htlperg
innovativ | kreativ | praxisnah

Catrin Extended - Firmeninternes Sitzplatzfindungssystem von Cloudflight

DIPLOMARBEIT

Höhere Abteilung für Informatik

01/07/2025 - 26/03/2026

Projektmitglieder: Fiona Houdek
Lea Joppich

Betreuerin: DI (FH) Gudrun Heinzelreiter-Wallner, MSc, BEd

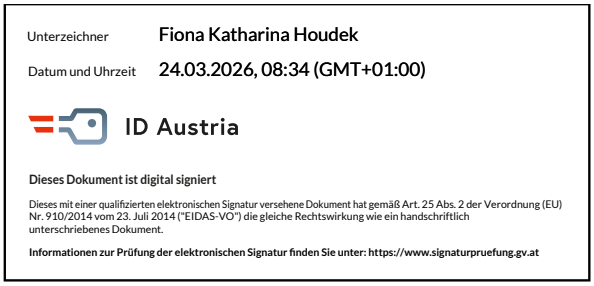
Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Bei der Erstellung der Arbeit haben wir keine generativen KI-Tools verwendet.

Perg, 24.03.2026

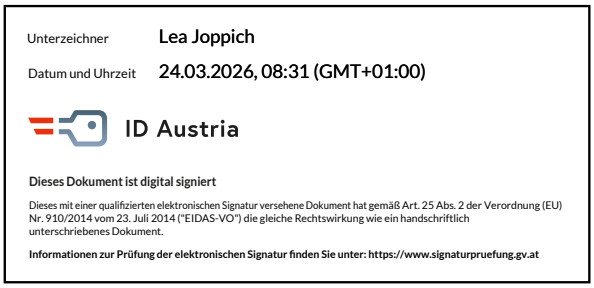
Ort, Datum



Fiona Houdek

Perg, 24.03.2026

Ort, Datum



Lea Joppich

Danksagung

Wir möchten uns bei sämtlichen Personen der HTL Perg sowie von Cloudflight bedanken, die diese Diplomarbeit möglich gemacht haben.

Insbesondere möchten wir uns bei unserer Diplomarbeitsbetreuerin DI (FH) Gudrun Heinzelreiter-Wallner, MSc, BEd bedanken, die uns zeitintensiv beim Schreiben dieser Arbeit unterstützt hat.

Ein großes Dankeschön auch an unsere Mentorin Pia Gerhofer und unseren Mentor Stefan Niederhumer von Cloudflight, sowie an Stefan Neumüller, der uns mit seinem umfassenden Know-How über das Catrin System bei der Entwicklung tatkräftig betreut hat.

Außerdem möchten wir uns bei unseren Eltern bedanken, die uns in unserer gesamten Schulzeit den Rücken gestärkt und uns vor allem mental unterstützt und motiviert haben.

Abstract

Catrin is a virtual seating assignment system, that simplifies the daily work routine for Cloudflight employees. The focus of Catrin Extended is on actively involving users in the process of enhancing the system. In a recent survey, users were given the opportunity to suggest new features or improvements they would like to have. The most frequently mentioned suggestions were then implemented. These include an expansion of the seat reservation feature, which now allows user to book a seat for a specific period of time, two-way Teams communication, and the display of an employee's availibility status on Catrin.

To achieve this goal and implement these enhancements, the existing code of the Catrin system will be updated and expanded. Catrin is designed as a web application and is divided into several different functional fields. A central component is the Catrin Client, which is implemented in Angular and defines the user interface. In this thesis, this user interface is extended to include the new features. The focus is on a clear and self-explanatory design, to enhance the usabilty. The Catrin Server is implemented in Spring Boot using Kotlin and is responsible for accessing the PostgreSQL database. The Backend also includes a C++/Kasvot system that integrates a machine learning model for facial recognition. While that is not essential for this thesis, it is helpful for understanding the system and is therefore described in the practical section. Through this expansion and adaptation, the application has gained significant added value, as the new features were developed with user-friendliness in mind and tailored to the needs of the employees.

Kurzfassung

Catrin ist ein virtuelles Sitzplatzfindungssystem und vereinfacht den Arbeitsalltag von Mitarbeiterinnen und Mitarbeitern von Cloudflight. Der Fokus bei Catrin Extended liegt darauf, User aktiv in den Prozess der Erweiterung einzubinden. In einer durchgeführten Umfrage haben Benutzerinnen und Benutzer die Möglichkeit erhalten, neue Features oder Optimierungen vorzuschlagen, die sie sich wünschen. Ziel ist es, die am häufigsten genannten Ideen umzusetzen. Darunter fallen die Features, einen Sitzplatz für einen Zeitraum buchen zu können, eine bidirektionale Teams-Kommunikation zu ermöglichen sowie den Status von Mitarbeitenden im Frontend anzuzeigen.

Um dieses Ziel zu erreichen und diese Erweiterungen zu verwirklichen, wird der bestehende Code des Catrin Systems verwendet und erweitert. Catrin ist als Webanwendung konzipiert und in mehrere funktionale Bereiche gegliedert. Ein zentraler Bestandteil ist der Catrin Client, der in Angular implementiert ist und die Benutzeroberfläche abbildet. Diese Ansicht wird in der vorliegenden Arbeit um die neuen Funktionen, welche auch im Backend implementiert werden, erweitert. Dabei wird der Fokus auf selbsterklärendes User Design und optimierte Usability gelegt. Der Catrin Server ist in Spring Boot mit Kotlin umgesetzt und für jegliche Datenbankzugriffe auf die PostgreSQL-Datenbank zuständig. Hier befindet sich ein C++/Kasvot-System, welches ein Machine Learning Model für die Gesichtserkennung integriert. Für diese Arbeit ist dieses nicht essenziell, jedoch für das Verständnis hilfreich und wird daher im praktischen Teil genauer ausgeführt. Die Anwendung hat durch die Erweiterung und Adaption einen großen Mehrwert erlangt, da die neuen Funktionen benutzerfreundlich entwickelt und auf die Bedürfnisse der User angepasst sind.

