Rechten verwaltet. Unterhalb einer Organisation werden Projekte erstellt. Für Projekte können ebenfalls Projektbeteiligte hinzugefügt werden und eben auch Beteiligte, welche nicht Teil der eigenen Organisation sind.

Im Kontext eines Projekts werden diverse Funktionalitäten geboten, welche die Arbeit in einem Bauprojekt digital stützen und erleichtern sollen.

Die **Kontakte** ermöglichen Kontaktinformationen für alle Projektbeteiligten einzusehen und zu pflegen. Die Projektbeteiligten können hier von fremden Organisationen hinzugefügt werden, wobei diesen auch eine Rolle – Bauherrschaft, Planer, etc. – zugewiesen wird.

Aufgaben bieten die Möglichkeit, anstehende Arbeiten zu erfassen. Eine Aufgabe kann eine effektive Aufgabe, sowie auch einen Mangel darstellen. Eine Aufgabe kann einer Person zugewiesen werden und entsprechend auch terminiert werden. Ebenso kann der Status für eine Aufgabe gepflegt werden, um diese beispielsweise als «Erledigt» zu markieren. In einer Aufgabe können *Dateien* verknüpft werden oder auch eine Konversation zur Aufgabe eröffnet werden. Zusätzlich können Aufgaben hierarchisch strukturiert werden, wobei einer Aufgabe weiter Unteraufgaben hinzugefügt werden können.

Mithilfe der **Protokolle** können einfach Sitzungen geplant und dokumentiert werden. Dabei werden verschiedene Sitzungstypen, wie Amtssitzungen oder Planersitzungen unterschieden. Sitzungen werden dann in sogenannte Sitzungsreihen aufgeteilt, wo sich die effektiven Einladungen, Termine und Protokolle wiederfinden.

Konversationen erlauben einen einfachen Austausch zwischen den Projektbeteiligten. Dabei wird für eine Konversation ein Titel und die Beteiligten bestimmt, welche es erlaubt, Nachrichten untereinander auszutauschen.

Für ein Projekt dürfen auch relevante **Dateien** nicht fehlen. Die Dateiablage erlaubt es diverse Dokumente, sowie auch Pläne und Modelle für ein Projekt zentral abzulegen und den Zugriff auf diese zu steuern.

Das **Journal** erlaubt es Einträge, Vorfälle oder Bilddokumentationen mit einem Datum festzuhalten. Auch hier können entsprechend relevante Dateien oder Dokumente verlinkt werden.

Für ein Bauprojekt sollte es auch möglich sein, **Abnahmen** in einer strukturierten Form zu erfassen und zu bearbeiten. Die Abnahme wird ebenfalls mit einem Datum und einer Adresse versehen, wobei auch die relevanten Teilnehmenden für die Abnahme definiert werden. Für eine Abnahme wird das Resultat mit potenziellen Unterschriften der Teilnehmenden festgehalten. Eine Mängelliste kann auch direkt für die Abnahme erstellt werden.

2.2 Technischer Kontext

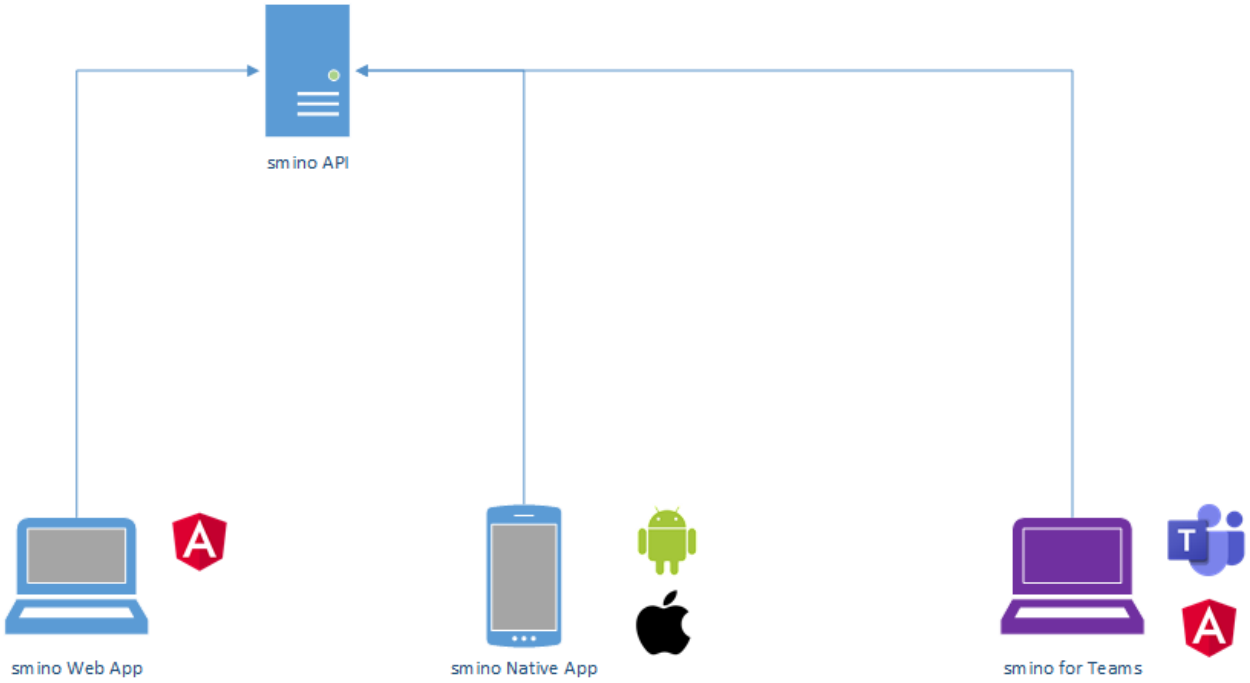


Abbildung 2: Technischer Kontext (eigene Darstellung)

Smino bietet eine Web-Applikation an, welche in Angular entwickelt wurde. Zudem wird auch eine native Applikation angeboten, welche über den Google Play Store und den Apple Store vertrieben wird. Die Applikationen beziehen die Daten über die smino API, welche dafür eine REST Schnittstelle zur Verfügung stellt. Die API fungiert auch als Security Token Service und bietet einen OpenID Connect Endpunkt[1], um Tokens für die Anwendung zu beziehen. Der Endpunkt erlaubt auch die Registration von neuen, externen Clients.

Grundsätzlich wird primär zwischen der Ninja Umgebung und der produktiven smino Umgebung unterschieden, welche über das öffentliche Netz verfügbar sind. Die Ninja Umgebung ist dabei für das Testen der Anwendung konzipiert. Diese bietet auch eine umfängliche Swagger Dokumentation[2], welche alle möglichen Operationen der API und die verwendeten Schemas dokumentiert.

Neu soll die Microsoft Teams Anwendung in die bestehende Architektur eingebunden werden.

[1] https://api.smino.ninja/.well-known/openid-configuration
[2] https://api.smino.ninja/swagger

3. Microsoft Teams Development

Die Microsoft Teams-Plattform bietet eine grosse Vielfalt an Funktionen an. Die verschiedenen Funktionen bieten diverse Integrations- und Interaktionsmöglichkeiten für Unternehmen, die ihre bestehenden Lösungen in Microsoft Teams integrieren wollen.

Teams Apps können Tabs, Bots, Messaging-Extensions oder auch irgendeine Kombination aus diesen drei Kategorien – auch «Capabilities» genannt – sein. Diese können persönliche oder auch geteilte Applikationen sein, welche in Teams oder Gruppen verwendet werden können.

Nachfolgend soll eine Übersicht über die Entwicklung und Funktionalitäten von Teams gegeben werden.

Abbildung 3: Microsoft Teams Apps Übersicht (Microsoft, 2022)

3.1 Teams Toolkit

Das Teams Toolkit ist ein Hilfsmittel für die Entwicklung mit Teams und unterstützt dabei alle Entwicklungsaspekte wie die Erstellung, Verwaltung sowie auch die Bereitstellung und Publizierung der Applikation. Das Teams Toolkit ist als Extension für Visual Studio Code, Visual Studio oder auch als CLI über NPM verfügbar. Im Besonderen bietet das Toolkit folgende Vorteile: (Microsoft, 2022)

- Integrierte Funktionalitäten für Microsoft Identity
- Zugriff auf Cloud Storage
- Zugriff auf Daten mittels Microsoft Graph
- Azure und Microsoft 3065 Service mit «zero-configuration approach"

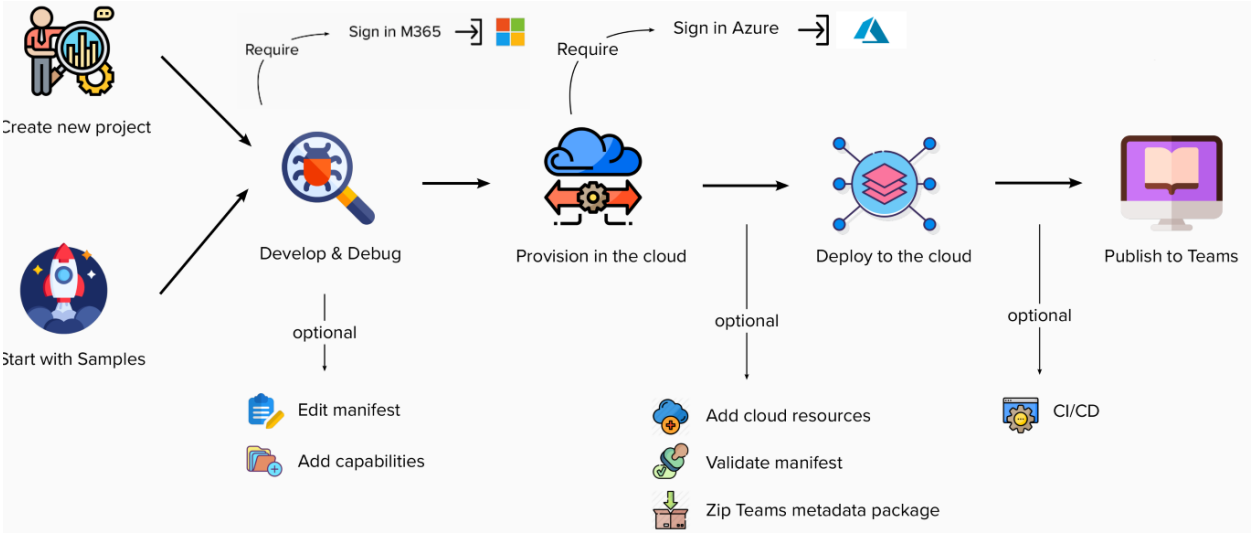


Abbildung 4: User journey of Teams Toolkit (Microsoft, 2022)

Eine erste stabile Version des Toolkit Plugins für Visual Studio Code wurde erst mit der Version 4.0.0 am 24.05.2022 veröffentlicht. Die zum Startzeitpunkt dieser Arbeit verfügbare Version 3.7.* war noch als «preview» markiert und es musste noch mit Bugs und grösseren Changes gerechnet werden.

Obwohl die Extensions für die Entwicklungsumgebungen sehr angenehm sind, werden teilweise bei Updates automatische Migrationen vorgenommen. Das Toolkit verwendet dabei nicht die installierte Version der CLI, wodurch die Kontrolle über die angewendete Version eher schwierig ist. Dies kann aber auch durch die Preview Versionen bedingt sein, wobei sich das Verhalten zukünftig ändern kann. Ansonsten würde sich die ausschliessliche Verwendung der CLI eher für die Entwicklung eignen.

3.2 Bots

Ein Bot, oder auch Chatbot genannt, ist eine Applikation, welche einfache und sich wiederholende Aufgaben von Benutzern ausführen. Beispielsweise kann dies ein Wetterdienst oder auch ein Supportdienst sein. In Teams unterstützen Bots auch komplexere Interaktionen, wobei diese durch Text, Interactive Cards oder auch durch Task Modules abgebildet werden können. Ebenso kann die Authentifizierung mit Drittanbietern im Bot gelöst werden. (Microsoft, 2022)

Die meisten Apps bieten sicher eine minimale Interkation mit einem Bot, wobei sicher eine Hilfestellung zur Benutzung der Applikation vorhanden ist, sowie eine Funktion für die An- und Abmeldung des Benutzers. Aufgrund der Komplexität und der Priorisierung wurde auf die Implementierung eines Bots verzichtet.

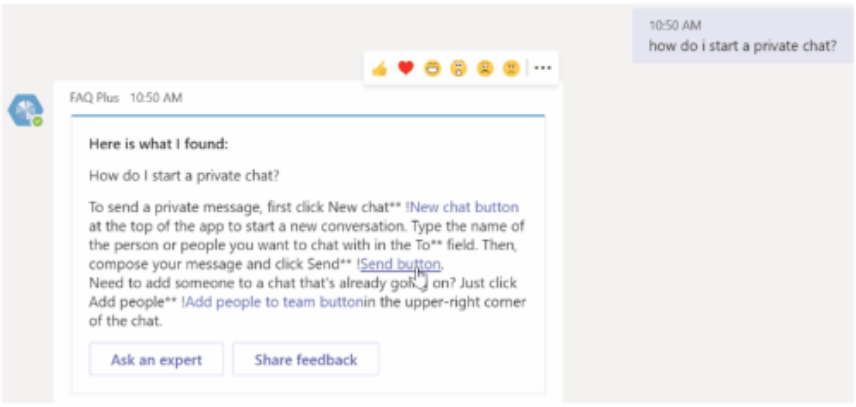


Abbildung 5: Bot Text (Microsoft, 2022)

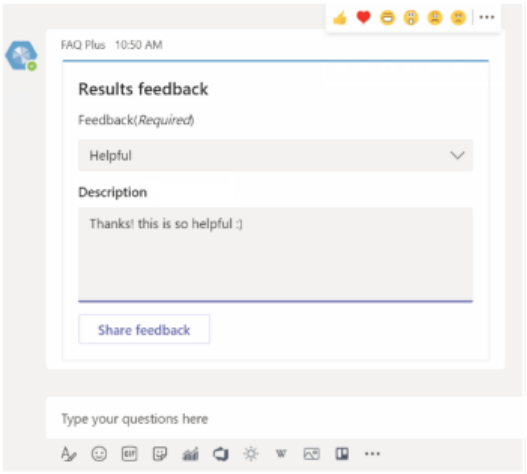


Abbildung 6: Bot Interactive Card (Microsoft, 2022)

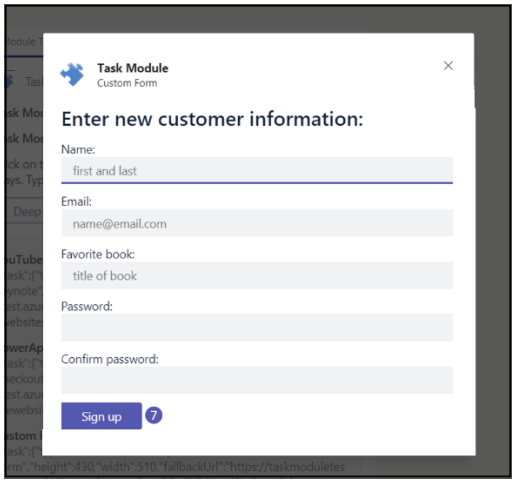


Abbildung 7: Bot Task Module (Microsoft, 2022)

3.3 Message Extensions

Message Extensions erlauben es dem Nutzer über Buttons und Formulare im Teams Client mit einem Webservice zu interagieren. Dabei sind die Funktionen beim Erfassen einer Nachricht verfügbar. Ergebnisse der Interaktion können in Form von «Rich Content» an den Teams Client zurückgegeben werden. Für die Message Extensions werden 3 Typen unterschieden: (Microsoft, 2022)

- **Action Commands:** Erlauben das Öffnen von Modalen/Task Modules, wobei die Antwort des Webservices in Form einer Message direkt in die Konversation oder den Erfassungsbereich eingefügt wird.
- **Search Commands:** Erlauben dem Benutzer Content von einem externen System zu durchsuchen und entsprechend für eine Nachricht zu verwenden.
- **Link Unfurling:** Wenn eine URL in das Nachrichtenfeld kopiert wird, kann ein Webservice für zusätzliche Informationen angefragt werden. Diese können dann ebenfalls wieder in Form von «Rich Content» in der Nachricht verwendet werden.

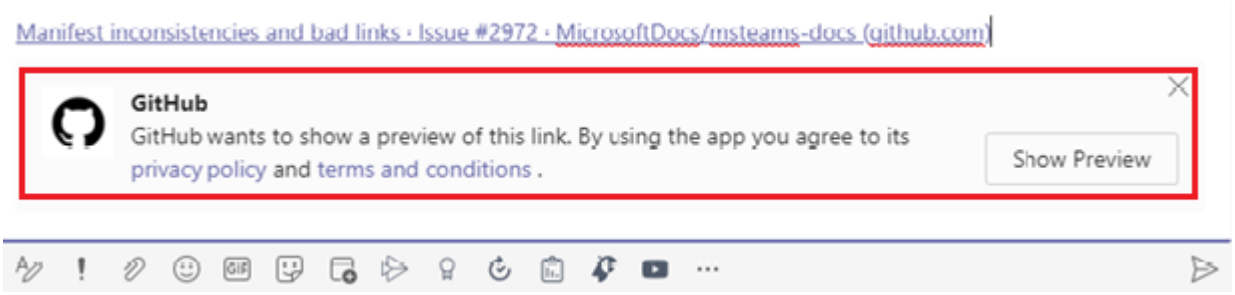


Abbildung 8: Link Unfurling (Microsoft, 2022)

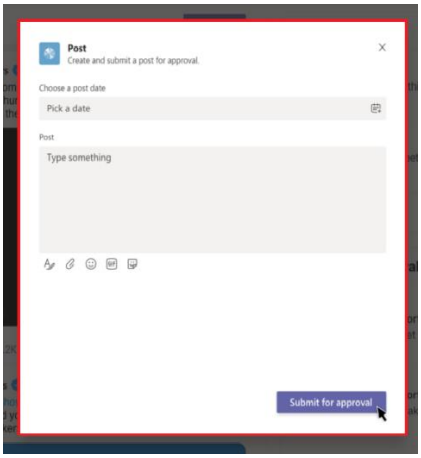


Abbildung 9: Action Commands (Microsoft, 2022)

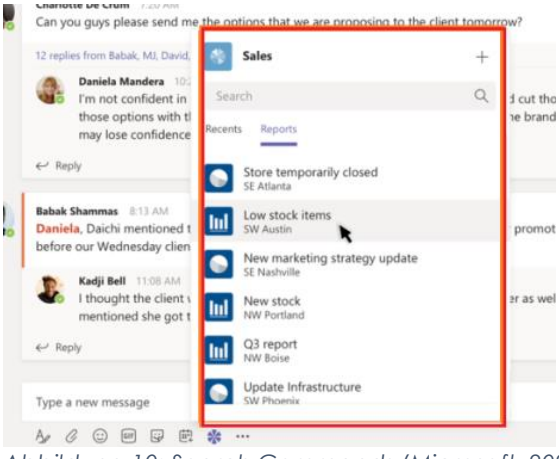


Abbildung 10: Search Commands (Microsoft, 2022)

3.4 Webhooks und Connectors

Webhooks und Connectors bieten die Möglichkeit, Channels in Microsoft Teams mit Webservices zu verbinden. Dabei können von Teams aus Nachrichten via @mention an einen externen Webservice versendet werden – Outgoing Webhooks –. Andererseits kann auch ein Endpunkt erstellt werden, mit dem ein Webservice Nachrichten an Teams senden kann, welche empfangen werden und weiterverarbeitet werden können – Incoming Webhooks –. (Microsoft, 2022)

3.5 Task Modules

Task Modules ermöglichen es, komplexe Modale in Teams zu öffnen. Dabei ist die Funktionsweise ähnlich wie bei Tabs in dem Sinn, dass in dem Task Module eine `iframe` zur Anzeige der Konfigurierte Page benutzt wird. Auf mobilen Geräten wird ein Task Module, bzw. das entsprechende Modal entsprechend als eigene «Seite» dargestellt. (Microsoft, 2022)

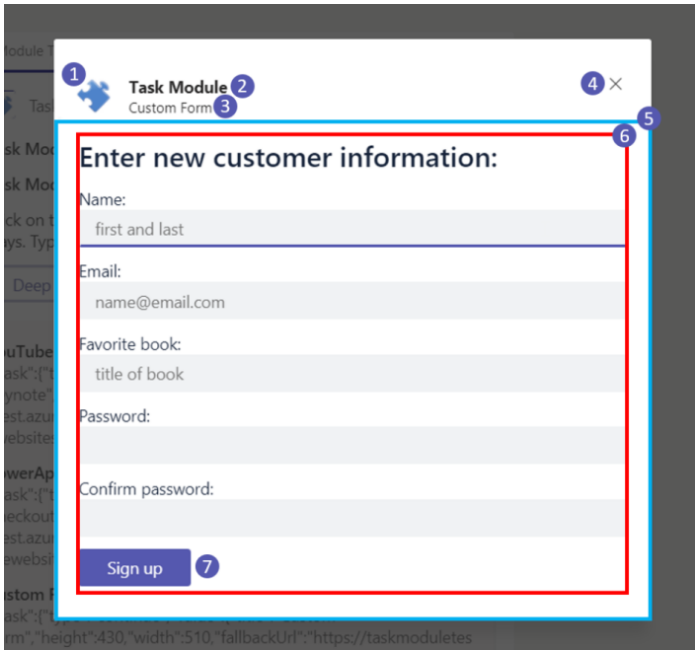


Abbildung 11: Task Module (Microsoft, 2022)

3.6 Tabs

Tabs sind «Teams-aware» Webseiten, welche in Microsoft Teams integriert werden. Schlussendlich besteht ein Tab aus einer Konfiguration und einem `iframe`, welches den entsprechenden Content der hinterlegten Url darstellt. Tabs können innerhalb eines Channels in einem Team, in einer Gruppenkonversation oder auch persönlich für einen Nutzer hinzugefügt werden. (Microsoft, 2022)

3.7 Teams JavaScript Client SDK

Das Teams JavaScript Client SDK ermöglicht es der Web-Applikation, welche über ein `iframe` in Teams, Office oder Outlook integriert werden, mit dem entsprechenden Kontext zu interagieren. Das SDK bildet eigentlich die Schnittstelle zwischen den entsprechenden Clients und der Applikation. Dabei wird die Authentifizierung in Teams für Tabs über das SDK angestossen und das entsprechende Resultat oder der Fehler empfangen.

Momentan wird für die Anwendung die Version 1.1.1 verwendet. Während der Arbeit wurde die Version 2.0.0 veröffentlicht. Die einzige wesentliche Änderung dabei ist jedoch nur, dass diese nun auch Outlook und Office als Clients unterstützt. (Microsoft, 2022)

3.8 Guidelines für die Entwicklung mit Tabs

Für das Veröffentlichen einer Teams App im Microsoft Teams Store müssen einige Guidelines berücksichtigt werden. Dabei wird zwischen «Mandatory Fix» und «Suggested Fix» unterschieden, wobei ein «Mandatory Fix» zwingend so umgesetzt werden muss, dass die Teams App im Store aufgenommen wird.

Diese Guidelines spielen aber nur dann eine Rolle, wenn die Teams App im offiziellen Store eingebunden werden soll. Wird die Teams App über die Organisationen vertrieben, können diese vernachlässigt werden. Dennoch ist es empfehlenswert, sich daran zu halten.

Nachfolgend sind die wichtigsten Guidelines festgehalten. Dies hilft auch dabei, ein Verständnis für die Funktionsweise der Tabs aufzubauen.

Die wichtigsten Punkte, die bei der Entwicklung mit Tabs nach Microsoft betrachtet werden sollten, sind: (Microsoft, 2022)

- **Kollaboration:** Es soll den Benutzern ermöglicht werden, mit dem Tab-Content zu interagieren und eine Konversation zu führen. Ein Tab sollte nicht wie eine weitere Webpage behandelt werden.
- **Navigation:** Tabs sollten spezifische Probleme lösen und für einen Anwendungsfall optimiert sein. Es sollte auf keinen Fall eine ganze Web-Applikation als Tab eingebunden werden.
- **Einrichtung:** Die Einrichtung des Tabs soll so einfach und schnell wie möglich gehen. Es sollten nur essenzielle Information für das Hinzufügen der Tabs gefordert werden.
- **Theming:** Es sollten Teams Color Tokens für das Theming verwendet werden, da Teams ein eigenes Farbschema definiert. Damit werden Änderungen am Theme automatisch übernommen.

3.8.1 Tab Konfiguration

Um ein Tab zu konfigurieren, existiert in Teams ein Konfigurationsdialog. Dieser kann frei gestaltet werden, wobei eine Beschreibung der App oder die Auswahl von Einstellungen behandelt werden. Die Authentifizierung der eignen App sollte ebenfalls an diesem Punkt behandelt werden. (Microsoft, 2022)

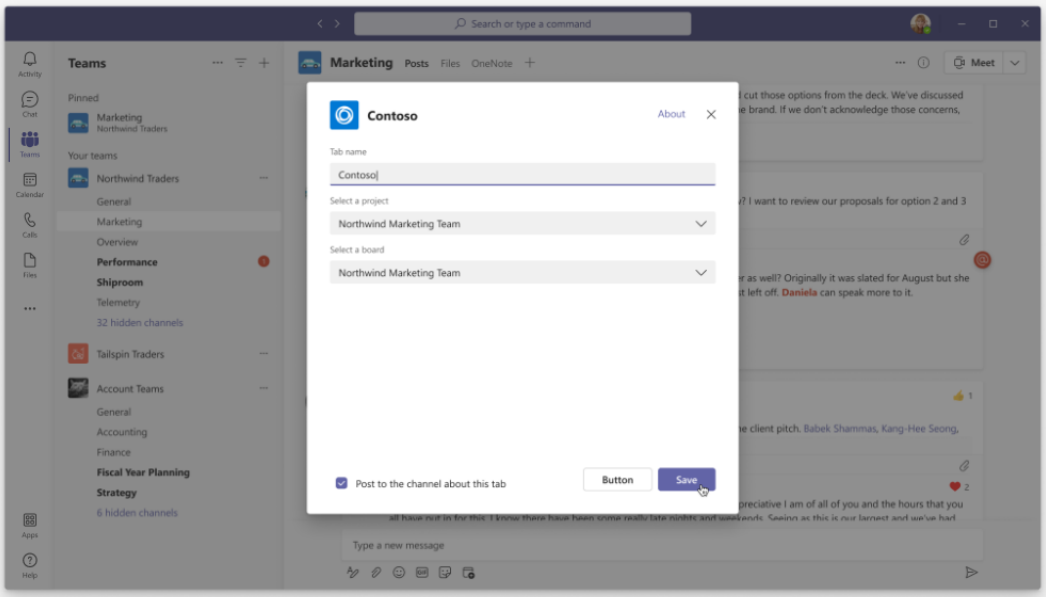


Abbildung 12: Tab Konfiguration Beispiel (Microsoft, 2022)

3.8.1.1 Anatomie

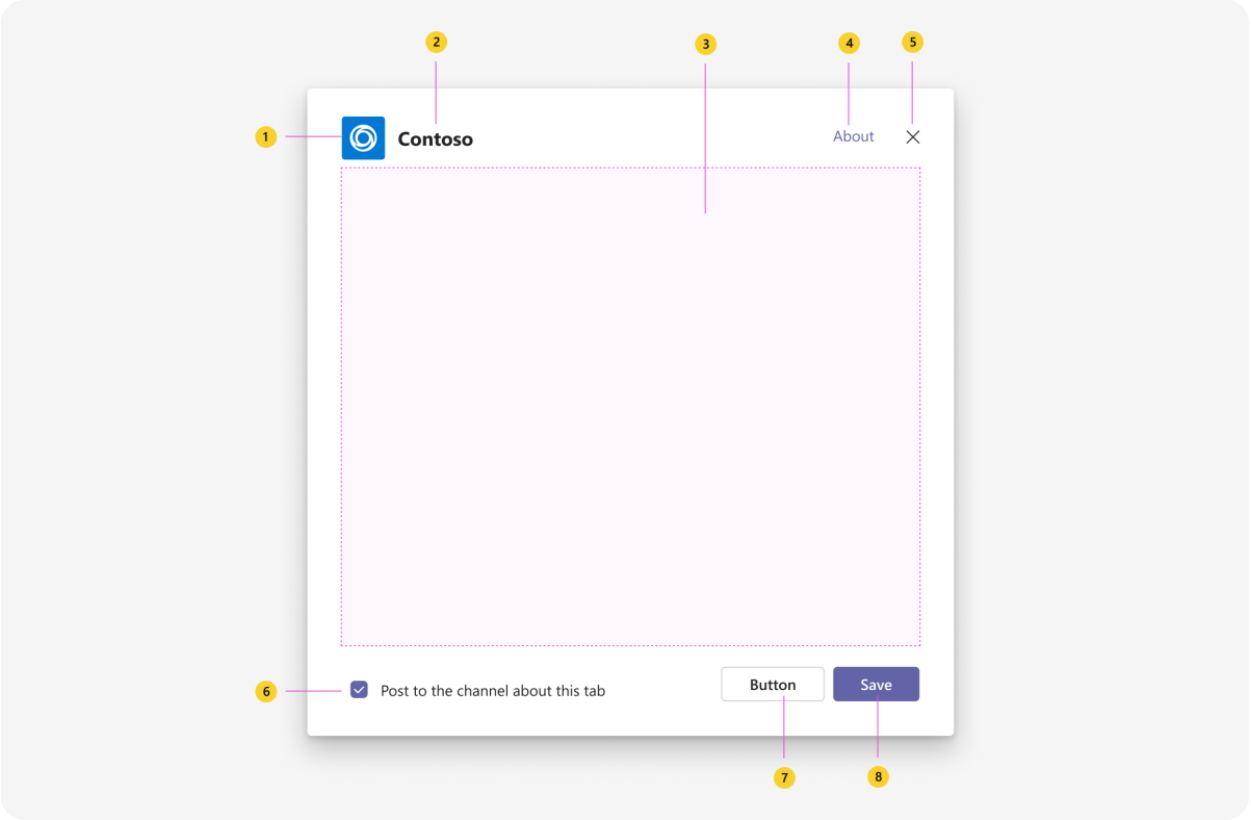


Abbildung 13: Tab Konfiguration Anatomie (Microsoft, 2022)

Nr.	Name	Beschreibung
1	**App Logo**	Farbenreiches, komplettes Logo der App
2	**App Name**:	Vollständiger Name der App
3	`iframe`	Responsiver Bereich für den Inhalt der App. (z.B. für Tab-Einstellungen oder Authentifizierung)
4	**About link**	Öffnet einen Dialog mit weiteren Informationen über die App. Unter anderem die komplette Beschreibung, die benötigten Berechtigungen, sowie die Links zu den «privacy policy» und «terms of service» der App.
5	**Close button**	Schliesst den Dialog.
6	**Notify team members option**	Informiert andere Benutzer dieses Kanals über der Erstellung des neuen Tabs.
7	**Back button**	Rücksprung zum vorherigen Schritt der Konfiguration / Authentifizierung
8	**Save button**	Stellt die Konfiguration des Tabs fertig.

Tabelle 1: Tab Konfiguration Anatomie (Microsoft, 2022)

3.8.2 Tab

Tabs sind im Grundsatz full-screen "Webseiten", welche in Teams integriert werden. Diese können genutzt werden, um mit kollaborativen Content zu interagieren oder wichtige Informationen anzuzeigen.

3.8.2.1 Anatomie

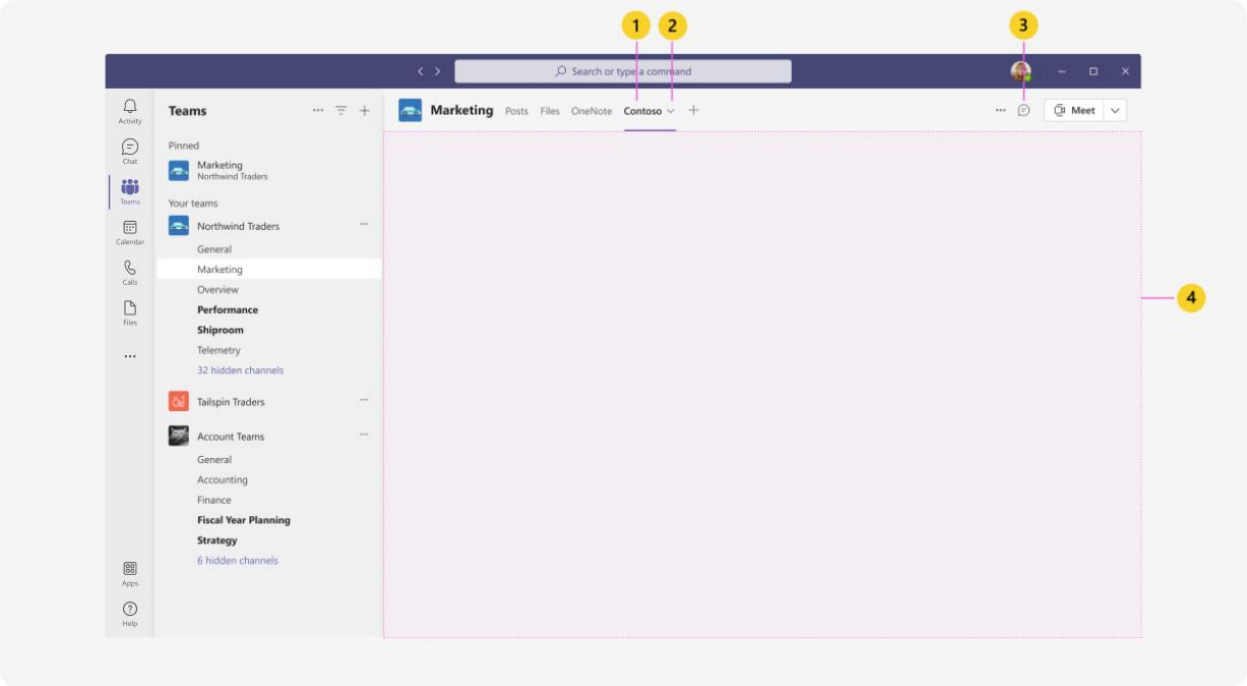


Abbildung 14: Tab Anatomie (Microsoft, 2022)

Nr.	Name	Beschreibung
1	**Tab name**	Navigationslabel für den Tab
2	**Tab overflow**	Öffnet die Tab-Aktionen, wie z.B. Tab umbenennen, Tab neu laden oder Tab löschen
3	**Tab chat**	Öffnet einen Chat zur rechten Seite, welcher es den Benutzern erlaubt, eine Konservation in Bezug auf diesen Tab zu führen
4	`iframe`	Responsiver Bereich für den Inhalt der App.

Tabelle 2: Tab Anatomie (Microsoft, 2022)

3.8.3 Tab Chat

Benutzer können Konversationen führen, welche im Zusammenhang mit dem Tab Content stehen.

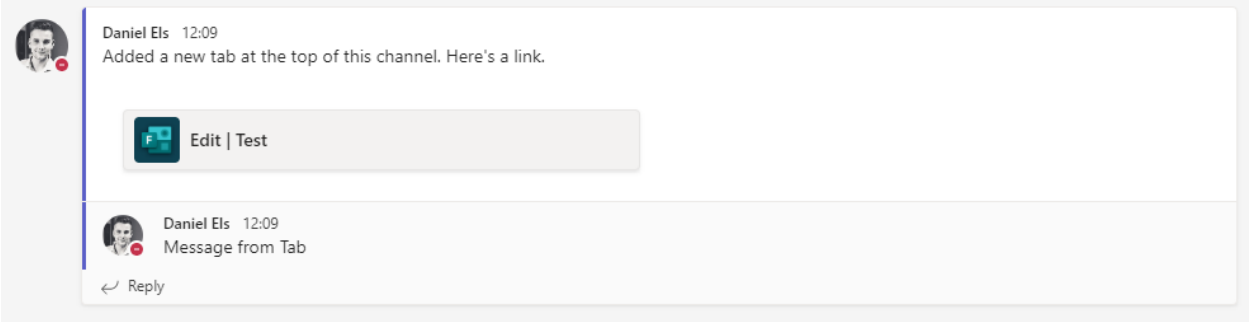


Abbildung 15: Tab Chat Beispiel (eigene Darstellung)

3.8.4 Manage a Tab

Für ein Tab können Einstellungen auch nach dem Einbinden konfiguriert werden oder der Tab kann umbenannt oder entfernt werden.

3.8.4.1 Anatomie

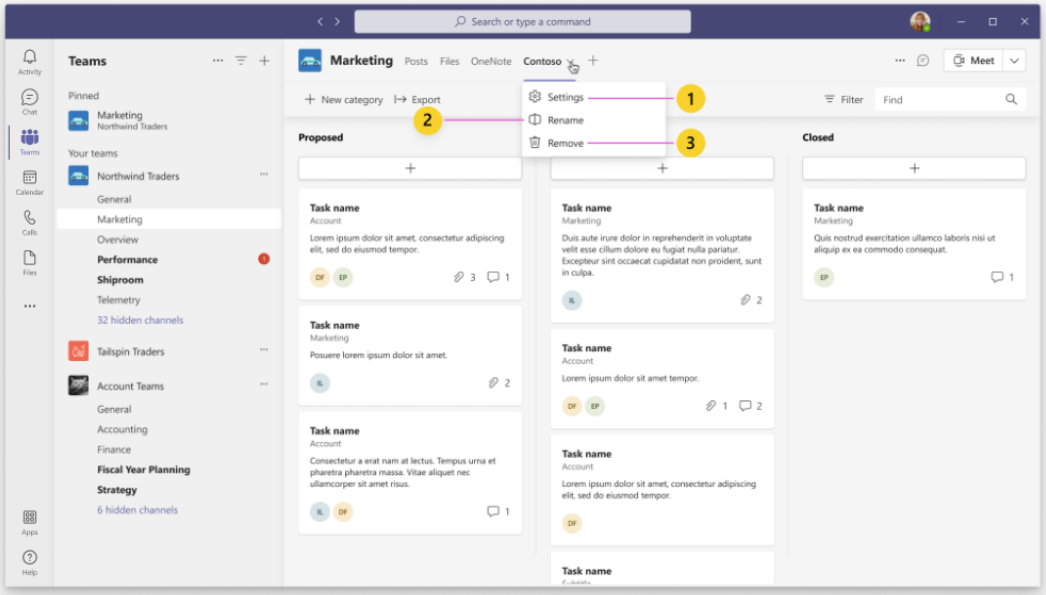


Abbildung 16: Verwaltung von Tabs Anatomie (Microsoft, 2022)

Nr.	Name	Beschreibung
1	**Settings** *(Optional)*	Erlaubt dem Benutzer die Tab-Einstellungen nach dem Hinzufügen des Tabs zu ändern.
2	**Rename**	Benutzer können dem Tab einen eigenen Namen geben
3	**Remove**	Entfernt den Tab aus dem Kanal, Chat oder Meeting

Tabelle 3: Verwaltung von Tabs Anatomie (Microsoft, 2022)

3.9 Anwendungsbeschreibung

Die Anwendungsbeschreibung wird beim Hinzufügen einer Applikation und auf der Info- (About-) Page der eingebundenen Applikation von Microsoft Teams dargestellt.

Diese Texte müssen die gleichen Inhalte haben, wie die Texte für die "AppSource" (Beschreibungen der Applikation im Store)

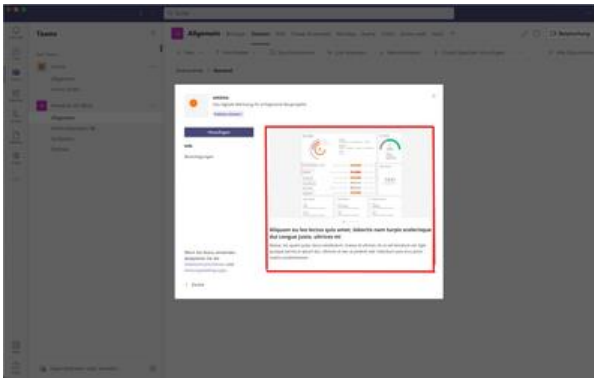


Abbildung 17: Anwendungsbeschreibung - Neue App hinzufügen (eigene Darstellung)

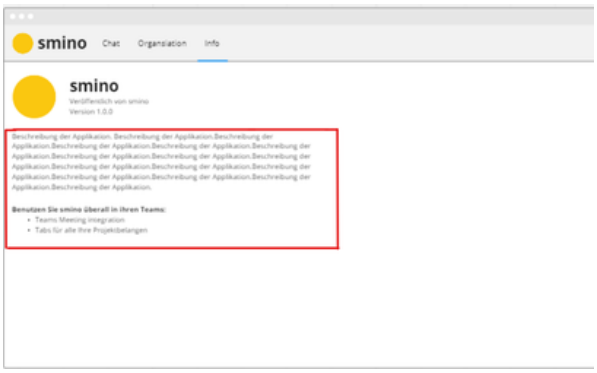


Abbildung 18: Anwendungsbeschreibung - App Info-Seite (eigene Darstellung)

Diese Texte werden aus dem Manifest von Microsoft Teams aus dem "description" Schema gelesen.

Es wird keine statische HTML Webseite geladen, die ein Styling der Seite mit HTML, css oder JavaScript ermöglichen.

(Microsoft, 2022; Microsoft, 2022)

3.9.1 Short description

Eine prägnante Zusammenfassung der Applikation, die originell und ansprechend sein sollte und sich an die smino Zielgruppe richtet. Diese Beschreibung sollte ungefähr die Länge eines Satzes haben.

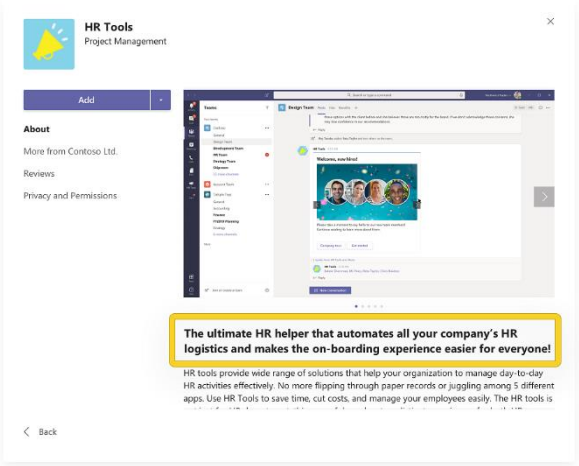


Abbildung 19: Anwendungsbeschreibung - Short description (Microsoft, 2022)

3.9.2 Long description

Die ausführliche Beschreibung sollte Information enthalten, die die Anwendung hervorhebt:

- Hauptmerkmale
- Die Probleme, die sie löst
- Zielpublikum

Die Beschreibung darf bis zu 4'000 Zeichen enthalten, es wird aber empfohlen jeweils nur 300 - 500 Wörter anzugeben.

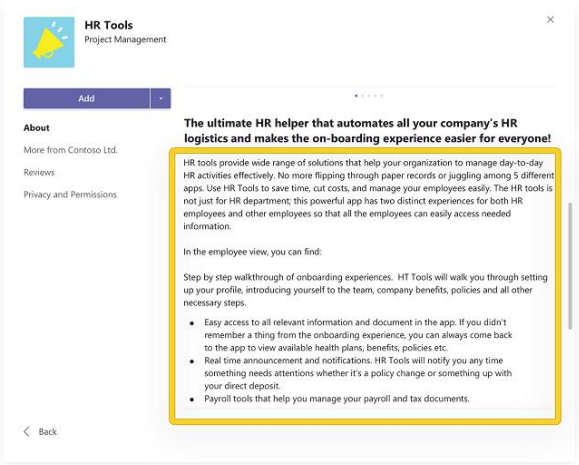


Abbildung 20: Anwendungsbeschreibung - Long description (Microsoft, 2022)

3.9.3 Styling

Für das Styling der Beschreibung kann auf eine vereinfachte Version von Markdown zurückgegriffen werden.