Inhaltsverzeichnis

1 Motivation und Zielsetzung

In der vorliegenden Diplomarbeit wird das bestehende Sitzplatzfindungssystem Catrin des Unternehmens Cloudflight Austria GmbH erweitert.

Das Unternehmen entstand laut Cloudflight GmbH [3] im Jahr 2019 durch die Fusionierung der österreichischen Softwareentwicklungsfirma Catalysts und dem deutschen Marktforschungs- und Beratungsunternehmen Crisp Research zur Cloudflight GmbH. Das Unternehmen beschäftigt derzeit rund 800 qualifizierte Mitarbeiterinnen und Mitarbeiter an über 20 Standorten in vier Ländern Europas. Der Hauptsitz befindet sich in München. Cloudflight GmbH bietet keine standardisierten Produkte an, sondern entwickelt individuelle und maßgeschneiderte Softwarelösungen in den Bereichen Digital Commerce, Cloud Services sowie Daten- und KI-Services. Die Kunden des Unternehmens stammen aus verschiedenen Branchen, darunter Luft- und Raumfahrt, Mobilität, Gesundheitswesen, Transport, Handel sowie Produktion.

Im Unternehmen wird ein Sitzplatzfindungssystem namens Catrin eingesetzt. Laut der firmeninternen Dokumentation von Coudflight, hilft dieses Besucherinnen und Besuchern sowie Mitarbeiterinnen und Mitarbeitern dabei, grundlegende Aufgaben wie die Suche nach einer Person oder die Reservierung von Sitzplätzen in einem Büro zu bewältigen. Personen werden erkannt und unterschieden, indem das Videosignal der Webcam kontinuierlich analysiert und durch eine speziell entwickelte Echtzeit-Gesichtserkennungs-Engine geleitet wird.

Das Diplomarbeits-Team, bestehend aus Fiona Houdek und Lea Joppich, hat folgende User Stories für die Erweiterung implementiert:

- *Als Mitarbeiterin oder Mitarbeiter möchte ich einen Sitzplatz gleich für eine ganze Woche reservieren können, damit ich diesen nicht jeden Tag erneut buchen muss.*
- *Als Mitarbeiterin oder Mitarbeiter möchte ich Teams-Benachrichtigungen, welche mittels Catrin stattgefunden haben, beantworten können. Meine Antwort wird via Catrin der benachrichtigenden Person angezeigt.*
- *Als Mitarbeiterin oder Mitarbeiter möchte ich auf dem System Catrin sehen, ob mein Arbeitskollege im Office ist.*

Hierzu hat Fiona Houdek das Backend in SpringBoot mit Kotlin und Lea Joppich das Frontend mit Angular ausgebaut. Dabei liegt der Fokus auf UI/UX, um den Endanwenderinnen und Endanwendern ein System zu bilden, das eine einfache und effiziente Bedienung sicherstellt. Basierend auf einer Erhebung und UI-/UX-Richtlinien wird das grafische und funktionale Konzept umgestaltet. Gemeinsam wurde der bereits bestehende Teams-Bot, welcher eine unidirektionale Kommunikation ermöglicht, um eine bidirektionale Kommunikation erweitert und im System integriert.

2 Theoretische Grundlagen

In der modernen Software-Entwicklung ist eine agile Vorgehensweise eine häufige Form der Projektorganisation. Da das Unternehmen Cloudflight SCRUM als agiles Framework einsetzt, wird dieses nachstehend im Detail beschrieben. Für die weitere Ausführung der Anwendung werden alle zugrunde liegenden Technologien des Backends und Frontends diskutiert.

2.1 SCRUM

Scrum ist laut Rubin [40, S. 47] kein Framework, welches jeden Schritt bereits im vorhinein plant. Es gibt lediglich einen Rahmen für die Organisation und Verwaltung an. Dieser beruht auf Werte, Praktiken und Prinzipien, die in der unterstehenden Tabelle (siehe Tabelle 1) angeführt werden, die Zusammenhänge sind aus Atlassian [1] entnommen. Diese können nicht von Grund auf geändert oder ganz weglassen werden. Sie bilden das Fundament von Scrum. Dadurch entsteht ein sehr flexibeles System, das sich leicht an spezielle Unternehmensstrukturen anpassen lässt.

Prinzipien	**Unterstützende Werte**	**Umsetzende Praktiken**
Transparenz	Offenheit, Ehrlichkeit	Artefakte
Überprüfung	Fokus	Review
Anpassung	Mut, Zusammenarbeit	Retrospektive

Tabelle 1: Zusammenhang von Scrum-Prinzipien , -Werten und -Praktiken

Alle Rollen, Abläufe und Aktivitäten aus den nachfolgenden Absätzen sind aus Rubin [40, S. 48–61] entnommen, sofern keine andere Quelle explizit angegeben ist.

2.1.1 Scrum Rollen

In jeder Umsetzung eines Software-Projekts, in dem als agiles Vorgehensmodell Scrum eingesetzt wird, sind drei fixe Rollen vertreten. Es können aber auch mehr definiert werden, weniger als diese drei sind jedoch nicht möglich.

Der **Product Owner (PO)** trägt die Verantwortung für das Projekt und den Ablauf des Projekts. Dazu gehört die Aufgabe den Backlog zu definieren und zu priorisieren. In dieser Rolle ist das Ziel immer klar vor Augen zu haben und zu gewährleisten, dass das Team dieses erfolgreich erreicht. Die Person, welche diese Rolle inne hat, ist für den Gesamterfolg des Projekts zuständig und muss das auch seinem Team vermitteln. Ein PO arbeitet stets mit dem Team und der ScrumMistress oder dem ScrumMaster zusammen, um ihre Fragen so schnell wie möglich beantworten zu können.

In der Rolle der **ScrumMistress** oder des **ScrumMasters (SM)** wird kontrolliert, ob alle Bestandteile des Scrum-Frameworks und des angepassten Prozesses eingehalten werden. Die ausübende Person dieser Rolle muss mit allen Werten, Prinzipien, Praktiken und Abläufen vertraut sein und steht auch dem Team zur Verfügung, sofern irgendwelche Fragen in diesem Bereich aufkommen. Diese Rolle kann auch als Moderator oder Trainer bezeichnet werden. Eine weitere Aufgabe ist die Förderung der Produktivität des Teams, wie zum Beispiel alle möglichen Hindernisse für das Team aus dem Weg zu schaffen. Die ScrumMistress oder der ScrumMaster hat jedoch keine Autoritäten gegenüber dem Team.

Das **Entwicklungsteam** ist verantwortlich, das zu liefern, was ihnen der PO auftragt. Sie entscheiden selbst, wie sie das umsetzen und was sie dafür benötigen. Die technischen Rollen wie Datenbankadministrator oder Tester werden bei Scrum nicht explizit definiert, aber das Team muss alle technischen Kenntnisse haben, um das Projekt erfolgreich umsetzen zu können. Ein Team besteht meist aus fünf bis neun Entwicklerinnen und Entwicklern. Für ein größeres Projekt ist der beste Weg mehrere Teams zu erstellen, anstatt ein Team mit vielen Mitgliedern zu bestimmen.

2.1.2 Scrum Ablauf

Die Abbildung 1 zeigt grafisch alle wichtigen Scrum Aktivitäten und Scrum Artefakte. Die Abläufe und Zusammenhänge sind anhand eines Zeitstrahls visuell aufbereitet und werden in Folge im Detail beschrieben.

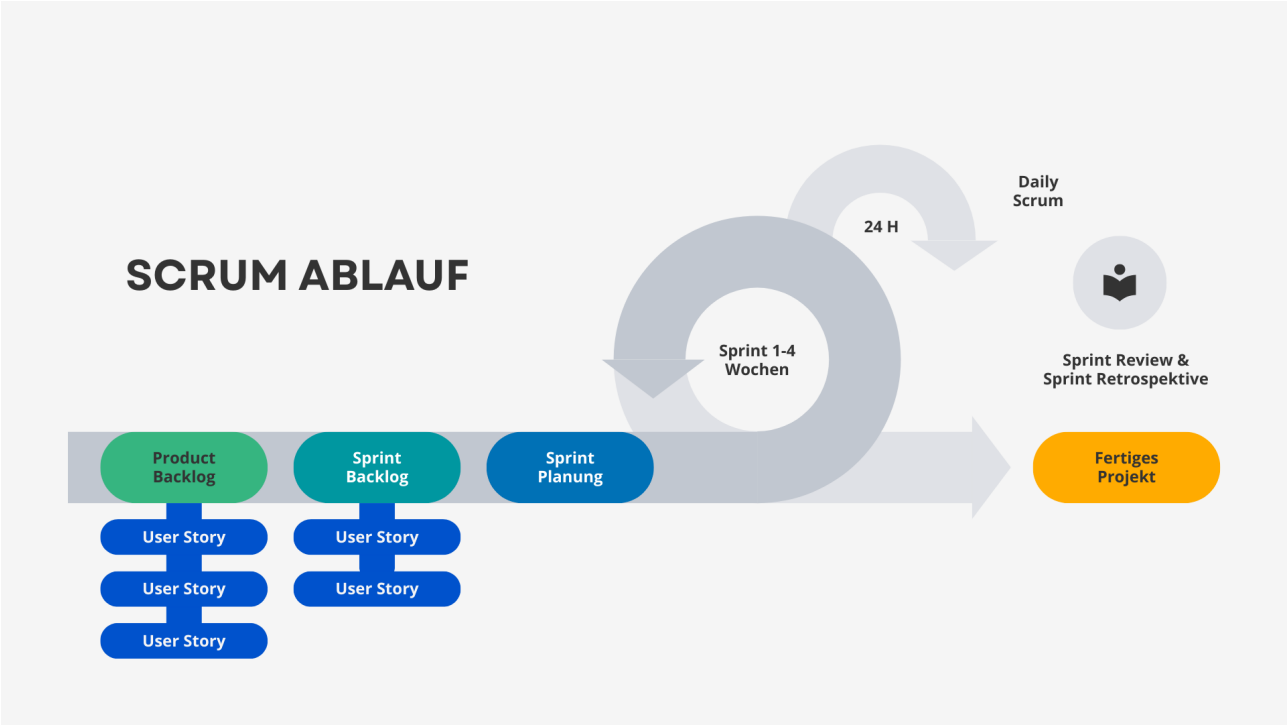


Abbildung 1: Scrum Ablauf (erstellt von Lea Joppich mithilfe des Mediums Canva)

2.1.3 Scrum Aktivitäten

Scrum definiert eine Reihe von Aktivitäten, die kontinuierlich innerhalb des Projekts mehrfach durchlaufen werden.