Dieses vereinfachte Markdown bietet die folgenden Optionen:

- **Bold Text**
- *Italic Text*
- Nummerierte Liste
- Unnummerierte Liste
- Hyperlinks
- Block- und Inline-Code
- Einen einfachen Titel

(Microsoft, 2022)

4. Grundlegende Design Decisions

Grundsätzlich ist die Entwicklung nicht auf die Verwendung eines spezifischen Technologie-Stacks beschränkt. Die Entwicklung von Tabs in Microsoft Teams ist schlussendlich durch die technologischen Mittel im Web-Bereich beschränkt. In ihrer Dokumentation lassen sich Guidelines und Beispiel-Implementationen für die folgende Technologien finden:

- JavaScript (incl. React)
- SharePoint Framework (SPFx)
- C#
- Node.js

4.1 Vorgehen

Für die jeweiligen Technologien wurde, wo nötig, eine Analyse vorgenommen, wobei ausgewählte Technologien miteinander verglichen wurden, um Vor- und Nachteile auszuarbeiten. Dies bildet die Basis für die Entscheidungsfindung des Projektteams. Ein Hauptpunkt bildet dabei die Wahl zwischen React und Angular. In der Arbeit wurden zwei Prototypen erarbeitet, welche die jeweiligen Funktionalitäten demonstrieren und validieren. Die Prototypen wurden zudem verwendet, um die folgenden Aspekte in Relation zu setzen

- **Setup**: Komplexität des Setups und Integration in die bestehenden Entwicklungsumgebung.
- **Dokumentation:** Inwiefern bereits Dokumentationen, Guidelines, Code-Beispiele und weitere Hilfestellungen vorhanden sind.
- **Styling Framework:** Die verschiedenen Möglichkeiten von Styling Frameworks.

4.2 Gegenüberstellung React und Angular

Das Microsoft Teams Toolkit verwendet standardmässig React für neue Projekte, wobei das Template auch auf die Entwicklung damit ausgelegt ist. React ist ein «First-Class Citizen» im Teams Bereich, wobei das Styling Framework von Teams auch direkt unterstützt wird. Die smino Applikation wird jedoch mit Angular entwickelt. In diesem Abschnitt werden jeweils die Vor- und Nachteile dieser beiden Technologie-Stacks erläutert, um einen Entscheid treffen zu können.

Die grundsätzliche Vorgabe von Teams ist, dass man den Teams Style / das Teams Theme verwenden soll. Dabei ist aber zu beachten, dass dies ein Suggested Fix ist. Ein eigenes Theming / Styling Framework zu entwerfen wird hier nicht weiter berücksichtigt, da der dafür geschätzte Aufwand zu gross ist.

4.2.1 React

React ist eine einfache JavaScript Library für das Erstellen von Single-Page-Applikationen. Das Framework muss dabei selbst definiert werden und setzt stark auf externe Libraries und Frameworks, die eingebunden werden.

4.2.1.1 Setup

Aufgrund des «*First-Class Citizen*» Status von React ist das Generieren einer neuen Teams App mit dem Teams Toolkit sehr einfach. Auch das Debugging ist integriert und gestaltet die Entwickler-Experience somit sehr positiv.

4.2.1.2 Dokumentation

Die Teams Development Dokumentation, Guidelines und Code-Beispiele basieren ebenfalls auf React.

4.2.1.3 Styling Frameworks

React bietet das First-Class Citizen Package `@fluentui/react-northstar` an. Das Package beinhaltet diverse Komponenten, welche durch FluentUI definiert werden. Zusätzlich enthält es auch bereits ein Theme für Teams.

4.2.1.4 In Kürze

Pro	**Contra**
«First-Class citizen» von Teams	Komponenten können nicht von smino übernommen werden
Dokumentationen, Guidelines und Code-Beispiele basieren meist auf React	Unbekannte Technologie für sowohl das Projektteam als auch die Entwickler von smino
Teams Theme integriert	

Tabelle 4: Evaluation React in Kürze (eigene Darstellung)

4.2.2 Angular

Angular ist ein komplettes Framework und eine Entwicklungsplattform für die Erstellung von anspruchsvollen Single-Page-Applikationen.

4.2.2.1 Setup

Grundsätzlich kann für das Setup einer Angular Anwendung mit Teams das Template für React Applikationen verwendet werden, wobei die effektive React Anwendung durch eine Angular Anwendung ersetzt werden kann. Dies stellt keine grosse Herausforderung dar, wobei das Debugging und die Entwicklungsexperience analog zur React basierten Variante sind.

Ebenso wäre ein manuelles Setup denkbar. Dies kann anhand der Beispiele für generelle Node.js Applikationen übernommen werden. Dabei muss das Build-System aber komplett selbst – z.B. mittels Gulp – konfiguriert werden.

4.2.2.2 Dokumentation

Es gibt keine expliziten Dokumentationen, Beispielprojekte und Guidelines für Angular, sondern lediglich einige allgemeine für Node.js Applikationen. Die Konzepte können jedoch weiterhin aus der React basierten Dokumentation genommen werden.

4.2.2.3 Styling Frameworks

Die `@fluentui/web-components` sollen eine Grundlage bieten, um FluentUI Komponenten unabhängig von einem UI-Framework anzuwenden. Dabei ist zu beachten, dass sich das Basis-Theme, sowie auch das Styling von den React-Komponenten für Teams klar unterscheiden. Aktuell können auch keine vorgefertigten Themes für Teams gefunden werden. Andererseits ist auch nur für React dokumentiert, wie Themes erstellt und angewendet werden können. Soweit ersichtlich, werden die Web-Components auch nicht als Grundlage für das entsprechende React Package verwendet. Siehe auch: Anhang Tabelle 1: Vergleich React und Web-Components Komponenten (eigene Darstellung)

Angular bietet zudem einen First-Class Support für Material mit `@angular/material`. Dabei können die gesnutzten Farben auch selbst konfiguriert werden und es wird ein Light und Dark Theme von Haus aus unterstützt werden. Die Komponenten, welche FluentUI bietet, sollten zum grossen Teil abgedeckt sein.

4.2.2.4 In Kürze

Pros	**Cons**
Angular Komponenten von smino können teilweise übernommen werden	Keine native Unterstützung für das Theming und Komponenten von Microsoft Teams
Bekannte Technologie für sowohl das Projektteam als auch die Entwickler von smino	Keine bestehende Unterstützung durch das Teams Toolkit und mangelnde Dokumentation

Tabelle 5: Evaluation Angular in Kürze (eigene Darstellung)

4.2.3 Visueller Vergleich React-Northstar und Web-Components

Im visuellen Vergleich ist klar ersichtlich, dass der «Sign In» Button im React Dialog (Abbildung 21: React Dialog with @fluentui/react-northstar (eigene Darstellung)) dem «Back» Button Teams selbst entspricht. Wohingegen bei «Sign In» Button im Angular Dialog (Abbildung 22: Angular Dialog with @fluentui/web-components (eigene Darstellung)) nur eine Annäherung dessen erreicht werden konnte.

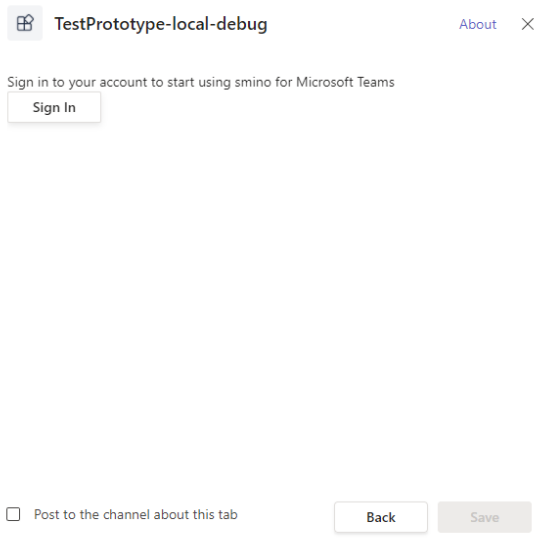


Abbildung 21: React Dialog with @fluentui/react-northstar (eigene Darstellung)

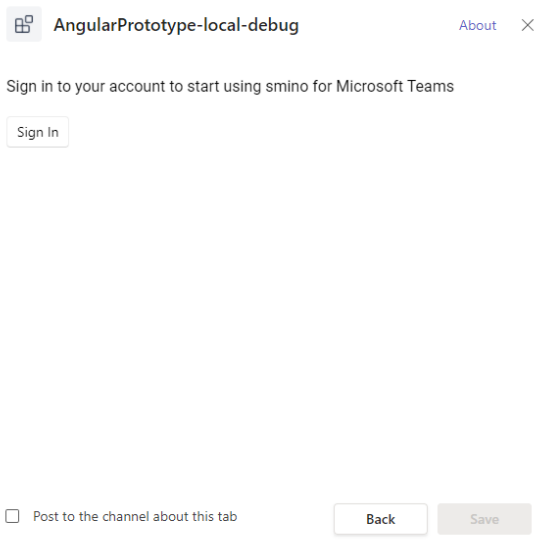


Abbildung 22: Angular Dialog with @fluentui/web-components (eigene Darstellung)

4.2.4 Entscheidung

Für die Entscheidung zur Auswahl des zu verwenden Frameworks und Stylings wurden 3 Vorschläge erarbeitet. Die Vorschläge basieren darauf, wo die Priorität für dieses Projekt liegen soll.

Die Verwendung der Teams Komponenten steht im Vordergrund
Hier würde sich die Verwendung von React mit den FluentUI Komponenten anbieten, da dies schon "von Haus aus" unterstützt wird und sich nahtlos in Teams integrieren lässt.

Die Verwendung von smino Komponenten steht im Vodergrund
Das Farbschema für React/FluentUI kann konfiguriert werden, um eine Wiedererkennung zur smino Web-Applikation zu bieten. Wenn man Angular verwendet, würde es sich aber sicher anbieten, die bestehenden smino Komponenten wiederzuverwenden. Dies würde auch aus Entwicklungssicht viel Arbeit reduzieren.

Styling ohne Relevanz und wird später miteingebunden
Für die Verwendung mit Angular würde sich dann Material anbieten, da dies nahtlos in Angular integriert wird und auch eine grosse Verbreitung geniesst. Die Komponenten haben auch bereits Accessibility umgesetzt, wobei das Farbschema ebenfalls konfigurierbar ist und entweder an Teams oder smino angepasst werden könnte.

***Entscheid:* Angular mit Verwendung der bestehenden smino Komponenten**
Smino hat von Anfang an den Wunsch geäussert, die Tabs Applikation aufgrund der bestehenden Kompetenzen mit Angular umzusetzen. Eine weitere Entscheidung ist, dass die effektiven Tabs die bestehenden smino Komponenten verwenden sollen und entsprechend das Styling übernommen wird. Ebenso soll die Architektur so nah wie möglich an smino angeglichen werden.

(Color schemes, 2022; Microsoft, 2022; Microsoft, 2022; Fluent UI Team, 2022)

4.3 Testing

Die Komplexität des automatisierten Testing im Microsoft Teams Kontext wurde bereits früh erkannt und evaluiert. Der in der Microsoft Teams Dokumentation[3] vorhandene Abschnitt bezüglich Testing bezieht sich ausschliesslich auf das manuelle Testen einer Teams Applikation im Microsoft Teams Kontext selbst. Unter anderem ist der Fokus darauf angelegt, wie ein Teams-Tenant dafür aufgesetzt wird. Für das Schreiben von automatisierten Tests gibt es keinerlei Dokumentationen oder Beispiele.

Für Angular, sowie generell für JavaScript Frameworks, gibt es diverse Testing Frameworks, die in Betracht gezogen werden können. Im Rahmen dieser Arbeit wurden, aufgrund ihrer Bekanntheit im Projektteam, jedoch nur die Frameworks Jasmine, Jest und Cypres, berücksichtigt.

4.3.1 Jasmine

Jasmine[4] ist das First-Class Citizen Testing Framework von Angular und somit Out-of-the-Box verfügbar. Es ist ein behavior-driven Testing Framework, das keine weiteren Referenzen auf andere JavaScript Frameworks benötigt.

Der Fokus von Jasmin ist das Testen einzelner Components, Services, Directives und Pipes.

Pro	**Contra**
First-Class Citizen	Low Performance
Gute Dokumentation	Unfreundliche Fehlermeldungen
Test laufen im realen Browser	Snapshot Testing ist nicht integriert
Eine allfällige Migration auf Jest ist verhältnismässig einfach	

Tabelle 6: Evaluation Jasmine (eigene Darstellung)

[3] https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/build-and-test/test-app-overview

[4] https://jasmine.github.io/

4.3.2 Jest

Das Jest Testing Framework[5] verhält sich sehr ähnlich wie das Jasmin Testing Framework. Auch hier stehen die Komponententests im Vordergrund. Ebenso ist die Syntax für das Schreiben der Tests identisch.

Pros	Contra
Wenig Konfiguration notwendig	Snapshots sind nur in einem kleinen Rahmen effizient
Gute Dokumentation	
Einfaches Mocking von Objekten und Services	
Tests werden isoliert ausgeführt	
Snapshot Testing integriert	

Tabelle 7: Evaluation Jest (eigene Darstellung)

4.3.3 Cypress

Anders als Jasmine oder Jest wird Cypress in derselben Ausführungsschleife wie die Angular Applikation ausgeführt und ist auf e2e Tests ausgerichtet. Die Idee der e2e-Tests ist das Testen aller Komponenten im Zusammenspiel, ohne auf zu viele Mocks zugreifen zu müssen. API-Calls können mit dem Intercept-Command einfach abgefangen werden und anschliessend kann die Response mit dem Fixture-Command gefaked werden. Das Mocken des Microsoft Teams SDK hingegen ist nicht ohne grossen Aufwand zu realisieren.

4.3.3.1 End-To-End Testing im Microsoft Teams Environment

Cypress *kann* e2e-Tests gegen die Web-App-Version von Microsoft Teams ausführen.[6] Dabei gibt es aber einige Probleme, die nur mittels instabilen Workarounds gelöst werden können.

Das grösste Problem dabei ist, dass Cypress die Web-App in einem `iframe` lädt, was einerseits die Microsoft-Login-Page nicht zulässt, andererseits Microsoft Teams selbst nicht zulässt.

Positive e2e-Tests ausserhalb des Microsoft Teams Kontexts lassen sich mit Cypress einfach erstellen. Da sie aber komplett entkoppelt von den Komponenten sind und der Test-Case anhand von HTML-Attributen (`id`, `class`, `data-*`, etc.) ausgeführt wird, ist ihr Wartungsaufwand als sehr hoch einzustufen. Daher sind e2e Tests nur für die wichtigsten Abläufe zu empfehlen.

[5] https://jestjs.io/
[6] https://www.eliostruyf.com/e2e-testing-microsoft-teams-solutions-cypress/

4.3.4 Fazit

Für das Testing der Components eignet sich das Jest Framework am besten. Die Einbindung in eine Angular Applikation ist sehr einfach und macht das Schreiben von Tests dank den Snapshots gegenüber dem Jasmine-Framework einiges effizienter.

Das e2e-Testing im Microsoft Teams Kontext ist noch nicht genügend ausgereift und die Workarounds sind sehr instabil. Aus diesem Grund wird für dieses Projekt auf das e2e Testing im Microsoft Teams Environment verzichtet. e2e Tests im Rahmen der Web-Applikation von Angular sind an und für sich realisierbar.

Die Einbindung im nativen Teams-App kann nur mit sehr viel Aufwand und unter Verwendung eines Emulators automatisiert getestet werden.

(Microsoft, 2022; Microsoft, 2022; Ravindranath, 2022; Struyf, 2022; Google, 2022; Parakramasinghe, 2022; Jaaidi, 2022)

4.4 Monitoring

Für die Überwachung der Web-Applikation wird Application Insights[7] verwendet. Application Insights bietet eine einfache Integration für JavaScript basierte Clients und ermöglicht es einfach Metriken und Fehler zu überwachen. Zudem ist es ideal im Microsoft Umfeld integriert. Application Insights wird aktuell bereits für die Überwachung der API und Web-Applikation bei smino eingesetzt.

4.5 State Management Angular

Für die Angular Anwendung soll das Redux Pattern, bzw. ein zentrales State Management verwendet werden. Dies erlaubt es einfach State global in der Applikation zu teilen oder diesen auch in einem Storage – z.B. LocalStorage – zu persistieren. Das State Management soll konsequent verwendet werden, wobei sich alle von der API bezogenen Daten im State wiederfinden sollten.

Aufgrund der bisherigen Erfahrungen wurde im Prototypen NgRx als State Management Lösung verwendet. Dabei geniesst NgRx eine grosse Verbreitung und eine breite Community. Zudem werden bereits viele Erweiterungen angeboten, wie eine Erweiterung für das Persistieren des State in einen Storage oder eine Erweiterung für die vereinfachte Verwaltung von Entitäten.

Die smino Web-Applikation verwendet NGXS als State Management Lösung. Die etwas neuere Implementierung zielt vor allem auf eine vereinfachte Verwaltung des States ab. Dabei können auch einfach native Angular Funktionen– wie Dependency Injection – in den States benutzt werden, wobei auch eine Reduzierung von «Boilerplate Code» im Vordergrund steht.

Aufgrund der Entscheidung, sich an der Architektur von smino anzupassen, wird für die produktive Lösung NGXS verwendet.

4.6 Pipelines CI/CD

Für Continuous Integration und Continuous Deployment werden Azure YAML Pipelines[8] verwendet. Diese sind bereits in Azure DevOps integriert, wobei Azure DevOps bereits für die Versionskontrolle und das Backlog-Management eingesetzt wird. YAML Pipelines sind der Nachfolger von den klassischen Build und Release Pipelines und können sehr einfach weitergegeben werden.

[7] https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
[8] https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/pipelines-get-started

4.7 @smino/api als Angular Library

Die smino API soll als separate Angular Library in das Projekt eingebunden werden. Dies soll auch einem ersten Schritt aufzeigen, wie diese paketiert werden könnte, um sie auch anderen Client Applikation zur Verfügung zu stellen. Andererseits vereinfacht es die Architektur der Tabs-Applikation selbst und bildet eine klare Separierung.

4.8 smino-components

Die Komponenten, welche von der smino Web-Applikation übernommen werden, werden auch klar als solche in der Applikation abgelegt. Ziel soll es sein, die Komponenten so weit als möglich von der Tabs-Applikation selbst zu entkoppeln, so dass diese später ebenfalls in eine oder mehrere Libraries ausgelagert werden könnten. Es soll aber versucht werden, nur das anzupassen oder zu ergänzen, was für die aktuellen Anforderungen und Architektur notwendig ist.

4.9 Dark-Theme

Der Teams-Client bietet ein Light und ein Dark-Theme an. Die bestehenden smino Komponenten unterstützen bis anhin kein Theming, wobei entsprechend auch kein Dark-Theme unterstützt wird. Für dies soll ein Theming Konzept erstellt und umgesetzt werden.

4.10 Verwendung Teams Toolkit

Für die Erstellung und Verwaltung der Teams Applikation soll strikt das Teams Toolkit verwendet werden. Dies ermöglicht eine schnelle Migration bei technologischen Veränderungen oder die Nutzung von neuen Features. Ebenso wird der Aufwand für das Setup und die Pflege massiv vereinfacht.

4.11 Linting und Code Qualität

Zur Sicherstellung der Code Qualität wird ESLint und Stylelint verwendet. Dabei werden für ESLint die Rules von smino übernommen. Für die Formatierung der Dateien wird Prettier verwendet. Um sicherzustellen, dass die entsprechende Formatierung und Rules angewendet werden, wird lint-staged verwendet. Dabei ist sichergestellt, dass die Linter und die Formatierungsregeln vor jedem Commit validiert und angewendet werden. Zudem sollen die Linter in der Continuous Integration Pipeline ausgeführt werden.

4.12 OpenID Client Library

Für die Authentifizierung mit smino wird OpenID Connect verwendet. Für die Anbindung wurden hauptsächlich zwei Libraries in Betracht gezogen: oidc-client-ts[9] und angular-auth-oidc-client[10].

Die angular-auth-oidc-client Library ist eine native Angular Library, welche den Authentication Flow in der Anwendung fest vorgibt und sich einfach konfigurieren lässt. Die Library ist OpenID Certified und wird auch aktiv gewartet. Bei der Erstellung des Prototypen wurde jedoch festgellt, dass der Nutzer oder das Resultat der Authentifizierung nur schwer manuell hinterleget werden kann. Dies ist jedoch erforderlich, da die Authentifizierung in einem separaten Popup von Microsoft Teams abgehandelt wird.

Die oidc-client-ts Library ist der Nachfolger der OpenID Certified oidc-client-js Library, welche die Weiterentwicklung im Juni 2021 eingestellt hat. Diese wurde initial portiert und wird aktiv, um neue Funktionalitäten erweitert. Dem Autor nach ist die Library selbst aus Kostengründen nicht OpenID Certified. Dieser Client ist minimalistischer als die native Angular Library, wobei der User mit seinen Tokens manuell abgelegt werden kann.

Für die Anwendung wird die oidc-client-ts Library verwendet, da deren Funktionalität völlig ausreichend für das Setup ist und das Resultat der Authentifizierung einfach von einem anderen Kontext hinterlegt werden kann.

4.13 Accessibility

Im Rahmen der Arbeit werden keine expliziten Funktionen für eine Unterstützung der Accessibility umgesetzt. Momentan ist dies durch die Komponenten von smino noch nicht gegeben, wobei eine Umsetzung einer kompletten Accessibility den Rahmen des Projekts sprengen würde. Die Autoren würde aber eine Umsetzung und Unterstützung der Accessibility befürworten.

[9] https://github.com/authts/oidc-client-ts
[10] https://github.com/damienbod/angular-auth-oidc-client

5. Applikationsarchitektur

Nachfolgend soll die Architektur der Anwendung etwas näher beschrieben werden. Einerseits die Top-Level Architektur der Teams Anwendung, sowie auch die Tabs Anwendung im Detail.

5.1 Teams Applikation

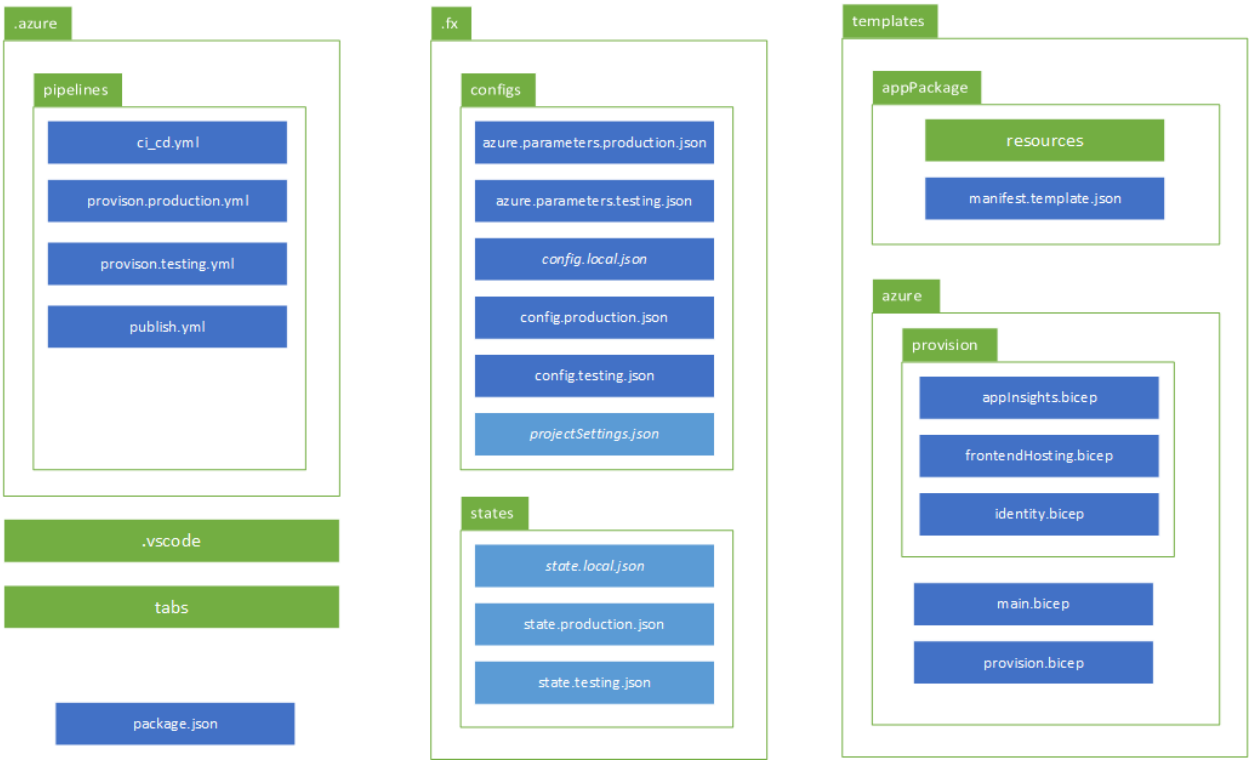


Abbildung 23: Teams Applikation Ordnerstruktur (eigene Darstellung)

Wenn eine Teams Anwendung für Tabs erstellt wird, werden folgende Ordner und Dateien standardmässig angelegt:

- **.fx:** Konfigurationen und Statusinformation für das Toolkit
- **.vscode:** Einstellungen und Konfiguration für Visual Studio Code
- **tabs:** Source-Code für die Tabs Applikation
- **templates:** Manifest und Konfigurationen für die Provisionierung
- **package.json:** Pakete und Scripts welche als Basis für die Entwicklung dienen

Der **.azure** Ordner dient zur Ablage der YML Pipelines für Azure DevOps und wird nicht standardmässig durch das Toolkit angelegt.

Nachfolgend werden die einzelnen Ordner und Dateien etwas näher erklärt.

5.1.1 .azure

Enthält die YAML Pipelines und sonstige Ressourcen, welche von Azure DevOps verwendet werden.

ci_cd.yml

Für CI und CD der Teams App existiert eine "teams-ci-cd" Pipeline. Die Pipeline kann für Pull Requests verwendet werden, um die Integrität des Codes sicherzustellen. Das effektive Deployment wird nur auf dem "main" Branch durchgeführt.

provision.{Umgebung}.yml

Pro Umgebung existiert eine Pipeline für die Provisionierung der Applikation. Also die Bereitstellung und Einrichtung aller notwendigen Ressourcen, um die Applikation später bereitstellen zu können.

publish.yml

Die Publish Pipeline kümmert sich um die Publikation der Teams Applikation in den Organizational Store.

5.1.2 .vscode

Konfigurationen und Einstellungen für Visual Studio Code. Die erforderlichen Tasks und Debug-Konfigurationen werden dabei automatisch durch das Teams Toolkit angelegt.

launch.json

Enthält Informationen, um eine Debugging-Session für die Anwendung zu starten. Dabei werden Konfigurationen angelegt, um die Anwendung zu starten und anschliessend den Debugger an den gewählten Browser anzuhängen. Standardmässig werden die Konfigurationen für Google Chrome und Microsoft Edge angelegt.

settings.json

Enthält Einstellungen für Code, welche für den aktuellen Workspace berücksichtigt werden sollten. Beispielsweise wird hier eine Einstellung für das ESLint Plugin vorgenommen, damit die Dateien oder Source-Ordner richtig erkannt werden.

tasks.json

Enthält die Tasks, um die Teams Anwendung über das Teams Toolkit Plugin zu starten.

5.1.3 tabs

Der Tabs Ordner enthält den Source-Code für die Tabs Applikation. Im Teams Toolkit werden die Bereiche «Capabilities» genannt. Beispielsweise wäre ein Bot eine weitere «Capability», wodurch ein neuer Ordner «bot» angelegt werden würde. Die Tabs Applikation selbst wird im nächsten Abschnitt genauer beschrieben.

5.1.4 .fx

Der **configs** Ordner enthält alle Konfigurationen, welche vom Teams Toolkit für die Provisionierung und die Bereitstellung der Applikation notwendig sind. Hier können Parameter für die Provisionierung oder auch für das Manifest definiert werden. In den **states** werden die Zustände nach der Provisionierung der Applikation festgehalten.

azure.parameters.{Umgebung}.yml

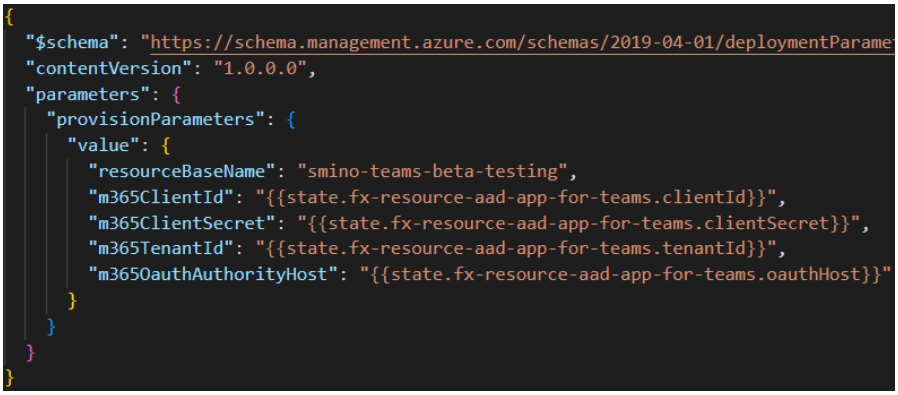


Abbildung 24: azure.parameters.{Umgebung}.yml (eigene Darstellung)

Pro Umgebung können hier Parameter für die Provisionierung festgelegt werden. Beispielsweise wird aktuell der Basisname für die Ressourcen konfiguriert. Ebenso werden Parameter von den aktuellen **states** für M365 an die Provisionierung weitergereicht. Diese Konfiguration wird primär von den Bicep Dateien verwendet.

config.{Umgebung}.yml

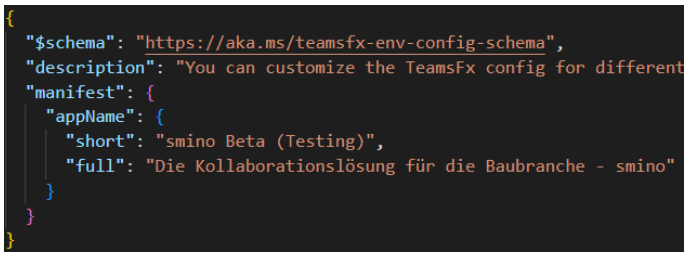


Abbildung 25: config.{Umgebung}.yml (eigene Darstellung)

Mit diesen Dateien kann das Teams Toolkit pro Umgebung konfiguriert werden. Beispielsweise können Parameter für das Manifest definiert werden, welche später ersetzt werden.

projectSettings.json

Abbildung 26: projectSettings.json (eigene Darstellung)

In dieser Datei werden die Projekteinstellungen für das Teams Toolkit verwaltet. Zum Beispiel wird der Name, die verwendete Programmiersprache oder auch die «Capabilities» konfiguriert. Die Datei wird durch das Toolkit verwaltet und sollte **nicht durch den Benutzer verändert werden**.

state.{Umgebung}.json

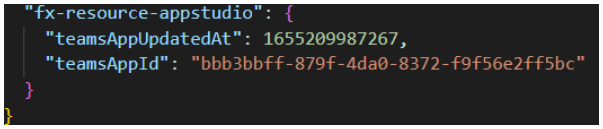


Abbildung 27: state.{Umgebung}.json (eigene Darstellung)

Hier werden die States für die Provisionierung der Teams Applikation abgelegt. Einerseits können die Werte dann für das Deployment weiterverwendet werden und andererseits kann validiert werden, ob beispielsweise der M365 Mandant verändert wurde.

5.1.5 templates/appPackage

Abbildung 28: manifest.template.json (eigene Darstellung)

Hier ist das Template für das Teams App-Package Manifest – **manifest.template.json** – enthalten. Jegliche Konfiguration für die App selbst sind hier vorzunehmen, wobei darauf geachtet werden soll, Werte, die pro Umgebung unterschiedlich sind, auszulagern. Als Ressourcen sind für das Manifest die entsprechenden Logos für das App hinterlegt.

5.1.6 templates/azure

Für Azure sind hier die Bicep Konfigurationen abgelegt, welche für die Provisionierung der entsprechenden Cloud Ressourcen verwendet werden. In **provision** finden sich Konfiguration für Application Insights, Frontend Hosting (Storage Account) und Identity.

Die Datei **provision.bicep** bildet ein Modul, welche alle Ressourcen der Provisionierung zusammenfasst. Als Einstiegspunkt wird **main.bicep** verwendet.

5.2 Tabs Web-App Architektur

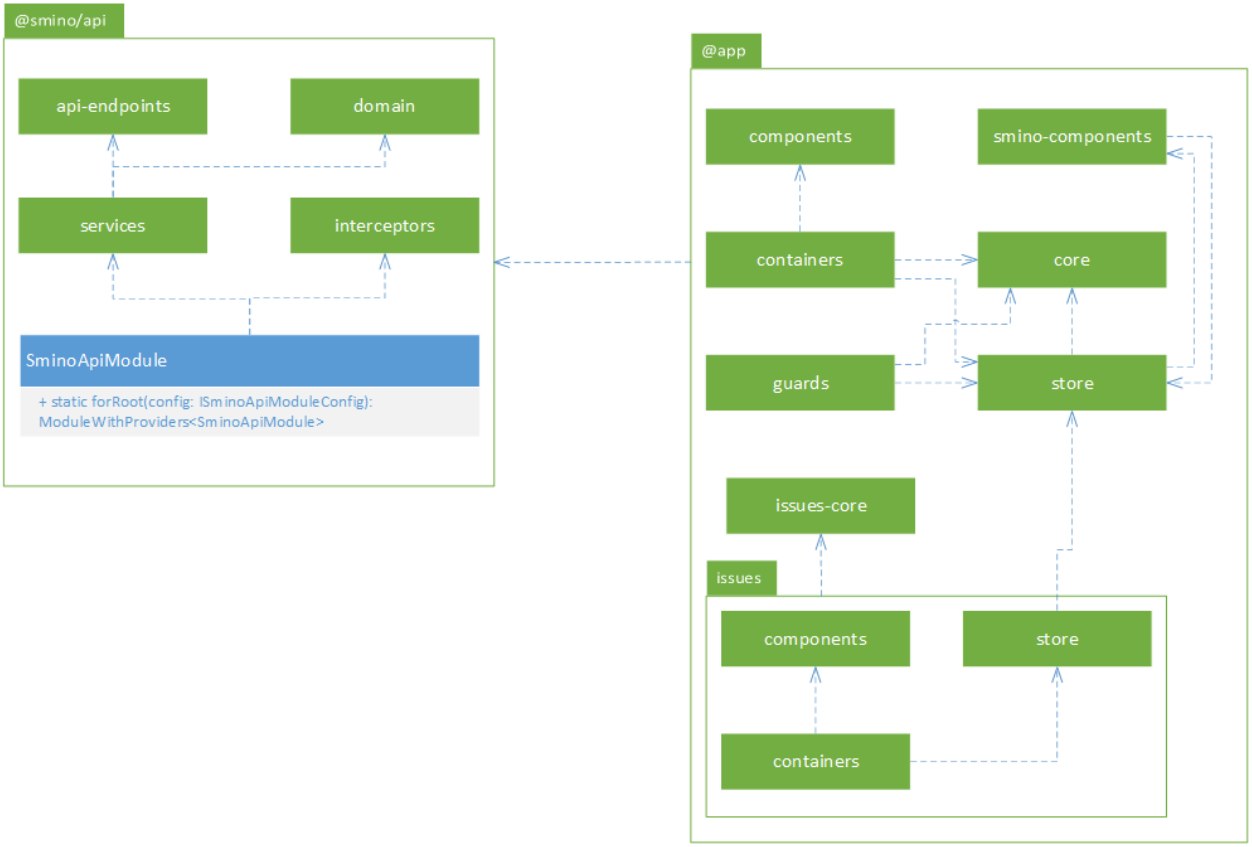


Abbildung 29: Tabs Web-App Architektur (eigene Darstellung)

Die Tabs Applikation ist in die eigentliche Web-Applikation selbst sowie in Projekte unterteilt, welche von der Applikation benutzt werden. Eine Auslagerung in ein Projekt ist immer dann sinnvoll, wenn das Modul Potential für eine Paketierung aufweist und ohne Abhängigkeiten zur Applikation selbst verwendet werden kann. In der Abbildung sind die wichtigsten Module dargestellt, wobei diese nachfolgend genauer erläutert werden.

5.2.1 @smino/api

Das `@smino/api` Projekt enthält alle nötigen Models und Services, um mit der API von smino zu kommunizieren. Langfristig könnte dies paketiert und über NPM bezogen werden. Dies würde die Duplizierung des Codes massiv verringern.

Aktuell wurden nur die Aufrufe übernommen, welche für die Umsetzung auch notwendig sind und es wurde nicht die gesamte API abgebildet.

- **api-endpoints:** Enthält Komponenten, um vereinfacht URI's für die API zu erstellen.
- **domain:** Enthält alle Models, um Requests und Responses der API abzubilden.
- **services:** Enthält Services, welche die Interaktion mit der API ermöglichen. Dies soll es vereinfachen, Aufrufe direkt mit einer korrekten Typisierung zu verwenden.
- **interceptors:** Verschiedene Interceptors, welche zur Kommunikation mit der API notwendig sind. Um die entsprechenden Interceptors zu verwenden, müssen Services (Providers) zur Verfügung gestellt werden, welche die entsprechenden Werte liefern.
    - **AccessTokenInterceptor:** Fügt ein AccessToken an die Requests zur API hinzu.
    - **ApiUrlInterceptor:** Ersetzt den API Url Parameter in Requests.
    - **TranslationApiUrlInterceptor:** Ersetzt den Translation API Url Parameter in Requests.
    - **UserIdInterceptor:** Ersetzt den User-Id Parameter in Requests.

5.2.2 @app

Im Bereich `@app` ist die eigentliche Web-Applikation enthalten.

- **components:** Basis-Komponenten für die Container der Applikation.
- **containers:** Basis-Pages der Applikation, welche für Login und Tab-Konfiguration verwendet werden.
- **guards:** Guards, welche für das Routing der Applikation verwendet werden. Hier sind Guards für Authentifizierung oder auch zur Validierung des selektierten Projekts enthalten.
- **core:** Enthält Kernfunktionalitäten der Applikation. Hier findet sich beispielsweise auch ein AuthService. Das CoreModule kümmert sich um die Initialisierung der Applikation mit den korrekten Modulen.
- **store:** Enthält die zentrale Store Logik, welche in der Applikation verwendet wird.
- **smino-components:** Hier befinden sich alle wiederverwendbaren Komponenten der smino Web-Applikation. Ziel soll es sein, dass dies später ebenfalls in ein oder mehrere separate Projekte/Pakete ausgelagert werden kann.
- **issues-core:** Im issues-core Module sind zentrale Services und Komponenten für die Aufgaben enthalten. Dies wurde analog zur smino Web-Applikation übernommen.
- **issues:** Repräsentiert das Issues Feature-Module. Grundsätzlich soll für jeden Bereich in der Teams App ein Feature Module erstellt werden. Ein Feature Module definiert ebenfalls seine Containers, Components und kann auch neue States definieren, welche nur innerhalb des Features verwendet werden.

5.3 Verwendete Patterns

5.3.1 Smart and Presentational Components

In der Angular Applikation selbst wird zwischen Smart und Presentational Components unterschieden. Dies hilft massiv bei der Aufteilung der Komponenten und verhindert, dass übergrosse Komponenten entstehen.[11]

Dabei kümmern sich Smart Components um die Interaktion mit dem Service-Layer, also beispielsweise um das Laden von Daten oder die Speicherung dieser. Die entsprechenden Daten werden den Presentational Components übergeben, welche sich ausschliesslich um die Darstellung kümmern. Aus diesem Grund werden Presentational Components auch Dumb Components genannt.

Eine Hauptfrage für die Erstellung von Presentational Components ist, ob die Logik für die Anzeige eventuell wiederverwendet wird. Denn die Wiederverwendbarkeit der Komponenten ist ebenfalls einer der Hauptvorteile dieses Patterns.

Ein weiterer Vorteil ist, dass Änderungen am Service-Layer oder Business Logik in den Komponenten selbst nur direkte Auswirkungen auf die Smart Components haben.

In der Applikation werden Smart Components als «Container» und Presentational Components als «Components» deklariert.

5.3.2 Redux

NGXS ist schlussendlich eine für Angular konzipierte Implementation des Redux Patterns. Dabei wird der gesamte State der Applikation in einem zentralen Store abgelegt und über sogenannte Actions mutiert.

[11] https://blog.angular-university.io/angular-2-smart-components-vs-presentation-components-whats-the-difference-when-to-use-each-and-why/

5.3.3 Facade Pattern für States

Die Interaktion wird auf zwei Arten abstrahiert. Einerseits wird der Store selbst, durch den `AppState` abstrahiert. Dies erlaubt es, einfach die aktuelle Store Lösung durch eine andere Implementation, wie NGRX, zu ersetzen. Andererseits werden Facades für das Aktualisieren der einzelnen States verwendet. Seiteneffekte, wie der Aufruf einer API, werden dabei in der Facade behandelt. Ebenso können aus einem Aufruf der Facade mehrere Actions im Store versendet werden oder weitere Facades aufgerufen werden. Durch die Abstraktion der Facade werden die verwendenden Komponenten vom State Management und der konkreten Implementation dessen entkoppelt.

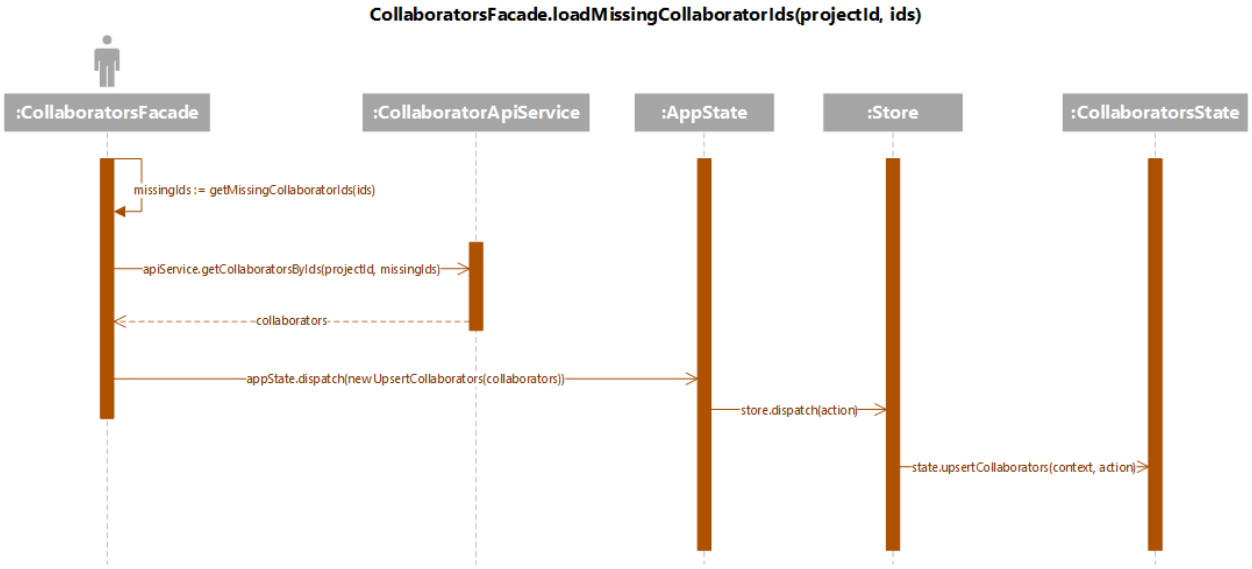


Abbildung 30: loadMissingCollaborators Ablauf (eigene Darstellung)

Als Beispiel ist hier die Funktion `loadMissingCollaborators` der `CollaboratorsFacade` vereinfacht illustriert. Die Facade bezieht zuerst die fehlenden Collaborator Ids. Dafür werden alle Ids gesucht, welche nicht bereits im State vorhanden sind. Die API wird dann entsprechend für die fehlenden Collaborator Ids angefragt. Bei der erfolgreichen Rückgabe der Collaborators wird die Action `UpsertCollaborators` auf dem `AppState` und entsprechend auf dem `Store` versendet. Der Store führt dann die entsprechenden Handler auf den States aus, welche auf diese Action registriert sind. Der `CollaboratorState` aktualisiert aufgrund der Action die aktuellen Collaborators in seinem State.