Die Projektdauer wird unter Scrum in **Sprints** unterteilt. Diese Zyklen dauern maximal einen Monat und die Dauer ist für alle Zyklen immer gleich. Jeder Sprint muss als Ergebnis ein Endprodukt oder ein Inkrement des Endprodukts vorweisen, sodass die Kundin oder der Kunde den kontinuierlichen Fortschritt sehen kann. Während eines laufenden Sprints dürfen sich weder Umfang noch Mitarbeitende ändern. Für jede Iteration wird ein Sprint Backlog definiert. Es werden die Aktivitäten Sprint-Planung, Sprint Review und Sprint Retrospektive genau einmal durchgeführt.

Als erste Aufgabe in einem neuen Sprint, steht die **Sprint-Planung** an. Dafür treffen alle Rollen aufeinander (PO, ScrumMistress/ScrumMaster, Entwicklungsteam). Es wird das Sprint-Ziel definiert und aufgrund dessen wird dann entschieden, welche Funktionen aus dem Product Backlog in den Sprint Backlog übernommen werden. Dafür ist wichtig, stets ein realistisches Ziel vor Augen zu haben, das theoretisch in der festgelegten Zeit erreicht werden kann. Danach wird für jede Aufgabe noch die voraussichtliche Dauer der Implementierung in Stunden vom Entwicklungsteam geschätzt. Nach der Planung organisiert sich das Team selbst und auch die weitere Reihenfolge ist dem Team überlassen.

An jedem Tag in einem laufenden Sprint wird ein **Daily Scrum Meeting** abgehalten. Dieses Treffen ist zeitlich auf 15 Minuten begrenzt. Damit diese Zeit eingehalten werden kann, wird empfohlen, das Meeting im Stehen abzuhalten. Daher wird es auch oft als „Standup“ bezeichnet. Die Moderation übernimmt die ScramMistress oder der ScrumMaster und jedes Teammitglied erklärt kurz:

- Was seit dem letzen Standup erreicht wurde.
- Woran weitergearbeitet wird.
- Welche Probleme aufgekommen sind.

Zum Schluss muss jede und jeder einen Überblick haben, wie weit das gesamte Team ist, und wer Hilfe benötigt.

Am Ende der vorgegebenen Zeit eines Sprints steht immer das **Sprint Review**. Hier wird das Produktinkrement getestet und überprüft. Der PO kontrolliert, ob das Produkt den definierten Vorgaben entspricht und er entscheidet für jede Aufgabe, ob diese in Ordnung ist, oder ob sie in einer nächsten Iteration noch einmal überarbeitet werden muss.

Nach dem Sprint Review kommt noch die **Sprint Retrospektive**. Hier steht nicht das Produkt, sondern der Prozess im Fokus. Die ScrumMistress oder der ScrumMaster, der PO und das Entwicklungsteam besprechen, was alles gut gelaufen ist, besonders in Bezug auf das Scrum-Framework, und was beim nächsten Sprint verbessert werden muss. Das Ziel ist eine kontinuierliche Verbesserung. Am Ende sind konkrete Aktionen zur Verbesserung definiert. Nach der Retrospektive beginnt der ganze Sprint-Ablauf wieder von vorne mit der Sprint-Planung.

2.1.4 Scrum Artefakte

Die Scrum Artefakte werden für die Aktivitäten benötigt und zeigen transparent den Fortschritt für das gesamte Projektteam.

Der **Product Backlog** ist eine Liste aller Fähigkeiten und Funktionalitäten, die das fertige System können muss. Es können auch Fähigkeiten definiert werden, die nicht unbedingt implementiert werden müssen, sondern eher als optional betrachtet werden. Nach dem Definieren wird der Backlog priorisiert, damit zuerst die wichtigeren Aufgaben erledigt werden können. In jeder Iteration entscheidet sich das Entwicklungsteam für eine gewisse Anzahl an Funktionen aus dem Product Backlog und versucht diese dann in dem jeweiligen Sprint zu implementieren.

In der Sprint-Planung werden die entsprechenden Aufgaben für die Iteration vom Product-Backlog in den **Sprint-Backlog** übernommen. Die Liste wird laufend vom Entwicklungsteam

aktualisiert. Für eine übersichtliche Anzeige der Aufgaben, sowie später im Sprint Review ist die Liste essentiell.

Eine **User Story** beschreibt laut Wikipedia [55] eine Funktionalität aus der Sicht eines Users. Es darf somit keine fachspezifische Sprache verwendet werden. In der User Story wird die Benutzerin oder der Benutzer kurz beschrieben (zum Beispiel eine 20-jährige Fußballspielerin). Dann wird angegeben, welche Funktionen sich diese Person aus welchen Grund wünscht. Eine Begründung bietet einen Einblick in die Notwendigkeit der jeweiligen Funktion.

2.2 Frontend

Im Frontend wird Angular für die Programmierung eingesetzt und essentielle Bestandteile einer guten Entwicklung der Benutzeroberfläche sind eine einfache und übersichtliche Visualisierung und eine verlässliche Usability geachtet.

2.2.1 Angular-Framwork

Ein großes Frontend Projekt kann sehr schnell sehr groß werden. Aufgaben wie Routing, Daten Binding und Validierung bleiben dabei vollständig bei der Programmiererin oder dem Programmierer hängen. Außerdem muss der Quellcode überschaubar und leicht zu warten sein. All diese Kriterien sind schwer umzusetzen, deswegen ist die Verwendung von Frontend-Frameworks laut Steyer und Softic [46, S. 31] oft unverzichtbar. Eines davon ist AngularJS, dass von Google entwickelt worden ist.

Seit das Web eine Wandlung von statisch nachladbaren Seiten hin zu dynamisch austauschbaren Elementen genommen hat, kommen **Single-Page-Applications (SPA)**, wie in Steyer und Softic [46, S. 31–33] erklärt, immer mehr zum Einsatz. Auch Angular beruht auf diesem Prinzip. Ein großer Vorteil der SPA ist ihre Plattformunabhängigkeit. Da heute viele verschiedene Plattformen am Markt sind und stets neue erscheinen, ist diese Eigenschaft wichtiger denn je. Die Seite ist damit auch viel angenehmer zu bedienen, weil der User nicht ständig auf neue Teile der Website warten muss, sondern die Daten dynamisch aktualisiert werden.

Die Hauptbestandteile jedes Angular-Projektes sind nach Moiseev und Fain [34, Kap. 2.1 *Components*] die **Komponenten**. Eine Komponente besteht meist aus mehreren Dateien:

- Eine TypeScript Klasse für die Definition und Logik
- Eine CSS-Datei für die Layout-Definition
- Eine HTML-Datei als Template

- Eine Datei für Tests

Es ist aber auch möglich alle benötigten Informationen in eine Datei zu geben, darunter leidet aber die Überschaubarkeit enorm. Nachdem eine Komponente definiert wurde, kann sie sofort an einer anderen Stelle eingebunden werden.

Bei einer strukturierten Programmierung werden Logik-Aufgaben in **Services** ausgelagert. Moiseev und Fain [34, Kap. 2.2 *Services*] erklären dies und nennen Beispiele, wie unter anderem Datenmanipulation, sowie das Aufrufen von externen Daten über eine API. Diese Services werden in den Komponenten importiert und aufgerufen. So wird auch sichergestellt, dass eine Aufgabe nicht an mehreren Stellen im Code doppelt implementiert wird. Ein Service wird nur durch Komponenten aufgerufen und nie direkt auf der Benutzeroberfläche angezeigt.

Die gesamte Architektur von Angular ist laut Moiseev und Fain [34, Kap. 2.5 *Modules*] in **Module** unterteilt. Ein Modul ist ein Container für zusammengehörige Elemente wie Komponenten und Services. Es ist auch möglich, dass alle Elemente in dem von Angular vorgeschriebenen *Root-Modul* liegen. Selbsterstellte Container werden *Feature-Module* genannt. Auch Angular hat mehrere Module und diese können im *Root-Modul* importiert werden, zum Beispiel die Angular Forms API.

Es ist möglich das Angular-Framework mit **JavaScript oder TypeScript** zu verwenden. Moiseev und Fain [34, Kap. 1.2 *Why Develop in TypeScript and not in JavaScript?*] beschreiben, warum TypeScript die sinnvollere Variante bei der Entwicklung von größeren Projekten ist. Zum Ersten ist das gesamte Framework mit TypeScript geschrieben worden und es unterstützt, im Gegensatz zu JavaScript, Typen. Zum Zweiten hat es eine bessere Fehlerkorrektur als JavaScript und die Programmiererin oder der Programmierer wird besser durch die IDE (*Integrated Development Environment* oder integrierte Entwicklungsumgebung) unterstützt.

2.2.2 Visualisierung

Visualisierung oder Datenvisualisierung bezeichnet laut Nazemi et al. [35, S. 1] die grafische Darstellung von Daten in einem zweidimensionalen Raum. Nachdem Computer für die generelle Bevölkerung zugänglich wurden und nicht mehr nur von Spezialistinnen und Spezialisten bedient werden konnten, wurde das Thema Visualisierung immer bedeutender, wie von Preim und Dachselt [37, S. 1] beschrieben. Auch Menschen ohne besondere Vorkenntnisse im technischen Bereich müssen Webseiten bedienen können. Wenn komplexe Programme nur mit Handbüchern oder Schulungen zu bedienen sind, verlieren sie an Bedeutung und werden von den Konsumentinnen und Konsumenten nicht mehr akzeptiert.

Arten der Visualisierung

Informationen können je nach Zweck und Inhalt auf unterschiedliche Weise auf Webseiten visuell dargestellt werden. Die wichtigsten Methoden werden in den nächsten Absätzen erläutert.

Um numerische Werte oder Vergleiche zwischen Werten verständlich darzustellen, sind laut Duisburg-Essen [7, S. 6] auf **Diagramme** die beste Option. Hier werden komplexe Informationen leicht lesbar dargestellt und sie sind oft verständlicher als ein langer erklärender Text. Wichtig ist nur die Wahl des richtigen Diagramms, um Daten nicht zu verfälschen oder ein falsches Bild an die Nutzerinnen und Nutzer zu vermitteln.