Ein Grundpfeiler bei der Implementation von Aufrufen der Facaden ist, dass die Aufrufe dem Prinzip von Fire-and-forget folgen. Also der Aufrufende nicht gezwungen ist, allfällig auf einen Return Wert zu warten oder sich darauf zu registrieren. Das Resultat und vor allem Fehler seitens der API werden dem Aufrufenden aber in Form eines «Observables» zurückgegeben. Dies ermöglicht es vor allem auf Fehler zu reagieren.

5.4 Konfiguration von Applikationseinstellungen

Im Speziellen soll hier noch die Konfiguration der Applikationseinstellungen erläutert werden. Angular bietet die Möglichkeit umgebungsbasierte Einstellungen mittels Environment Dateien bereitzustellen. Der Nachteil daran ist, dass ein Build der Applikation pro Umgebung erstellt werden muss. Dies widerspricht dem Prinzip von «Build Once, Deploy Anywhere».

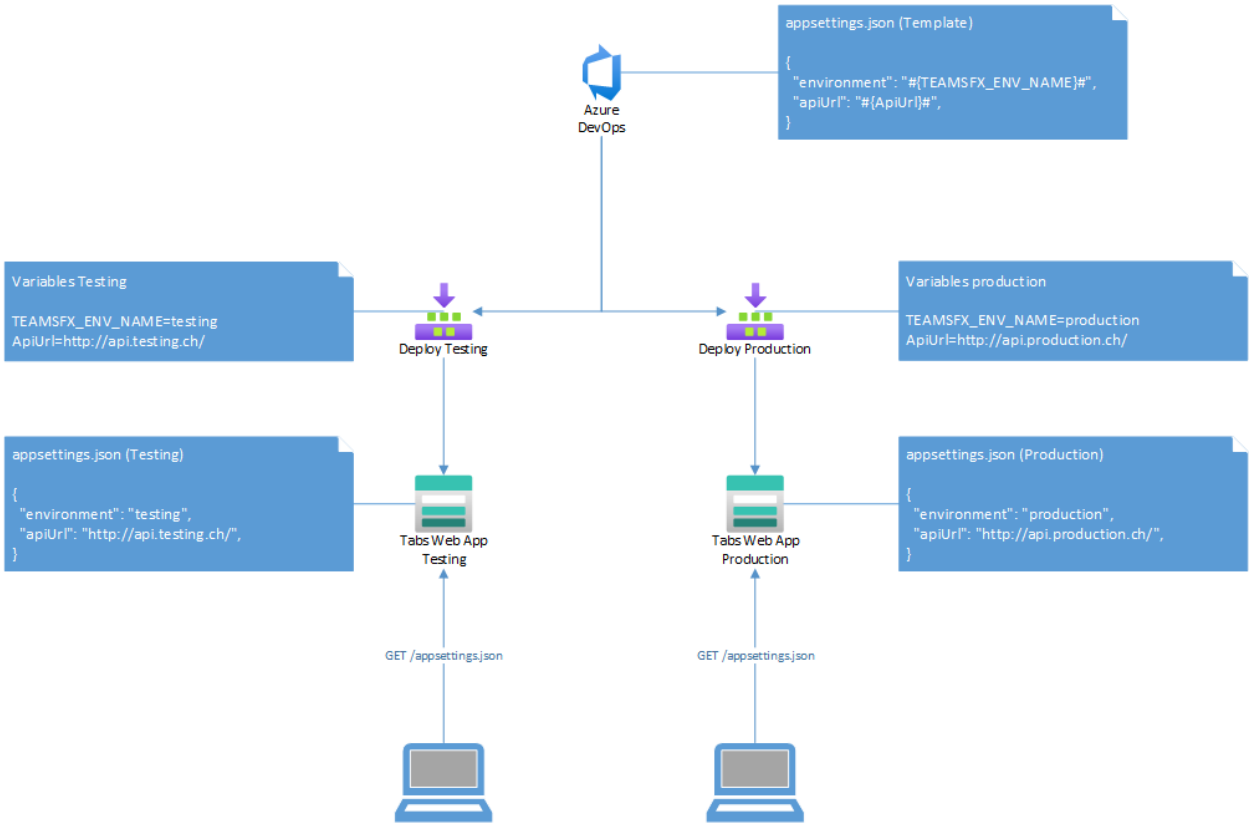


Abbildung 31: Applikationseinstellungen (eigene Darstellung)

Um das Problem zu lösen, wird in der Tabs Web-Applikation zwei JSON Files definiert, welche die Konfiguration der Applikation beinhalten. Die Standardkonfiguration wird für die Entwicklung verwendet, wobei eine zweite Datei als Template angelegt wird. Für den produktiven Build einer Applikation wird die Template Datei verwendet. Wie in der Abbildung zu sehen ist, können dann pro Umgebung die Platzhalter in der entsprechenden Konfiguration aufgrund von Variablen ersetzt werden.

Die Angular Applikation lädt die entsprechenden Einstellungen bei der Initialisierung der Applikation und wartet die Initialisierung der Einstellungen ab. Ebenso wird ein `AppSettingsService` bereitgestellt, welcher es erlaubt, die geladenen Einstellungen zu beziehen.

6. smino Integration

Im folgenden Abschnitt werden die in die smino Team App integrierten Features beschrieben. Es wird insbesondere auf die Implementierung der Authentifizierung, der Tab-Konfiguration und die Aufgaben eingegangen. Die einzelnen Schritte oder Anforderungen zur Integration der Aufgaben wurden in Form von «Backlog Items» und Screen-Designs festgehalten.

6.1 Nicht-Funktionale Anforderung

Nicht-Funktionale Anforderungen wurden von smino nur wenige vorgegeben:

- Die Ladezeit der smino Aufgaben muss gleich oder besser als diejenige der Aufgaben in der smino Web-Applikation sein.
- Die Qualität der übernommenen smino Module muss gleich oder besser sein.

Auf weitere explizite Nicht-Funktionale Anforderungen wurde vom Projektteam verzichtet. Dennoch wurden einige präventive Schritte unternommen:

- Konservative Coding-Style Regeln in allen Code-Files *(.eslintrc, .lintstagedrc, .prettierrc, .stylelintrc*) mit einer *git commit pre-hook*.
- Verwendung des NGXS - State Management für vorhersehbare Zustandsveränderungen.
- Jegliche Pull-Request in dem main-Branch wurden durch smino geprüft.

6.2 Authentifizierung und Autorisierung

Die Authentifizierung in Teams wird mit OAuth 2.0 gemacht. Ein grundlegendes Verständnis von OAuth 2.0 ist also Voraussetzung für die Arbeit mit Authentifizierung im Teams Kontext. Der Authentication-Flow ist dabei für Bots und Tabs unterschiedlich. Da Tabs sehr ähnlich wie normale Webseiten aufgebaut sind, kann hier direkt mit OAuth 2.0 gearbeitet werden. Für die Bots gibt es ein paar Unterschiede, dass zugrundeliegende Konzept ist jedoch das Gleiche. Da im Rahmen dieser Arbeit nur Tabs verendet werden, wird nur die Tab Authentifizierung vertieft.

6.2.1 Authentication flow

1. Der Benutzer interagiert mit dem Inhalt der Tab-Konfiguration, meistens mit einem Anmelden-Button.
2. Der Tab konstruiert die URL für seine *auth start page*.
3. Der Tab führt die `microsoftTeams.authentication.authenticate()` Methode aus und registriert die `successCallback` und `failureCallback` Funktionen.
4. Die *auth start page* wird in einem externen Popup `iframe` geöffnet.
5. Der Benutzer loggt sich auf Seiten des Providers ein und gibt die Rechte für den Tab frei.
6. Der Provider ruft die OAuth 2.0 *redirect page* mit dem *access token* auf.

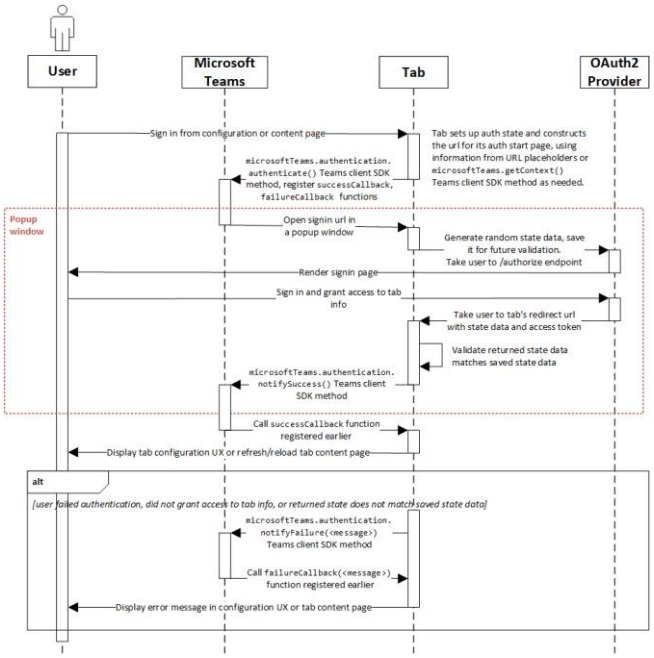


Abbildung 32: Authentication flow (Microsoft, 2022)

7. Der Tab prüft den retournierten state, ruft die `microsoftTeams.authentication.notifySuccess()` Methode auf, welche wiederum die vorgängig registrierte `successCallback` Funktion aufruft.
8. Teams schliesst das Popup Fenster wieder.
9. Der Tab lädt entweder weitere Konfigurationen oder lädt den Tab neu.

(Microsoft, 2022)

6.2.2 Login

Abbildung 33: Miro - Tab-Konfiguration Login (eigene Darstellung)

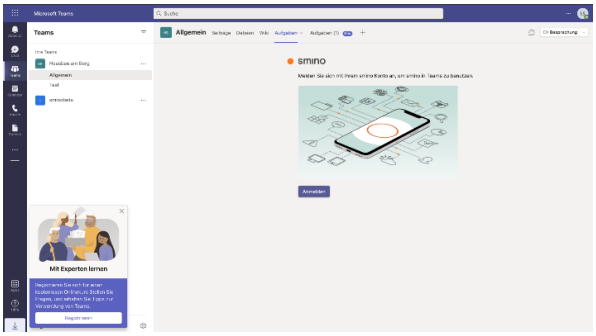


Abbildung 34: Figma - Tab Login (smino, 2022)

Beim Einbinden einer neuen smino Teams App in einem Tab oder beim ersten Aufruf auf einen Tab, dass die smino Teams App einbindet, muss sich der Benutzer bei smino anmelden.

Der im Popup geöffnete Authentication-Flow ist nicht Teil dieser Arbeit und wird daher nicht weiter beschrieben.

6.2.3 Unauthorized

Ist ein Benutzer bei smino eingeloggt und er versucht auf ein Tab zuzugreifen, das Aufgaben aus einem Projekt darstellt, für welches der Benutzer keine Berechtigung hat, muss eine «Unauthorized Project Access» Seite dargestellt werden.

Auf dieser muss ein Benutzerwechsel möglich sein.

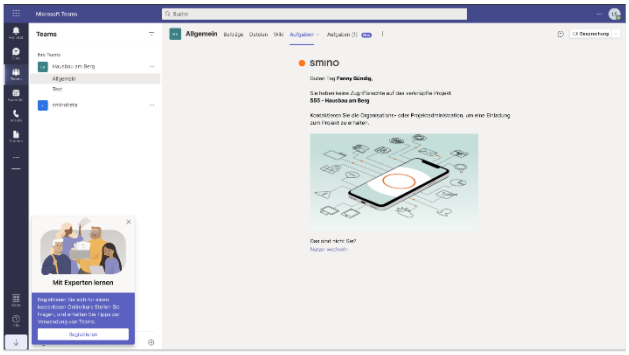


Abbildung 35: Figma - Tab unauthorized acces (smino, 2022)

6.2.4 Authentication Error

Entsteht im Authentication-Flow auf Seite des Providers ein gewollter oder ungewollter Fehler und der Teams Kontext erhält kein gültiges Resultat, muss der Benutzer auf eine entsprechende «Authentication Error» Seite weitergeleitet werden.

Auf dieser muss eine erneute Anmeldung möglich sein.

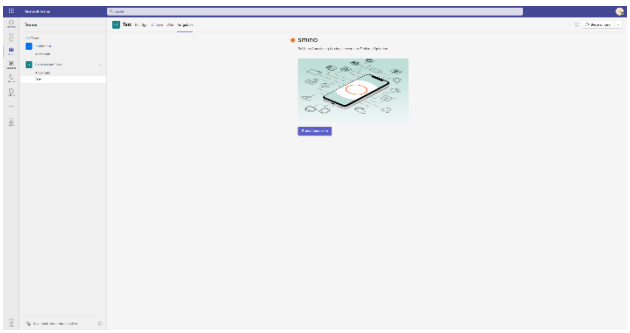


Abbildung 36: Figma - Tab authentication error (smino, 2022)

6.3 Tab-Konfiguration

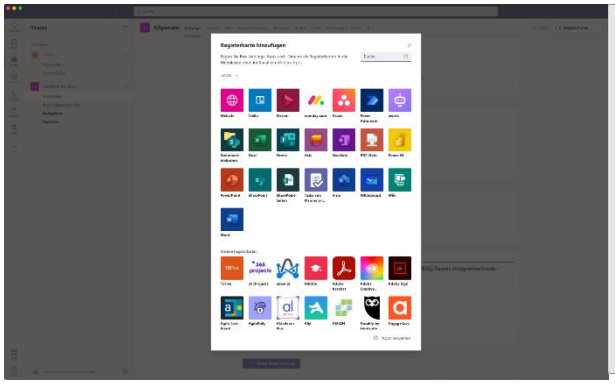


Abbildung 37: Figma - smino Teams App hinzufügen (smino, 2022)

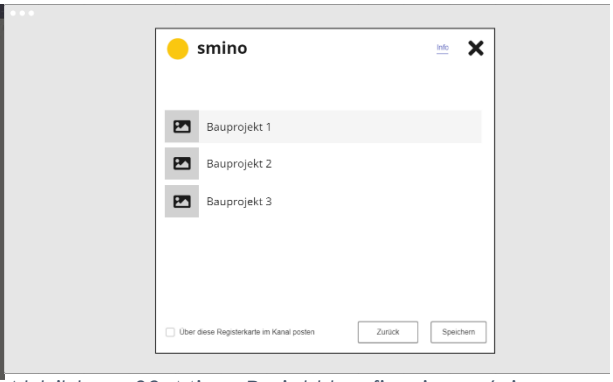


Abbildung 38: Miro - Projekt konfigurieren (eigene Darstellung)

Die smino Teams App kann über die Standard Teams Funktion «Tab Hinzufügen» einem Teams Kanal oder einer Konversation hinzugefügt werden.

Nach der allfälligen Authentifizierung wird dem Benutzer eine Tab-Konfiguration Seite dargestellt, in der das Projekt gewählt werden muss, für welches die Aufgaben im Teams Tab geladen werden sollen.

Für das selektierte Projekt muss jeweils ein separates *acces token* für den «User Collaborator» angefragt werden. Damit können Projektspezifische Informationen angefragt werden. Wechselt das Projekt, muss auch der «User Collaborator» neu geladen werden.

Wenn weitere Features dazu kommen, wie zum Beispiel Protokolle oder Journale, welche in einem Tab eingebunden werden können, muss ein weiterer Konfigurationsschritt eingefügt werden, um den gewünschten Bereich zu selektieren.

6.4 Aufgaben

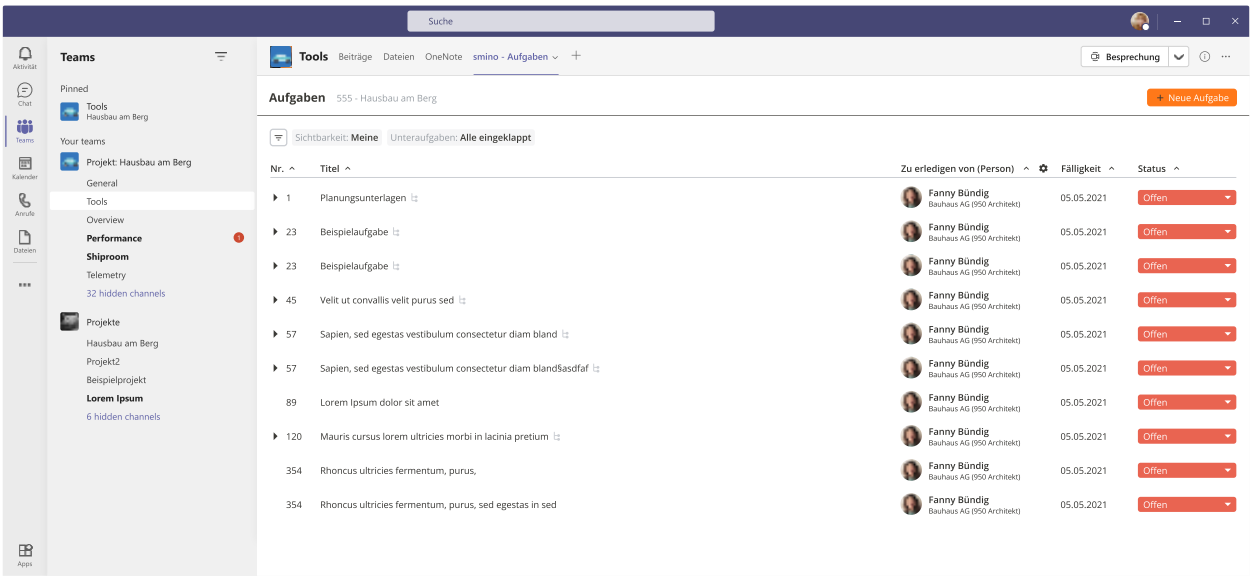


Abbildung 39: Figma – Aufgaben (smino, 2022)

Der smino Aufgaben-Tab ist eine vereinfachte Ansicht der sich auf der smino Web-Applikation befindenden Aufgabenansicht.

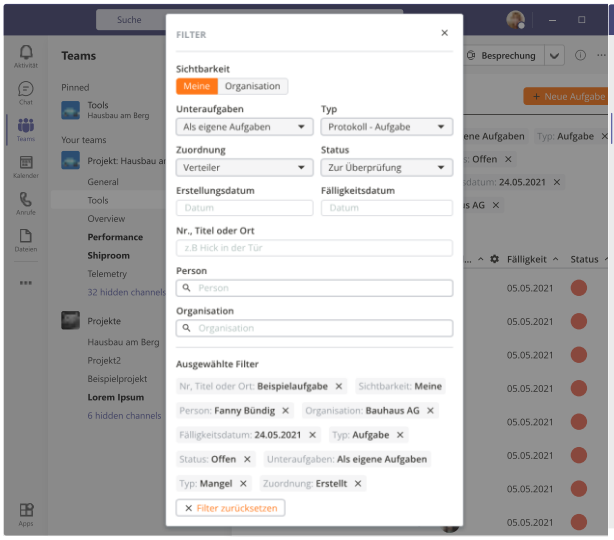


Abbildung 40: Figma - Filter-Modal (smino, 2022)

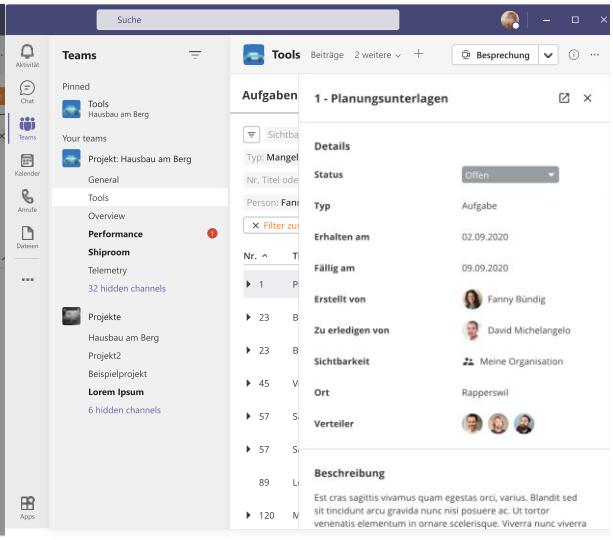


Abbildung 41: Figma – Aufgabendetails (smino, 2022)

Die Filtermöglichkeiten sind ebenfalls die gleichen wie in der smino Web-Applikation, jedoch in einer neuen Form, dem Filter-Modal. Die Details hingegen kommen in einer neuen Sheet- und Tab-Darstellung daher. Dabei werden die aktuell ausgewählten Filter im Modal, sowie auch in der Ansicht selbst, mittels «Chips» dargestellt.

6.4.1 Änderungen

Nachfolgend werden die wichtigsten Änderungen bezüglich der Aufgaben erläutert.

6.4.1.1 Aufgaben im State

Die Issues (Aufgaben) in der smino Teams App sind neu in einem NGXS Feature State.

Dadurch wird die Logik für das Laden von den Issues und den dazugehörigen Collaborators von den Components entkoppelt und im Store behandelt.

Da diverse smino Module `@Input` nicht Immutable behandeln, oder nicht genügend entkoppelt voneinander sind, müssen diese korrigiert werden. Zwei, die stark davon betroffen sind, sind die Sort und Filter Module.

6.4.2 Sort Module

Das herkömmliche `SortModule` hat die zu sortierenden Elemente als Liste erhalten und bei einer Sortierungsänderung diese Liste entsprechend aktualisiert. Auch die mitgegeben `ISortingConfiguration` wird vom `SortModule` aktuell gehalten.

Neu werden nur noch die `ISortingConfiguration` als Immutable Input Variablen benötigt und die zu sortierenden Elemente müssen nicht mehr übergeben werden. Das Module selbst ändert keine Input-Daten mehr, sondern Teilen die gewünschten Sortierungsänderungen, über einen `EventEmitter`. Diese Änderungen werden neu im Store verarbeitet.

Damit ist die Wiederverwendung des Modules vereinfacht.

Ein grosser Vorteil davon ist zudem, dass es dem `SortModule` keine Rolle mehr spielt, ob Client- oder Serverseitig sortiert wird.

6.4.3 Filter Module

Das `FilterModule` hat die `IFilterContainerConfig` und die zu filternde Elemente als Liste erhalten und deren State bei Bedarf aktualisiert.

Neu werden nur noch die `IFilterContainerConfig` als Immutable Inputs benötigt, sowie eine Liste der gefilterten Filter-Values und nicht mehr die zu filternde Elemente. Jegliche Änderungen an den Filter werden über einen `EventEmitter` mitgeteilt. Die gewünschte Filterung wird anschliessend im Store behandelt.

Ein grosser Vorteil davon ist zudem, dass es dem `FilterModule` keine Rolle mehr spielt, ob Client- oder Serverseitig gefiltert wird.

6.5 Übersicht der smino Module

Module Name	Kurzbeschreibung	Änderung	Übernahme
Alert Label	Darstellung von Alerts mit Tooltips	Minimal	Komplett
*Algorithms****	Verschiedene Hilfsalgorithmen	Minimal	Partiell
Attachments	Darstellung der verschiedenen Dateitypen (Bilder, Dokumente, etc.)	Diverse	Partiell
Avater	Darstellung Benutzerprofile als Bild	Minimal	Komplett
Blueprint Marker	Darstellung der Blueprint Marker Dateien	Minimal	Partiell
Collaborator Avatar List	Darstellung der Collaborator als Liste von Avataren	Minimal	Komplett
Collaborator Company Pipes	Diverse Pipes für die Darstellung der Firma eines Collaborators	Minimal	Komplett
Collaborator Name Pipe	Pipe für die Darstellung des Namens eines Collaborators	Minimal	Komplett
*Common**	Allgemeine Klassen, Methoden und Konstanten	Minimal	Partiell
Date Core	Allgemeine Datumspezifische Komponenten	Minimal	Komplett
Dropdown	Darstellung von Buttons als Dropdowns.	Wenige	Komplett
*Events**	Diverse JavaScript Event Interception	Minimal	Komplett
File Upload	Input für den Upload von Dateien	Diverse	Partiell
Filter	Diverse Filter	Komplett	Komplett
Image	Bilder spezifische Komponenten	Minimal	Partiell
Internationalization	Funktionalitäten für Übersetzungen	Diverse	Partiell
*List Node**	Hilfskomponenten für die hierarchische Darstellung von Aufgaben	Minimal	Komplett
Loading Container	Darstellung einer Loading-Animation und allfälligem Loading-Error	Minimal	Komplett
Mailto	Funktionalitäten für Mailings	Minimal	Partiell
Modal	Darstellung eines Modals	Minimal	Komplett
Monitoring	Funktionalitäten fürs Monitoring, wie z.B fürs Logging oder für Analytics	Diverse	Komplett
Property Transition	Darstellung von Änderungsverläufen	Minimal	Partiell
Sanitation	Dom sanitation helpers	Minimal	Komplett
Search	Komponenten für Personensuchen	Minimal	Partiell
Sheet	Darstellung von Inhalten in einer Seitenansicht (Sheet)	Neu	-
Sort	Auf- und Absteigendes sortieren von Listen	Komplett	Komplett

Module Name	Kurzbeschreibung	Änderung	Übernahme
Storage	Client-Seitiges Speichern von Objekten in der Applikation (Local Storage, Session Sotrage, etc.)	Minimal	Partiell
Support	Komponenten für Support Referenzen	Diverse	Partiell
Tabs	Darstellung von Inhalten in Tabs	Neu	-
Toast	Darstellung von Inhalten in einem Toast (auch bekannt als Snackbar)	Minimal	Komplett
Tooltip	Darstellung von Inhalten in einem Tooltip	Minimal	Komplett
UI	Diverse Hilfskomponenten für die Darstellung von Inhalten	Minimal	Partiell
User Name Pipe	Pipe für die Darstellung eines Benutzernamen	Minimal	Komplett

Tabelle 8: Übersicht Anpassung smino Module (eigene Darstellung)

**Keine effektiven Module*

6.5.1 Übernahme

Bei der Übernahme wird zwischen «Komplett» und «Partiell» unterschieden.

Eine komplette Übernahme ist dann gegeben, wenn alle Funktionalitäten und Komponenten eines Modules in die smino Teams App übernommen wurden.

Bei den partiellen Übernahmen sind nur diejenigen Funktionalitäten kopiert, welche auch eine Anwendung in der smino Teams App haben.

6.5.2 Änderung

Module mit minimalen Änderungen sind grundsätzlich komplett aus dem smino Web-Applikation übernommen und enthalten nur kleinere Refactorings wie zum Beispiel:

- Umstrukturierungen der Ordnerstrukturen
- Strict-Null Checking Ergänzungen
- Ergänzungen von Barrel Module-Files

Die Beschreibungen zu den komplett überarbeiteten Modulen finden sich in dem Kapitel Sort Module, sowie Filter Module.

Für die neuen Module gibt es keine Änderungsübersicht. Ihre Anwendungen und Darstellungen sind aber im Kapitel Aufgaben ersichtlich.

Nachfolgend werden die Änderungen beschrieben, die als Module mit diversen Änderungen markiert sind.

6.5.2.1 Attachments

Die eingebundenen Komponenten sind ohne ihre Input-Funktionalitäten übernommen. Sie werden lediglich für die Darstellung ihrer Inhalte benötigt.

6.5.2.2 Dropdown

Die Abhängigkeit zum Core Module wurde entfernt und die notwendigen Komponenten und Services direkt im Dropdown Module eingebunden.

6.5.2.3 File Upload

Die File Upload Komponente wurde nur mit der Download Funktionalität übernommen. Die weiteren Funktionen, wie das Editieren oder Löschen von Dateien, wurde entfernt, da sie in der smino Teams App zurzeit noch nicht benötigt werden.

6.5.2.4 Internationalization

Für die Internationalization sind die folgenden Änderungen vorgenommen worden:

- Anpassung des Module-Namen.
- Aufteilung von `directives`, `models`, `pipes`, `services` und `store` in eine dementsprechende Ordnerstruktur.
- Der `DefaultLanguageService` wurde entfernt und neu werden die Konfigurationen über den `InternationalizationConfigProvider` zur Verfügung gestellt.
- Die **Store** Logik ist neu in einem eigenen `InternationalizationStoreModule` entkoppelt.
- Der `TranslationsState` kann sich neu alle verwendeten Sprachen merken.
- Der `TranslationCacheService` wurde entfernt, das Caching wird nun über den NGXS Store und der entsprechenden LocalStorage Lösung gehandelt.
- Der `TranslationService` wurde dementsprechend angepasst und lädt neu alle Übersetzungen aus der `TranslationsFacade`.

6.5.2.5 Monitoring

Für das Monitoring sind die folgenden Änderungen vorgenommen worden:

- Der `LoggingService` wurde abstrahiert, so dass nun jeweils die spezifische Implementation zur Verfügung gestellt werden kann (z.B. als `ConsoleLoggingService`)
- Das Module verwendet neu einen `AnalyticsConfigProvider` für die Konfigurationen

6.5.2.6 Support

Die Konfigurationen werden neu über den `SupportConfigProvider` zur Verfügung gestellt und nicht mehr in den Environment-Variablen.

6.6 Theming

6.6.1 Ausgangslage

smino verendet für das Styling die Sass Extension mit der scss- Syntax. Dabei wird nicht das Angular Component-Styles eingesetzt, sondern eine eigene definierte Module-Styles-Architektur verwendet (Abbildung 42: Styles-Architektur).

Verschiedene Themes sind bis anhin nicht implementiert. Die verschiedenen verwendeten Farben sind als Sass-Variablen in einem config-File hinterlegt.

Die einzelnen Sass Module beinhalten sämtliche Styling-Logik. Das heisst, es wird kein Unterschied zwischen Typographie, Farb- oder Layoutstyling gemacht.

```scss
$color_dark: #3b3a36;
$color_dark-alpha-90: rgba(59, 58, 54, 0.9);
$color_dark-alpha-80: rgba(59, 58, 54, 0.8);
$color_dark-alpha-40: transparentize($color_dark, 0.6);
$color_bright: #fff;
$color_brand: #ff7819;

// ...

h1.panel__title {
  font-size: $h2_font-size; // Typography-Styling
  font-weight: $font-weight-bold; // Typography-Styling
  padding-top: rem(15); // Layout-Styling
  padding-bottom: rem(15); // Layout-Styling
  color: $color_brand; // Color-Styling
}
```

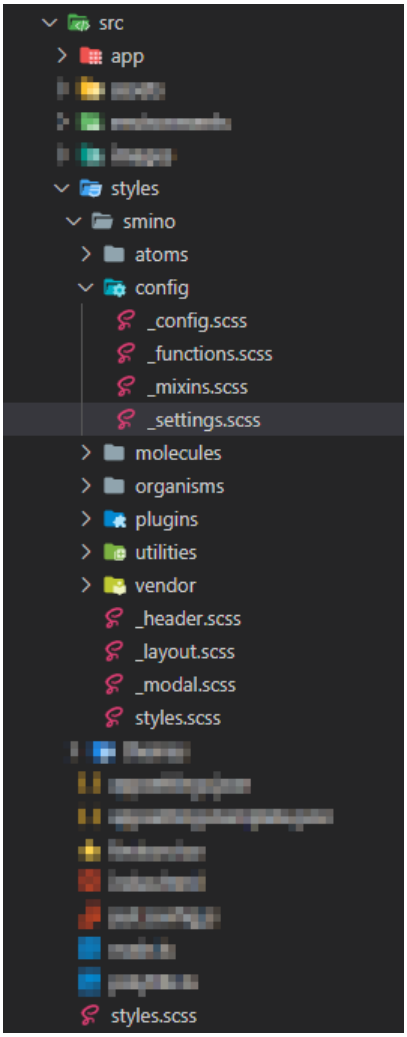


Abbildung 42: Styles-Architektur (eigene Darstellung)

6.6.2 Anpassungen für das Dark-Theme im Rahmen dieser Arbeit

Das Theming der smino-Components ist nicht im Scope dieser Bachelorarbeit. Da aber Microsoft-Teams ein Light-, Dark- und Contrast-Mode für Teams zur Verfügung stellt und die beiden Modi, Light und Dark, regelmässig verwendet werden, muss sichergestellt werden, dass zumindest alle smino-Components, welche in der Teams-Applikation verwendet werden, einen Dark-Mode unterstützen. Dies wurde wie folgt realisiert:

Farbkonzept:

Die Farben des Dark-Theme sind in einer sehr einfachen Sass-Map abgebildet und können mit `sass:map` ausgelesen werden:

```
@use "sass:map";

$dark-theme: (
  background: #000000,
  teams-blue: #7479dc,
  smino-orange: #ff7819,

  primary: #ffffff,
  primary-25: #fafafa,
  primary-50: #e1e1e1,
  primary-250: #c7c7c7,
  primary-300: #b3b3b3,
  primary-350: #949494,
  primary-550: #3d3d3d,
  primary-700: #292929,
  primary-800: #1f1f1f,
  primary-900: #0f0f0f,
);

// Auslesen einer Farbe:
p {
  color: map.get($dark-theme, primary-50);
}
```

Theming

Das Theming der smino-Components ist mit einem Basic-`@mixin`-Approach realisiert. Dazu sind aus den verwendeten smino-Styles die Color-Stylings in einer duplizierten Module-File-Architektur herausgearbeitet. In diesen Modulen gibt es jeweils ein `@light`- und `@dark`-mixin, welche sich um die korrekte Farbcodezuweisungen der Attribute kümmern.

Im Light-Theme sind die bestehenden Farbcodes hinterlegt, im Dark-Theme die neuen Farbcodes mittels ausgelesen aus der definierten Sass-Map.

Das aktive Theme muss jeweils als Css-Class auf einem beliebigen Parent `HTML<element>` hinterlegt werden. Für unsere Zwecke wird hierzu auf dem `HTML<body>` entweder die Light-Theme oder Dark-Theme Klasse gesetzt:

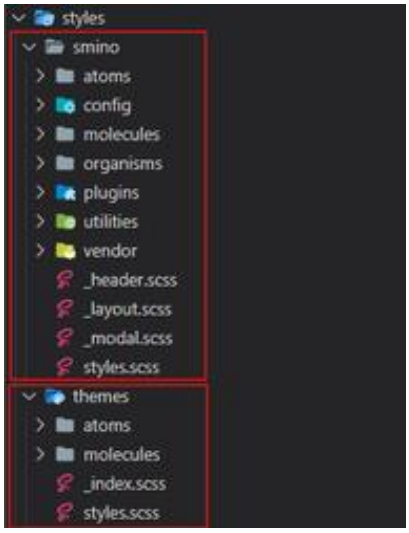


Abbildung 43: Theming-- Architektur (eigene Darstellung)

```
<body class="light-theme">...</body>
```

```
/** styles.scss  */
@use "./smino/index" as smino;
.light-theme {
  @include smino.all-component-themes($light-theme: true);
}
.dark-theme {
  @include smino.all-component-themes($light-theme: false);
}

/** smino/_index.scss */
@use "./panel";
@mixin all-component-themes($light-theme: true) {
  @include panel.theme-colors($light-theme);
}

/** smino/_panel.scss */
@use "sass:map";
@use "config/settings";

@mixin theme-colors($light-theme: true) {
  @if $light-theme {
    @include light;
  } @else {
    @include dark;
  }
}

@mixin light {
  .panel {
    background: settings.$color__bright;
  }
}

@mixin dark {
  .panel {
    background: map.get(settings.$dark-theme, primary-800);
  }
}
```

Bekannte Probleme

- **Farbkonzept**: Die Farbcodes sind in keiner einheitlichen Form hinterlegt. Dies kann zu Duplikationen und verwirrenden Namensgebungen führen.
- **Theming**: Mit dem gewählten Basic-`@mixin`-approach müsste je neuem Theme je Modul eine weitere `@mixin` für dieses Theme definiert werden, was einen grösseren Aufwand bedeutet.
- **Typografie*:** Die Typografie ist ebenfalls mit den Layout-Styles gekoppelt. Diese müssten in gleicher Art und Weise wie die Farbcodes aus den bestehenden Styles herausgearbeitet werden.

**Da smino aber nur eine Typografie verwendet, kann dies als ein "nice-to-have" eingestuft werden.*

6.6.3 Zukunft: Theming smino-Components

In diesem Abschnitt wird beschrieben, wie ein allfälliges, ausgereifteres Konzept für das Theming der smino-Components aussehen könnte.

Farbkonzept

Um das Generieren von verschiedenen Themes zu vereinheitlichen, wird eine Theme-Konfiguration notwendig. Dabei wird eine Konfiguration für jedes Theme benötigt. Zurzeit wären dies die beiden Konfigurationen Light-Theme-Config und Dark-Theme-Config. Angelehnt an die Material Design Guidelines empfehlen wir die folgenden Farben zu definieren:

- **Background** Farbe, die hinter scrollierbaren Elementen (Container) erscheint.
- **Contrast** Standard Textfarbe (Entspricht der Materia Design "On Background" Farbe)
- **Error** Farbe für Fehlermeldung
- *Warn** Farbe für Warnungen
- *Info** Farbe für Infos
- *Success** Farbe für Erfolge
- **Primary** Farbenpalette
- **Secondary** Farbenpalette

**Abweichende Definitionen vom Material Design Guide*

Farbenpalette

Die Farbenpalette wird für die Abstufung von verschachtelten Elementen benötigt. Dabei lohnt es sich, pro Abstufung noch eine zusätzliche Kontrastfarbe (contrast) anzugeben, welche für die Darstellung des Textes verwendet wird (diese entspricht der Material Design "On Primary | On Secondary" Farbe). Damit wird sichergestellt, dass die Schrift auf jeglichem Hintergrund gut lesbar ist.

Für die Accessabillity der Farben und Kontrastfarben stellt Material ein hilfreiches Color Tool[12] zur Verfügung.

Eine allfällige Theme-Konfiguration könnte wie folgendermassen aussehen:

```
// Theme-Konifuration
$dark-theme-config: (
  background: $black,
  contrast: $white,
  error: $error,
  warn: $warn,
  info: $info,
  success: $success,
  primary: $greys, // sass:map color-palette
  secondary: $smino-orange, // sass:map color-palette
);
```

12 https://material.io/resources/color/#!/?view.left=0&view.right=0&secondary.color=ff7819&primary.color=212121

Theming

Pro Theming-Module muss nun nur noch ein `@mixin` definiert werden. Dieses erhält jedoch nicht mehr nur die Info, um welches Theming es sich handelt, sondern die komplette Konfiguration (`$dark-theme-config`)

```
// Theme-Konfigurationen
$light-theme-config: (/* ... */);
$dark-theme-config: (/*...*/);

// styles.scss
.light-theme {
  @include smino.all-component-themes($light-theme-config);
}
.dark-theme {
  @include smino.all-component-themes($dark-theme-config);
}

// smino/_index.sxss
@mixin all-component-themes($theme-config) {
  @if $theme-config == null {
    @error 'No theme configuration specified.';
  }

  @include layout.theme-colors($theme-config);
}

// smino/_layout.scss
@mixin theme-colors($config) {
  body {
    color: map.get($config, contrast);
    background-color: map.get($config, background);

    .container {
      $color: map.get($config, primary, contrast, 600);
      $background-color: map.get($config, primary, 600);
    }
  }
}
```

(Implementing your theme, 2022)

6.7 Mobile Ansicht Teams Ansatz

Der grundsätzliche Ansatz für Teams wäre, dass anstatt der Implementation einer Ansicht für mobile Geräte, die native smino Applikation benutzt wird.

Der Vorteil dabei liegt darin, dass für die mobile Ansicht ein anders Navigationskonzept verwendet werden müsste. Ebenso sind auch die aktuellen Komponenten nicht auf die Benutzung mit einem mobilen Gerät ausgelegt, wodurch diese wesentlich überarbeitet werden müssten.

Ein anderer Ansatz wäre, eine komplett separate Web-Applikation oder komplett separate Pages für die mobilen Geräte zu erstellen. Dies würde aber ebenso einen enorm hohen Aufwand bedeuten, zumal bereits eine native App für die Geräte existiert, welche für diese auch entsprechend optimiert ist.

Um dies zu erreichen, wären folgende Schritte notwendig:

1. Detektieren der Plattform in der Teams Applikation. Entscheidung treffen, ob man sich auf einem mobilen Gerät befindet.
2. Wenn man sich auf einem mobilen Gerät befindet, darf die Anwendung entsprechend nicht geladen werden und es darf keine Authentifizierung angestossen werden.
3. Identifizierung des Projekts und des Bereiches, den man gerne öffnen würde.
4. Folgend müsste auf dem Gerät ein Deferred Deep Link auf die mobile App geöffnet werden. Dieser soll bewirken, dass die App geöffnet wird oder man zum Store für eine Installation weitergeleitet wird.

6.7.1 Detektieren der Plattform

Über das Teams SDK, kann der aktuelle Kontext in Teams abgefragt werden. Dabei ist auf dem Kontext hinterleg, auf welcher Host Plattform man sich aktuell befindet.

```
this.teamsService.teamsSdk.getContext((context) => {
      switch (context.hostClientType) {
        case microsoftTeams.HostClientType.android:
          break;
        case microsoftTeams.HostClientType.desktop:
          break;
        case microsoftTeams.HostClientType.ios:
          break;
        case microsoftTeams.HostClientType.surfaceHub:
          break;
        case microsoftTeams.HostClientType.web:
          break;
      }
    });
```

Abbildung 44: Detektieren der Plattform über Teams SDK (eigene Darstellung)

6.7.2 Laden der Anwendung abbrechen

Hierfür würde sich ein zentraler Guard anbieten, welcher vor der Authentifizierung aufgerufen wird. Wenn die entsprechende Plattform nicht dem Desktop oder einer nicht unterstützende Plattform entspricht, wird die Aktivierung abgebrochen. In einem Fehlerfall könnte dem Nutzer auch eine Seite mit den nötigen Informationen angezeigt werden.

6.7.3 Detektieren des Projekts und des Bereichs

Die Url der Tabs-Applikation, welcher ein Benutzer versucht aufzurufen, ist grundsätzlich bekannt. Entweder könnte hier die Url geparsed oder eine Art Mapping hinterlegt werden, welche die Tab-Url auf eine Url der mobilen Applikation konvertiert.

6.7.4 Öffnen der mobilen Applikation

Grundsätzlich muss für die mobile Anwendung ein Deep Link erstellt werden. Dies erlaubt es, Links über die Applikation zu öffnen. In einem Blogeintrag mit dem Thema «Open Mobile Application From The Browser» wird das Thema etwas angeschnitten.[13] Die korrekte Erstellung der Links, sowie eine entsprechende Weiterleitung und Fehlerbehandlung birgt eine gewisse Komplexität.

Hierbei könnte überlegt werden, ob dies in einem eigenen Micro-Service behandelt wird, um die Komplexität auszulagern und für diverse Integrationen verfügbar zu machen.

Ebenso besteht die Möglichkeit, diese Funktionalität einzukaufen, wobei solche Provider auch noch weiter Funktionalitäten bieten. AppsFlyer wäre ein solcher Provider, welcher die Deep Linking Möglichkeiten auch im Detail beschreibt.[14]

[13] https://vhudyma-blog.eu/open-mobile-application-from-the-browser/
[14] https://www.appsflyer.com/resources/guides/deep-linking-101/

7. Bereitstellung der Applikation

In diesem Kapitel wird die Bereitstellung der Applikation etwas näher erläutert. In einem ersten Schritt werden die verschiedenen Varianten dargestellt, wie eine Teams Applikation in einem Teams Client verfügbar gemacht werden kann. In einem nächsten Schritt wird die konkrete Architektur der smino Teams Applikation dargestellt und die verwendeten Pipelines erläutert, welche das Deployment ermöglichen.

7.1 Store der Organisation

Pro Organisation können "Custom Apps" hochgeladen werden. Das App kann von einem Entwickler kommen und muss entsprechend durch die O365 Administration bewilligt werden. Dafür existiert auch eine entsprechende Submission API. Alternativ kann das App auch direkt durch die O365 Administration hochgeladen werden. Dabei kann zwischen dem effektiven "Publish" der App und dem "Upload" einer App unterschieden werden. Beim Hochladen wird die App nur lokal für den Anwender verfügbar und nicht für andere Personen. Ist das App aber einmal in einem Team hinzugefügt, ist dies auch für andere Anwender sichtbar. (Microsoft, 2022)

Der direkte «Upload» der App wird auch «Sideloading» genannt und muss für die Organisation aktiv freigegeben werden. Für die Entwicklung mit Teams ist es essenziell, dass diese Option verfügbar ist. (Microsoft, 2022)

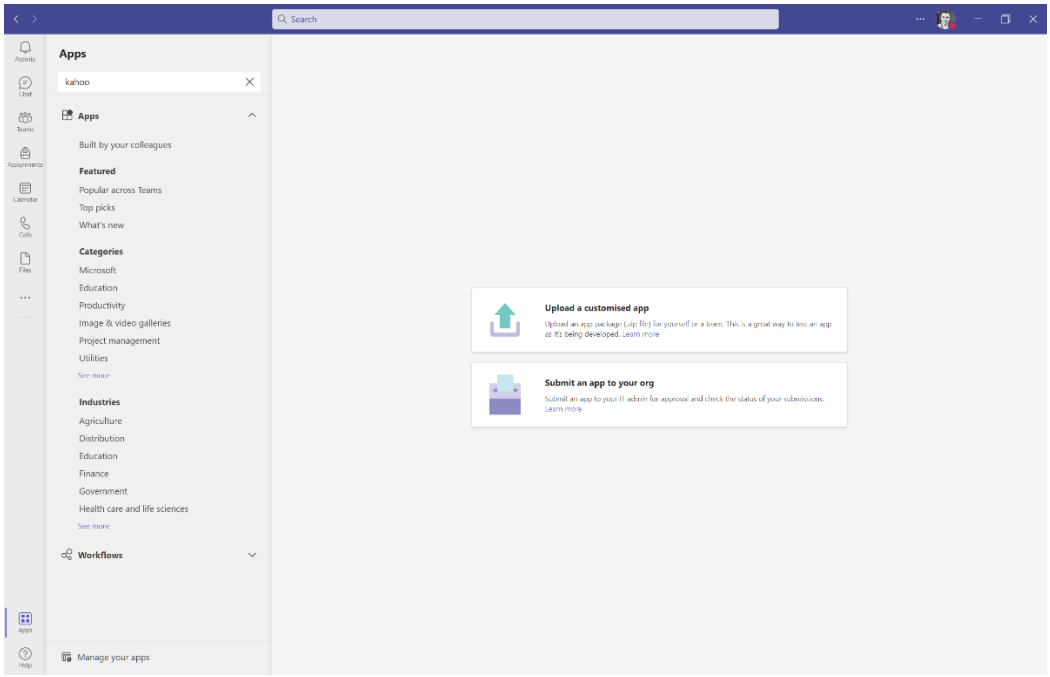


Abbildung 45: Teams Client App Upload (Microsoft, 2022)

7.2 Öffentlicher Microsoft Teams Store

Um die Applikation für alle Nutzer weltweit zur Verfügung zu stellen, kann diese im öffentlichen Store publiziert werden. Applikationen im Teams Store werden automatisch in Microsoft AppSource, der offizielle Marketplace für Microsoft 365 Applikationen, gelistet. Wichtig bei der Veröffentlichung der Applikation ist, dass diese durch Microsoft intensiv überprüft und entsprechend auch freigegeben wird. (Microsoft, 2022)

Der Prozess zur Publikation der Applikation kann in folgende Schritte aufgeteilt werden:

1. Überprüfung der Teams Store Validierungsrichtlinien. Dies soll sicherstellen, dass Teams-App-Standards und Store-Standards erfüllt werden.[15]
2. Partner Center Entwicklerkonto erstellen.[16]
3. Store Übermittlung vorbereiten.[17] Dazu gehört das eigentliche Testen der Applikation, sowie die Vorbereitung des Überprüfungsprozesses für Microsoft.
4. Übermittlung der App über das Partner Center.[18]
5. Wenn bei der Publikation der Applikation Fehler auftreten oder bei der Überprüfung durch Microsoft Mängel festgestellt werden, können diese in Zusammenarbeit mit Microsoft behoben und die App erneut übermittelt werden.[19]

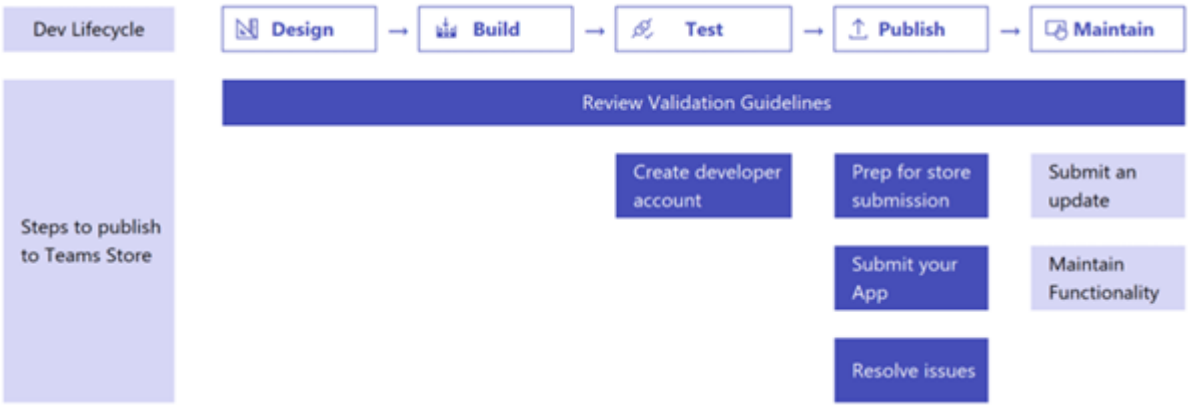


Abbildung 46: Publish Process für Teams (Microsoft, 2022)

[15] https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/appsource/prepare/teams-store-validation-guidelines
[16] https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/appsource/prepare/create-partner-center-dev-account
[17] https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/appsource/prepare/submission-checklist
[18] https://docs.microsoft.com/en-us/office/dev/store/add-in-submission-guide
[19] https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/appsource/resolve-submission-issues

7.3 Fokus der Arbeit

Im Rahmen dieser Arbeit wurde der Fokus vor allem auf die Entwicklung und das Testing der Applikation gelegt, wobei das Ziel eine Betaversion der Applikation ist. Die Publizierung in den öffentlichen Store wird durch smino im Anschluss an die Arbeit durchgeführt. Dies ist vor allem auch durch die zeitliche Limitierung bedingt, wobei die Publizierung der App und eine allfällige Überarbeitung ein hohes Risiko birgt. Dieses Risiko entsteht vor allem durch die Abhängigkeit von Microsoft und den Mitteln, welche für die Publizierung in den Store notwendig sind. Um dies zu berücksichtigen, müsste die Entwicklung viel früher abgeschlossen werden und die Übergabe an smino bereits erfolgt sein.

Für die Publikation in den Store ist auch ein Partner Account notwendig, welcher die Schnittstelle zwischen einem Unternehmen und Microsoft bildet. Im Rahmen der Arbeit wäre es schwierig, einen separaten Partner Account zu erstellen, um die Publizierung der Applikation zu testen.

7.4 Vorschlag Architektur smino

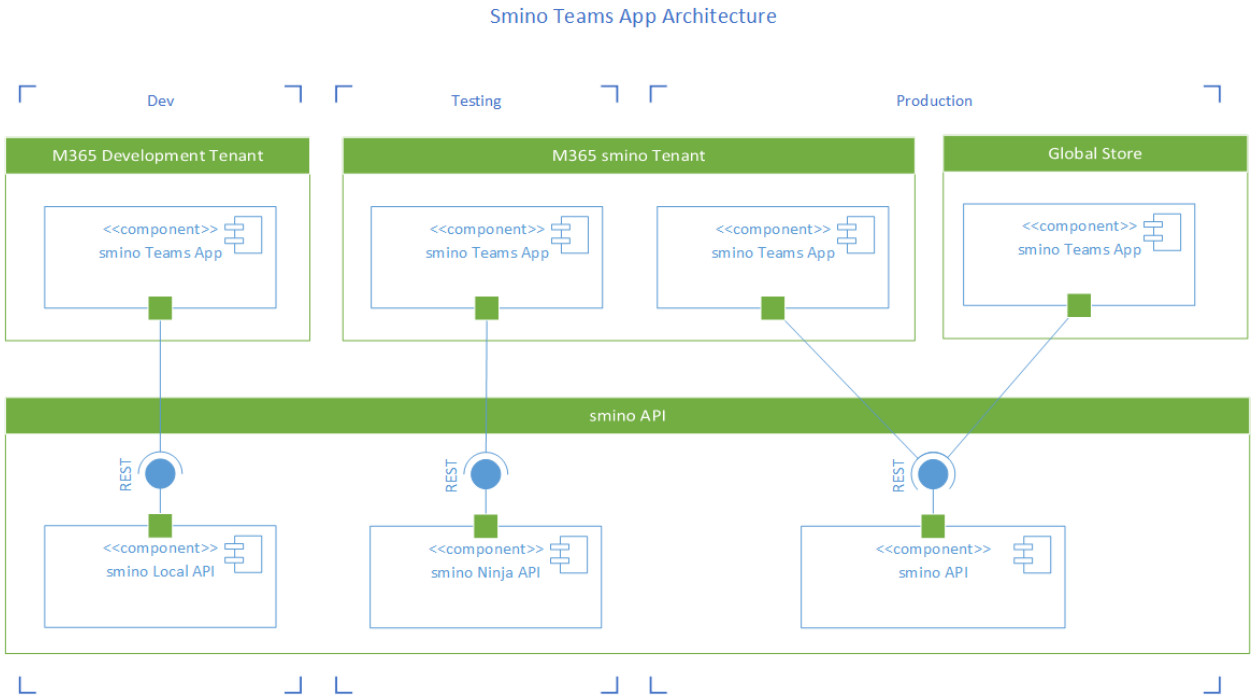


Abbildung 47: Vorschlag Systemarchitektur smino (eigene Darstellung)

Für die Architektur wird zwischen drei Umgebungen unterschieden: Development, Testing und Production. Dies richtet sich auch nach den verfügbaren Umgebungen der smino API.

Es können drei Versionen der App unterschieden werden, welche zu einem gewissen Zeitpunkt verfügbar sind:

- **Development:** Die App wird entwickelt und über «Sideloading» im aktuellen Teams Client getestet. Die App sollte grundsätzlich eine lokale Version der API verwenden.
- **Testing:** Ein Testversion der Teams App, welche die Ninja Umgebung von smino benutzt. Die App ist effektiv im Organization-Store publiziert.
- **Production:** Die produktive Version der App, welche auch die produktive API von smino verwendet. Die App sollte einerseits im Organization-Store, sowie auch im Global-Store publiziert werden.

Für M365/Teams werden drei Umgebung unterschieden:

- **Development Tenant:** Tenant, der für die Entwicklung der Teams App zur Verfügung steht. Dies erlaubt es einfach an der App zu entwickeln, ohne eine produktive Umgebung zu beeinflussen.
- **smino Tenant:** Dies ist der bereits existierende M365 Tenant von smino. Die Testing und Production Version werden hier im Organization-Store bereitgestellt, wodurch das App entsprechend auch getestet werden kann.
- **Global Store:** Wenn die App für die Produktivierung bereit ist, kann diese manuell über den Partner Center Account im Global-Store publiziert werden.

7.5 Azure Deployment

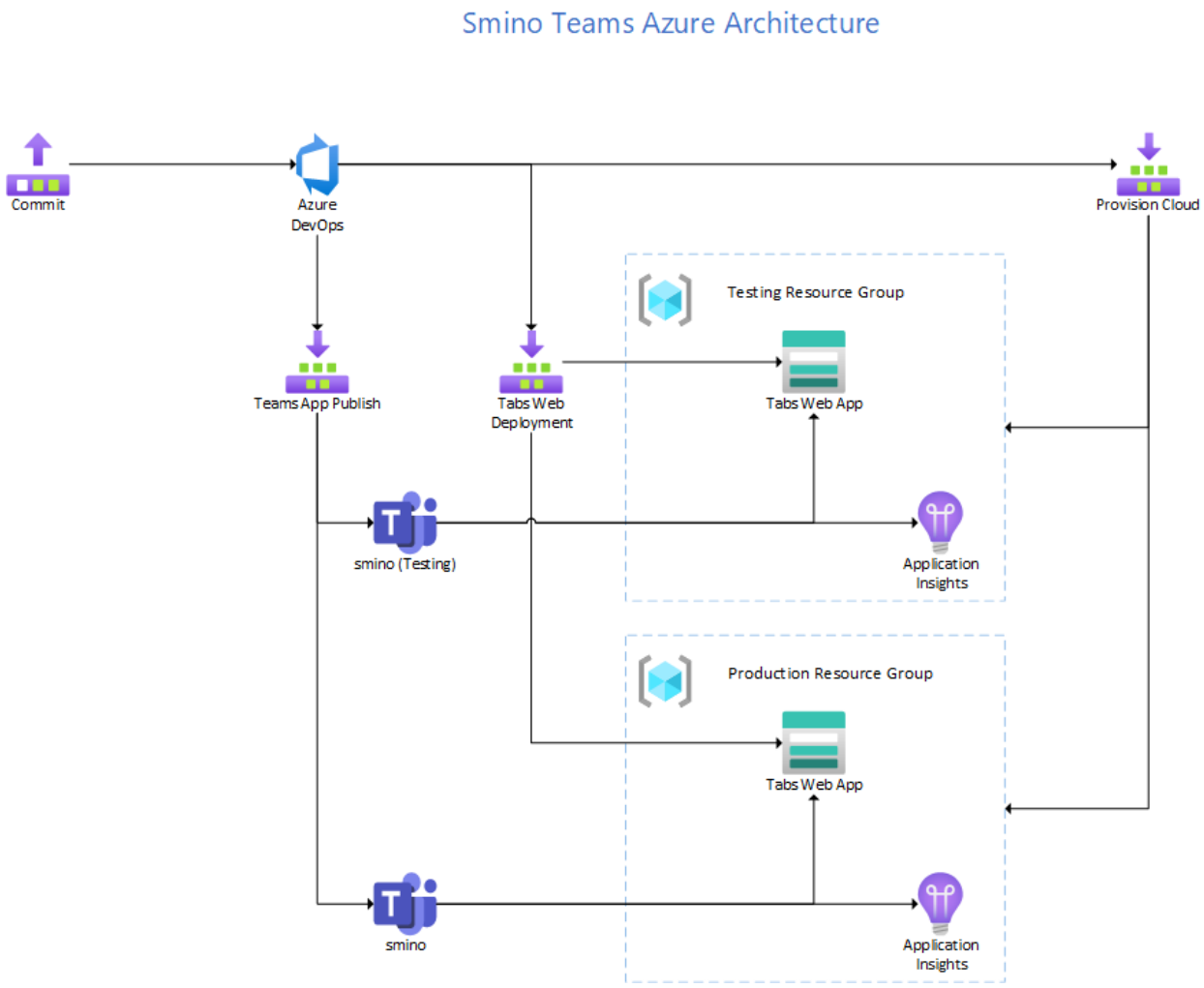


Abbildung 48: Azure Deployment (eigene Darstellung)

Die Provisionierung und das Deployment der Cloud Ressourcen, sowie das «Publishing» der Teams Applikation erfolgt über Azure DevOps. Dabei wird pro Umgebung – Testing und Production – eine Ressourcengruppe angelegt. Dies Architektur kann einfach um weitere Umgebungen ergänzt werden, wobei dafür nur eine neue Ressourcengruppe und die entsprechende Konfiguration der Applikation erforderlich ist.

7.5.1 Commit

Commit entspricht dem effektiven Stand oder einer konkreten Änderung der Versionskontrolle, welche aktuell auf dem Hauptbranch – main – stattfindet. Dies ist die Ausgangslage für die weiteren Schritte.

7.5.2 Azure DevOps

Die entsprechende Azure DevOps Instanz, auf welcher der Source-Code sowie die Pipelines abgelegt sind und entsprechend auch ausgeführt werden.

7.5.3 Ressourcengruppe

Eine Ressourcengruppe ist ein Container, um entsprechende Ressourcen einer Azure Lösung zu gruppieren. Jede Organisation kann selbst entscheiden, wie sie Ressourcen gruppieren möchte, jedoch bietet es sich an, Ressourcen mit dem gleichen Lebenszyklus zu gruppieren. Damit können diese einfach als Gruppe erstellt, aktualisiert oder gelöscht werden. Entsprechend wird für die Teams Applikation eine Ressourcengruppe pro Umgebung angelegt.

7.5.4 Tabs Web-Applikation

Die Ressourcen «Tab Web-Applikation» entsprechen Azure Storage Accounts[20]. Azure Storage Accounts bieten eine einfache Möglichkeit Blobs, File Shares, Queues oder auch Tables zu speichern. Vor allem bietet der Blob Storage eine einfache Möglichkeit, eine statische Website bereitzustellen und wird standardmässig vom Teams Toolkit für das Hosting der Applikation verwendet. Pro Umgebung wird ein separater Storage Account verwendet.

7.5.5 Application Insights

Application Insights wird für das Monitoring der Tabs Applikation verwenden. Application Insights bietet eine einfache Integration in JavaScript basierte Clients, wobei Fehler oder eigene Metriken einfach überwacht werden können. Pro Umgebung wird eine separate Instanz von Application Insights verwendet.

7.5.6 Provision Cloud

«Provision Cloud» ist die Pipeline, welche sich um die Provisionierung der entsprechenden Azure Ressourcen kümmert. Die entsprechende Konfiguration und Definition der Ressourcen wird mithilfe von Bicep[21] definiert und kann über das Teams Toolkit provisioniert werden. Diese Aktion sollte besser manuell ausgeführt werden und ist nur bei Änderungen an den Ressourcen notwendig.

[20] https://docs.microsoft.com/en-us/azure/storage/common/storage-account-overview
[21] https://docs.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview

7.5.7 Tabs Web Deployment

Die Pipeline «Tabs Web Deployment» stellt die Tabs Web-Applikation im Blob Storage bereit. Im Prinzip bedeutet dies, einen Build für die Angular Applikation auszuführen und die entsprechenden Artefakte in den Blob Storage zu kopieren. Das Deployment der Teams Applikation ist in das Teams Toolkit integriert und kann damit pro Umgebung entsprechend ausgeführt werden.

7.5.8 Teams App Publish

Diese Pipeline kümmert sich um die Publikation der Teams Applikation, also des App Packages, in den definierten Organizational Store. Zudem wird pro App Package ein Artefakt hinterlegt, womit der «Upload» jederzeit manuell ausgeführt werden kann. Dies ist vor allem für die produktive Version der App relevant, wenn diese in den Global Store publiziert werden soll. Das Publishing ist ebenfalls im Teams Toolkit intergiert und kann pro Umgebung ausgeführt werden.

8. Testing

Dieser Abschnitt erläutert die umgesetzten Tests. Er zeigt auf, welche Bereiche mit Tests abgedeckt und welche aussen vor gelassen wurden.

smino selbst hat keine oder nur sehr wenige Tests für ihre Komponenten und dementsprechend auch keine Vorgaben diesbezüglich für das smino Teams App. Im folgenden Teil werden diejenigen Tests beschrieben, die im Rahmen dieses Projekts als sinnvoll erachtet wurden.

8.1 Components Tests

Der Fokus für das Testing lag auf den langlebenden, grundlegenden Services und Komponenten der Applikation. Zu den ausführlich getesteten Bereichen gehören die folgenden Module / Services:

- `SortModule`: Komplette Überarbeitung des Basismodules
- `FilterModule`: Komplette Überarbeitung des Basismodules
- `InternationalizationModule`: Komplette Überarbeitung des Basismodules
- `SheetModule` Eigenständiges Basismodule
- `TabModule` Eigenständiges Basismodule
- `IssueDetailsFacade`: Basis service für alle Store-Logik

Aus zeitlichen und wartbarkeitstechnischen Gründen wurde darauf verzichtet für folgende Bereiche Tests zu schreiben:

- Übernommene, unveränderte smino Komponenten
- Kurzlebende und / oder einfache Komponenten

8.2 e2e Tests mit Cypress

API-Calls an die smino API sowie der Microsoft Teams Kontext wurden durch Mocks ersetzt. Einerseits, um die Komplexität der Tests zu reduzieren, andererseits sollen auch nicht die Microsoft Teams Funktionen getestet werden.

Aufgrund des hohen Wartungsaufwandes wurde nur für die, aus der Sicht des Projektteams, wichtigsten Workflows jeweils ein e2e Test-Case erstellt.

Nachfolgend ist eine Aufzählung an umgesetzten e2e Test Cases mit einer Kurzbeschreibung aufgeführt. Es wird dabei aufgezeigt, was die Tests zu Testen versuchen und dessen Begründung, warum es davon eine Implementation gibt.

8.2.1 Login Workflows

Unter dem «Login Workflow» werden alle Abläufe verstanden, die sich um das Authentifizieren als smino Benutzer und die Autorisierung von Projekten in der smino App drehen.

8.2.1.1 Bad authentication

Beschreibung

Bei einem fehlerhaften beabsichtigten oder unbeabsichtigten Benutzerlogin muss dem Benutzer eine dementsprechende Seite dargestellt werden. Unter anderem muss ein neuer Login-Versuch ausgeführt werden können.

Begründung

Das Login wird von der smino Web-Applikation bearbeitet und ist zurzeit noch nicht für externe Applikationen ausgelegt. Daher ist hier mit einer erhöhten Fehlerzahl zu rechnen.

8.2.1.2 Successful authentication

Beschreibung

Nach einem erfolgreichen Login soll der Benutzer wieder auf die von ihm vorgängig aufgerufene Seite der Web-Applikation weitergeleitet werden. Im Umfang dieser Arbeit wird dieser Workflow ausschliesslich für die Tab Konfiguration benötigt.

Begründung

Dies ist das Standardprozedere für das Einbinden eines neuen Tabs für smino Aufgaben.

8.2.1.3 Bad authorization

Beschreibung

Beim Zugriff auf einen smino Aufgaben Tab, welcher auf ein Projekt zeigt, auf das der zurzeit eingeloggte Benutzer keinen Zugriff hat, muss dem Benutzer eine entsprechende Unbefugte-Projektzugriff-Seite dargestellt werden, in der unter anderem ein Benutzerwechsel ausgeführt werden kann.

Begründung

Ein Tab für die Aufgaben eines Projektes kann pro Microsoft Teams Kanal eingebunden werden. Es ist aber nicht gegeben, dass die Benutzer des Microsoft Teams Kanals auch alle das Zugriffsrechts für das eingebundene smino Projekt haben.

8.2.2 Aufgaben Tab Workflow

Unter dem «Aufgaben Tab Workflow» werden alle Workflows verstanden, die sich um das Filtern, Sortieren oder Anzeigen von Aufgaben drehen.

8.2.2.1 Empty list

Beschreibung

Wird eine leere Liste dargestellt, so muss dem Benutzer eine entsprechende Info dargestellt werden, dass keine Aufgaben gefunden werden konnten.

Begründung

Da komplexe Filterkombinationen möglich sind, ist damit zu rechnen, dass Resultate keine Einträge liefern.

8.2.2.2 Logout

Beschreibung

Auf dem Aufgaben-Tab ist ein Logout aus dem smino Kontext möglich. Nach einem erfolgreichen Logout muss es dem Benutzer möglich sein, sich erneut anzumelden.

Begründung

Dem Benutzer muss es aus Sicherheitsgründen möglich sein, sich aus dem smino Kontext abzumelden.

8.2.2.3 Sorting

Beschreibung

Das Sortieren muss für alle Spalten, wo sinnvoll, möglich sein. Beim Anklicken eines Spaltentitels soll wahlweise auf- oder absteigend sortiert werden.

Begründung

Dies ist eine Hauptfunktionalität der Aufgabenliste.

8.2.2.4 Filtering

Beschreibung

Der Benutzer kann das Filter-Modal öffnen, um entsprechende Filter zu setzen. Diese Filter müssen auf die geladenen Aufgaben angewendet werden und beim Schliessen des Filter-Modal weiterhin bestehen.

Begründung

Dies ist eine Hauptfunktionalität der Aufgabenliste.

8.2.2.5 Reload

Beschreibung

Aufgaben müssen neu von der smino API geladen werden können, ohne dass der bestehende Filter oder die Sortierung verloren gehen.

Begründung

Zurzeit gibt es noch keine Synchronisation zwischen der smino Teams App und dem smino Backend, daher muss es dem Benutzer möglich sein, diese manuell zu starten.

8.2.2.6 Toggle Details

Beschreibung

Die Details einer einzelnen Aufgabe werden in einem Side-Sheet übersichtlich dargestellt, wenn die Aufgabe angeklickt wird.

Begründung

Dies ist eine Hauptfunktionalität der Aufgabenliste.

8.2.2.7 Sub issue tab

Beschreibung

In einem offenen Side-Sheet für eine Aufgabe mit Unteraufgaben werden dem Benutzer diese Unteraufgaben in einem weiteren Tab des Side-Sheets dargestellt. Beim Wechseln auf den Unteraufgaben-Tab soll dem Benutzer eine aktuelle Liste der Unteraufgaben dargestellt werden.

Begründung

Dies ist eine Hauptfunktionalität der Aufgabenliste.

8.2.2.8 Update status

Beschreibung

Beim Aktualisieren eines Status einer Aufgabe in der Aufgabenliste soll dem Benutzer die Aufgabenliste mit den neuen Werten dargestellt werden.

Begründung

Dies ist eine Hauptfunktionalität der Aufgabenliste.

8.3 User Testing

smino hat für das MVP der Aufgabenliste ein User Testing in ihrer Firma organisiert, welches wir begleiten durften.

8.3.1 Ablauf

Die Tester sind jeweils zuerst über ihr bereits vorhandenen Microsoft Teams Kenntnisse befragt worden.

Anschliessend sind Tester, die in der **Fehler! Verweisquelle konnte nicht gefunden werden.Fehler! Verweisquelle konnte nicht gefunden werden.** aufgeführten Aufgaben gestellt worden, die der Tester unter Beobachtung und lautdenkend ausführen musste.

Abschliessend konnten die Tester noch ein allgemeines Feedback zum ersten Eindruck und Funktionsumfang abgeben.

8.3.1.1 Testprotokoll

Aufgabe	**Erwartung**	**Resultat / Bemerkung**
smino Teams App für das Projekt «Hausbau am Berg» in einem Microsoft Teams Kanal einbinden	Projekteinbindung über Teams – Tab hinzufügen und den darauffolgenden Tab konfigurieren.	Test erfolgreich.
Filtern nach einem bestimmten Aufgabetitel mit einem Filter auf einen *falschen* Status.	Filter-Modal via '⧩'-Icon öffnen, Filter anwenden und mittels '×'-Icon wieder schliessen.	Das Öffnen-Icon ist etwas zu unscheinbar. Unklar, ob der Schliessen-Icon Filter diese wieder entfernt.
Den *falschen* Filter auf den Status der Aufgabe entfernen.	Filter entweder via '×'-Icon des Filter-Chips in der Übersicht oder im Filter-Modal entfernen.	Test erfolgreich.
Die Planmarkierungen einer Aufgabe in den Details anzeigen.	Öffnen der Details einer Aufgabe im Side-Sheet mit dem Side-Sheet Scrolling.	Test erfolgreich.
Unteraufgaben der Aufgabe finden und deren Details anzeigen lassen.	Tab-Navigation im Side-Sheet und Unter- zu Hauptaufgabe-Navigation und visa verse.	Tab und Unter- zu Hauptaufgaben-Navigation erfolgreich. Haupt- zu Unteraufgaben-Navigation unklar.
Die Attachments der Aufgabe finden.	Attachments in den Details finden und Absprung in die smino Web-Applikation verstehen.	Test erfolgreich.
Den Status der Aufgabe ändern.	Status in der Aufgabenliste oder in den Aufgabendetails ändern.	Test erfolgreich.
Die Sichtbarkeit der Aufgabe ändern.	Status der Sichtbarkeit einer Aufgabe in den Details ändern.	Test erfolgreich.
Die Aufgaben von der smino API neu laden.	Bewusst die Aufgaben neu laden und verstehen, dass die Synchronisation (noch) nicht automatisch ist.	Implementierter Refresh-Button zu unscheinbar. Teams Funktion (Tab wechseln, Tab neu laden) anstelle genutzt.
Die Beschreibung der Aufgabe aktualisieren.	Absprung in die smino Web-Applikation verstehen.	Test erfolgreich.

Tabelle 9: User Testing – Testprotokoll (eigene Darstellung)

8.3.2 Fazit

Das User Testing war für das Projektteam eine neue und interessante Erfahrung. Es wurden einige UX-Probleme aufgedeckt, die ohne Testing nicht erkannt worden wären.

Dennoch war das Tester-Feedback sehr positive ausgefallen und hat gezeigt, dass mit dieser Arbeit ein erster Schritt in die richtige Richtung getan werden konnte.

Das aus unserer Sicht Wichtigste aus diesem User Testing war, dass während dem ganzen Testing kein einziger technischer Fehler aufgetreten ist.

9. Resultat

Die Beta-Version der smino Teams App ist mittels User-Testing in der Microsoft Teams Web-Applikation erfolgreich getestet worden.

Die Einbindung der smino Teams App, die Authentifizierung des smino Benutzers und die Projektauswahl aus den für den Benutzer zur Verfügung stehenden Projekten kann elegant in der von Microsoft Teams vorgegebenen Workflow abgehandelt werden.

Auch die verschiedenen *Unauthenticated-*, *Unauthorized-* oder *Forbidden*-Ansichten sind implementiert und ermöglichen dem Benutzer jeweils eine entsprechende Aktion auszuführen.

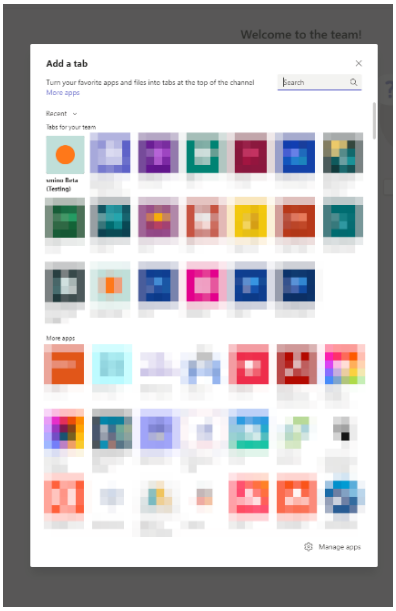


Abbildung 49: smino Teams App hinzufügen (eigene Darstellung)

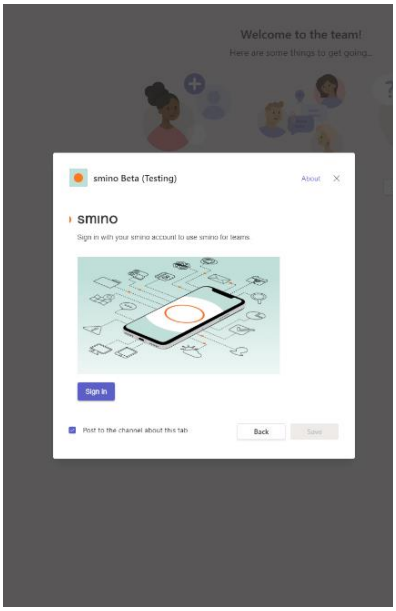


Abbildung 50: Tab-Konfiguration Login (eigene Darstellung)

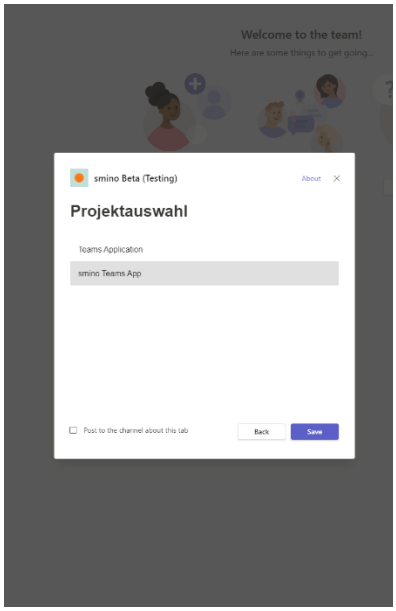


Abbildung 51: Tab-Konfiguration Projektwahl (eigene Darstellung)

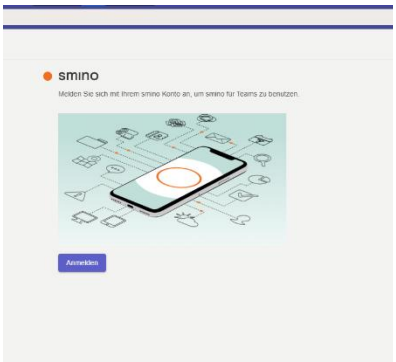


Abbildung 52: Tab - Unauthenticated (eigene Darstellung)

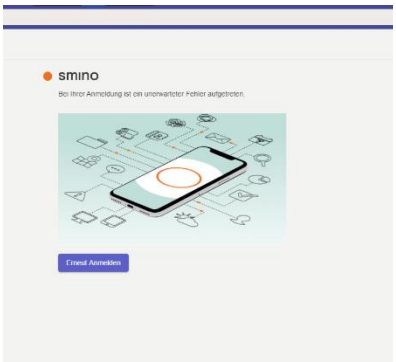


Abbildung 53: Tab - Unauthorized (eigene Darstellung)

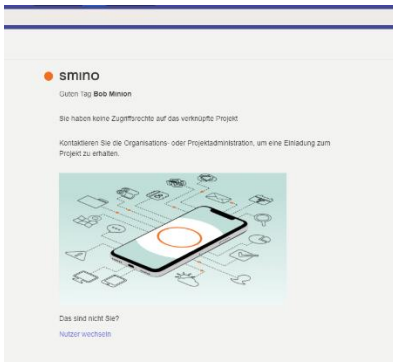


Abbildung 54: Tab - Forbidden (eigene Darstellung)

Der Aufgaben-Tab zeigt alle Aufgaben des in der Projektauswahl konfigurierten Projektes. Eine vereinfachte Übersicht der angewendeten Filter ist ersichtlich und allenfalls entfernbar, die einzelnen Spalten können nach Belieben auf- oder absteigend sortiert werden, der Tooltip für eine Aufgabe wird bei einem «Hoven» dargestellt und die inline-edit Methode für das Aktualisieren eines Status einer Aufgabe funktioniert.

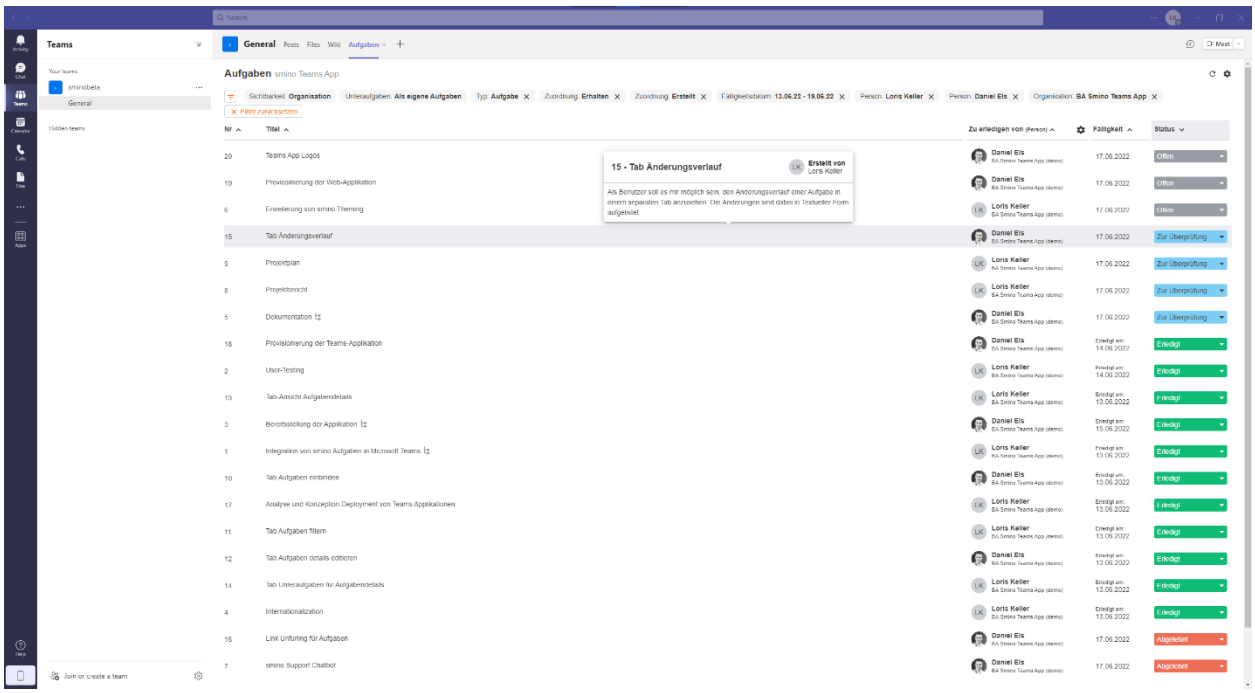


Abbildung 55: Aufgaben (eigene Darstellung)

Über das ↻ Refresh-Icon kann eine Aktualisierung der Aufgaben vom Server erzwungen werden. Mit dem ⚙ Settings-Icon kann ein Menu geöffnet werden, in welchem der Benutzer die Möglichkeiten hat, sich aus dem smino Kontext auszuloggen oder die Ansicht in der smino Web-Applikation zu öffnen.

Über das ⏷ Filter-Icon kann das Filter-Modal geöffnet werden. Die Filter können hier ergänzt oder auch entfernt werden. Diese Änderungen werden direkt angewandt, so dass beim Schliessen des Modal die Aufgaben bereits mit den neuen Kriterien gefiltert sind.

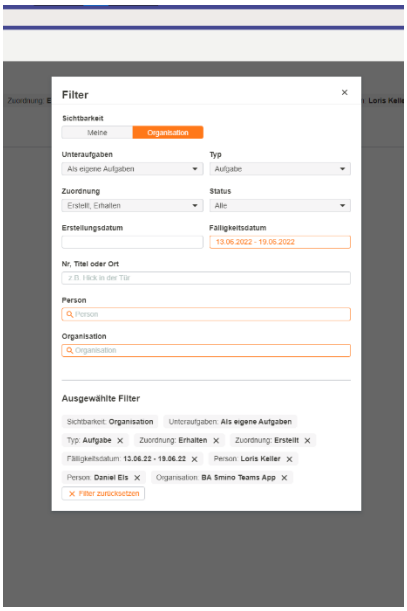


Abbildung 56: Aufgaben - Filter (eigene Darstellung)

Die Details einer Aufgabe werden in einem Side-Sheet dargestellt. Beim Öffnen einer Aufgabe werden diese und all seine zusätzlichen Informationen automatisch vom Server neu geladen.

Der Info-Tab stellt die allgemeinen Details, die Beschreibung sowie die vorhandenen Anhänge der Aufgaben dar. Auch hier kann der Status der Aufgabe geändert werden und zusätzlich kann die Sichtbarkeit editiert werden. Ist die Aufgabe eine Unteraufgabe einer anderen Aufgabe, wird dies in einem interaktiven Info-Alert dargestellt.

Falls eine Aufgabe Unteraufgaben besitzt, werden diese in dem entsprechenden Tab dargestellt. Hier wird nur eine grobe, interaktive Übersicht ohne zusätzliche Unter-Hierarchien dargestellt.

Im Änderungsverlauf-Tab können alle Änderungen in textueller Form nachvollzogen werden.

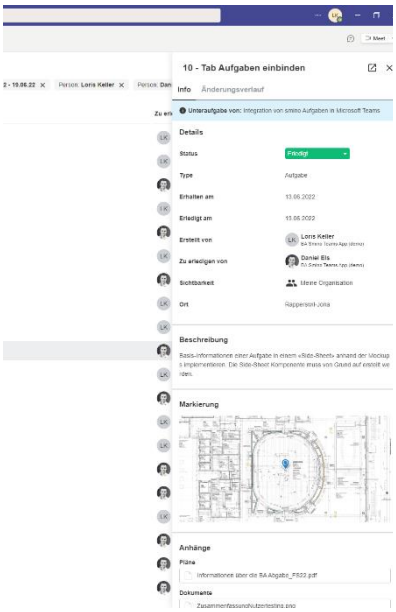


Abbildung 57: Aufgaben - Info (eigene Darstellung)

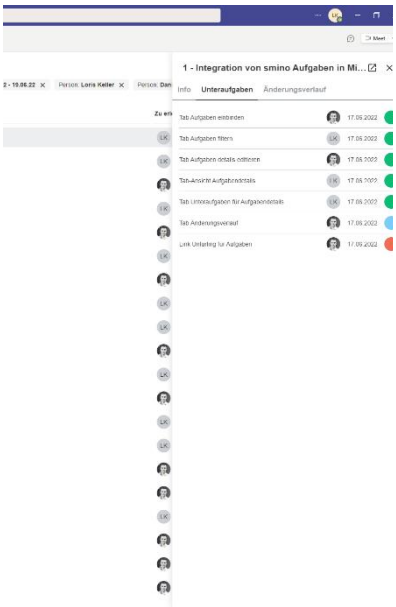


Abbildung 58: Aufgaben - Unteraufgaben (eigene Darstellung)

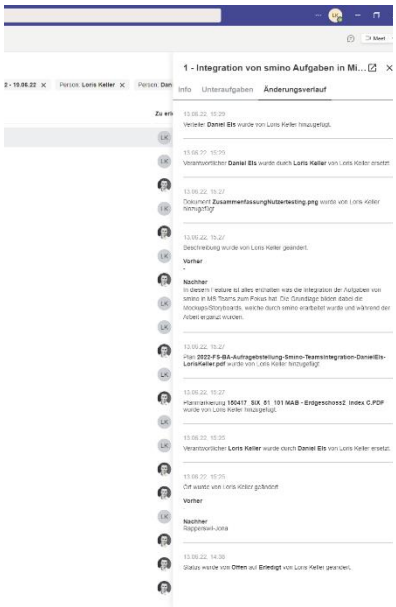


Abbildung 59: Aufgaben - Änderungsverlauf (eigene Darstellung)

Die smino Teams App kann auch in von smino definierten Sprachen (Deutsch, Englisch, Französisch und Italienisch) übersetzt werden. Die Sprache wird nach nachfolgender Priorisierung selektiert:

1. Die allfällig definierte Sprache im Profil der smino Web-Applikation.
2. Die im Microsoft Teams verwendete Sprache, falls von smino unterstützt.
3. Die von smino gesetzte Standartsprache (Deutsch).

Für Benutzer, die Teams im Dark-Mode verwenden ist auch ein entsprechender smino Dark-Mode implementiert.