Arten von Diagrammen:

- Säulen-/Balkendiagramm: bei Kategorien oder Aufzählungen
- Kreis-/Tortendiagramm: bei Wertvergleichen
- Kurvendiagramm: bei Entwicklungen über einen definierten Zeitraum
- Streu-/Punktdiagramm: um Zusammenhänge und Trends zu erkennen
- Blasendiagramm: verwandt mit den Streudiagrammen, wobei hier die Größe der verschiedenen Blasen angepasst werden kann

Eine besondere Art von Grafiken sind nach Hahn [12, Kap. 11.6 *Icons*] **Icons**. Diese ähneln den Symbolen und bestehen meist aus einem Objekt oder einer Aktion und helfen den Usern bei möglichen Interaktionen. Gängige Icons wie ein Pfeil als Zurück-Symbol, eine Lupe als Darstellung einer Suche, ein Zahnrad für die Einstellungen oder ein Haus für die Startseite werden besonders oft eingesetzt, da hier die Verwendung bereits vertraut ist. Somit wird ein zusätzlicher erklärender Text nicht mehr benötigt.

Hahn verdeutlicht in [12, Kap. 11.2 *Bilder im Web*], dass **Bilder** mehrere Aufgaben erfüllen. Sie können einerseits die Stimmung der Webseite verändern und zum Beispiel als reine optische Komponente im Hintergrund eingebunden sein. Andererseits können sie auch eine echte Funktionalität erfüllen und als Beispiel zur Orientierungshilfe verwendet werden, wie um zwischen Auswahlmöglichkeiten zu wählen, die nicht durch einen einfachen Button mit Text, sondern durch Bilder dargestellt sind. Ein weiteres Beispiel ist die Vorschau auf ein Produkt in einem Onlineshop, um dieses dem User visuell präsentieren zu können.

Gestaltungsprinzipen

Bestimmte Gestaltungsprinzipien helfen um eine effektive Visualisierung ermöglichen zu können. Einige verschiedene Ansätze werden in den nachfolgenden Absätzen vorgestellt.

Software muss nach Preim und Dachselt [37, S. 231–234] für viele verschiedene User anwendbar sein, um eine **barrierefreie Nutzung** gewährleisten zu können. So dürfen auch die Bedürfnisse von körperlich oder geistig eingeschränkten Menschen nicht vernachlässigt werden. Beispiele hierfür sind Farbenblindheit und Leseschwächen. Alle diese Benutzergruppen müssen bereits in der Planung berücksichtigt werden.

Farben sind laut Stapelkamp [44, S. 184, 198] ein wichtiges Medium, um Informationen zu übertragen und miteinander in Verbindung zu bringen. Es ist aber größte Vorsicht geboten, da durch Farben auch leicht eine Fehlinterpretation von Sachlagen passieren kann. Rottöne werden bereits von Natur aus mit Gefahr oder Warnung assoziiert, das findet zum Beispiel auch im Straßenverkehr Anwendung. Es ist deshalb sehr wichtig auf einen korrekten und verständlichen Umgang mit Farben zu achten. Wie von Webdesignratgeber.de [52] erklärt, lösen Farben auch unterschiedliche Gefühle in uns aus und auf diese Farbenpsychologie darf in der Designphase nicht vergessen werden.

Ein weiterer Punkt für die Lesbarkeit ist außerdem ein guter Kontrast zwischen der Hintergrundfarbe und dem Text. Schließlich muss auch die Konsistenz bedacht werden. Es ist förderlich ein Farbschema für eine Website zu entwickeln und dieses dann konsistent zu verwenden. Das fördert ein einheitliches Design und eine Abgrenzung zu anderen Websites.

Wie in Webdesignratgeber.de [52] beschrieben, müssen auch alle Inhalte in einem Raster übersichtlich aufgeteilt sein. Eine logische Reihenfolge ist ebenfalls essentiell. Die Navigation muss leicht verständlich sein. In der Praxis haben sich Menüleisten mit den wichtigsten Seiten durchgesetzt, mit Markierungen, wo sich ein User gerade befindet. Eine implementierte Suchfunktion kann die Benutzung zusätzlich erleichtern.

2.2.3 Usability

„Gute Usability wird selten von Nutzer*innen wahrgenommen, schlechte Usability allerdings schon. Eine gute Usability erleichtert den Prozess, trägt zur Zufriedenheit bei und verhindert unnötige Fehler.“ [47]

Die Technikum Wien Academy [47] beschreibt dazu drei wichtige Faktoren, die es zu beachten gibt: Erlernbarkeit, Effektivität und Zufriedenheit. Die Übersichtlichkeit und wie schnell sich ein User mit der Software zurecht findet, wird mit der Erlernbarkeit beschrieben. Die Effektivität definiert, ob einem User bei seinem Ziel geholfen werden kann und die Zufriedenheit geht auf die Erfahrungen eines Users nach der Verwendung der Software ein.

Ist eine hohe Benutzerfreundlichkeit eigentlich sinnvoll? Wie Erfahrungen zeigen, wird in vielen Projekten schnell auf Usability-Tests oder ein Feedback von Usern verzichtet. Dieser grundlegender Fehler und seine Folgen werden von der Technikum Wien Academy [47] erklärt. Selbst die innovativste Software ist ohne Anwenderinnen und Anwender nichts wert und ein User besucht nur Seiten, auf denen er sich wohl fühlt. Klare Störfaktoren auf einer Website sind laut Tillmanns [49] zum Beispiel zu große Pop-Ups, die nur den User Flow stören und nichts zum Erreichen des eigentlichen Zieles beitragen.

Die **Navigation** ist wichtig, damit sich ein User schnell auf der Webseite zurechtfinden kann. Zum Ersten ist die Platzierung von großer Bedeutung. Standardmäßig wird erwartet, die Navigationsleiste oben oder an der Seite zu finden. Befindet sie sich dort nicht, kann es schnell das Gefühl von Unübersichtlichkeit im User auslösen. Zum Zweiten ist das Design der eigentlichen Leiste nicht zu vernachlässigen, dies wird im Webdesignratgeber.de [52] beschrieben. Maximal drei Ebenen sind in den meisten Fällen noch übersichtlich und der aktuelle Standort auf der Webseite wird üblicherweise deutlich markiert.