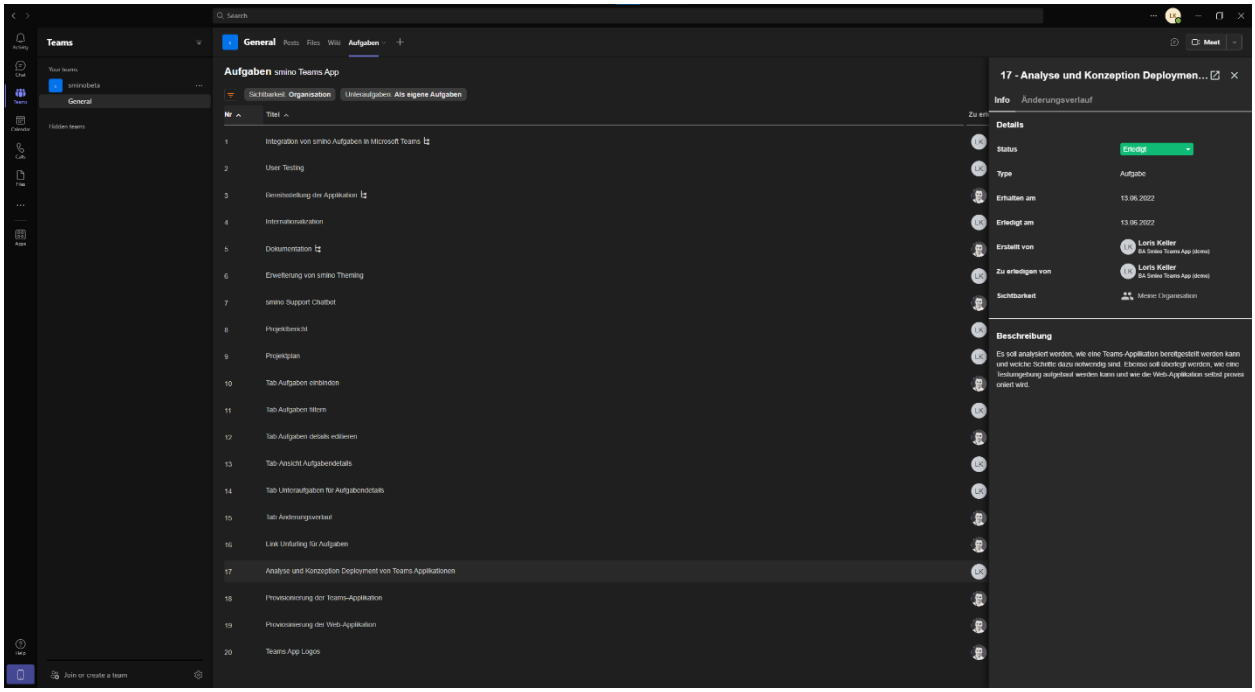


Abbildung 60: Aufgaben - Info - Dark-Theme (eigene Darstellung)

10. Zielerreichung

Architektur Prototyp zur Entwicklung von Microsoft Teams App (inklusive Authentifizierung)

Es wurden zwei Prototypen zwei Prototypen für Angular und React erarbeitet, welche bereits die Authentifizierung mit smino implementiert haben. Mit diesen konnten erste Erfahrungen in der Entwicklung mit Teams gemacht werden und das Konzept für die Integration validiert werden.

Testkonzept für Microsoft Teams App erstellen

Die wichtigsten Klassen und Methoden, welche eigenständig erstellt wurden, sind mit Unit- / Component-Tests abgedeckt. Dabei ist die Code-Coverage dieser getesteten Files mit mindestens 85% Abdeckung sehr gut. Zudem wurden die häufigsten Benutzer-Workflows mit e2e Tests abgedeckt.

Provisionierung der Applikation (Hosting/Store/O365 Plattform)

Die Applikation wurde auf Azure bereitgestellt. Die Bereitstellung der Applikation, sowie der Infrastruktur wurden automatisiert und mittels Azure Pipelines bereitgestellt. Ebenso wurde eine Architektur für den Aufbau der Teams Mandanten in Relation mit den verschiedenen Umgebungen erarbeitet.

Erarbeitung einer Dokumentation für die Entwicklung mit Microsoft Teams

In dieser Arbeit wurde die Entwicklung mit Teams und die Architektur der erstellten Anwendung näher beschrieben. Ebenso wurde auf die Bereitstellung und Publizierung der Applikation im Detail eingegangen. Für die Übergabe an smino wurden die wichtigsten Erkenntnisse und Ausschnitte dieser Arbeit in einem separaten Dokument bereitgestellt und mit dem Source-Code übergeben.

Integration von Funktionalitäten der smino Applikation in Teams als Tabs (MVP)

Die smino Aufgabenansicht konnte erfolgreich mit den definierten Funktionalitäten in Teams integriert werden und als installierbare Beta-Version bereitgestellt werden. Diese konnte auch durch ein User-Testing validiert werden.

Architektur für eine Production-Ready Applikation mit der Integration von smino Komponenten

Es wurde eine geeignete Applikationsarchitektur für eine langfristige und produktive Entwicklung der Applikation aufgebaut und dokumentiert. Dabei wurde versucht, bereits erste Schritte für eine bessere Aufteilung der einzelnen Komponenten, bzw. zur Paketierung und Auslagerung der Projekte vorzunehmen.

(Optional) Erarbeitung weiterer Integrationsmöglichkeiten (Bot, Meeting Extensions, Chat Extensions)

Weitere Integrationsmöglichkeiten wurden einerseits im Allgemeinen für Teams bestimmt, wobei weitere Ideen durch die Studierenden, sowie auch durch smino in den Screen-Designs festgehalten wurde.

11. Ausblick

Nach Beendigung der Arbeit wird smino die Applikation in ihre eigene Infrastruktur übernehmen und weitere Funktionalitäten oder Verbesserungen vornehmen, um die Applikation über den öffentlichen Teams Store vertreiben zu können.

Für die Aufgabenansicht selbst, würde als nächstes die Integration der Konversationen anstehen. Zudem könnte mittels einer Message Extension und Link Unfurling eine weitere Interaktionsmöglichkeit für den Benutzer zur Referenzierung einer spezifischen Aufgabe über den Chat geschaffen werden. Ebenso können weitere Bearbeitungsmöglichkeiten für die Aufgaben in Teams mittels Task Modules integriert werden.

Mit einem Bot und einer allgemeinen Chat-Page für die Applikation könnte initial eine Hilfestellung zur Bedienung der Applikation angeboten werden. Ebenso könnte man sich auch über diese Page in Teams mit seinem smino Konto An- oder Abmelden.

Weiter können die angewendeten Konzepte und die gelernten Erfahrungen verwendet werden, um die weiteren Bereiche eines smino Bauprojekts in Teams verfügbar zu machen.

12. Projektplan

12.1 Einführung

12.1.1 Zweck

Dieses Dokument gibt eine Übersicht über den Projektplan des Projekts «MS Teams smino App» geben. Einerseits soll beschrieben werden, was das Ziel der Bachelorarbeit ist, andererseits soll eine Übersicht über die Projektorganisation, Management Abläufe, Versionskontrolle und das Release Management gegeben werden.

12.1.2 Aufgabenstellung

Die Arbeit umfasst eine Recherche, welche die Integrationsmöglichkeiten von Microsoft Teams, auch anhand bestehender Integrationen, aufzeigt. Anhand dieser Recherche soll, in Kombination mit einer Analyse der User Prozesse von smino, ein Konzept erstellt werden, wie ein sinnvoller Teil von smino auf Microsoft Teams integriert werden kann.

Eine mögliche Integration kann verschiedene Funktionalitäten beinhalten wie z.B. eigene Tabs für smino Projekte, ein Chatbot, welcher Aufgaben erstellen kann oder eine Kalenderintegration, welche Ereignisse von smino abbildet. Auch muss berücksichtigt werden, wie ein Benutzer sich auf Microsoft Teams mit smino verbindet.

Ziel ist das erarbeitete Konzept als MVP umzusetzen und den Kunden von smino eine erste Teams Integration anbieten zu können.

12.2 Projekt Übersicht

12.2.1 Wer ist smino?

Smino ist ein motiviertes IT-Startup aus Rapperswil mit grossen Ambitionen. Mit http://smino.ch ist die Firma gerade dabei, die Baubranche fundamental zu revolutionieren. Sie entwickeln moderne, digitale Werkzeuge für Planungs- und Bauprojekte und vernetzen die Bauherrschaft, Behörden, Planer und Unternehmen zu einer effizienten Einheit.

12.2.2 Motivation

Immer mehr Menschen arbeiten von zuhause, was dazu geführt hat, dass Programme wie Microsoft Teams enorm in den Vordergrund gerückt sind, um remote kommunizieren zu können. Um einem Nutzer ein angenehmes Arbeiten mit anderen Tools zu ermöglichen, möchten wir gerne herausfinden, wie smino in Microsoft Teams integriert werden kann, um den Workflow mit smino für unsere Nutzer zu verbessern.

12.2.3 Ziel

Die Arbeit umfasst eine Recherche, welche die Integrationsmöglichkeiten von Microsoft Teams, auch anhand bestehender Integrationen, aufzeigt. Aufgrund dieser Recherche soll, in Kombination mit einer Analyse der User Prozesse von smino, ein Konzept erstellt werden, wie ein sinnvoller Teil von smino auf Microsoft Teams integriert werden kann.

Eine mögliche Integration kann verschiedene Funktionalitäten beinhalten wie z.B. eigene Tabs für smino Projekte, ein Chatbot, welcher Aufgaben erstellen kann oder eine Kalenderintegration, welche Ereignisse von smino abbildet. Auch muss berücksichtigt werden, wie ein Benutzer sich auf Microsoft Teams mit smino verbindet.

Ziel ist das erarbeitete Konzept als MVP umzusetzen und eine erste smino Teams Integration den Kunden von smino anbieten zu können.

12.2.4 Zielformulierung

1. Architektur Prototyp zur Entwicklung von Microsoft Teams App (inklusive Authentifizierung)
2. Testkonzept für Microsoft Teams App erstellen
3. Provisionierung der Applikation (Hosting/Store/O365 Plattform)
4. Erarbeitung einer Dokumentation für die Entwicklung mit Microsoft Teams
5. Integration von Funktionalitäten der smino Applikation in Teams als Tabs (MVP)
6. Architektur für eine Production-Ready Applikation mit der Integration von smino Komponenten
7. (Optional) Erarbeitung weiterer Integrationsmöglichkeiten (Bot, Meeting Extensions, Chat Extensions)

12.3 Projektorganisation

Die grobe Planung des Projekts ist an den Rational Unified Process (RUP) angelehnt, wobei Scrum für die Detailplanung verwendet wird. Die Sprints werden den entsprechenden RUP-Phasen zeitlich zugeordnet. Grundsätzlich schlagen wir eine stark agile Vorgehensweise vor, wobei pro Sprint ein MVP definiert und umgesetzt wird. Dabei sollen konzeptionelle Arbeiten jeweils auf einen Sprint-Start vorbereitet werden.

Im Sprint 0 geht es vor allem darum, das Projekt aufzusetzen, den initialen Projektplan zu erstellen und um die Einarbeitung in das Projekt. Die effektive Planung der Sprints beginnt im Sprint 1.

Die konkrete Planung wird Iterativ mit dem Projektmanagement Framework Scrum gemacht. Die Projektplanung wird ab Sprint 1 jeweils in 2-Wochen Sprints stattfinden. Dabei werden die zu erreichende Ziele pro Sprint festgehalten.

12.4 Roadmap

Die Roadmap stellt den aktuellen geplanten Ablauf der Sprints und Meilensteine dar und widerspiegelt immer den gegenwärtigen Fortschritt und kann während dem Projekt Änderungen unterliegen.

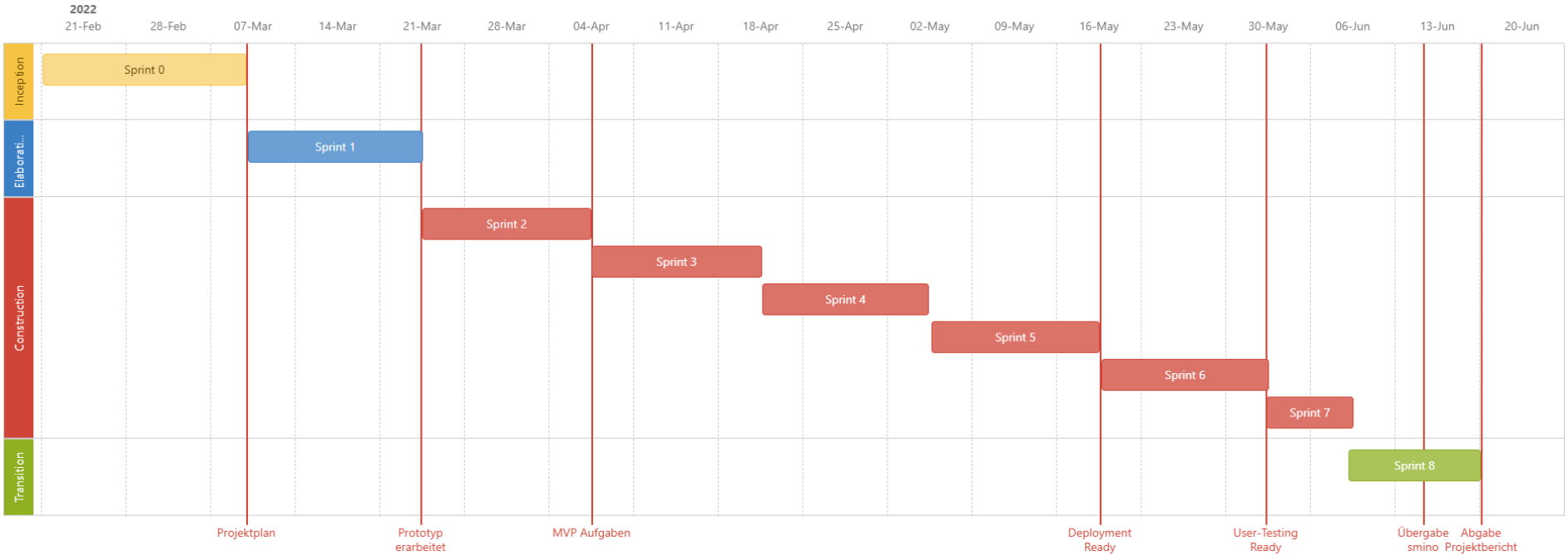


Abbildung 61: Roadmap (eigene Darstellung)

12.4.1 RUP-Phasen

12.4.1.1 Inception

Diese Phase umfasst vor allem den Projektstart, also die Projektidee und die Einigung auf ein gemeinsames Vorgehen. Im Weiteren nutzen die Entwickler diese Zeit, um eine erste Annäherung an die Entwicklung mit Microsoft Teams vorzunehmen.

12.4.1.2 Elaboration

In dieser Phase gehet es darum, die initialen Anforderungen an das Projekt und das Vorgehen zu definieren. Ebenso soll die technologische Grundlage geschaffen werden, womit dann effektiv entwickelt werden kann.

12.4.1.3 Construction

Construction ist die effektive Umsetzung des Projekts. Hier soll ein stark Agiles Vorgehen gelebt werden, wobei bei Sprint ein MVP definiert und umgesetzt werden soll. Dies soll es erlaube, fortlaufend neu zu priorisieren und bestehende Funktionalitäten auszubauen oder neue einzubinden. Während eines Sprints sollen auch gleichermassen Anforderungen und Vorbereitungen für den nächsten Sprint getätigt werden.

12.4.1.4 Transition

Zum Schluss soll das Produkt von einer "Entwicklung" zum Betrieb, bzw. Produktion übergehen. In diesem Fall beinhaltet dies vor allem die Übergabe und Integration des Produkts in smino. Dazu gehört auch die notwendige Dokumentation für Endbenutzer und Entwicklung. Ebenso wird hier der Projektbericht für die Bachelorarbeit fertiggestellt und abgegeben.

12.5 Meilensteine

12.5.1 Projektplan (07.03.2022)
Abnahme des Projektplans und Einigung auf das gemeinsame Vorgehen.

12.5.2 Prototyp erarbeitet (21.03.2022)
Ein erster Architekturprototyp steht zur Verfügung. Mit dessen Fertigstellung sind folgende Punkte geklärt:

- Technologie-Stack ist evaluiert.
- Authentifizierungs- / Autorisierungs-Möglichkeiten sind evaluiert
- Test and Debugging-Möglichkeiten sind evaluiert

Das Minimum Viable Product (MVP) ist definiert und kann in dem nächsten Sprint umgesetzt werden.

12.5.3 MVP Aufgaben (04.04.2022)
Das Minimum Viable Product (MVP) ist umgesetzt. Testing, Bugfixing und das Ergänzen weitere optionale Features kann in Erwägung gezogen werden.

Das MVP für die Aufgaben beinhaltet:

- Authentifikation und Autorisierung des Benutzers bei smino.
- Tabs
    - Konfiguration: Auswahl für ein Projekt
    - Übersicht aller Aufgaben für ein Projekt mit Filteroptionen
    - Detailansicht einer Aufgabe

12.5.4 Übergabe an smino (13.06.2022)
Übergabe des Sourcecode und der technischen Dokumentation an smino.

Abgabe des BA-Abstracts auf dem Online-Tool für die Publikation auf der OST-Webseite.

12.5.5 Abgabe Projektbericht (17.06.2022)
Abgabe der publizierbaren Version der Arbeit für eprints.ost.ch (Bericht als PDF und Plain-Text Abstract), Eigenständigkeitserklärung, Urheber- und Nutzungsrechte, Einverständniserklärung, Poster A0, sowie den Archiv-Dateien mit den von Betreuern und Experten verlangten Unterlagen.

12.6 Sprints

RUP	Sprint	Periode	Ziel	Effort
Inception	Sprint 0	21.02.2022 - 07.03.2022	Projektplan, «ready for first sprint planning»	-
Elaboration	Sprint 1	07.03.2022 - 21.03.2022	Prototypen erstellt, def. Technologie Stack, Authentifizierung validiert	-
Construction	Sprint 2	21.03.2022 - 04.04.2022	smino Teams App mit Authentifizierung, Tab-Konfiguration und den smino Aufgaben	20
	Sprint 3	04.04.2022 - 18.04.2022	Umstellung der smino API, Umstellung auf smino Komponenten, Authentifizierung von Kollaborators	19
	Sprint 4	18.04.2022 - 02.05.2022	Aufgaben filtern, inline editieren und anzeige der Aufgabendetails	22
	Sprint 5	02.05.2022 - 16.05.2022	Theming Konzept, Dark-Theme implementation und Deployment	28
	Sprint 6	16.05.2022 - 30.05.2022	Logout, Provisionierung der Teams-Applikation, Tabs für Unteraufgaben	36
	Sprint 7	30.05.2022 - 06.06.2022	UX Improvements, Bug fixes und Testing	35
Transition	Sprint 8	06.06.2022 - 17.07.2022	Dokumentation, Source-Code für Übergabe bereitstellen, Projektbericht	33
Total				**193**

Tabelle 10: Sprints (eigene Darstellung)

12.7 Anpassungen Roadmap und Meilensteine

06.04.2022: Architekturwechsel und genauere Bestimmung des Resultats

Nach dem Review der Prototypen stellt sich heraus, dass die Entwicklung von Tabs in Teams selbst keine ausserordentliche Herausforderung darstellt. Eine wichtige Erkenntnis ist, dass ein Teams-Tab im Prinzip nur eine URL, bzw. ein `iframe` ist und nicht als native Applikation angesehen werden kann. Daraus folgt auch, dass grundsätzlich jede Web-Technologie zur Entwicklung eines Tabs in Teams eingesetzt werden kann.

Aufgrund dieser Erkenntnisse würde es nur wenig Sinn ergeben eine prototypische Implementation, als Ziel der Bachelorarbeit anzustreben. So wurde ein Prototyp mit einer vereinfachten Aufgabenansicht bereits in Sprint 2 bereits realisiert.

Neu ist das Ziel der Arbeit eine produktive, verwendbare Umsetzung der smino Applikation in Teams, wobei ein User Testing gegen Ende der Arbeit eingeplant werden kann. Als minimale Anforderung sind die definierten Funktionalitäten für die Aufgaben umzusetzen. Dadurch ist auch die Bindung zu smino sehr klar definiert, da grundsätzlich die Komponenten, Styles und Architektur der smino Web-Applikation verwendet werden sollen. Diese Entscheidung hat folgende Konsequenzen:

- Architektur der Anwendung soll an smino angeglichen werden. (Komponenten, Styles, NGXS Store Management)
- smino stellt den Studierenden (Loris Keller und Daniel Els) den Source-Code der smino Web-Applikation zur Verfügung. Um dies zu ermöglichen, ist es nötig, dass die Studierenden ein Non Disclosure Agreement (NDA) unterzeichnen.
- Der Source-Code der Arbeit darf nicht mehr veröffentlicht werden, da smino interne Komponenten verwendet werden und die Anwendung 1:1 durch smino übernommen wird.
- Das MVP der Aufgaben muss nochmals mit den neuen Anforderungen überarbeitet werden.
- Erarbeitung eines Konzepts für das Theming der smino Komponenten für Dark- und High-Contrast-Themes.
- Genauere Definition der Architektur der Teams App und inwiefern diese von der smino Architektur abweichen darf.
- Genauere Definition / Dokumentation der Funktionalitäten in Teams
- Warum / wofür wird die Ansicht in Teams verwendet? Welche Vorteile bringt die Ansicht gegenüber der Web-Applikation?
- Welche Schritte und Komponenten sind zur Erfüllung des Szenarios notwendig?
- Welche Komponenten sind bereits existent, welche müssten definiert / erarbeitet werden?

12.8 Stakeholder

Die folgende Tabelle stellt die Stakeholder des smino Teams App und ihre jeweilige Intention dar.

- **Product Owner**: Ist äquivalent zum Auftraggeber, wobei das Verwalten der Backlog-Items vom Projektteam übernommen wird.
- **Scrum Master**: Ist nicht vorhanden.
- **Developers**: Ist äquivalent zum Projektteam.

Funktion	Person	Erwartungshaltung
Projektteam	Daniel Els	Entwickelt die Applikation, so dass diese allen Anforderungen der restlichen Stakeholder gerecht wird
Projektteam	Loris Keller	Entwickelt die Applikation, so dass diese allen Anforderungen der restlichen Stakeholder gerecht wird
Betreuungsperson BA	Prof. Dr. Markus Stolze	Überwacht den Arbeitsfortschritt und interveniert bei grossen Abweichungen bezüglich des Ziels dieser Arbeit
Mitarbeiter smino AG / Auftraggeber	Raphael Eissler	Unterstützt das Projektteam mit Fachwissen im technischen Bereich und hat die Entscheidungsgewalt bei der Implementation
Mitarbeiter smino AG / Stv. Auftraggeber	Mischa Trecco	Unterstützt das Projektteam mit Fachwissen im technischen Bereich und hat die stellvertretende Entscheidungsgewalt bei der Implementation
Mitarbeiterin smino AG	Lina Stadelmaier	Unterstützt des Projektteam mit Fachwissen im UX Bereich und hat die Entscheidungsgewalt bei den UX / UI Implementation

Tabelle 11: Stakeholder (eigene Darstellung)

12.9 Backlog Verwaltung

Das Backlog wird mithilfe von Azure DevOps erstellt und verwaltet. Dabei werden folgende Kategorien unterschieden:

Die Arbeitspakete sind im Azure DevOps mit den folgenden Kategorien erfasst:

- **Epics:** Ohne Schätzung. Sollten aufgrund der relativ kurzen Dauer des Projekts keine Verwendung finden.
- **Features:** Grössere Funktionalitäten, ohne Schätzung erfasst.
- **Product Backlog Item**: Konkrete Funktion, Schätzung von *Effort* in Form von Story-Points anhand Fibonacci Skala.
- ***Tasks***: *Einzelner Arbeitsschritt, Schätzung von Remaining Work in Stunden.*

Da die Arbeitspakete jeweils sehr klein gehalten sind, kann bei einigen auf die Deklarierung von *Tasks* verzichtet werden. Für den Fall, dass dennoch ein *Task* erfasst wird, ist das Ergänzen der Schätzung auf Stundenbasis optional und jeweils dem Entwickler überlassen.

Die folgenden Status werden für die *Product Backlog Items* verwendet:

- **New:** Das Item wurde neu erstellt.
- **Approved:** Das Item ist definiert und potenziell bereit zur Umsetzung
- **Committed:** Arbeit am Item hat begonnen, *«Commitment made by the team»*
- **Done:** Das Item ist umgesetzt.

Für jeden Sprint wird ein Scrumboard mit 3 Swimlanes für die Tasks geführt:

- **To Do:** Arbeiten, die noch zu erledigen sind.
- **In Progress:** Arbeiten, die gerade erledigt werden.
- **Done:** Erledigte Arbeiten.

12.10 Management Abläufe

12.10.1 Kostenvoranschlag

Für die Bachelorarbeit werden pro Student 12 ECTS vergeben, wofür ein Aufwand von 360 Stunden erwartet wird. Insgesamt entspricht dies einem Aufwand von 720 Stunden.

Das Projekt beginnt mit der Kick-off-Sitzung am 21. Februar und endet am 17.06.22 (24.06.22?) mit der Abgabe aller Dokumente bis um 17 Uhr.

12.10.2 Meeting Organisation / Timeboxing

Auf Daily Stand-Up Meetings wurde verzichtet, da die Mitglieder nicht täglich an dem Projekt arbeiteten.

Das Team hat sich jeweils jeden Montag von 16:00 bis 17:00 Uhr getroffen. Im Falle eines Feiertages oder aufgrund von zu vielen wichtigen Absenzen wurden die Meetings frühzeitig auf ein anderes Datum verschoben.

Diese Meetings wurden jeweils auf 1 Stunde limitiert, wobei immer darauf geachtet wurde, dass das geplante Zeitlimit nicht überschritten wurde.

Da mit einem 2-Wochen Sprint-Rhythmus gearbeitet wurde, war folgerichtig jedes zweite Meeting ein Sprintmeeting mit den folgenden Traktanden:

- **Review:** *(~ 10')* Kurze Präsentation über das Erreichte im letzten Sprint.
- **Retrospektive:** *(~ 5')* Einholen gegenseitiger Feedbacks für Verbesserungen.
- **Offene Fragen:** *(~ 15')* Klären von offenen Fragen.
- **Planning:** *(~ 30')* Planung und Priorisierung des nächsten Sprints.

Während eines Sprints diente diese Besprechung primär zur Klärung von offenen Fragen.

Aufgrund von den Erkenntnissen in der jungen Microsoft Teams Entwicklung und der anschliessenden Komplexität diverser Features waren mehrere zusätzliche Meetings ausserhalb der regulären Meetings sowie Workshops notwendig. Diese wurden mit einer eigenen Traktandenliste und einer angepassten Zeitlimitierung abgehalten.

Für alle Meetings wurde ein Protokoll geführt. Diese wurden jeweils auf *Confluence* erfasst und für alle Stakeholders zur Verfügung gestellt.

12.11 Versionskontrolle und Release Management

12.11.1 Versionskontrolle

Für die Verwaltung des Source-Codes werden Git Repositories im Azure DevOps Team Projekt erstellt. Dies erlaubt eine einfache Zugangssteuerung, wobei eine Verfügbarkeit von 99,9 % garantiert wird.

12.11.2 Release Management and CI/CD

Für das Builden und Deployment können Azure Pipelines verwendet werden. Diese unterstützen eine Definition über eine grafische Oberfläche, sowie eine Definition über YAML. Ebenso bieten sie Möglichkeiten für zeitlich gesteuerte Deployments, wie auch die Definition eines Genehmigungsprozesses.

12.11.3 Berechtigungen und Privacy

Zugang zu Azure DevOps und entsprechend auch dem Source-Code wird ausschliesslich dem Projektteam erteilt und ist somit der Öffentlichkeit nicht zugänglich. Dies ist während dem ganzen Projektzeitraum garantiert. Der Source-Code darf frühestens 6 Monate nach dem Projektende, also nicht vor dem 18.01.2023, publiziert werden.

12.11.4 Qualitätsziele

Massnahme	Zeitraum	Ziel
Retrospektive	Fortlaufend, am Ende eines Sprints	Verbesserung der zukünftigen Sprints
Dokumentation	Fortlaufend	Gewährleistung der Gültigkeit und Aktualität
CI / CD	Bei jedem Pull-Request	Fehlerverhinderung, Verbesserung der Code-Qualität
Code Reviews	Bei jedem Pull-Request	Fehlerverhinderung, Verbesserung der Code Qualität
Automatisierte Tests	Fortlaufend	Fehlerverhinderung, Verbesserung der Code Qualität
Linter	Fortlaufend	Einhaltung des Code-Styles

Tabelle 12: Qualitätsziele (eigene Darstellung)

12.11.5 Dokumentation

Die aktuelle Dokumentation wurde in *Confluence* hinterlegt. Dokumente wie Projektbericht, User-Dokumentation und Developer-Dokumentation sind schlussendlich als PDFs zur Verfügung gestellt.

12.11.6 Entwicklung

Die Applikation ist in einem Azure DevOps Repo mit git verwaltet. Es wird auf die Trunk Based Development Strategie gesetzt, wobei der Main-Branch als Trunk verwendet wird und davon die diversen Feature-Branches abgeleitet werden.

Für ein Pull-Request zurück in den Main-Branch ist jeweils ein required Reviewer aus dem smino Team, sowie auch eines Entwicklers aus dem Projektteam notwendig.

Bei der Entwicklung des Prototyps wurde auf den required Reviewer von Seiten der smino verzichtet und eher auf eine bilaterale Kommunikation gesetzt, wobei ein Review des Prototyps als Ganzes im Fokus stand. Dadurch konnten allfällige Verzögerung während der Prototypphase eliminiert werden und das smino Team wurde nicht mit einer Vielzahl an Änderungen strapaziert.

Für die entsprechenden Features und Bugs sind die entsprechenden Work Items im Azure erfasst. Diese wurden jeweils auch mit den entsprechenden Pull-Requests verknüpft.

12.11.7 Arbeitspapiere

Für die Erarbeitung von gewissen Themenfeldern oder Anforderungen wurden sogenannte Arbeitspapiere erstellt. Während die Use-Cases spezifische Fälle aus Sicht eines Actors beschreiben, sollen die Arbeitspapiere allgemeingültige Definitionen enthalten, welche auch von mehreren Use-Cases geteilt werden können. Zudem werden auch verschiedene Lösungen dargestellt, oder Befunde aus der Recherche festgehalten.

Zu den erarbeiteten Arbeitspapieren gehören die folgenden Themengebiete:

- Anwendungsbeschreibung
- Authentifizierung für Tabs
- Bereitstellung der App
- Design Decisions
- Guidelines MS Teams
- Integrationskonzept
- Plan your app
- Styling Framework
- Testing
- Theming

12.11.8 Architektur und Frameworks

Die Architektur der BA smino Teams Applikation soll möglichst nah an die Architektur der smino Applikation angelehnt werden, um die Einbindung der smino-Komponenten möglichst einfach zu gewährleisten.

Wenn möglich werden auch auf die gleichen Frameworks eingesetzt. So wird zum Beispiel für das State-Management die NGXS Library verwendet

12.11.9 Testing

Die Grundlage für das Testing bilden Component / Unit Tests. Weitern Möglichkeiten für das Testing wurden ihm Rahmen dieser Arbeit abgeklärt und im Bericht dokumentiert.

13. Literaturverzeichnis

Color schemes. (21. März 2022). Von Fluent UI: https://fluentsite.z22.web.core.windows.net/0.51.7/color-schemes abgerufen

Fluent UI Team. (20. März 2022). *Theming*. Von GitHub: https://github.com/microsoft/fluentui/wiki/Theming abgerufen

Google. (21. März 2022). *Testing*. Von Angular: https://angular.io/guide/testing abgerufen

Implementing your theme. (9. April 2022). Von Material Design: https://material.io/design/material-theming/implementing-your-theme.html abgerufen

Jaaidi, Y. (21. März 2022). *Testing Angular Components Using Cypress*. Von Marmicode: https://marmicode.io/blog/testing-angular-components-using-cypress abgerufen

Microsoft. (20. Mai 2022). *Bots in Microsoft Teams*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/bots/what-are-bots abgerufen

Microsoft. (25. Mai 2022). *Build Tabs for Microsoft Teams*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/tabs/what-are-tabs abgerufen

Microsoft. (10. Juni 2022). *Building tabs and other hosted experiences with the Microsoft Teams JavaScript client SDK*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/tabs/how-to/using-teams-client-sdk abgerufen

Microsoft. (03. März 2022). *Design your tab for Microsoft Teams*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/tabs/design/tabs abgerufen

Microsoft. (14. Juni 2022). *Enable authentication using third-party OAuth provider*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/tabs/how-to/authentication/auth-flow-tab abgerufen

Microsoft. (9. Juni 2022). *Manage Teams apps in the Microsoft Teams admin center*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/MicrosoftTeams/manage-apps abgerufen

Microsoft. (20. Mai 2022). *Message extensions*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/messaging-extensions/what-are-messaging-extensions abgerufen

Microsoft. (20. März 2022). *Microsoft Teams store validation guidelines*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-

us/microsoftteams/platform/concepts/deploy-and-publish/appsource/prepare/teams-store-validation-guidelines#tabs abgerufen

Microsoft. (21. März 2022). *Plan your app with Teams features*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/app-fundamentals-overview abgerufen

Microsoft. (21. März 2022). *Prepare your Microsoft 365 tenant*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/build-and-test/prepare-your-o365-tenant abgerufen

Microsoft. (9. April 2022). *Prepare your Microsoft Teams store submission*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/appsource/prepare/submission-checklist#long-description abgerufen

Microsoft. (27. Mai 2022). *Publish your app to the Microsoft Teams store*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/appsource/publish abgerufen

Microsoft. (9. April 2022). *Reference: Manifest schema for Microsoft Teams*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/resources/schema/manifest-schema abgerufen

Microsoft. (27. Mai 2022). *Task modules*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/task-modules-and-cards/what-are-task-modules abgerufen

Microsoft. (27. Mai 2022). *Teams Toolkit Overview*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/toolkit/teams-toolkit-fundamentals abgerufen

Microsoft. (21. April 2022). *Test your app*. Von Teams | Microsoft Docs: https://docs.microsoft.com/bs-latn-ba/microsoftteams/platform/concepts/build-and-test/test-app-overview abgerufen

Microsoft. (9. Juni 2022). *Upload your app in Microsoft Teams*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/apps-upload abgerufen

Microsoft. (9. April 2022). *Use Markdown formatting in Teams*. Von support.microsoft.com: https://support.microsoft.com/en-us/office/use-markdown-formatting-in-teams-4d10bd65-55e2-4b2d-a1f3-2bebdcd2c772?ui=en-us&rs=en-us&ad=us#ID0EBBD=Desktop abgerufen

Microsoft. (17. Mai 2022). *Webhooks and connectors*. Von Teams | Microsoft Docs: https://docs.microsoft.com/en-us/microsoftteams/platform/webhooks-and-connectors abgerufen

Parakramasinghe, S. (21. März 2022). *Angular 9 Testing: Why We Chose Jasmine Over Jest and Mocha*. Von Bits and Pieces: https://blog.bitsrc.io/angular-9-testing-a-comparison-between-jasmine-jest-and-mocha-acc57bcab836 abgerufen

Ravindranath, H. (21. März 2022). *Jest vs Mocha vs Jasmine: Comparing The Top 3 JavaScript Testing Frameworks*. Von LambdaTest: https://www.lambdatest.com/blog/jest-vs-mocha-vs-jasmine/ abgerufen

smino. (2022). smino. Rapperswil-Jona, Schweiz.

Struyf, E. (21. März 2022). *End-to-End testing your Microsoft Teams solutions with Cypress*. Von Elio Struyf: https://www.eliostruyf.com/e2e-testing-microsoft-teams-solutions-cypress/ abgerufen

14. Abbildungsverzeichnis

15. Tabellenverzeichnis

16. Anhang

Anhangsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Danksagung

Die Autoren möchten sich am Schluss herzlich bei dem Projektteam für die angenehme Zusammenarbeit und die erfolgreiche Umsetzung dieser Arbeit bedanken. Ebenso möchten wir allen Personen Danken, die uns bei der Arbeit unterstützt haben und zum Erfolg dieser beigetragen haben.

Raphael Eissler Wir möchten Raphael für seine aktive Betreuung und Unterstützung während der Arbeit danken. Raphael hat immer Zeit gefunden, um uns Fragen zu beantworten, technische Probleme zu diskutieren oder auch Pull-Requests zu überprüfen.

Lina Stadelmeier Wir möchten Lina für die Erarbeitung der Screen-Designs und für die Konzepte zur Integration von smino in Teams danken. Sie hat uns während der Arbeit aktiv vor allem in Belangen der User-Experience und des UI-Designs unterstützt und uns die entsprechenden Ressourcen zur Verfügung gestellt.

Mischa Trecco Wir danken Mischa für die Betreuung während der Abwesenheit von Raphael und seinen wertvollen Inputs. Ebenso für die Überprüfung der vielen Pull-Requests die in dieser Zeit entstanden sind.

Markus Stolze Wir danken Markus Stolze für die Betreuung dieser Bachelorarbeit und die aktive Unterstützung der Autoren. Wir haben seine Anmerkungen, Verbesserungsvorschläge und sein allgemeines Feedback stehts sehr geschätzt.

Jonathan Winter Auch vielen Dank an Johnny, der uns ebenfalls bei der Überprüfung der Pull-Requests in Abwesenheit von Raphael aktiv unterstützt hat.

Serge Ljubenovic, Beartice Kurth Wir möchten auch besonders Serge Ljubenovic und Beatrice Kurth danken, die unsere Arbeit gegengelesen und auf die Rechtschreibung geprüft haben.

2. Team Report

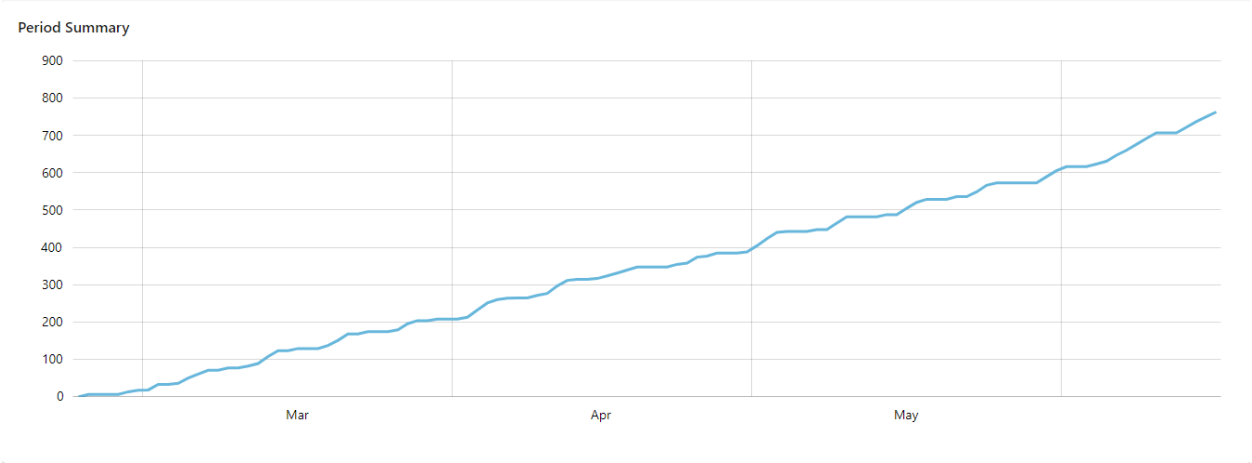
Die Microsoft Teams-Entwicklung ist noch in seinen Anfängen und war somit zu Beginn der Arbeit auch für uns Neuland. Einerseits war das erste Zurechtfinden in der Teams-Entwicklung mit viel Aufwand verbunden, andererseits gab es in der Zeitspanne dieser Arbeit immer wieder Minor-Changes, sowie einen Major-Change, der Microsoft Teams Entwicklung und Dokumentation, was jeweils von uns aufgearbeitet werden musste. Abgesehen davon war unsere grösste Erkenntnis, dass die Entwicklung von Teams-Tabs nicht viel anders ist als die Web-Entwicklung.

Mit dem agilen Vorgehen waren wir sehr zufrieden. So konnten wir nach dem Erstellen der Prototypen ohne allzu grossen Aufwand einen doch eher starken Richtungswechsel vornehmen. Die Sprints hatten jeweils eine gute Grösse, so dass wir bis am Ende dieser Arbeit immer in ungefähr gleich grossen Arbeitsblöcken arbeiten konnten.

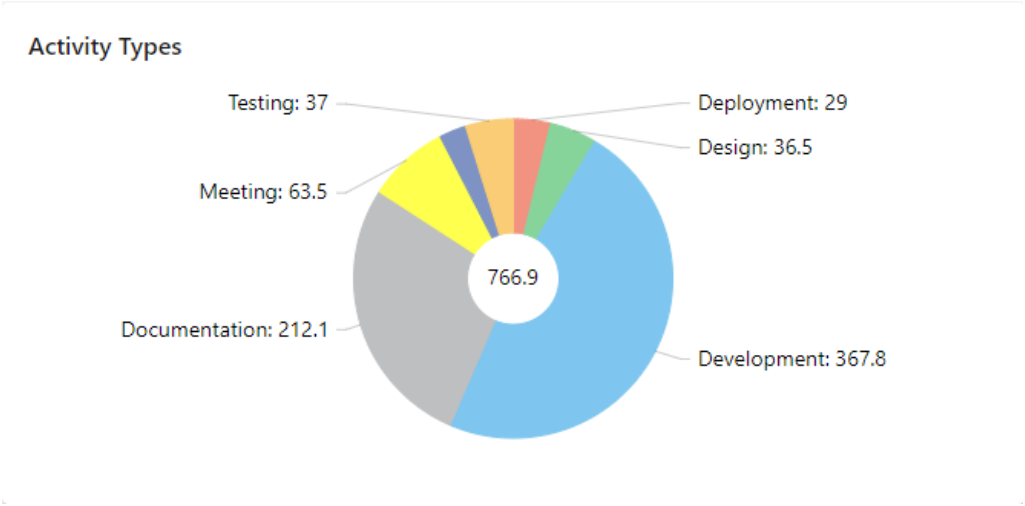
Die Zusammenarbeit im Projektteam war für uns beide sehr angenehm. An diesem Punkt möchten wir auch ein grosses Dankeschön an Raphael, Lina, Mischa und Johnny aussprechen, die uns während der ganzen Arbeit, in den regelmässigen sowie ausserordentlich Workshops, unterstützt haben. Ihr Feedback war für uns immer hilfreich und sehr wertvoll.

Mit unserem Resultat dieser Arbeit sind wir äusserst zufrieden. Der minimale Anspruch war ein Prototyp, der unter anderem die smino Aufgaben in einem Teams-Tabs darstellt. Das übergebene Produkt ist eine produktive Beta-Version der smino Teams App, welche bereits ein User-Testing durchlaufen konnte. In diesem sind keine technischen Fehler aufgetreten und das allgemeine User-, sowie smino-Feedback war sehr positiv.

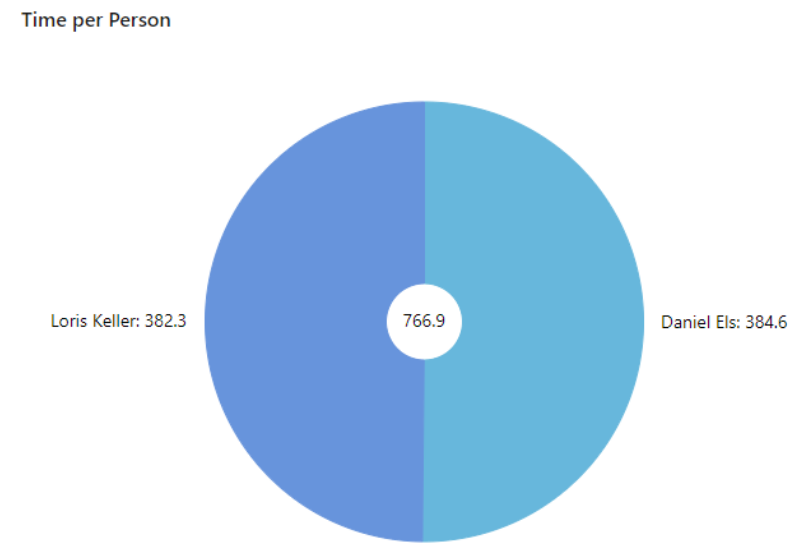
3. Time-Tracking



Anhang Abbildung 1: Period Summary Timetracker (eigene Darstellung)



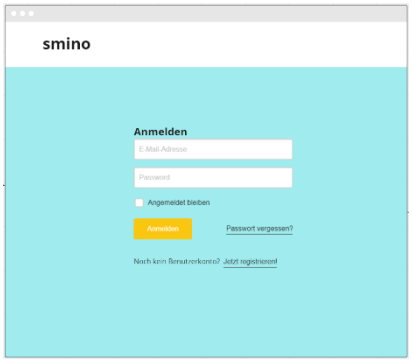
Anhang Abbildung 2: Activity Types Timetracker (eigene Darstellung)



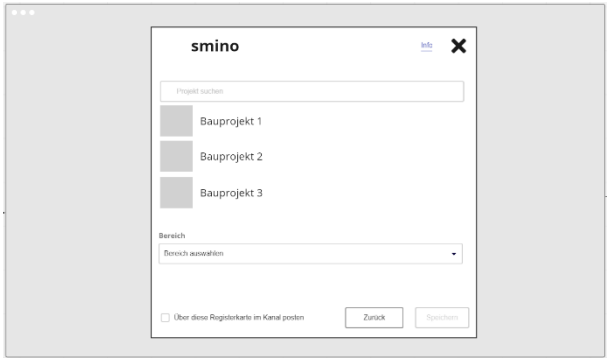
Anhang Abbildung 3: Time per Person (eigene Darstellung)

4. Miro Skizzen

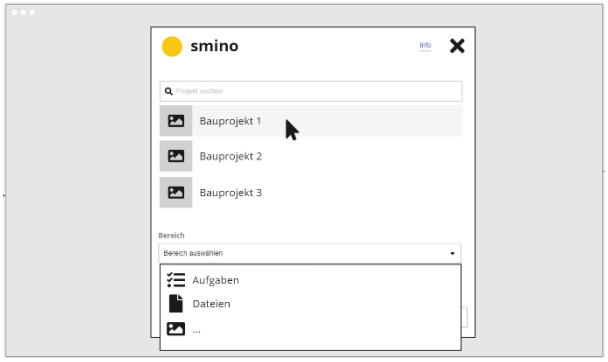
Anhang Abbildung 4: Schritt 1 - Tab-Konfiguration Login (eigene Darstellung)



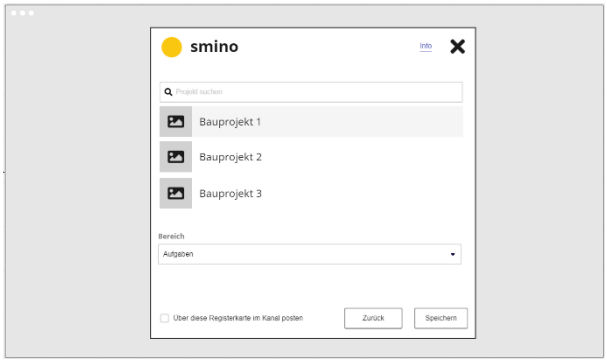
Anhang Abbildung 5: Schritt 2 - smino Login (Authorize Endpunkt) (eigene Darstellung)



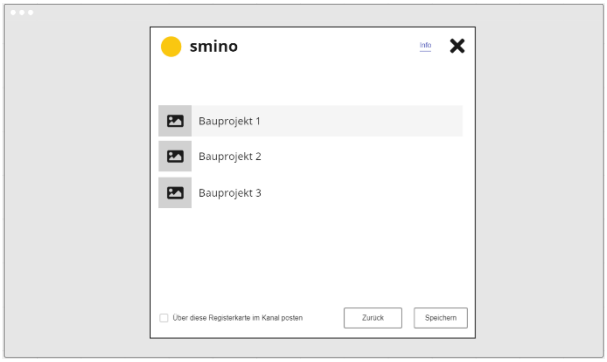
Anhang Abbildung 6: Schritt 3 - Tab-Konfiguration (eigene Darstellung)



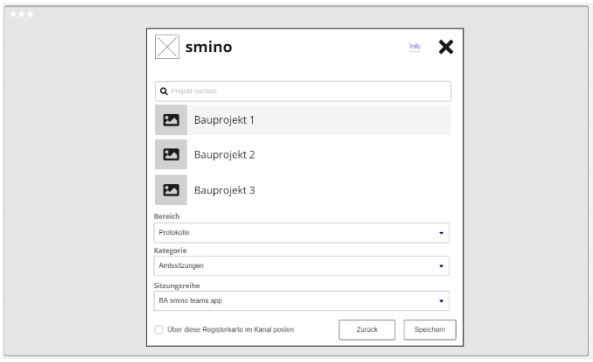
Anhang Abbildung 7: Schritt 4 - Tab-Konfiguration (Optionen) (eigene Darstellung)



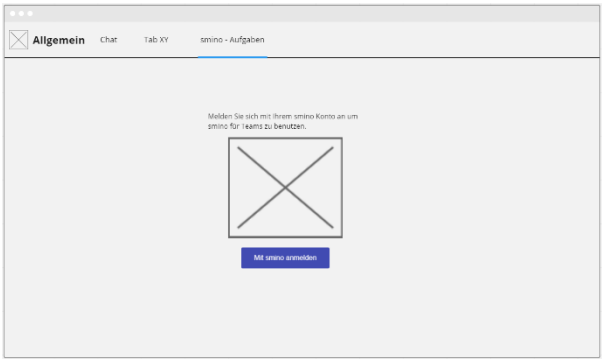
Anhang Abbildung 8: Schritt 5 - Tab-Konfiguration (Aufgaben) (eigene Darstellung)



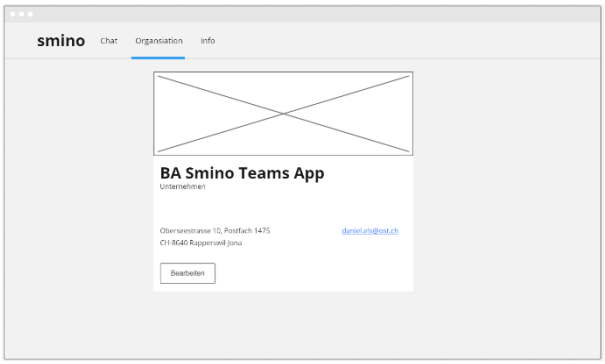
Anhang Abbildung 9: Tab-Konfiguration der Aufgaben ohne Optionen (eigene Darstellung)



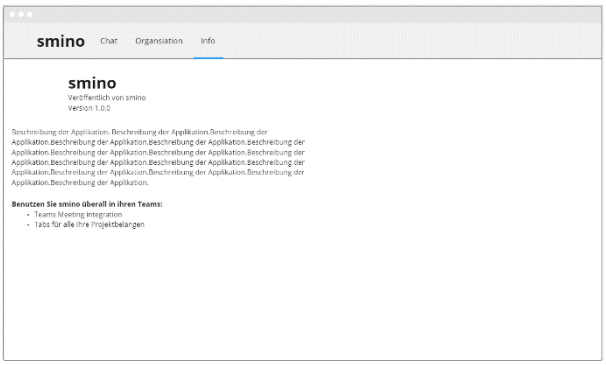
Anhang Abbildung 10: Tab-Konfiguration mit Kategorien (eigene Darstellung)



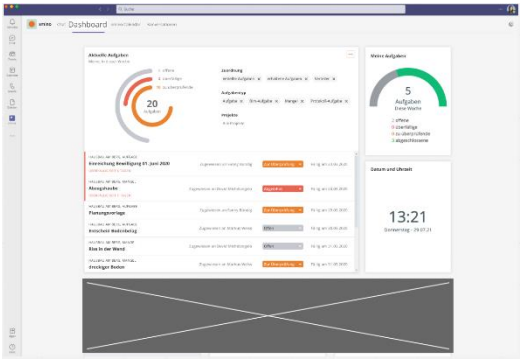
Anhang Abbildung 11: Tab Login (eigene Darstellung)



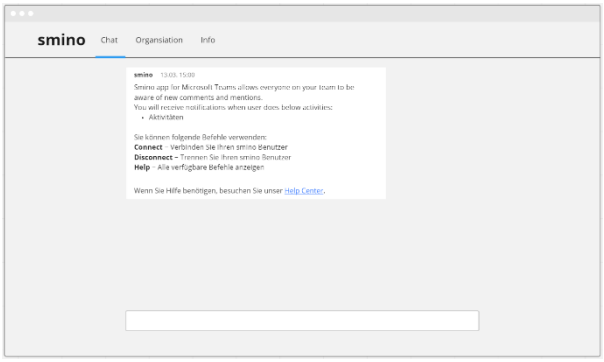
Anhang Abbildung 12: Organisationsbeschreibung (eigene Darstellung)



Anhang Abbildung 13: Teams App Info (eigene Darstellung)

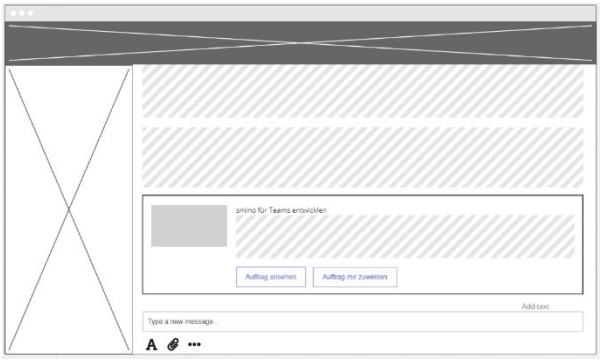


Anhang Abbildung 14: smino Teams App Dashboard (eigene Darstellung)

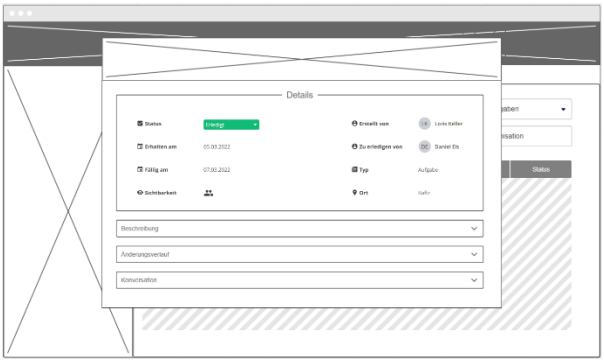


Anhang Abbildung 15: Chat-Bot (eigene Darstellung)

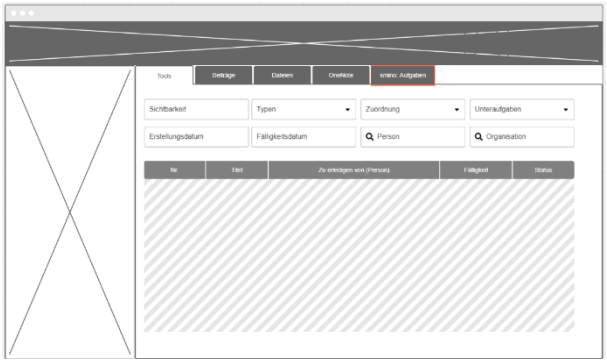
Anhang Abbildung 16: Chat-Extension (eigene Darstellung)



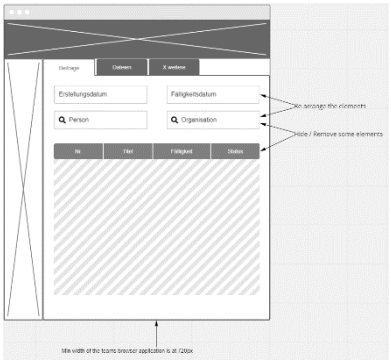
Anhang Abbildung 17: Link Unfurling (eigene Darstellung)



Anhang Abbildung 18: Aufgabe hinzufügen (eigene Darstellung)

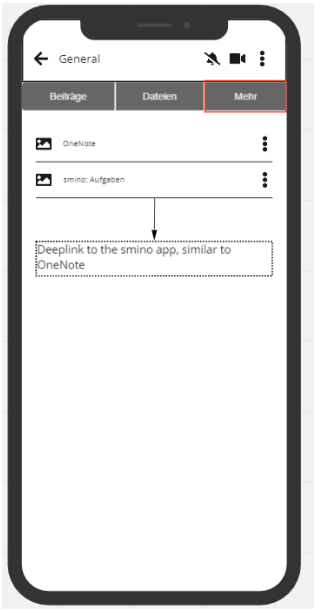


Anhang Abbildung 19: Tab Desktop (eigene Darstellung)



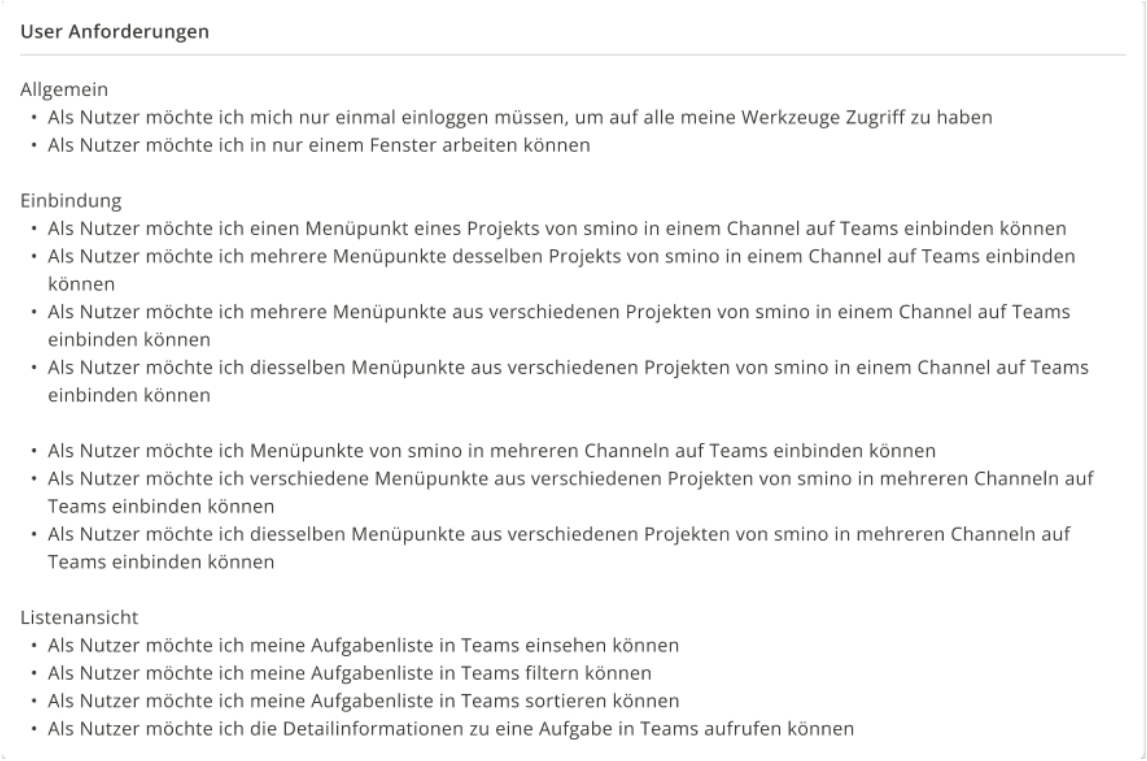
Anhang Abbildung 20: Tab Desktop small (eigene Darstellung)

Anhang Abbildung 21: Tab Mobile (eigene Darstellung)



Anhang Abbildung 22: Tab Mobile Deeplink (eigene Darstellung)

5. Figma Storyboards



User Anforderungen

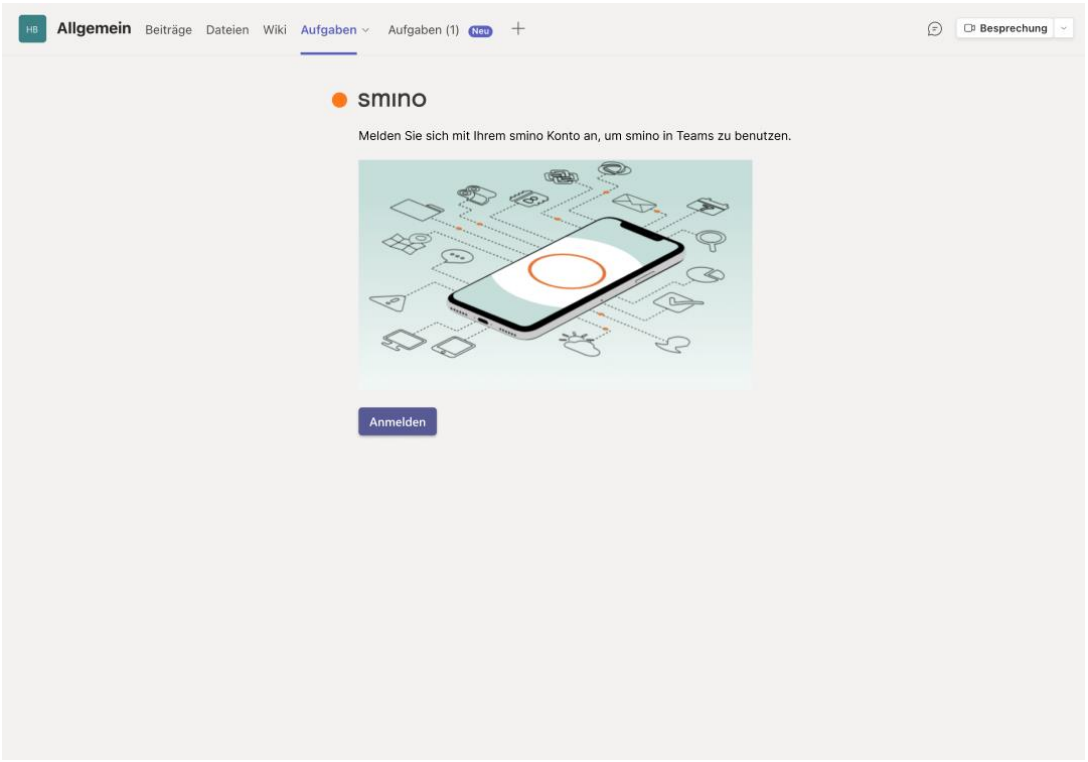
Allgemein
- Als Nutzer möchte ich mich nur einmal einloggen müssen, um auf alle meine Werkzeuge Zugriff zu haben
- Als Nutzer möchte ich in nur einem Fenster arbeiten können

Einbindung
- Als Nutzer möchte ich einen Menüpunkt eines Projekts von smino in einem Channel auf Teams einbinden können
- Als Nutzer möchte ich mehrere Menüpunkte desselben Projekts von smino in einem Channel auf Teams einbinden können
- Als Nutzer möchte ich mehrere Menüpunkte aus verschiedenen Projekten von smino in einem Channel auf Teams einbinden können
- Als Nutzer möchte ich diesselben Menüpunkte aus verschiedenen Projekten von smino in einem Channel auf Teams einbinden können

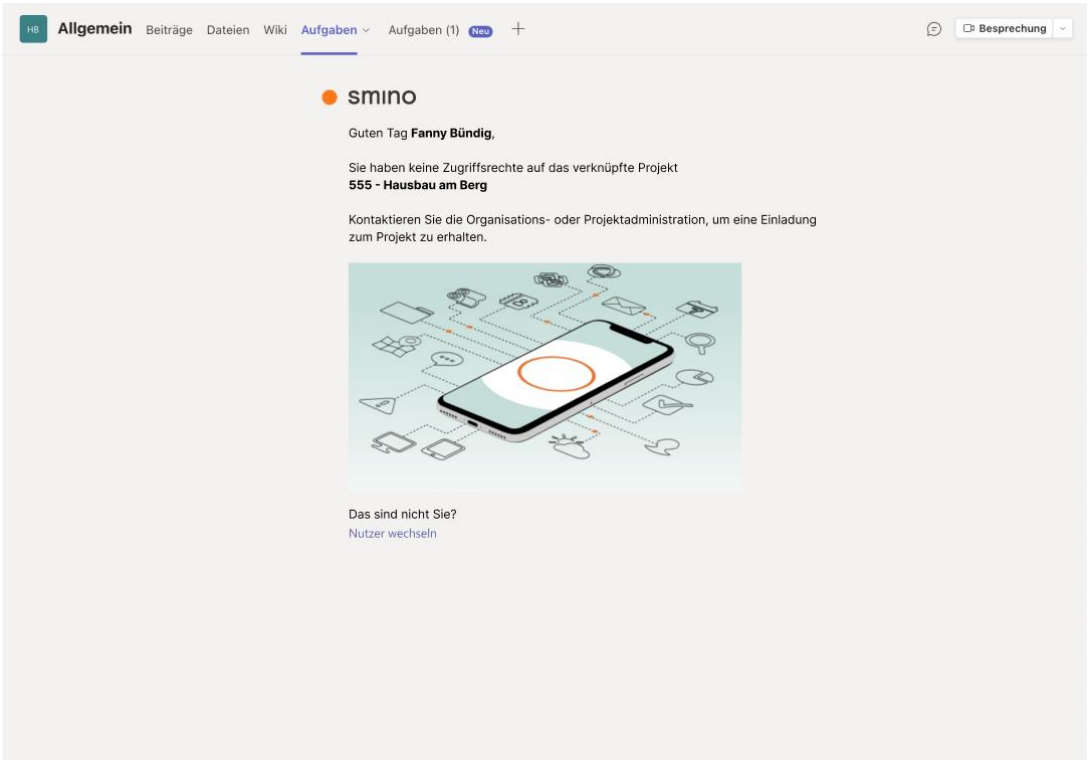
- Als Nutzer möchte ich Menüpunkte von smino in mehreren Channeln auf Teams einbinden können
- Als Nutzer möchte ich verschiedene Menüpunkte aus verschiedenen Projekten von smino in mehreren Channeln auf Teams einbinden können
- Als Nutzer möchte ich diesselben Menüpunkte aus verschiedenen Projekten von smino in mehreren Channeln auf Teams einbinden können

Listenansicht
- Als Nutzer möchte ich meine Aufgabenliste in Teams einsehen können
- Als Nutzer möchte ich meine Aufgabenliste in Teams filtern können
- Als Nutzer möchte ich meine Aufgabenliste in Teams sortieren können
- Als Nutzer möchte ich die Detailinformationen zu eine Aufgabe in Teams aufrufen können

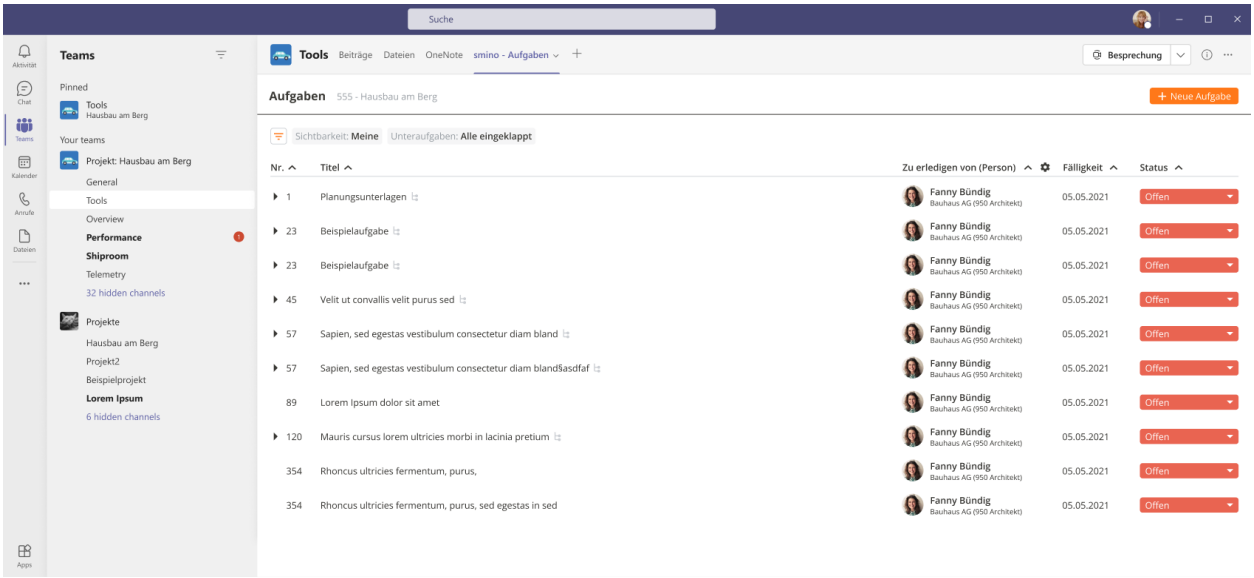
Anhang Abbildung 23: Figma - User Anforderung (smino, 2022)



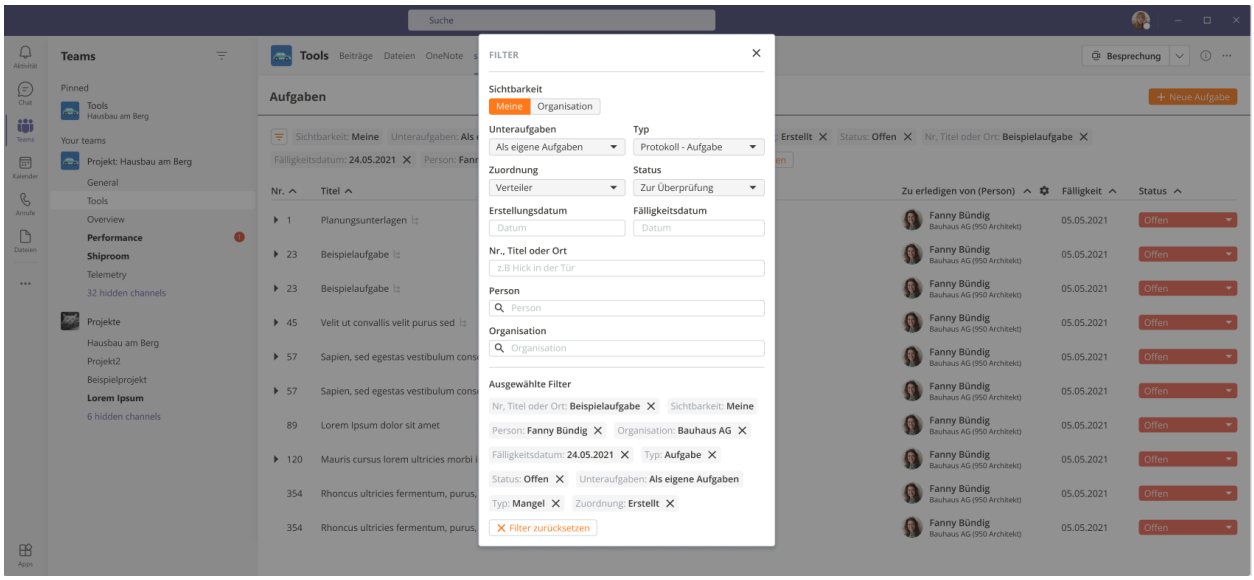
Anhang Abbildung 24: Figma - Tab Login (smino, 2022)



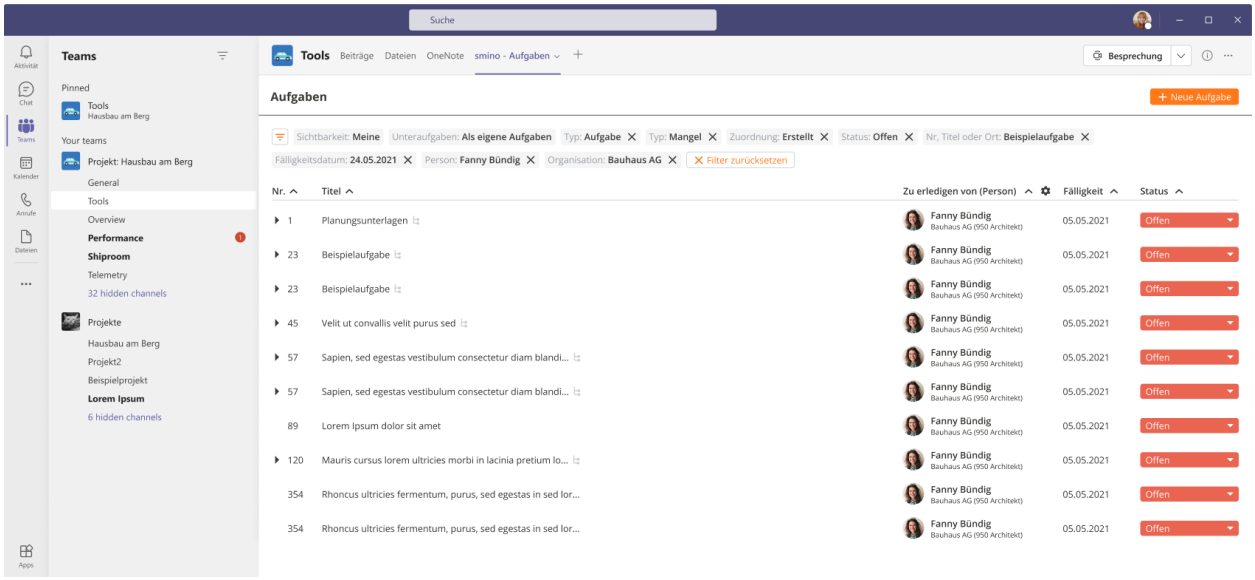
Anhang Abbildung 25: Figma – Forbidden (smino, 2022)



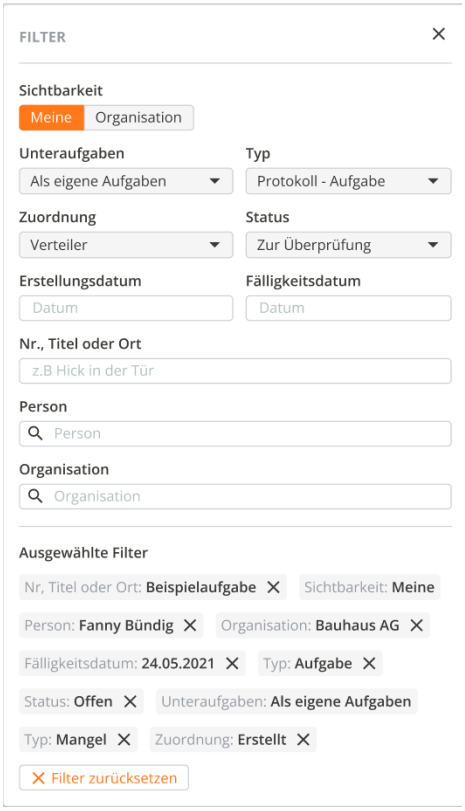
Anhang Abbildung 26: Figma – Aufgaben (smino, 2022)



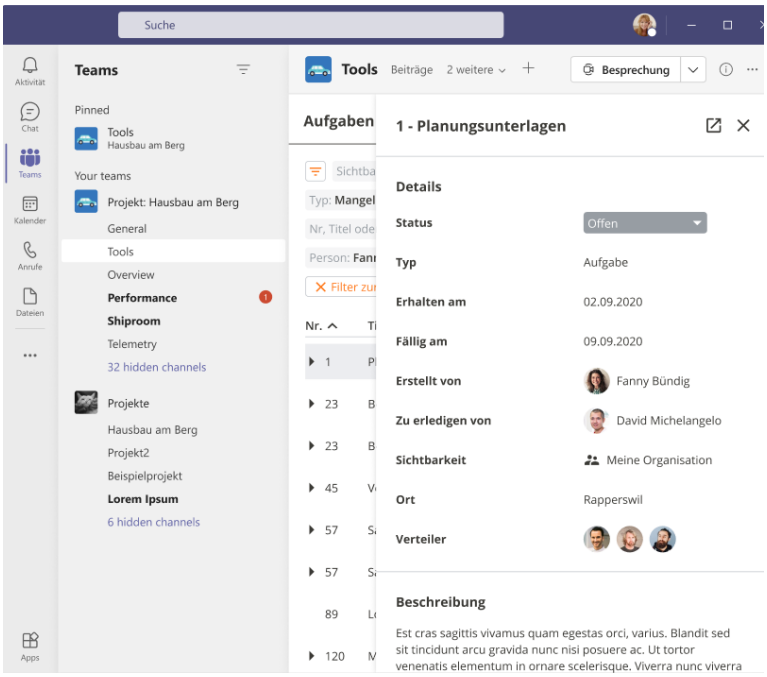
Anhang Abbildung 27: Figma – Aufgabenfilter (smino, 2022)



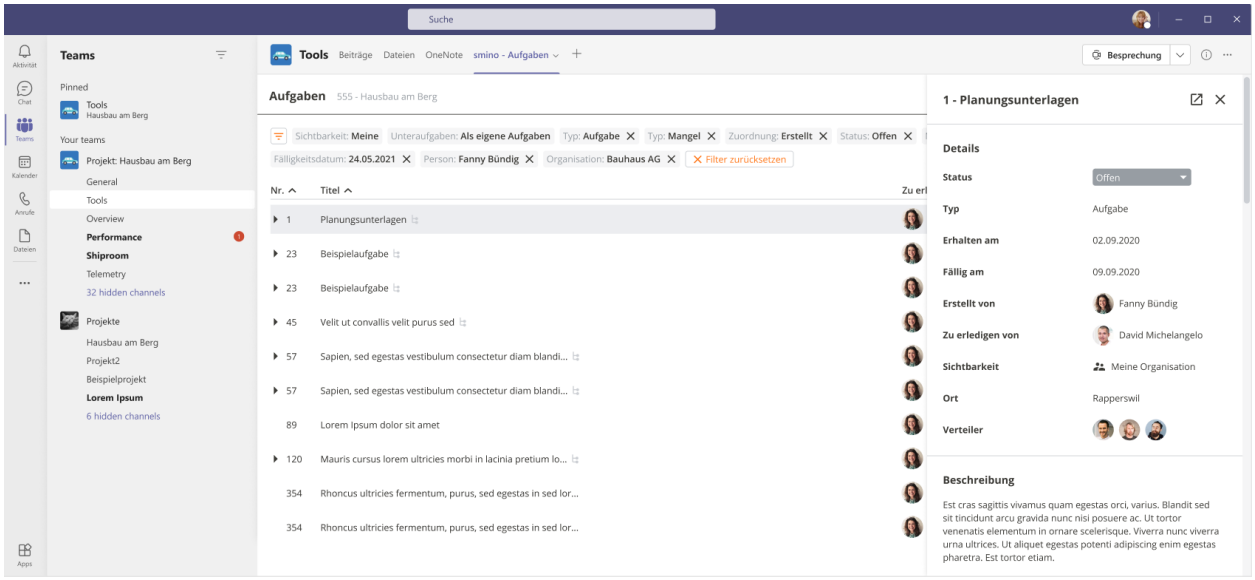
Anhang Abbildung 28: Figma - Aufgaben gefiltert (smino, 2022)



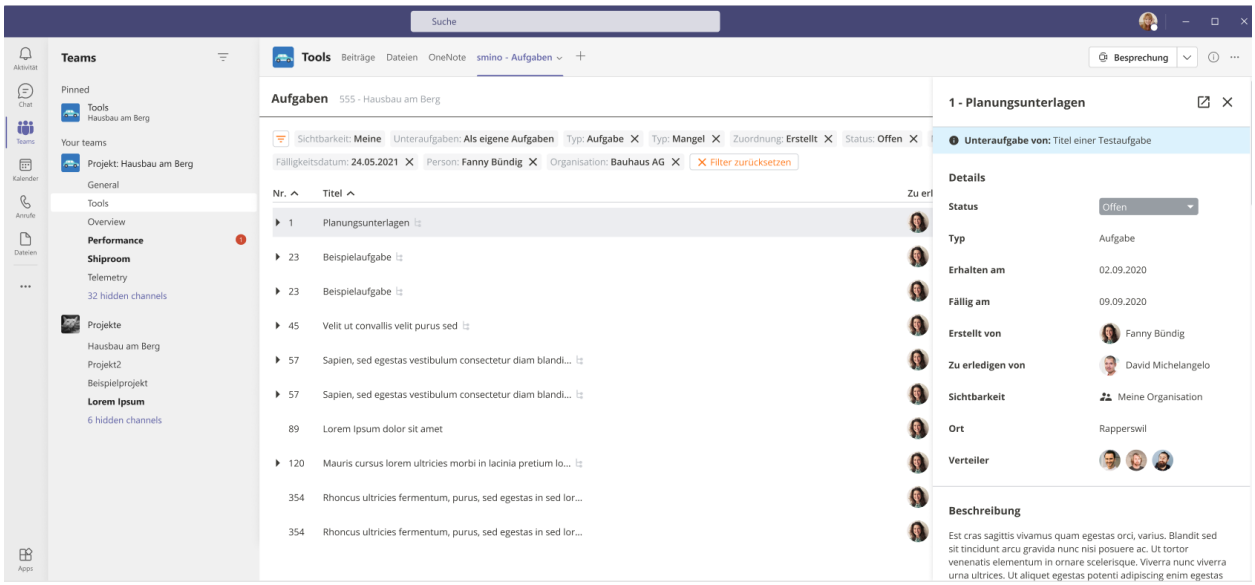
Anhang Abbildung 29: Figma – Filter (smino, 2022)



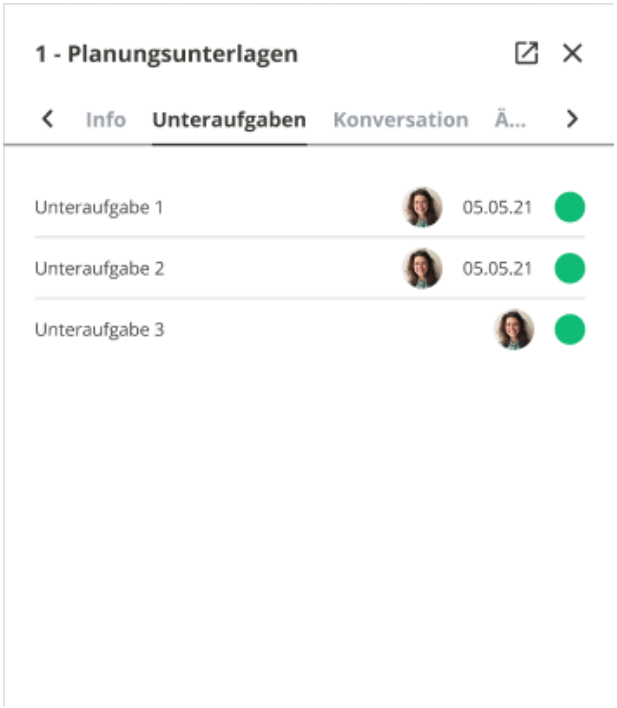
Anhang Abbildung 30: Figma – gefilterte Aufgaben und Detailansicht (small) (smino, 2022)



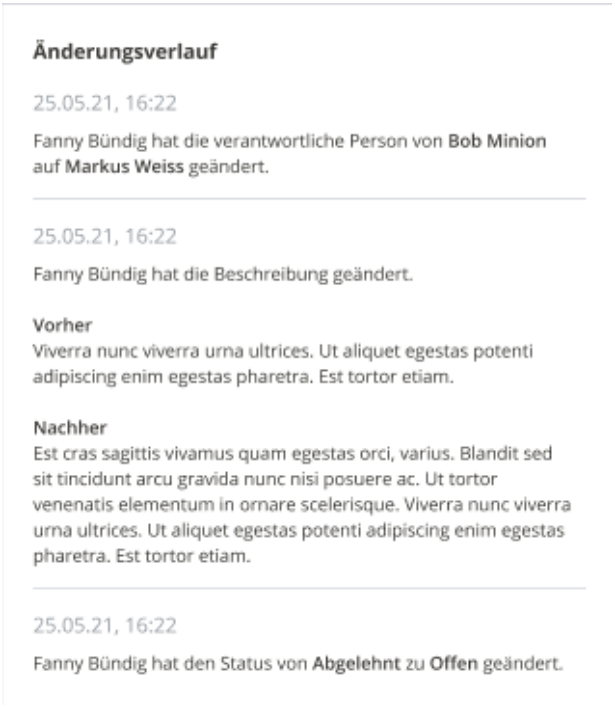
Anhang Abbildung 31: Figma – gefilterte Aufgaben und Detailansicht (smino, 2022)



Anhang Abbildung 32: Figma - Detailansicht einer Unteraufgabe (smino, 2022)



Anhang Abbildung 33: Figma - Detailansicht Unteraufgaben (smino, 2022)



Anhang Abbildung 34: Figma - Detailansicht Änderungsverlauf (smino, 2022)

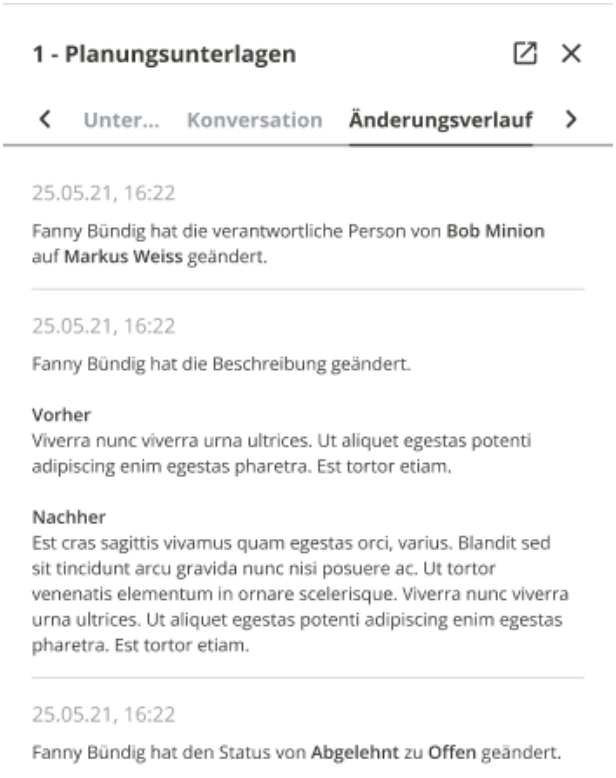
Anhang Abbildung 35: Figma - Detailansicht Info (smino, 2022)

Anhang Abbildung 36: Figma - Detailansicht Konversation (smino, 2022)

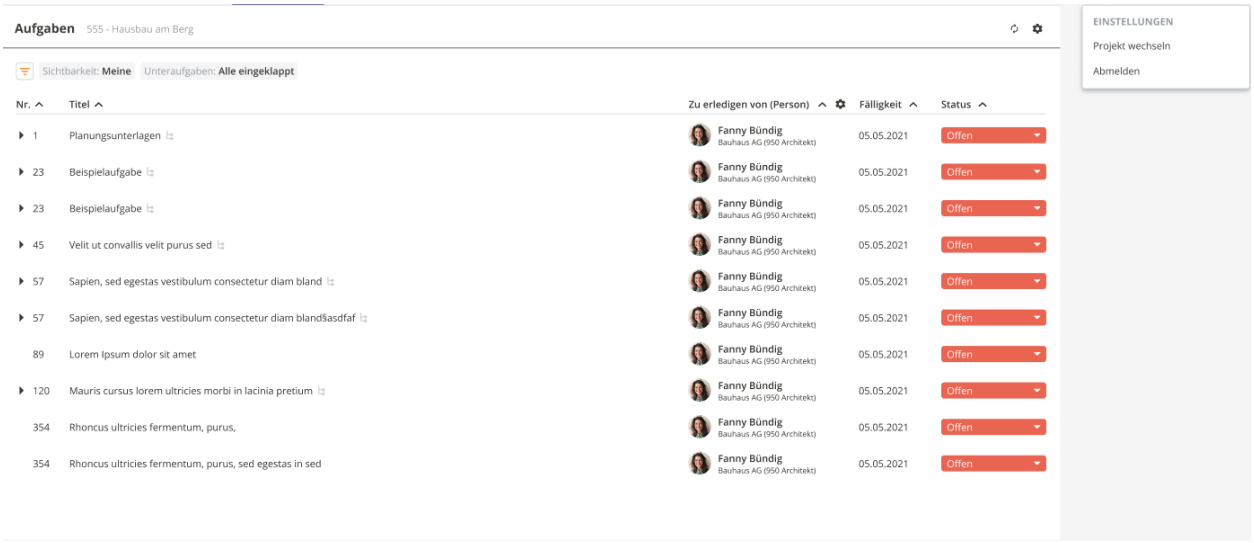
Anhang Abbildung 37: Figma – Detailansicht Info mit Tabs (smino, 2022)

Anhang Abbildung 38: Figma - Detailansicht Unteraufgabe mit Tabs (smino, 2022)