Unter der 3-Klick-Regel ist laut Webdesignratgeber.de [52] die schnelle Navigation zu verstehen. Jeder Inhalt muss innerhalb von drei Klicks erreichbar sein. Diese Regel ist aber nur bei kleineren Websites überhaupt umsetzbar und auch dort nicht immer sinnvoll. Oft ergeben mehrere eindeutige Klicks mehr Sinn als drei schwer auffindbare Links. Deswegen wird diese Regel oft mehr als Richtwert verstanden.

Barrierefreiheit

Laut Statistik Austria [45] gibt es in Österreich 781.821 Personen mit registrierter Behinderung. Das sind mehr als acht Prozent der österreichischen Bevölkerung. Rein ethisch betrachtet müsste jede und jeder Zugriff auf digitale Inhalte haben. Aber auch wirtschaftlich betrachtet ist das ein enormer Vorteil, da sich dadurch die Zielgruppe deutlich vergrößert. Weiters dürfen, wie in UXMA [51] beschrieben, auch die kurzzeitig eingeschränkten Personen nicht vergessen werden. Als Beispiel können alle im Lauf ihres Lebens eine gebrochene Hand oder eine Augenoperation haben und somit ist das Thema für jede und jeden persönlich wichtig.

Seit dem 28. Juni 2025 gibt es laut der Wirtschaftskammer Österreich [56] auch das Barrierefreiheitsgesetzt (BaFG). Es werden dadurch ein Minimum von Maßnahmen rechtlich vorgeschrieben. Barrierefreiheit ist nicht mehr nur ein Extrafeature, sondern eine verpflichtende Anforderung an Softwareprojekte.

Usability-Test

Keine Entwicklerin und kein Entwickler kann alle Tricks kennen, die zu einer sehr hohen Benutzerfreundlichkeit führen. Deswegen gibt es die sogenannten Usability-Tests. Es werden laut Usability.de [50] verschiedene Personen aus der Zielpersonengruppe befragt und die Software wird stichprobenartig getestet, um Feedback einzusammeln. In Folge können die gewünschten Verbesserungen durchgeführt werden. Empfehlenswert ist dies nicht nur am Ende des Entwicklungsprozesses, sondern iterativ. Jedes Mal nachdem ein neues Inkrement abgeschlossen ist, kann dieses durch Stichproben getestet werden. Dazu können immer die gleichen Personen befragt werden oder stets neue Testerinnen und Tester gesucht werden.

2.3 Backend

Für die technische Umsetzung des Backends kommt das Framework Spring Boot in Kombination mit der Programmiersprache Kotlin zum Einsatz. Spring Boot wird verwendet, da es sich besonders für REST APIs eignet. Produktionsreife Anwendungen können aufgrund des Coding-by-Convention-Ansatzes schnell und mit minimaler Konfiguration entwickelt werden.

2.3.1 Framework Spring Boot und Kotlin

Spring Boot ist nach Microsoft [30] ein gängiges Open-Source-Tool, das auf dem Spring-Framework aufbaut und unter anderem die Implementierung von Webanwendungen sowie Microservices erleichtert. Ein Microservice ist ein kleiner, eigenständig entwickelbarer und deploybarer Softwarebaustein, der eine klar abgegrenzte Aufgabe erfüllt und über definierte APIs mit anderen Services kommuniziert.

Das Spring-Framework basiert nach Spring Boot [43, S. 15-18] auf zwei Prinzipien:

Inversion of Control (IoC, Kontrollumkehr) bedeutet, dass der Aufruf eines Programmcodes vom Framework kontrolliert wird. Dies wird unter anderem bei „Callbacks“ verwendet, also wenn eine Funktion selbst programmiert wird und dann vom Framework zum passenden Zeitpunkt aufgerufen wird.

Dependency Injection (DI) kann nicht ganz gleich gesetzt werden mit Inversion of Control. Es ist eine konkrete Umsetzung davon. Das bedeutet, dass Spring Objekte (Beans) erstellt, verwaltet und dort injiziert werden, wo sie benötigt sind. Dabei geht das Framework so vor, dass es das Projekt nach Annotationen scannt, Instanzen von diesen Klassen erstellt und diese im Application Context verwaltet.

Der Application Context ist nach Spring Boot [43, S. 19] ein Container, der von Spring beim Start einer Anwendung erstellt wird und sich um die Beans-Erstellung, Dependency Injection und Verwaltung von Bean-Lifecycles kümmert.

Ein wichtiger Vorteil dieses Ansatzes ist nach Spring Boot [43, S. 17] die lose Kopplung der Komponenten. Klassen müssen ihre Abhängigkeiten nicht selbst instanziieren, sondern nur deklarieren, welche Typen sie brauchen. Wird eine Klasse verändert, wirkt sich das nicht auf andere Klassen aus.

Durch die zentrale Objektverwaltung im Application Context nach Yuyang W. (Compile N Run) [57], entscheidet Spring wann, wie und wie viele Instanzen einer Bean erstellt werden. Dadurch werden Inkonsistenzen, doppelte Instanzen und unkontrollierte Abhängigkeiten vermieden.