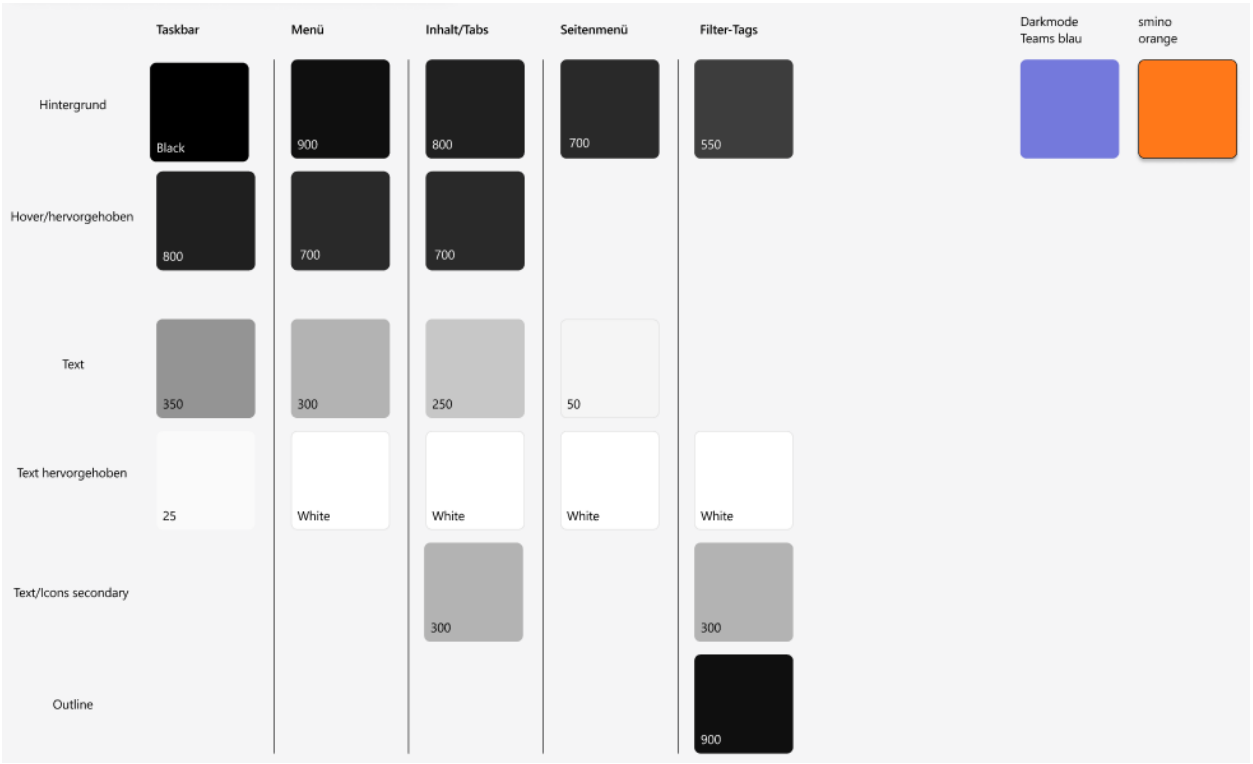
Anhang Abbildung 39: Figma - Detailansicht Konservation mit Tabs (smino, 2022)



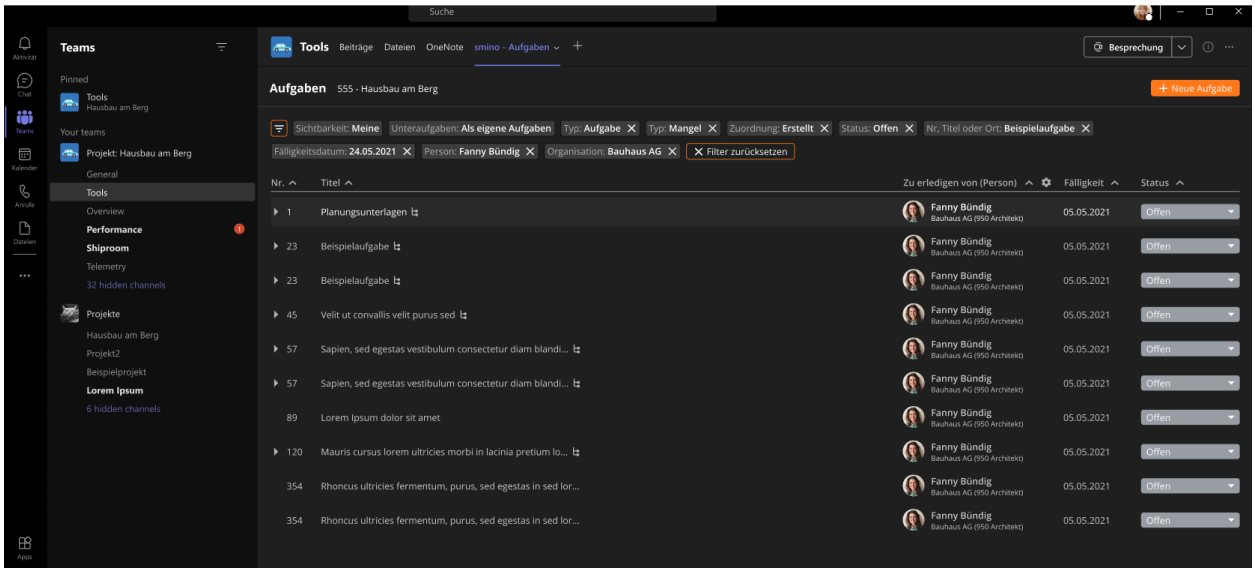
Anhang Abbildung 40: Figma - Detailansicht Änderungsverlauf mit Tabs (smino, 2022)



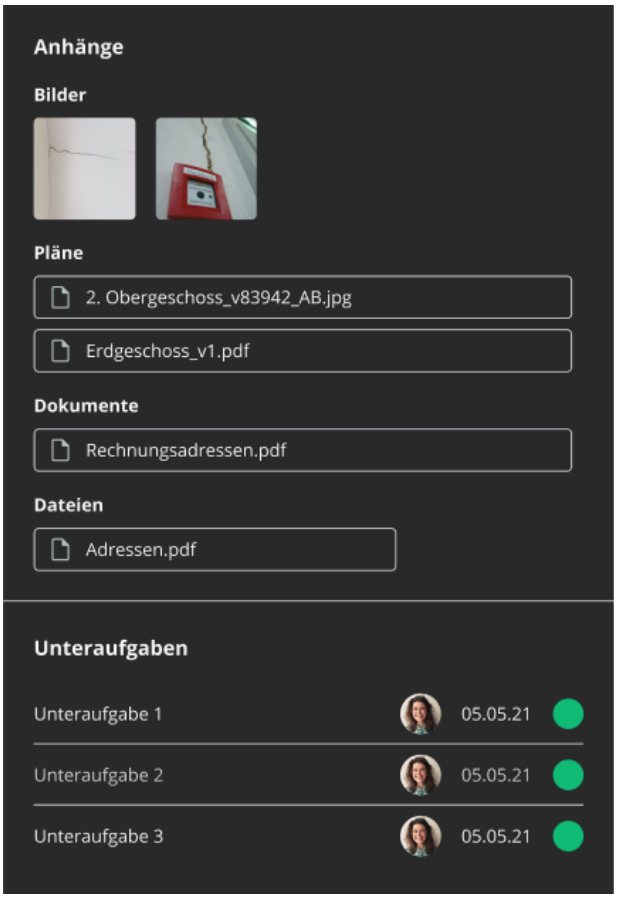
Anhang Abbildung 41: Figma - Refresh- und Settings-Button (smino, 2022)



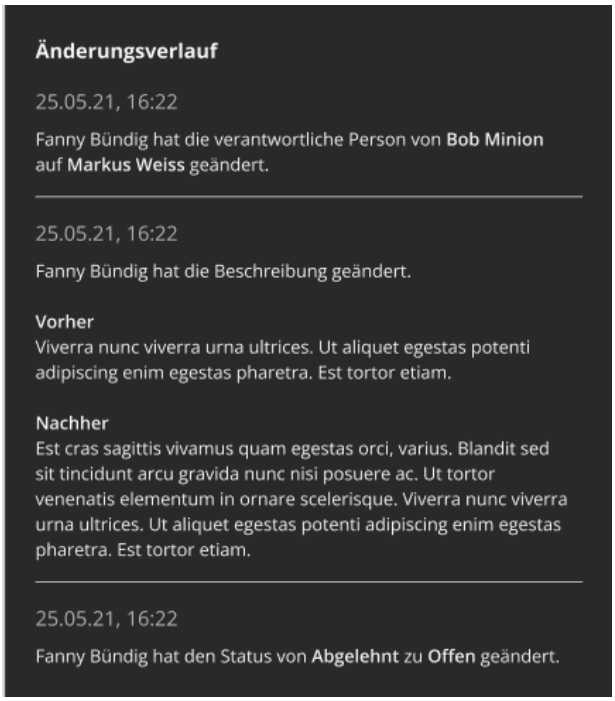
Anhang Abbildung 42: Figma - Dark-Theme Color-Palette (smino, 2022)



Anhang Abbildung 43: Figma - Aufgaben (Dark-Mode) (smino, 2022)



Anhang Abbildung 44: Figma - Detailansicht Unteraufgaben (Dark-Mode) (smino, 2022)



Anhang Abbildung 45: Figma - Detailansicht Änderungsverlauf (Dark-Mode) (smino, 2022)

Anhang Abbildung 46: Figma - Aufgabendetails Info (Dark-Mode) (smino, 2022)

6. Vergleich React und Web-Components Komponenten

Die grün hinterlegten Zellen markieren, welche Komponenten in beiden Libraries vorhanden sind.

@fluentui/web-components	@fluentui/react-northstar
Accordion	Accordion
Anchor	Alert
Anchored Region	Animation
	Attachment
	Avatar
	Box
Breadcrumb	Breadcrumb
Button	Button
Card	Card
	Carousel
	Chat
Checkbox / Switch	Checkbox
	Datepicker
Dialog	Dialog
Divider	Divider
Select / Combobox	Dropdown
	Embed
Flipper	Flex
	Form
	Grid
	Header
	Image
Number Field / Text Field	Input
	Item Layout
	Label
	Layout
ListBox	Lost
Progress / Progress Ring	Loader
Menu	Menu
	Menu Button
Badge	Pill
	Popup
	Portal
	Provider
Radio / Radio Group	Radio Group
	Reaction
	Ref
	Segment
Skeletion	Skeleton
Slider	Slider
	Split Button
	Status
Data Grid	Table
Tabs	Text

Text Area	Text Area
	Toolbar
Tooltip	Tooltip
Tree View	Tree
	Video

Anhang Tabelle 1: Vergleich React und Web-Components Komponenten (eigene Darstellung)

7. Backlog Sprints

Rechtlicher Hinweis:
Alle nachfolgenden Tabellen sind Auszüge aus dem vom Projektteam verwalteten Azure DevOps Backlog und sind eigene Darstellungen

Sprint 0 (21.02.2022 – 07.03.2022)

ID	Name	Effort
3	Projektplan erstellen	-
4	Risikoanalyse erstellen	-
7	Initiale Einarbeitung in MS Teams Entwicklung	-
2	Confluence einrichten	-
6	DevOps Einrichten	-
Total		**-**

Sprint 1 (07.03.2022 – 21.03.2022)

ID	Name	Effort
8	Dokumentation und Analyse Teams Integrationsmöglichkeiten	-
9	Authentifizierung/Autorisierung in MS Teams	-
13	Vorbereitung Workshop Anforderungen Integration Aufgaben in Teams	-
15	Authentifizierung Prototyp	-
16	Beispielansicht Aufgaben	-
33	Evaluation Angular Fluent UI	-
37	Pre-Commit Hooks, Linter und CI Pipeline	-
24	Analyse Testkonzept für MS Teams Entwicklung	-
32	Integration von Tests in Prototyp	-
Total		**-**

Sprint 2 (21.03.2022 – 04.04.2022)

ID	Name	Effort
20	App Template erstellen	5
38	Initiale smino API Services abbilden	3
19	Authentifizierung smino in App einbinden	5
26	Tab-Configuration Page erstellen	3
25	Tab Aufgaben einbinden	3
28	Privacy und Terms of Use	1
56	Zwischenpräsentation	-
Total		**20**

Sprint 3 (04.04.2022 – 18.04.2022)

ID	Name	Effort
64	Umstellung ESLint auf smino Rules	1
62	Models, API-Endpoints und Services anhand von smino übernehmen	3
61	Umstellung NGRX auf NGXS	5
58	Authentification Flow mit Collaborators	3
63	Umstellung der Aufgaben Komponenten von Material auf smino Komponenten	5
73	Error Handling und Loading Page	2
Total		**19**

Sprint 4 (18.04.2022 – 02.05.2022)

ID	Name	Effort
39	Tab Aufgaben filtern	5
40	Tab Aufgaben inline editieren	3
27	Details für Aufgabe anzeigen	8
77	Popup beim hovern einer Aufgabe	1
87	Tab Aufgaben loading performance	3
89	SharedModule entfernen	1
92	Dokumentation zum lokalen Ausführen der Anwendung	1
Total		**22**

Sprint 5 (02.05.2022 – 16.05.2022)

ID	Name	Effort
99	Bugs Issue Details	2
11	Analyse und Konzeption Deployment von Teams Applikationen	8
67	Technisches Konzept Theming für smino Komponenten	5
88	Tab Aufgaben unauthorized project acccess	1
45	Mehrsprachigkeit einbinden	3
66	Design/Farben für Dark Theme	2
94	Übersetzungen cachen	2
95	Aufgaben – Darstellung Unteraufgaben in der Aufgabenliste	2
96	Aufgaben – Darstellung Unteraufgaben in den Details	2
114	Upgrade Teams Toolkit to Version 3.8	1
Total		**28**

Sprint 6 (16.05.2022 – 30.05.2022)

ID	Name	Effort
110	Logout in Tabs ermöglichen	5
43	Provisionierung der Web-Applikation	3
104	Pläne und Dateien in Teams öffnen	2
105	Tab-Ansicht Aufgabendetails	3
44	Provisionierung der Teams-Applikation	5
106	Tab Unteraufgaben für Aufgabendetails	3
131	Bug: Bilde-Anzeige über Teams SDK	2
148	UX Improvements	2
147	Bug: Sub-Tasks Einträge werden dupliziert	0
140	Bug: Aufgaben Tab endlos loop	1
143	Bug: Gelöschte Filter werden erst übernommen, wenn andere Option angepasst wird	1
144	Bug: Dropdown Filter "Alle" Option funktioniert nicht korrekt	1
117	Teams App Logos	1
124	Smino Teams Umgebung einrichten	3
130	Konvertierung von klassischen Release Pipelines zu YML Pipelines	3
135	Upgrade Teams Toolkit to Version 4.0	1
Total		**36**

Sprint 7 (30.05.2022 – 06.06.2022)

ID	Name	Effort
151	Filter UX Improvements	1
152	Bug: Filter Gruppierung entfernen	2
88	Tab Aufgaben unauthorized project acccess	1
141	Bug: Anmeldung wenn man keinen Zugriff auf ein Projekt hat	0
110	Logouts in Tabs ermöglichen	5
119	Sprache sollte primär von User übernommen werden	2
107	Tab Änderungsverlauf	1
125	Filter Caching	2
136	Bug: «Webseite öffnen» öffnet den falschen Link	1
154	App Insights	2
146	Aufgabe beim Öffnen der Details neu laden	2
142	Bug: Feld "zu erledigen von" wird nicht gesetzt	1
80	E2E Tests erstellen	5
79	Tests für Issue Module implementieren	3
108	Testing Sorting Module	2
109	Testing Filter Module	3
153	Refresh Button für Aufgaben	1
155	Tab Scrolling	1
Total		**35**

Sprint 8 (06.06.2022 – 17.07.2022)

ID	Name	Effort
112	Mobile «Ansicht» für Aufgaben Dokumentation	2
144	Dokumentation Architektur Teams Anwendung	2
149	Dokumentation «Technical Debt»	2
150	Vorlage Test-Protokoll erstellen	2
161	Projektbericht	13
162	Merging Main branch	3
163	Projektplan	3
164	Abstract	3
165	Poster	2
166	Person verschwindet aus der List nach Filterung und Laden der Details	1
Total		**33**

8. Backlog Offenes

Rechtlicher Hinweis:
Alle nachfolgenden Tabellen sind Auszüge aus dem vom Projektteam verwalteten Azure DevOps Backlog und sind eigene Darstellungen

ID	Name
Integration der app basics	
21	Chat Page mit Commands erstellen
Integration von smino Aufgaben in MS Teams	
111	Link Unfurling für Aufgaben
113	Tab Konversationen
Refactoring	
75	Tokens für Collaborator in Storage vermerken
76	Separater Signin Callback
98	CDK Virtual Scrolling
167	Datepicker Options nicht übersetzt
User-Testing	
156	Tabs Spacing von Unten
157	Erster Tab Titel und Spacing der Inhalte sollten auf einer Linie sein
159	Empty States für Unteraufgaben Titel und Avatar
160	Banner für Elternaufgabe komplett Anklickbar
Internationalization	
82	Localization (Formate)

9. Backlog Items Auszug

Rechtlicher Hinweis:
Alle nachfolgenden Tabellen sind Auszüge aus dem vom Projektteam verwalteten Azure DevOps Backlog und sind eigene Darstellungen

Feature «Projekt initialisieren»

Das Feature «Projekt initialisieren» behandelt die Tätigkeiten, welche notwendig sind um initial in das Projekt zu starten und alle notwendigen Mittel für die Bearbeitung bereitzustellen.

PRODUCT BACKLOG ITEM 2			
2	Confluence einrichten		
State	Done	**Effort**	-
Beschreibung			
Um die Dokumentation und Sitzungsprotokolle zu verwalten soll ein Confluence Workspace eingerichtet werden und die Projektbeteiligten berechtigt werden.			

PRODUCT BACKLOG ITEM 3			
3	Projektplan erstellen		
State	Done	**Effort**	-
Beschreibung			
Erstellung eines Projektplans. Der Projektplan soll den Auftrag, sowie das Vorgehen im Projekt beschreiben.			

PRODUCT BACKLOG ITEM 4			
4	Risikoanalyse erstellen		
State	Done	**Effort**	-
Beschreibung			
Für das Projekt sollen die grössten Risiken analysiert und eine Strategie definiert werden, wie mit diesen umgegangen werden soll.			

PRODUCT BACKLOG ITEM 6			
6	DevOps einrichten		
State	Done	**Effort**	-
Beschreibung			
Für die Versionskontrolle und die Backlogverwaltung wird Azure DevOps verwendet. Dies soll entsprechend aufgesetzt werden und die relevanten Projektmitglieder berechtigt werden.			

Feature «Einarbeitung Teams Entwicklung»

Für das Projekt müssen zuerst Kompetenzen im Bereich der MS Teams Entwicklung aufgebaut werden. Einerseits soll generell ein Überblick über MS Teams und die entsprechende Entwicklung geschaffen werden, als auch schon konkrete Integrationsmöglichkeiten oder auch die Authentifizierung.

PRODUCT BACKLOG ITEM 7			
7	Initiale Einarbeitung in MS Teams Entwicklung		
State	Done	**Effort**	-
Beschreibung			
Initial müssen die Entwicklungskonzepte für MS Teams erarbeitet werden. Dafür stellt Microsoft eine Dokumentation zur Verfügung, welche die verschiedenen Integrationsmöglichkeiten in Teams beschreibt. Ziel soll es sein einen initialen Überblick über die Entwicklung und Möglichkeiten von MS Teams zu gewinnen und die festzuhalten.			

PRODUCT BACKLOG ITEM 8			
8	Dokumentation und Analyse Teams Integrationsmöglichkeiten		
State	Done	**Effort**	-
Beschreibung			
Im speziellen soll dokumentiert werden, welche Integrationsmöglichkeiten in MS Teams bestehen und wie die Integration für smino aussehen könnte.			

PRODUCT BACKLOG ITEM 9			
9	Authentifizierung/Authorisierung in MS Teams		
State	Done	**Effort**	-
Beschreibung			
Die Authentifizierung bildet eine fundamentale Anforderung in der Integration von smino in Teams. Es soll erarbeitet werden, wie das Authentifizierungsverfahren in Teams aussieht und was dafür eventuell noch notwendig ist.			

PRODUCT BACKLOG ITEM 9			
9	Authentifizierung/Authorisierung in MS Teams		
State	Done	**Effort**	-
Beschreibung			
Die Authentifizierung bildet eine fundamentale Anforderung in der Integration von smino in Teams. Es soll erarbeitet werden, wie das Authentifizierungsverfahren in Teams aussieht und was dafür eventuell noch notwendig ist.			

PRODUCT BACKLOG ITEM 24			
24	Analyse Testkonzept für MS Teams Entwicklung		
State	Done	**Effort**	-
Beschreibung			
Für die Qualitätssicherung ist es von grosser Wichtigkeit, dass die Anwendung getestet wird. Gerade für den Kontext MS Teams soll erarbeitet werden, wie die Anwendung vor allem auch automatisiert getestet werden kann.			

PRODUCT BACKLOG ITEM 33			
33	Evaluation Angular Fluent UI		
State	Done	**Effort**	-
Beschreibung			
Es muss evaluiert werden, ob Angular mit den Fluent UI Web-Components verwendet werden kann und ob ein Theme für Teams, analog zu den React Komponent (Fluent UI (windows.net))[22] erstellt werden kann.			

Feature «Prototypentwicklung»

Für die weitere Entwicklung soll ein Prototyp erstellt werden, welche die geplanten Konzepte bereits demonstriert und sicherstellt, dass die geplanten Funktionalitäten für die Umsetzung machbar sind. Es wird je einen Prototyp für Angular und für React erstellt.

PRODUCT BACKLOG ITEM 15			
15	Authentifizierung Prototyp		
State	Done	**Effort**	-
Beschreibung			
Im Prototyp soll die Authentifizierung möglichst einfach eingebunden werden. Dafür kann initial ein beliebiger OpenID Connect Endpunkt verwendet werden, wobei in einem weiteren Schritt sicher die Authentifizierung mit dem OpenID Connect Endpunkt von smino selbst getestet werden muss.			

PRODUCT BACKLOG ITEM 16			
16	Beispielansicht für Aufgaben		
State	Done	**Effort**	-
Beschreibung			
Im Prototyp soll bereits ein Tab mit einer Beispielhaften Ansicht der Aufgaben eingebunden werden, um die Funktionalität eines Tabs zu demonstrieren. Ebenso wird hiermit die Konfiguration eines Tabs getestet.			

[22] https://fluentsite.z22.web.core.windows.net/

PRODUCT BACKLOG ITEM 32			
32	Integration von Tests in Prototyp		
State	Done	**Effort**	-
Beschreibung			
Im Prototyp sollen das gewählte Testkonzept und Technologien mit Beispielhaften Tests validiert werden.			

PRODUCT BACKLOG ITEM 37			
37	Pre-Commit Hooks, Linter und CI Pipeline		
State	Done	**Effort**	-
Beschreibung			
Zur Sicherstellung der Code-Qualität sollen entsprechende Code-Formatter und Linter aufgesetzt werden. Diese sollten vor jedem Commit entsprechend durchlaufen, um eine einheitliche Formatierung sicherzustellen und um Fehler zu vermeiden. Entsprechend soll auch eine CI Pipeline aufgesetzt werden, welche gerade zur Validierung von Pull-Requests verwendet werden kann.			

Feature «Integration der app basics»

Die «App Basics» umfassen das initiale Aufsetzen der eigentlichen Teams Applikation, sowie das Umsetzen grundlegender Funktionalitäten wie Authentifizierung oder Tab-Konfiguration.

PRODUCT BACKLOG ITEM 19			
19	Authentifizierung smino in App einbinden		
State	Done	**Effort**	5
Beschreibung			
Einbindung der Authentifizierung via OpenId-Connect an der smino Testumgebung https://api.smino.ninja/.well-known/openid-configuration. Für die Tab-Konfiguration, sowie der Zugriff auf Tabs, welche eine Authentifizierung erfordern, soll ein Login-Fenster implementiert werden. Im Fehlerfall muss auch eine Fehlerpage angezeigt werden, welche es ermöglicht den Login Prozess erneut anzustossen. Die entsprechenden Tokens werden im Local Storage abgelegt. Wenn ein Refresh Token vorhanden ist, soll der Nutzer automatisch (silent) wieder authentifiziert werden.			

PRODUCT BACKLOG ITEM 20			
20	App Template erstellen		
State	Done	**Effort**	5
Beschreibung			
Es soll ein grundlegendes App-Template erstellt werden, welches die Basis für die Entwicklung der smino App für Teams bildet.			

PRODUCT BACKLOG ITEM 21			
21	Chat Page mit Commands erstellen		
State	Approved	**Effort**	8
Beschreibung			
Für die Applikation selbst soll eine Chat-Page erstellt werden, welche es erlaubt sich mit seinem smino Account anzumelden oder abzumelden. Zusätzlich soll auch ein Command für "Hilfe" erstellt werden, welche mögliche Interaktionen aufzeigen soll.			

PRODUCT BACKLOG ITEM 26			
26	Tab-Configuration Page erstellen		
State	Done	**Effort**	3
Beschreibung			
Implementation der Tab-Konfiguration Page anhand der MockUps. Grundsätzlich sollen in der Konfiguration das Projekt und der gewünschte Bereich (Tab) angegeben werden. Wenn die Konfiguration abgeschlossen ist, soll der Tab im Channel hinzugefügt werden.			

PRODUCT BACKLOG ITEM 28			
28	Privacy und terms-of-use Pages		
State	Done	**Effort**	1
Beschreibung			
Verlinkung der Geschäfts- und Datenschutzbedingungen von smino.			

PRODUCT BACKLOG ITEM 38			
38	Initiale smino API services abbilden		
State	Done	**Effort**	3
Beschreibung			
Initiale Anbindung der smino API mit den wichtigsten Services, welche für die Integration der Aufgaben notwendig sind.			

PRODUCT BACKLOG ITEM 58			
58	Authentication Flow mit Collaborators		
State	Done	**Effort**	3
Beschreibung			
Der Authentication Flow muss angepasst werden, wobei für den Collaborator in einem Projekt, also für seine entsprechende Id, ein weiteres Token bezogen werden muss. Dieses Token muss auch verwendet werden, um auf Projektspezifische Daten zuzugreifen.			

PRODUCT BACKLOG ITEM 110			
110	Logout in Tabs ermöglichen		
State	Done	**Effort**	5
Beschreibung			
Dem Benutzer soll es ermöglicht werden, sich von seinem bestehenden smino-Account in der Teams Tab-Ansicht abzumelden.			

BUG 136			
136	«Website öffnen» öffnet den falschen Link		
State	Done	**Effort**	1
Beschreibung			
"Webseite öffnen" auf dem Tab öffnet den Link in der lokal gehosteten App, wobei der Link die smino Web-Applikation aufrufen sollte.			

PRODUCT BACKLOG ITEM 154			
154	App Insights		
State	Done	**Effort**	2
Beschreibung			
Für das Monitoring soll App Insights verwendet werden. Dabei soll die Application Insights mit der Teams Applikation provisioniert werden. Ebenso soll der der bestehende Logging-Service für Application Insights ergänzt werden, um Fehler und Telemetriedaten entsprechend weiterzugeben.			

Feature «smino Architektur übernehmen»

Mit dem Entscheid, die Komponenten von smino direkt zu verwenden und das Projekt möglichst einfach für smino weiterverwendbar zu machen, ist es auch sinnvoll die Architektur so nah wie möglich an smino anzugleichen. Dies soll eine einfache Transition ermöglichen.

PRODUCT BACKLOG ITEM 61			
61	Umstellung von NGRX auf NGXS		
State	Done	**Effort**	5
Beschreibung			
Für den Prototypen und das initiale App-Template wurde NGRX als State Management Lösung für Angular verwendet. Die smino Web-Applikation verwendet NGXS als Lösung, weshalb eine Umstellung notwendig ist.			

PRODUCT BACKLOG ITEM 62			
62	Models, API-Endpoints und Services anhand von smino übernehmen		
State	Done	**Effort**	3
Beschreibung			
API-Endpoint Builders und Models anhand der smino Web-Applikation übernehmen. Services für API-Aufrufe erstellen.			

PRODUCT BACKLOG ITEM 63			
63	Umstellung der Aufgaben Komponenten von Material auf smino Komponenten		
State	Done	**Effort**	5
Beschreibung			
Für die Darstellung der Aufgaben wurden Komponenten von Material verwendet. Um die Ansicht an smino anzugleichen, sollen die bestehenden Komponenten von smino übernommen und nötigenfalls angepasst werden.			

PRODUCT BACKLOG ITEM 64			
64	Umstellung ESLint auf smino Rules		
State	Done	**Effort**	1
Beschreibung			
Bestehende ESLint Rules von smino verwenden.			

PRODUCT BACKLOG ITEM 73			
73	Error Handling und Loading Page		
State	Done	**Effort**	2
Beschreibung			
Für das Error Handling sollen die entsprechenden Fehler über den LoggingService vermerkt werden. Zusätzlich soll die Loading Page benutzt werden, um einen Spinner anzuzeigen, sowie um die Supportmeldung in einem Fehlerfall anzuzeigen.			

PRODUCT BACKLOG ITEM 89			
89	SharedModule entfernen		
State	Done	**Effort**	1
Beschreibung			
SharedModule sollte nicht verwendet werden, da es zu generell ist. Besser wäre es erforderliche Module einzeln zu importieren.			

Feature «Integration von smino Aufgaben in MS Teams»

In diesem Feature ist alles enthalten was die Integration der Aufgaben von smino in MS Teams zum Fokus hat. Die Grundlage bilden dabei die Mockups/Storyboards, welche durch smino erarbeitet wurde und während der Arbeit ergänzt wurden.

PRODUCT BACKLOG ITEM 13			
13	Vorbereitung Workshop Anforderungen Integration Aufgaben in Teams		
State	Done	**Effort**	-
Beschreibung			
-			

PRODUCT BACKLOG ITEM 25			
25	Tab Aufgaben einbinden		
State	Done	**Effort**	3
Beschreibung			
Das "smino - Aufgaben" Tab soll anhand der Mockups als Tab in die Applikation eingebunden werden. Es sollen noch keine Funktionalitäten (details, update, delete, etc.) implementiert werden.			

PRODUCT BACKLOG ITEM 27			
27	Tab Aufgaben einbinden		
State	Done	**Effort**	8
Beschreibung			
Basis-Informationen einer Aufgabe in einem «Side-Sheet» anhand der Mockups implementieren. Die Side-Sheet Komponente muss von Grund auf erstellt werden.			

PRODUCT BACKLOG ITEM 39			
39	Tab Aufgaben filtern		
State	Done	**Effort**	5
Beschreibung			
Die Aufgaben sollten anhand der Mockups gefiltert werden können. Dabei werden die Aufgaben nur durch den Client und nicht mittels der API gefiltert. Die bestehende Filter-Komponente benötigt wesentliche Überarbeitungen, da diese Art von Filterung so noch nicht in der Web-App verwendet wird. Zusätzlich sollten die Daten über den Store, statt in der Komponente gefiltert werden.			

PRODUCT BACKLOG ITEM 40			
40	Tab Aufgaben inline editieren		
State	Done	**Effort**	3
Beschreibung			
In der Liste der Aufgaben soll es möglich sein den Status per Dropdown «inline» verändern zu können.			

PRODUCT BACKLOG ITEM 41			
41	Tab Aufgaben details editieren		
State	Done	**Effort**	3
Beschreibung			
In den Aufgabendetails soll der Status per Dropdown und die Sichtbarkeit per Toggle verändert werden können.			

PRODUCT BACKLOG ITEM 77			
77	smino Styling anwenden		
State	Done	**Effort**	1
Beschreibung			
Wenn die Maus über einer Aufgabe ist, soll ein Popup mit der Beschreibung der Aufgabe und dem Ersteller analog zur Web-Applikation angezeigt werden.			

PRODUCT BACKLOG ITEM 87			
87	smino Styling anwenden		
State	Done	**Effort**	3
Beschreibung			
Die Performance beim Laden von einem Projekt mit sehr vielen Aufgaben (1000+) soll verbessert werden. Einerseits soll nicht darauf gewartet werden, bis alle Aufgaben geladen sind. Aufgaben sollten vorzu, wenn Sie geladen wurden angezeigt werden. Wenn ein neuer «Load» der Aufgaben ausgelöst wird, sollte ein eventuell ältere Load abgebrochen werden. Auch bei der Anzeige sollten Projekte mit sehr vielen Aufgaben berücksichtigt werden, da dies ebenfalls zu Performance Problemen führen kann (Infinite/Virtual Scroll).			

PRODUCT BACKLOG ITEM 88			
88	Tab Aufgaben unauthorized project acccess		
State	Done	**Effort**	1
Beschreibung			
Beim Zugriff auf ein Tab (Projekt) ohne Berechtigung, sollte eine «Forbidden» Seite dargestellt werden, anstelle von der «Login» Seite.			

PRODUCT BACKLOG ITEM 95			
95	Aufgaben - Darstellung Unteraufgaben in der Aufgabenliste		
State	Done	**Effort**	2
Beschreibung			
Für die Aufgabenliste soll definiert werden können, ob Unteraufgaben untergeordnet oder als eigen Aufgaben dargestellt werden sollen (siehe Mockups).			

PRODUCT BACKLOG ITEM 96			
96	Aufgaben - Darstellung Unteraufgaben in der Aufgabenliste		
State	Done	**Effort**	2
Beschreibung			
In den Aufgabenteils soll ein «Banner» angezeigt werden, welcher angibt, ob die Aufgabe eine Unteraufgabe ist und auch den Namen der Elternaufgabe mit einem Verweis auf die entsprechende Aufgabe.			

PRODUCT BACKLOG ITEM 104			
104	Pläne und Dateien in Teams öffnen		
State	Done	**Effort**	2
Beschreibung			
Als Benutzer möchte ich die Anhänge (Pläne und Dateien) für eine Aufgabe auch in Teams ansehen können. Pläne und Dokumente sollen dabei in der Web-Applikation geöffnet werden. Dateien sollen direkt heruntergeladen werden.			
PRODUCT BACKLOG ITEM 105			

105	**Tab-Ansicht Aufgabendetails**		
State	Done	**Effort**	3
Beschreibung			
Die Aufgabendetails sollen anhand der Mockups mittels eigenen Tabs dargestellt werden. Die Basisinformationen befinden sich dabei im Tab "Info".			

PRODUCT BACKLOG ITEM 106			
106	Tab Unteraufgaben für Aufgabendetails		
State	Done	**Effort**	3
Beschreibung			
Für eine Aufgabe soll es möglich sein Unteraufgaben in einem separaten Tab anzusehen. Wenn eine Aufgabe keine Unteraufgaben besitzt, soll das entsprechende Tab nicht angezeigt werden.			

PRODUCT BACKLOG ITEM 107			
107	Tab Änderungsverlauf		
State	Done	**Effort**	3
Beschreibung			
Als Benutzer soll es mir möglich sein, den Änderungsverlauf einer Aufgabe in einem separaten Tab anzusehen. Die Änderungen sind dabei in Textueller Form aufgelistet.			

PRODUCT BACKLOG ITEM 111			
111	Link Unfurling für Aufgaben		
State	New	**Effort**	
Beschreibung			
Es wäre sinnvoll, wenn in Teams für einen Link zu einer smino Aufgabe entsprechende Informationen direkt angezeigt werden. Dies würde eine einfache Diskussion über Inhalte ermöglichen und dem Nutzer direkt einige Informationen zur Aufgabe anzusehen.			

PRODUCT BACKLOG ITEM 112			
112	Mobile "Ansicht" für Aufgaben Dokumentation		
State	Done	**Effort**	2
Beschreibung			
In Teams soll keine Mobile Ansicht für smino implementiert werden. Stattdessen soll die native App von smino benutzt werden. Beispielsweise wird dies für OneNote ebenfalls entsprechend gemacht. Es soll dokumentiert werden, wie eine mögliche Lösung dafür konzeptionell aussehen könnte.			

PRODUCT BACKLOG ITEM 113			
113	Tab Konversationen		
State	New	**Effort**	8
Beschreibung			
Einem Nutzer soll es ermöglicht werden in Teams mit den Konversationen einer Aufgabe zu interagieren. Dafür müsste definiert werden, wie dies in der Teams Applikation aussehen könnte oder welche weiteren Integrationsmöglichkeiten seitens MS Teams dafür sinnvoll wären.			

BUG 131			
131	Bilder-Anzeige über Teams SDK		
State	Done	**Effort**	2
Beschreibung			
Die Bilder werden in Teams nicht korrekt dargestellt, sondern sind um 50% nach Links-Oben verschoben. Dafür soll ein Bug für Teams-JS erstellt werden und der native Image-Viewer durch den Image-Viewer von smino ersetzt werden.			

PRODUCT BACKLOG ITEM 153			
153	Refresh Button für Aufgaben		
State	Done	**Effort**	1
Beschreibung			
Es soll möglich sein die Aufgaben in der Liste komplett über einen Button zu aktualisieren.			

Feature «Bereitstellung der Applikation»

Hier finden sich alle Features welche die Provisionierung oder das Deployment der Teams Applikation und der Notwendigen Infrastruktur zum Fokus haben.

PRODUCT BACKLOG ITEM 11			
11	Analyse und Konzeption Deployment von Teams Applikationen		
State	Done	**Effort**	8
Beschreibung			
Es soll analysiert werden, wie eine Teams-Applikation bereitgestellt werden kann und welche Schritte dazu notwendig sind. Ebenso soll überlegt werden, wie eine Testumgebung aufgebaut werden kann und wie die Web-Applikation selbst provisioniert wird.			

PRODUCT BACKLOG ITEM 43			
43	Provisionierung der Web-Applikation		
State	Done	**Effort**	3
Beschreibung			
Es sollen Ressourcen und eine Release Pipeline erstellt werden, welche es ermöglichen die Web-Applikation bereitzustellen.			

PRODUCT BACKLOG ITEM 44			
44	Provisionierung der Teams-Applikation		
State	Done	**Effort**	5
Beschreibung			
Die Teams Applikation selbst soll ebenfalls automatisch in den Store provisioniert werden. Dies sollte ebenfalls automatisiert stattfinden.			

PRODUCT BACKLOG ITEM 117			
117	Teams App Logos		
State	Done	**Effort**	1
Beschreibung			
Einbindung vom smino Logo für die Teams Applikation.			

PRODUCT BACKLOG ITEM 124			
124	Smino Teams Umgebung einrichten		
State	Done	**Effort**	3
Beschreibung			
Dokumentation und Unterstützung beim Aufbau der Teams Umgebung für smino.			

PRODUCT BACKLOG ITEM 130			
130	Konvertierung von klassischen Release Pipelines zu YML Pipelines		
State	Done	**Effort**	3
Beschreibung			
Initial wurden für den Aufbau der Release Pipelines die klassischen Release Pipelines von Azure DevOps verwendet. Für die weitergab, sowie für die Erzeugung von Artefakten würden die YAML basierten Pipelines eine enorme Erleichterung bringen.			

Feature «Erweiterung von smino Theming»

Für Teams sollten alle Komponenten mindestens einen Dark-Mode unterstützen, um eine Integration mit dem Dark-Theme für Teams zu gewährleisten. Dafür muss ein Farbkonzept erstellt werden und dies entsprechend technisch umgesetzt werden.

PRODUCT BACKLOG ITEM 66			
66	Design/Farben für Dark Theme		
State	Done	**Effort**	2
Beschreibung			
Definition/Farbkonzept der verwendeten smino Komponenten für ein Dark-Theme.			

PRODUCT BACKLOG ITEM 67			
67	Technisches Konzept Theming für smino Komponenten		
State	Done	**Effort**	5
Beschreibung			
Technisches Konzept für das Theming von smino Komponenten.			

Feature «Refactoring»

Refactoring enthält allfällige Refactorings oder Optimierung welche Code-Technisch optimiert werden könnten. Hier werden auch Belangen festgehalten, welche in PR's festgestellt wurden und eine grössere Anpassungen nach sich ziehen.

PRODUCT BACKLOG ITEM 75			
75	Tokens für Collaborator in Storage vermerken		
State	New	**Effort**	3
Beschreibung			
Momentan wird das Token für den Collaborator jedes Mal erneut geholt, selbst wenn noch ein gültiges Token existieren würde. Das Token für den Collaborator sollte daher im Storage abgelegt werden, um es wiederzuverwenden.			

PRODUCT BACKLOG ITEM 76			
76	Separater Signin Callback		
State	New	**Effort**	2
Beschreibung			
Für den OIDC Signin Callback sollte eine separate Route verwendet werden. Dies ermöglicht es einerseits einfacher zu prüfen oder zu erwarten, ob ein Callback stattfinden müsste. Andererseits vermeidet es Probleme mit dem Routing, bzw. erlaubt eine einfache Darstellung eines Spinners.			

PRODUCT BACKLOG ITEM 98			
98	CDK Virtual Scrolling		
State	New	**Effort**	3
Beschreibung			
Die aktuell verwendete Library für das Virtual Scrolling wird nicht mehr aktiv gewartet. Im besten Fall könnte man für das Virtual Scrolling @angular/cdk verwenden. Einige Features wie die dynamische Grösse der Items oder die Definition eines alternativ «Scroll»-Elements wird aber erst in kommenden Versionen produktiv verfügbar.			

PRODUCT BACKLOG ITEM 114			
114	Upgrade Teams Toolkit to version 3.8		
State	Done	**Effort**	-
Beschreibung			
Upgrade des Teams Toolkit auf Version 3.8 und anheben der CLI Version des Toolkits.			

PRODUCT BACKLOG ITEM 116			
116	Logo für die OpenId Connect Anwendung		
State	New	**Effort**	1
Beschreibung			
Für den OpenId-Connect Client sollte ein passendes Logo ausgewählt werden, welche die Teams-App wiedererkennbar macht.			

PRODUCT BACKLOG ITEM 118			
118	Beschreibung für den OpenId Client		
State	New	**Effort**	1
Beschreibung			
Überarbeitung der Beschreibung der Teams Applikation im OpenId-Connect Client.			

PRODUCT BACKLOG ITEM 119			
119	Sprache sollte primär von User übernommen werden		
State	Done	**Effort**	2
Beschreibung			
Die Priorisierung für die Auswahl der User-Sprache sollte angepasst werden, dabei soll die gespeicherte User-Sprache von smino die höchste Priorität bekommen. Prio: 1. User 2. Teams			

PRODUCT BACKLOG ITEM 120			
120	OpenId ClientId für Dev/Testing ändern		
State	Done	**Effort**	-
Beschreibung			
Umbenennung der ClientId des OpenID Clients für Dev/Testing.			

PRODUCT BACKLOG ITEM 125			
125	Filter Caching		
State	Done	**Effort**	2
Beschreibung			
Die Filter für die Aufgaben sollten pro Projekt im Local Storage cached werden, damit der Nutzer diese nicht bei jedem Aufruf des Tabs verliert.			

PRODUCT BACKLOG ITEM 135			
135	Upgrade Teams Toolkit to Version 4.0		
State	Done	**Effort**	1
Beschreibung			
Upgrade des Teams Toolkit auf Version 4.0 und anheben der CLI Version des Toolkits.			

PRODUCT BACKLOG ITEM 162			
162	Merging Main branch und Abgabebereinigung		
State	Done	**Effort**	3
Beschreibung			
Die offenen PR's müssen nach dem Approval gemerged werden. Zusätzlich sollen noch allfällige Bereinigungen für die Übergabe durchgeführt werden.			

BUG 167			
167	Datepicker Options nicht übersetzt		
State	New	**Effort**	3
Beschreibung			
Für den Datepicker werden aktuell die Optionen noch nicht übersetzt. Dafür müsste zuerst auch noch die Implementierung für das Übersetzen von Batches implementiert/integriert werden.			

Feature «Formalitäten»

Formalitäten beinhalten Termine oder Arbeiten, die in Zusammenhang mit den Anforderungen an die Bachelorarbeit erledigt werden müssen.

PRODUCT BACKLOG ITEM 56			
56	Zwischenpräsentation BA		
State	Done	**Effort**	-
Beschreibung			
Vorbereitung und Präsentation der Zwischenresultate für die Bachelorarbeit.			

Feature «Testing»

Erfasst alle Arbeiten, die im Zusammenhang mit dem Testing der Anwendung erledigt werden sollen, insbesondere deren Umsetzung.

PRODUCT BACKLOG ITEM 79			
79	Tests für Issue Module implementieren		
State	Done	**Effort**	3
Beschreibung			
Implementation von Unit-Tests für das «Issue» Module mit Fokus auf Business Logik/Services.			

PRODUCT BACKLOG ITEM 80			
80	E2E Test erstellen		
State	Done	**Effort**	5
Beschreibung			
E2E Tests erstellen, welche die Funktionsfähigkeit der wichtigsten Abläufe in der Anwendung garantieren. Dies umfasst vor allem die Authentifizierung/Autorisierung, die Tab Konfiguration, sowie den Tab «Aufgaben» mit Filterung, Sortierung und anzeige der Details.			

PRODUCT BACKLOG ITEM 108			
108	Testing Sorting Module		
State	Done	**Effort**	2
Beschreibung			
Unit-Testing der Komponenten für das Sorting Module.			

PRODUCT BACKLOG ITEM 109			
109	Testing Filter Module		
State	Done	**Effort**	3
Beschreibung			
Unit-Testing der Komponenten für das Filter Module.			

PRODUCT BACKLOG ITEM 150			
150	Test-Protokoll erstellen		
State	Done	**Effort**	2
Beschreibung			
-			

Feature «Internationalization»

Enthält alles was Mehrsprachigkeit oder die Formatierung von Culture-Sensitiven Werten umfasst.

PRODUCT BACKLOG ITEM 45			
45	Mehrsprachigkeit einbinden		
State	Done	**Effort**	3
Beschreibung			
Translation Service/Module von smino Einbinden und erforderliche Texte anhand der Benutzersprache übersetzen.			

PRODUCT BACKLOG ITEM 82			
82	Localization (Formate)		
State	Committed	**Effort**	2
Beschreibung			
Für die verschiedenen Cultures soll es möglich sein, dynamisch in Angular die Locale zu ändern, um Culture-Sensitive Werte im korrekten Format anzuzeigen. Die Registrierung der Locale und das Laden der entsprechenden Formate ist in Angular relativ starr, wobei fast ein Reload der Anwendung nötig wäre. Es müsste geprüft werden, ob es möglich ist, dynamisch das Format zu ändern oder ob auf eigene Pipes für die Formatierung gesetzt werden soll.			

PRODUCT BACKLOG ITEM 94			
94	Übersetzungen cachen		
State	Done	**Effort**	2
Beschreibung			
Übersetzungen sollen im Local Storage pro Sprache cached werden. Dies soll es ermöglichen, die Übersetzungen schneller zu laden.			

Feature «User-Testing»

Das Feature «User-Testing» enthält alle vorbereitenden Arbeiten für das User-Testing. Ebenfalls werden hier Änderungen oder Bugs erfasst, welche Aufgrund vom User-Testing oder der Vorbereitung enstehen.

BUG 140			
140	Aufgaben Tab endlos loop		
State	Done	**Effort**	1
Beschreibung			
Aufgaben Tab hängt im endlos loop fest, wenn keine Aufgaben im Projekt erfasst wurden.			

BUG 141			
141	Anmeldung wenn man keinen Zugriff auf ein Projekt hat		
State	Done	**Effort**	0
Beschreibung			
Wenn man aus dem Projekt entfernt wird aber für dieses Projekt bereits ein Aufgaben Tab besteht, kommt die "In smino anmelden" Maske und man kann sich so oft anmelden, wie man will. *Wird in Backlog Item 88 bereits behandelt.*			

BUG 142			
142	Feld «to be done by person» / «zu erledigen von» wird nicht gesetzt		
State	Done	**Effort**	1
Beschreibung			
Das Feld «to be done by person» / «zu erledigen von» wird nicht gesetzt, obwohl es in smino gesetzt ist.			

BUG 143			
143	Gelöschte Filter werden erst übernommen, wenn eine andere Filter Option angepasst wird		
State	Done	**Effort**	1
Beschreibung			
Gelöschte Filter werden erst übernommen, wenn eine andere Filter Option angepasst wird. Beispiel 1: Beim Abwählen einer Checkbox wird der letzte abgewählte Wert nicht aus dem Filter gelöscht. Beispiel 2: Beim Abwählen durch Klick auf «All» werden keine Werte aus dem Filter gelöscht.			

BUG 144			
144	Dopdown Filter "Alle" Option funktioniert nicht korrekt		
State	Done	**Effort**	1
Beschreibung			
Wenn die "Alle" Option in einem Dropdown ausgewählt wird, sollten wenn: • Alle Optionen ausgewählt sind: Alle Optionen abgewählt werden • Keine/teilweise Optionen auswegählt sind: Alle Optionen selektiert werden			

PRODUCT BACKLOG ITEM 146			
146	Aufgabe beim Öffnen der Details neu laden		
State	Done	**Effort**	2
Beschreibung			
Beim Öffnen der Aufgabendetails soll die Aufgabe nochmals neu geladen werden. Dadurch soll die Aktualität der Aufgabendaten verbessert werden.			

BUG 147			
147	Sub-Tasks duplizieren Einträge in der issue list		
State	Done	**Effort**	0
Beschreibung			
Pro Hierarchietiefe werden die Unteraufgaben entsprechend dupliziert angezeigt.			

PRODUCT BACKLOG ITEM 148			
148	UX Improvements		
State	Done	**Effort**	2
Beschreibung			

Diverse kleinere UX - Improvements:

- Tab configuration umbenennen zu "Projektauswahl" und Button: "hinzufügen"
- Aufgabenliste: Statusfarbe "offen": Text und Icon sollten immer weiss sein
- Tooltip Typo: Lineweight 400 anstatt 300
- Überschrift "Aufgaben", Seitenmenü, Filter in Lineweight 600
- Titel: Projekttitel ergänzen (siehe Mockups)
- Default: Organisation als Sichtbarkeit eingestellt
- Wenn das Seitenmenü aufgerufen wird, sollte das ausgewählte Listenelement im Hoverstate bleiben
- Sheet
    - fixe Breite Seitenmenü von 540px
    - Überschriften im Seitenmenü werden nur angezeigt, wenn Inhalte vorhanden sind (ausser Beschreibung und Details bis und mit Sichtbarkeit)
    - Tooltip Verteiler horizontal angeordnet
    - Titel: Markierung ist eingerückt im Vergleich zu den anderen Inhalten
    - Spacing im Seitenmenü überprüfen Titel/Inhalte/Trennlinien
    - Titel der Aufgaben werden mit "..." abgekürzt im Seitenmenü und die Aufgabennummer wird angezeigt
    - Icons im Seitenmenü in 24x24px grösse
    - wenn lange Beschreibungstexte nicht unterbrochen werden, müssen sie im Seitenmenü trotzdem umgebrochen werden
- Filter: Button "Sichtbarkeit" Aufteilung halb/halb
- Dark-Mode:
    - Hintergrund Tab in Farbe: #1F1F1F
    - Hoverfarbe Listenelement: #292929
    - Auswahl-Liste, Trennlinie: #949494
    - Buttonfarbe Schrift sollte weiss sein

PRODUCT BACKLOG ITEM 151			
151	Filter UX Improvements		
State	Done	**Effort**	1
Beschreibung			
Filterverhalten: Tags rutschen eine komplette reihe nach unten, wenn die Zeile neben dem Filter Button voll wird. Filter sollten nicht «Überlaufen», overflow ellipsis.			

BUG 152			
152	Filter Gruppierung entfernen		
State	Done	**Effort**	2
Beschreibung			
Jede Filteroption soll als einzelner Chip dargestellt werden, so dass einzelne Optionen wieder entfernbar sind und nicht nur jeweils die komplette Gruppe (Bsp. Organisation)			

PRODUCT BACKLOG ITEM 155			
155	Tab Scrolling		
State	Done	**Effort**	1
Beschreibung			
Tab Inhalte sollten Scrolling haben, wobei der Header fixiert bleiben soll.			

PRODUCT BACKLOG ITEM 156			
156	Tabs Spacing von Unten		
State	Approved	**Effort**	1
Beschreibung			
Tab Inhalte sollten am Ende der Seite ebenfalls ein Spacing von Unten haben.			

PRODUCT BACKLOG ITEM 157			
157	Erster Tab Titel und Spacing der Inhalte sollten auf einer Line sein		
State	Approved	**Effort**	1
Beschreibung			
Alignment der Texte in den Aufgabendetails mit dem ersten Tab Titel.			

PRODUCT BACKLOG ITEM 159			
159	Empty States für Unteraufgaben Titel und Avatar		
State	Approved	**Effort**	1
Beschreibung			
Wenn eine Unteraufgabe kein Titel oder Avatar hat, soll ein Platzhalter statt des Werts angezeigt werden, da dies für einen Benutzer ansonsten sehr verwirrend sein kann.			

PRODUCT BACKLOG ITEM 160			
160	Banner für Elternaufgabe komplett Anklickbar		
State	Approved	**Effort**	1
Beschreibung			
Der Banner für die Elternaufgabe sollte komplett anklickbar sein. Es würde auch Sinn machen, wenn noch auf eine zusätzliche Art die Interaktionsmöglichkeit mit dem Element dargestellt werden kann.			

BUG 166			
166	Person verschwindet aus der List nach Filterung und Laden der Details		
State	Done	**Effort**	1
Beschreibung			
1. Suche nach einer Person, welche nur als "Participant" in der Aufgabe vorkommt. 2. Auswählen der Aufgabe. 3. Die Aufgabe verschwindet aus der Liste.			

Feature «Dokumentation»

Enthält alle notwendigen Arbeiten für die Abgabe des Projektberichts, sowie für die Übergabe an smino.

PRODUCT BACKLOG ITEM 92			
92	Dokumentation zum lokalen Ausführen der Anwendung		
State	Done	**Effort**	1
Beschreibung			
Einfache Dokumentation/Hinweise, wie die Anwendung lokal gestartet werden kann.			

PRODUCT BACKLOG ITEM 145			
145	Dokumentation Architektur Teams Anwendung		
State	Done	**Effort**	2
Beschreibung			
Einfache Dokumentation/Hinweise, wie die Anwendung lokal gestartet werden kann.			

PRODUCT BACKLOG ITEM 149			
149	Dokumentation "Technical Debt"		
State	Done	**Effort**	2
Beschreibung			
Dokumentation von «Technical Debt» oder offenen Punkte für die Übergabe an smino.			

PRODUCT BACKLOG ITEM 161			
161	Projektbericht		
State	Done	**Effort**	13
Beschreibung			
Dokumentation des Projekts in Form eines Projektberichts.			

PRODUCT BACKLOG ITEM 163			
163	Projektplan		
State	Done	**Effort**	3
Beschreibung			
Dokumentation des Projektplans.			

PRODUCT BACKLOG ITEM 164			
164	Abstract		
State	Done	**Effort**	2
Beschreibung			
Erfassung des Abstracts für die Projektarbeit.			

PRODUCT BACKLOG ITEM 165			
165	Poster		
State	Done	**Effort**	2
Beschreibung			
Erstellung des Posters für die Ausstellung der Arbeit.			

10. Projektrisiken

Risiko	Beschreibung	Behebungsstrategie	Prio*
Technologieentscheidung	MS Teams Applikationen können diverse Technologien verwenden. Dabei sind gewisse Technologien besser unterstützt/integriert als andere. Je nach Wahl der Technologie kann ein enormer Mehraufwand bestehen, wobei auch weitere Einarbeitungsarbeiten nötig sein könnten.	Frühe Evaluation der Technologien und Entscheid	1
Authentifizierung / Autorisierung	Authentifizierung ist ein zentraler Teil, um mit externen APIs kommunizieren zu können. Zudem kann dies auch sehr fehleranfällig sein und eine korrekte Implementierung viel Zeit konsumieren.	Authentifizierung muss so früh als möglich im Prototyp implementiert werden und mit der konkreten Authentifizierungsplattform von smino getestet werden.	1
The big Microsoft Platform	Die Microsoft Plattform ist sehr breit. Zum Bereitstellen einer MS Teams Applikation sind diverse Komponenten relevant wie beispielsweise Hosting der Web-Applikation, der MS Teams Store oder die O365 Plattform.	Es muss frühzeitig beachtet werden, dass die Makroarchitektur eine grosse Relevanz hat. Dabei ist eine grobe Einarbeitung in die notwendige Komponente von Nöten. Zusätzlich sollte dies auch in einer Entwicklerdokumentation festgehalten werden.	2
Zugang zu smino internen Systemen und Test	Um Zugänge zu erhalten, kann es etwas Zeit brauchen. Wenn ein Zugang nicht funktioniert, wird dies vielleicht erst während dem Entwickeln erkannt und führt zu Blockaden.	Zugänge sollten frühzeitig beantragt und getestet werden.	2
Integrationsarten in MS Teams	Teams bietet diverse Arten an, wie "Content" in die Applikation integriert werden kann. Dabei sind aus aktuellem Wissensstand nur wenig Richtlinien	In einem ersten Schritt sollte eine Implementation so einfach wie möglich gestaltet werden, um schnell zu einem brauchbaren Resultat zu kommen. Auf	2

Risiko	Beschreibung	Behebungsstrategie	Prio*
	bekannt. Eine Evaluation der diversen Arten kann viel Zeit in Anspruch nehmen.	dieser Lösung kann dann aufgebaut werden.	
Neuigkeit der Technologie	Die Entwicklung mit Teams ist noch in einem sehr frühen Stadium, wobei die Dokumentation weit nicht alle Anwendungsfälle abdeckt und mehrheitlich den "geradeaus Fall" behandeln.	Frühe Evaluation von bestehenden Applikationen in Teams. Funktionen sollten immer konkret anhand einer prototypischen Implementierung auf ihre Machbarkeit geprüft werden.	3
Arbeitstätigkeit der Entwickler	Neben der Bachelorarbeit sind die Entwickler beruflich beschäftigt, wobei diese auch nicht unbedingt an den selbst Tagen arbeiten. Dies kann zu einer erschwerten Abstimmung oder auch zu einem Zeitdruck führen.	Meetings und Arbeiten sollten so detailliert als möglich geplant und abgestimmt werden. Auch ist es wichtig, dass Arbeiten dokumentiert werden und die wöchentlichen Arbeitszeiten genau eingehalten werden.	5
Verfügbarkeit der smino API	Die smino API kann aufgrund von Ausfällen oder Wartungsarbeiten nicht erreichbar sein. Dadurch kann eine allfällige Weiterentwicklung erschwert bis unmöglich sein.	Anfordern einer Testumgebung, welche explizit für dieses Projekt verwendet wird. Frühzeitiges kommunizieren von Wartungsarbeiten Seiten smino.	5

Anhang Tabelle 2: Projektrisiken (eigene Darstellung)

*Die Priorisierung wird in einer Skala von 1 – 5 angegeben. Dabei steht die 1 für die kleinste Priorität und 5 für die höchste.

11. Coverage Report

File		Statements		Branches		Functions		Lines	
access-token-provider.ts		100%	1/1	100%	0/0	100%	0/0	100%	1/1
access-token.interceptor.ts		100%	20/20	100%	6/6	100%	5/5	100%	18/18
api-url-provider.ts		100%	1/1	100%	0/0	100%	0/0	100%	1/1
api-url.interceptor.ts		100%	22/22	100%	4/4	100%	5/5	100%	20/20
user-id-provider.ts		100%	1/1	100%	0/0	100%	0/0	100%	1/1
user-id.interceptor.ts		100%	16/16	100%	6/6	100%	4/4	100%	14/14

Anhang Abbildung 47: Relevante Test-Coverage für @smino/api

File		Statements		Branches		Functions		Lines	
issues-core		100%	30/30	100%	0/0	100%	0/0	100%	28/28
issues-core/services		43.29%	71/164	38.66%	29/75	16.32%	8/49	40.9%	63/154
smino-components/filter		98.08%	308/314	87.69%	57/65	100%	95/95	97.86%	275/281
smino-components/filter/components		95.51%	213/223	90.47%	76/84	88.88%	48/54	94.87%	185/195
smino-components/internationalization		100%	16/16	100%	0/0	100%	1/1	100%	12/12
smino-components/internationalization/directives		100%	14/14	100%	0/0	100%	4/4	100%	12/12
smino-components/internationalization/pipes		100%	17/17	100%	3/3	100%	4/4	100%	15/15
smino-components/internationalization/services		87.03%	47/54	80%	20/25	92.3%	12/13	87.75%	43/49
smino-components/internationalization/store		100%	7/7	100%	0/0	100%	0/0	100%	5/5
smino-components/internationalization/store/translations		95%	76/80	88.88%	8/9	88%	22/25	94.44%	68/72
smino-components/internationalization/testing		86.95%	20/23	100%	4/4	33.33%	1/3	82.35%	14/17
smino-components/sheet		100%	9/9	100%	0/0	100%	0/0	100%	7/7
smino-components/sheet/components/sheet		100%	30/30	100%	5/5	100%	13/13	100%	28/28
smino-components/sheet/components/sheet-container		93.75%	15/16	33.33%	2/6	87.5%	7/8	92.85%	13/14
smino-components/sheet/components/sheet-content		100%	4/4	100%	0/0	100%	0/0	100%	2/2
smino-components/sort		100%	10/10	100%	0/0	100%	0/0	100%	8/8
smino-components/sort/components		97.82%	45/46	94.73%	54/57	100%	13/13	97.67%	42/43
smino-components/tabs		100%	14/14	100%	0/0	100%	0/0	100%	12/12
smino-components/tabs/components/tab-body		100%	30/30	100%	10/10	100%	6/6	100%	28/28
smino-components/tabs/components/tabs-header		100%	9/9	50%	1/2	100%	1/1	100%	7/7
smino-components/tabs/containers/tab		100%	12/12	100%	0/0	100%	3/3	100%	10/10
smino-components/tabs/containers/tabs		80%	16/20	50%	2/4	85.71%	6/7	77.77%	14/18
smino-components/tabs/directives		100%	27/27	100%	0/0	100%	8/8	100%	23/23

Anhang Abbildung 48: Relevante Test-Coverage
**issues-core/serivces besteht aus diversen übernommenen services*

issue-policy.service.ts		100%	43/43	96.66%	29/30	100%	8/8	100%	41/41

Anhang Abbildung 49: Relevante Test-Coverage für issues-core/services

12. Lines of Code

Language	files	blank	comment	code
TypeScript	479	3113	379	18713
SCSS	204	2192	204	11323
HTML	76	124	0	1865
JSON	39	0	0	1167
JavaScript	7	6	8	315
YAML	4	58	10	259
Markdown	3	45	0	84
CSS	2	2	17	4
SUM:	814	5540	618	33730

Anhang Abbildung 50: Lines of Code Gesamt (eigene Darstellung)

Language	files	blank	comment	code
SCSS	188	2145	203	11074
TypeScript	143	1021	238	6823
JSON	33	0	0	1102
HTML	31	76	0	873
JavaScript	5	6	7	269
YAML	4	58	10	259
Markdown	2	31	0	39
CSS	2	2	17	4
SUM:	408	3339	475	20443

Anhang Abbildung 51: Lines of Code ohne smino-components und @smino/api (eigene Darstellung)

13. Deployment Vorbereitungen

Development Tenant Einrichten

Um die App einfach lokal zu testen, sollte ein Development Tenant eingerichtet werden. In dem Tenant können dann entsprechende Testnutzer angelegt werden oder externe Nutzer als Gäste berechtigt werden. Weitere Informationen:

- Developer Program | Microsoft 365 Dev Center[23]
- Prepare your Microsoft 365 tenant - Teams | Microsoft Docs[24]

[23] https://developer.microsoft.com/en-us/microsoft-365/dev-program
[24] https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/build-and-test/prepare-your-o365-tenant

M365 Account einrichten/konfigurieren

Für das Deployment wird ein M365 Account benötigt, welcher die entsprechenden Rechte besitzt, um eine "Custom App" zu publizieren. Da die Pipeline ohne Nutzerinteraktion läuft, ist es wichtig zu beachten, dass für den Benutzer keine Multi-Faktor Authentisierung erforderlich sein darf. Ansonsten wird der Login fehlschlagen.

Azure Service Principal

Um von Azure DevOps zu deployen ist ein Service Principal notwendig, welches die nötigen Rechte hat, um Ressourcen zu erstellen oder zu löschen.

Dies kann über das Portal oder über PowerShell mit folgendem Command erstellt werden:

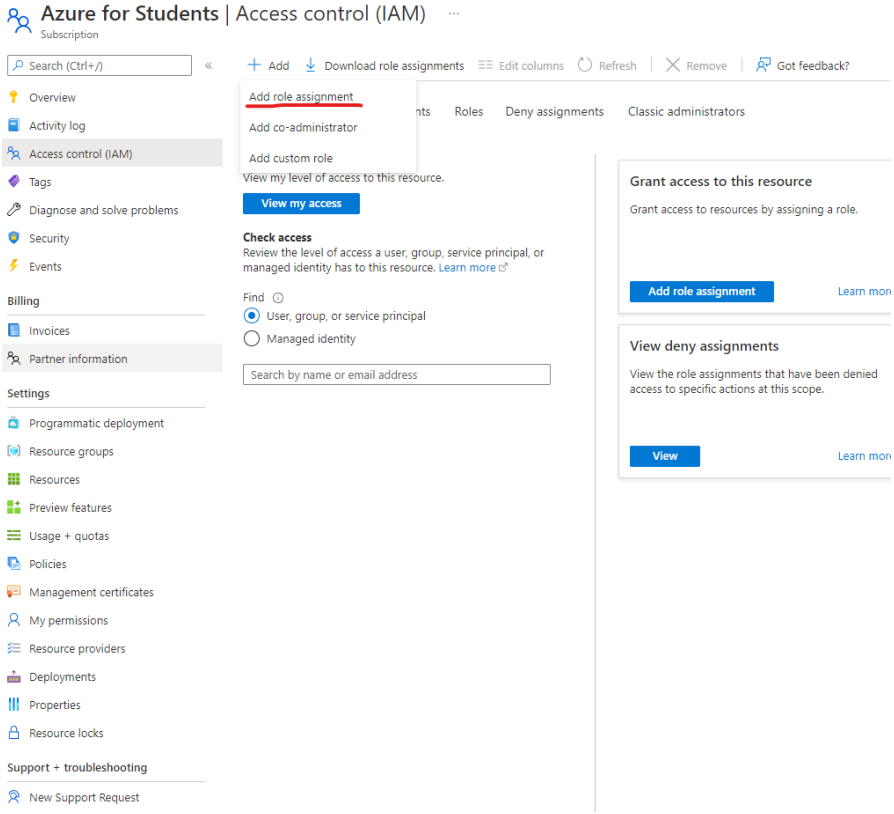
```
$sp = New-AzADServicePrincipal -DisplayName "<<DisplayName>>"
```

Dabei wird standardmässig bereits ein Passwort für das Service Principal erstellt (über $sp.PasswordCredentials ersichtlich).

In einem nächsten Schritt muss das Service Principal berechtigt werden. Hierfür stehen zwei Optionen zur Verfügung, abhängig davon, ob die Ressourcengruppe automatisch angelegt werden soll. Wenn dies automatisch bei der Provisionierung geschehen soll, muss das Service Principal für die gesamte Subscription für die Erstellung von Ressourcen berechtigt werden. Wenn die Ressourcengruppen vorzu manuell angelegt werden, kann das Service Principal auf der Ressourcengruppe berechtigt werden.

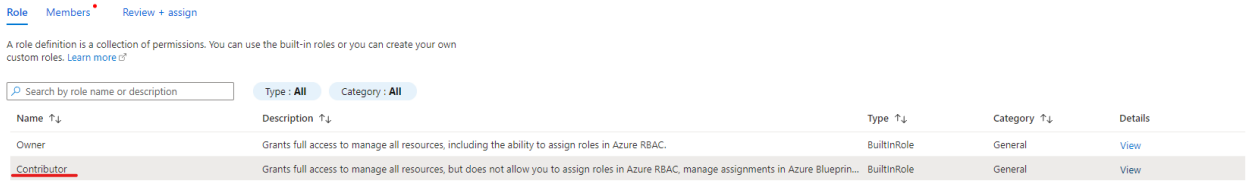
Nachfolgend ist gezeigt, wie das Principal für die Subscription berechtigt werden kann. Dies funktioniert für die Ressourcengruppe analog.

Role Assignment für Subscription



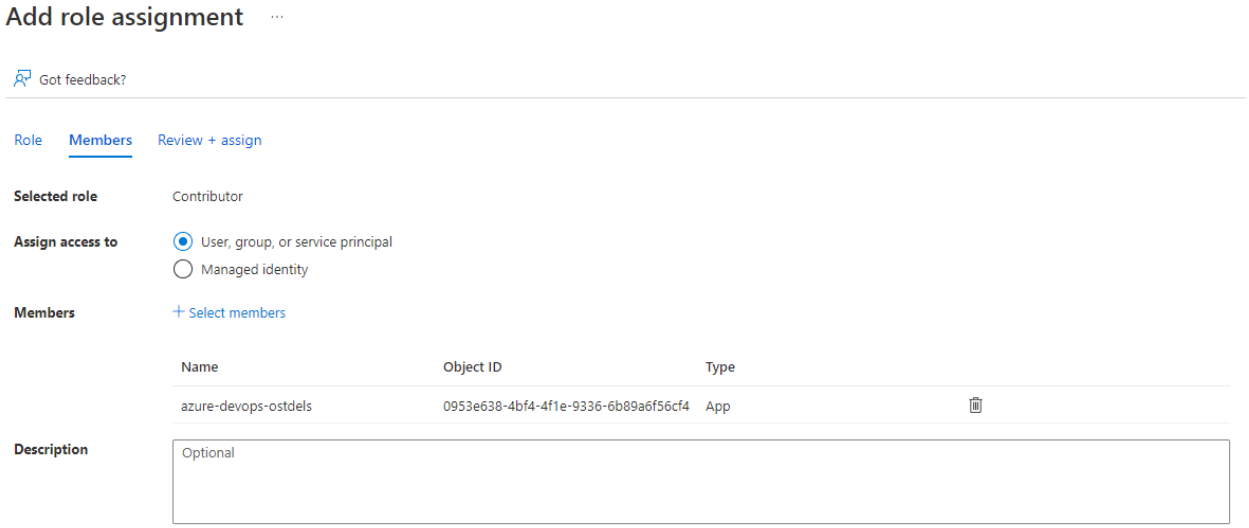
Anhang Abbildung 52: Role Assignments Azure (eigene Darstellung)

Contributor Role



Anhang Abbildung 53: Contributor Role Azure (eigene Darstellung)

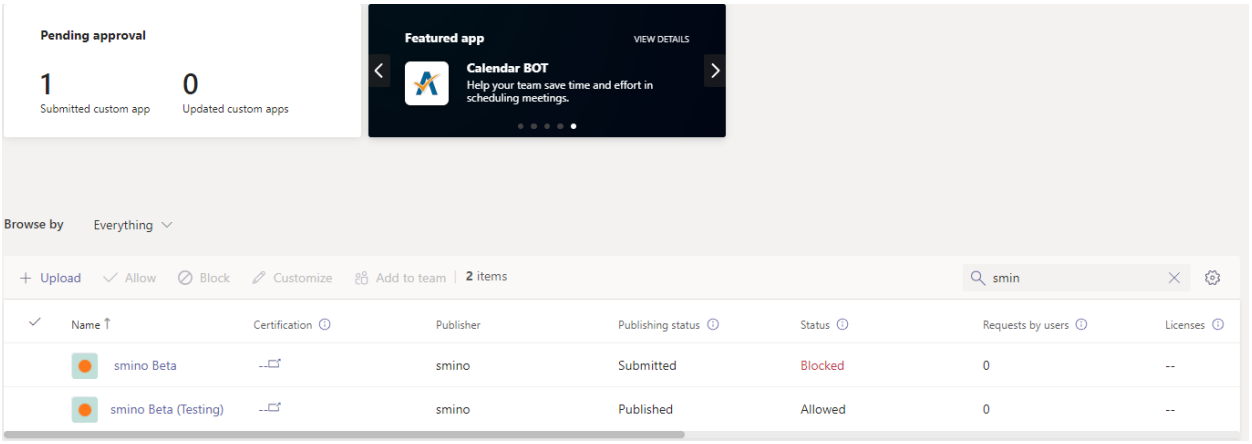
Service Principal hinzufügen



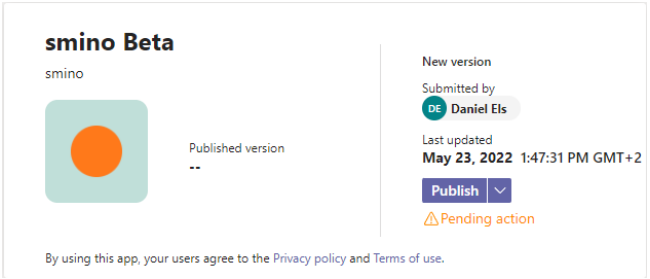
Anhang Abbildung 54: User zu Rolle hinzufügen Azure (eigene Darstellung)

Teams App Management

Die Apps für eine Organisation können im Teams Admin Center[25] verwaltet werden. Wenn eine "Custom App" initial eingereicht wird, muss diese zuerst von einem Admin freigegeben werden.



Anhang Abbildung 55: Teams Admin Center Apps (eigene Darstellung)



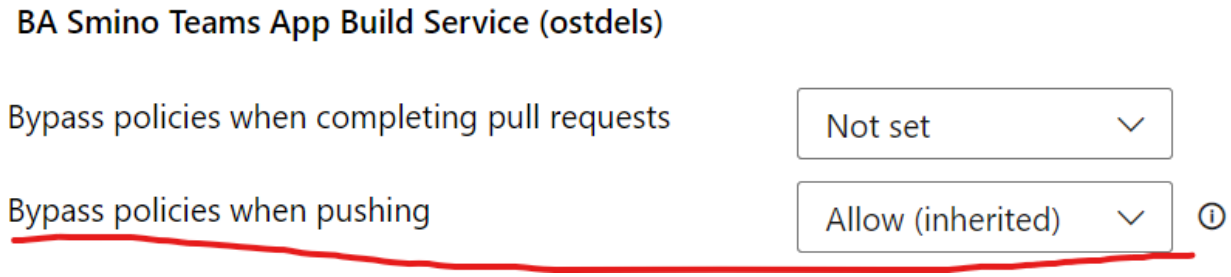
Anhang Abbildung 56: Teams Admin Center Publish (eigene Darstellung)

[25] https://admin.teams.microsoft.com/policies/manage-apps

Konfiguration für Commits von Pipelines

Damit von der Pipeline aus Code eingecheckt werden kann, muss der entsprechende DevOps Service Account für die Repositories berechtigt werden[26].

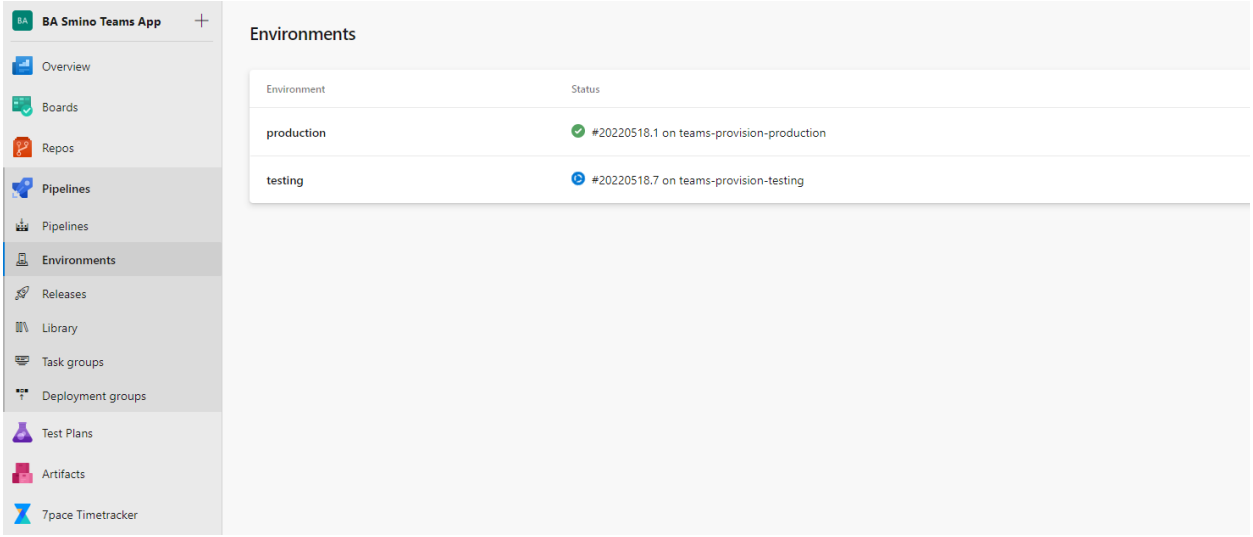
Falls der main Branch durch eine Policy geschützt ist und direkt auf dem main Branch provisioniert wird, muss zusätzlich sichergestellt werden, dass der Service Account die Policies umgehen kann.



Anhang Abbildung 57: Bypass policies DevOps (eigene Darstellung)

[26] https://docs.microsoft.com/en-us/azure/azure-portal/get-subscription-tenant-id

14. Pipelines



Anhang Abbildung 58: DevOps Environments (eigene Darstellung)

Für das Deployment müssen zwei Environments erstellt werden. Die Environments selbst müssen keinen Inhalt haben und können als "Leer" erstellt werden. Ein Environment kann für Approvals genutzt werden.

CI/CD

Für CI und CD der Teams App existiert eine eine "teams-ci-cd" Pipeline. Die Pipeline kann für Pull Requests verwendet werden, um die integrität des Codes sicherzustellen. Das effektive Deployment wird nur auf dem "main" Branch durchgeführt.

Steps

- ci
    - Install Node.js: Installiert Node Tooling mit Version 14
    - npm ci tabs: npm ci im "tabs" Ordner ausführen → Dependencies installieren
    - npm run build: Buildet alle "Projekte"
    - npm lint all: Führt den Linter für alle "Projekte" aus
    - npm test: Führt die Tests für alle "Projekte" aus
    - PublishTestResults: Publiziert die Testresultate als JUnit Report als Artefakt der Pipeline
    - PublishCodeCoverageResults: Publiziert Code-Coverage Resultat als Artefakt der Pipeline
- deploy_teams_app_testing: Deployment der Teams App für das testing Environment
    - Install Node.js: Installiert Node Tooling mit Version 14
    - npm ci: npm ci im Projekt-Root ausführen → Dependencies installieren
    - npm ci tabs: npm ci im "tabs" Ordner ausführen → Dependencies installieren
    - replacetokens: Ersetzt die Tokens für die App-Settings im "Tabs" Projekt anhand der vorhandenen Pipeline Variablen
    - Deploy Teams App: Provisioniert die Applikation auf Azure für das entsprechende Environment
- deploy_teams_app_production: Deployment der Teams App für das production Environment
    - Schritte analog zu deploy_teams_app_testing

Variablen

Name	Beschreibung	Wert	Secret
AZURE_SERVICE_PRINCIPAL_NAME	Nutzername für das Azure Service Principal		N
AZURE_SERVICE_PRINCIPAL_PASSWORD	Passwort für das Service Principal		Y
AZURE_TENANT_ID	Tenant Id vom gewünschten Azure Mandant		N
Production.ApiUrl	smino API Url	https://api.smino.ch/	N
Production.AppUrl	smino App Url	https://app.smino.ch/	N
Productrion.TranslationApiUrl	Url für den Translation Service	https://lang.smino.ch/	N
Production.Auth.Authority	Authority für den OIDC Client	https://api.smino.ch/	N
Production.Auth.ClientId	Client Id für den OIDC Client	-	N
Testing.ApiUrl	smino API Url	https://api.smino.ninja/	N
Testing.AppUrl	smino App Url	https://app.smino.ninja/	N
Testing.TranslationApiUrl	Url für den Translation Service	https://lang.smino.ninja/	N
Testing.Auth.Authority	Authority für den OIDC Client	https://api.smino.ninja/	N
Testing.Auth.ClientId	Client Id für den OIDC Client	-	N

Anhang Tabelle 3: CI / CD Variablen (eigene Darstellung)

Provisionierung

Für das Provisioning existieren zwei Pipelines für testing und production. Die Pipelines sind separat gehalten, da für den State ein Commit erzeugt wird, um die neuen States in der Versionskontrolle zu aktualisieren.

Steps

- Install Node.js: Installiert Node Tooling mit Version 14
- npm ci: npm ci im Projekt-Root ausführen → Dependencies installieren
- Provision Teams app: Provisioniert die nötigen Ressourcen für die Teams Applikation für das entsprechende Environment und erzeugt einen Commit den aktualisierten States

Variablen

Name	Beschreibung	Wert	Secret
AZURE_SERVICE_PRINCIPAL_NAME	Nutzername für das Azure Service Principal	-	N
AZURE_SERVICE_PRINCIPAL_PASSWORD	Passwort für das Service Principal	-	Y
AZURE_TENANT_ID	Tenant Id vom gewünschten Azure Mandant	-	N
AZURE_SUBSCRIPTION_ID	Id der zu verwenden Azure Subscription	-	N
M365_ACCOUNT_NAME	Benutzername für den M365 Account	-	N
M365_ACCOUNT_PASSWORD	Passwort für den M365 Account	-	Y
M365_TENANT_ID	Tenant Id für den M365 Mandant	-	N

Anhang Tabelle 4: Provisionierung Variablen (eigene Darstellung)

Teams Publish

Die Teams Publish Pipeline publiziert das Teams App Package in den angegeben Organization Store.

Steps

- publish_testing
    - Install Node.js: Installiert Node Tooling mit Version 14
    - npm ci: npm ci im Projekt-Root ausführen → Dependencies installieren
    - npm ci tabs: npm ci im "tabs" Ordner ausführen → Dependencies installieren
    - Package teams app: Erstellt das Teams App Package
    - PublishPipelineArtifact: Publiziert das App Package als Artefakt der Pipeline
    - Publish teams app: Publiziert das App Apckage in den angegeben Organization Store
- publish_production
    - Analog zu publish_testing

Variablen

Name	Beschreibung	Wert	Secret
M365_ACCOUNT_NAME	Benutzername für den M365 Account	-	N
M365_ACCOUNT_PASSWORD	Passwort für den M365 Account	-	Y
M365_TENANT_ID	Tenant Id für den M365 Mandant	-	N

Anhang Tabelle 5: Teams Publish Variablen (eigene Darstellung)

15. Offene Punkte, Verbesserungen und Known-Issues

Chat Page mit Commands erstellen

Für die Applikation selbst soll eine Chat-Page erstellt werden, welche es erlaubt sich mit seinem smino Account anzumelden oder abzumelden. Zusätzlich soll auch ein Command für "Hilfe" erstellt werden, welche mögliche Interaktionen aufzeigen soll.

Link Unfurling für Aufgaben

Es wäre sinnvoll, wenn in Teams für einen Link zu einer smino Aufgabe entsprechende Informationen direkt angezeigt werden. Dies würde eine einfache Diskussion über Inhalte ermöglichen und dem Nutzer direkt einige Informationen zur Aufgabe anzusehen.

Tab Konversationen

Einem Nutzer soll es ermöglicht werden in Teams mit den Konversationen einer Aufgabe zu interagieren. Dafür müsste definiert werden, wie dies in der Teams Applikation aussehen könnte oder welche weiteren Integrationsmöglichkeiten seitens MS Teams dafür sinnvoll wären.

Tokens für Collaborator in Storage vermerken

Momentan wird das Token für den Collaborator jedes Mal (bei jedem neuen Aufruf des Tabs) erneut geholt, selbst wenn noch ein gültiges Token existieren würde. Das Token für den Collaborator sollte daher im Storage abgelegt werden, um es wiederzuverwenden.

Separater Signin Callback

Für den OIDC Signin Callback sollte eine separate Route verwendet werden. Dies ermöglicht es einerseits einfacher zu prüfen oder zu erwarten, ob ein Callback stattfinden müsste. Andererseits vermeidet es Probleme mit dem Routing, bzw. erlaubt eine einfache Darstellung eines Spinners.

CDK Virtual Scrolling

Die aktuell verwendete Library für das Virtual Scrolling wird nicht mehr aktiv gewartet. Im besten Fall könnte man für das Virtual Scrolling @angular/cdk verwenden. Einige Features wie die dynamische Grösse der Items oder die Definition eines alternativ «Scroll»-Elements wird aber erst in kommenden Versionen produktiv verfügbar.

Localization (Formate)

Für die verschiedenen Cultures soll es möglich sein, dynamisch in Angular die Locale zu ändern, um Culture-Sensitive Werte im korrekten Format anzuzeigen.

Die Registrierung der Locale und das Laden der entsprechenden Formate ist in Angular relativ starr, wobei fast ein Reload der Anwendung nötig wäre. Es müsste geprüft werden, ob es möglich ist, dynamisch das Format zu ändern oder ob auf eigene Pipes für die Formatierung gesetzt werden soll.

Aufgrund der niedrigen Priorität seitens smino und aus Zeitgründen konnte dieses Feature nicht in der Arbeit umgesetzt werden.

UI Überarbeitungen Tabs

Tabs Spacing von Unten

Tab Inhalte sollten am Ende der Seite ebenfalls ein Spacing von Unten haben.

Erster Tab Titel und Spacing der Inhalte sollten auf einer Line sein

Alignment der Texte in den Aufgabendetails mit dem ersten Tab Titel.

Empty States für Unteraufgaben Titel und Avatar

Wenn eine Unteraufgabe kein Titel oder Avatar hat, soll ein Platzhalter statt des Werts angezeigt werden, da dies für einen Benutzer ansonsten sehr verwirrend sein kann.

Banner für Elternaufgabe komplett Anklickbar

Der Banner für die Elternaufgabe sollte komplett anklickbar sein. Es würde auch Sinn machen, wenn noch auf eine zusätzliche Art die Interaktionsmöglichkeit mit dem Element dargestellt werden kann.

Synchronisation der Aufgaben

Es kann passieren, dass ein Nutzer veraltete Aufgabendaten sieht, wenn er eine lange Zeit im Tab der Aufgaben verbringt, ohne das Tab zu wechseln. Einerseits haben wir das Problem temporär behoben, indem wir einen Refresh Button eingebaut haben, der die komplette Liste erneut lädt. Andererseits laden wir die Aufgabendetails beim Öffnen einer Aufgabe oder deren Unteraufgaben immer neu. Da für die mobile Applikation bereits ein Synchronisationsendpunkt erstellt wurde, könnte man sich überlegen, diesen auch für die Teams Applikation zu verwenden.

Übersetzungen für die Datepicker und Batch Translations

Für den Datepicker werden aktuell die Optionen noch nicht übersetzt. Dafür müsste zuerst auch noch die Implementierung für das Übersetzen von Batches implementiert/integriert werden.

Erweiterung der Tests

Es sollten noch weitere Unit-Tests für die Komponenten erstellt werden. Insbesondere würde es sich lohnen, zentrale Funktionalitäten innerhalb der smino-components zu testen, da diese sicher an vielen Orten verwendet werden.

Ebenso sollten für die Issues die Tests für die Services und Facades erweitert werden.

Unabhängigkeit der smino-components

Momentan existieren noch zwei Module innerhalb der smino-components, welche eine Abhängigkeit zur App selbst haben. Nämlich das InternationalizationModule und das ModalModule. Beide haben eine Abhängigkeit zum Store der Applikation selbst. Wenn diese Abhängigkeiten entfernt werden, könnten alle Module und Komponenten ausgelagert und potenziell paketiert werden.

Einbindung von cypress in CI Pipeline

Momentan wird cypress nicht automatisch mit der Continuous Integration Pipeline ausgeführt.