Spring Boot baut nach Microsoft [30] auf diesen Kernfunktionalitäten auf und bietet weitere Features, um die Entwicklung zu vereinfachen. Durch vordefinierte Abhängigkeitsbündel, sogenannte „Starter“ für beispielsweise Webanwendungen oder Datenbankzugriffe, müssen häufig benötigte Bibliotheken nicht einzeln gesammelt und konfiguriert werden. Dies erleichtert die Build- und Projektkonfiguration, während viele manuelle Einstellungen die automatische Konfiguration des Spring-Frameworks übernimmt. Dabei folgt Spring Boot dem Prinzip der Konvention vor Konfiguration (Coding by Convention). Das Framework hat sinnvolle Standardkonfigurationen, die automatisch angewandt werden, wenn keine anderen expliziten Einstellungen vorgenommen werden. Dadurch wird der Konfigurationsaufwand reduziert, während die Flexibilität des Frameworks erhalten bleibt.

Kotlin ist nach Domain Factory GmbH [6] eine funktionale und objektorientierte Programmiersprache mit statischer Typisierung, welche wie Java auf der Java Virtual Machine (JVM) ausgeführt wird. Mittels Kotlin kann im Vergleich zu Java der Code kompakter und ausdrucksstärker gestaltet werden. Somit wird weniger Code benötigt. Durch die integrierte Null-Safety werden nullable und non-nullable Datentypen unterschieden [53]. Dadurch wird das bekannte Java-Problem der NullpointerException vermieden und die Zuverlässigkeit des Codes gesteigert [2].

2.3.2 REST-API

Nach Geeks for Geeks [10] legt Represential State Transfer (REST) Architekturbeschränkungen fest. Application Programming Interface (API) ist eine Programmierschnittstelle, welche die Kommunikation zwischen Systemen über das Internet ermöglicht. Dabei sendet ein Client einen Request an den Server, der den Request verarbeitet und eine Response zurückschickt. Die

Anfrage vom Client wird über eine Web-URL übertragen und verwendet eine der folgenden HTTP-Methoden:

GET - ist eine idempotente Methode, um Daten vom Server abzurufen.

POST - ist eine nicht-idempotente Methode, um Daten an den Server zu senden oder neue Ressourcen zu erstellen.

PUT - ist eine idempotente Methode, um eine Ressource vollständig zu aktualisieren oder zu erstellen, wenn sie nicht existiert.

PATCH - ist eine (meist-)idempotente Methode, um eine Ressource teilweise zu aktualisieren.

DELETE - ist eine idempotente Methode, um eine Ressource zu löschen.

Idempotent bedeutet, dass es keinen Unterschied macht, ob eine Anfrage einmalig oder wiederholt ausgeführt wird [38].

Nach IBM [13] erfolgt die Antwort vom Server meist im JSON-Format, wobei viele Datenformate unterstützt werden. REST-APIs können mit praktisch jeder Programmiersprache erstellt werden.

Folgende Prinzipien sind laut Red Hat [39] und IBM [13] zu befolgen:

Einheitliche Schnittstelle - REST APIs haben Konventionen zu befolgen, wie konsistente URL-Pfade, sodass ähnliche Ressourcen entsprechend adressiert werden. Außerdem werden standardisierte HTTP-Methoden und Statuscodes verwendet.

Client-Server-Architektur - Die Kommunikation erfolgt zwischen dem Client, der auf Ressourcen zugreift, und dem Server, der die Ressource enthält. Dabei ist es dem Client nur möglich über die Adresse der gewünschten Ressource mit dem Server zu interagieren. Client- und Serverdienste müssen so konzipiert werden, dass sie unabhängig voneinander sind.

Zustandslosigkeit - Jeder Request enthält alle Informationen, die für die Verarbeitung benötigt werden. Der Server speichert keine Client-Informationen. Die einzelnen Anfragen erfolgen voneinander getrennt.

Cacheable - Der Server kann abgerufene Daten zwischenspeichern, um die Leistung auf der Seite des Clients zu erhöhen.

Mehrschichtige Systemarchitektur - Client und Server müssen keine direkte Verbindung haben. Die Kommunikation kann über mehrere Zwischenstufen erfolgen. Hierbei dürfen Client und Server nicht feststellen, ob die Kommunikation direkt oder über Zwischenschichten erfolgt.

2.3.3 Websockets

Wie in IONOS Digital Guide [14] erklärt, basiert die Standardkommunikation im Web auf HTTP-Protokolle beziehungsweise HTTP-Requests. Der Client fordert Daten vom Server an und erhält darauf eine Antwort. Die gebräuchlichste Variante ist die GET-Anfragemethode. Es wird definiert, welche Daten der Client benötigt und der Server antwortet dementsprechend mit den passenden Daten oder einer Fehlermeldung. Das kann, wie in Ryte Wiki [41] erwähnt, zu einigen Problemen führen. Das Hauptproblem ist die einseitige Kommunikation. Der Server kann nur dem Client antworten, er kann nie von sich aus Daten senden. Bei Anwendungsfällen, die Livedaten benötigen, ist dieses Prinzip nicht zielführend. Um solche Programme mit HTTP-Requests zu verwirklichen, muss der Client dauerhaft nach Updates fragen, was als „Polling“ bezeichnet wird. Das ist nicht nur ineffizient, sondern kann auch schnell zu einer Serverüberlastung führen.

Die Lösung sind Websockets. Ihr Hauptnutzen liegt darin, dass der Server auch ohne Anfrage Daten an den Client senden kann. Dazu wird eine Erweiterung der TCP-Verbindung genutzt. Die TCP-Verbindung wird von GDSYS [9] wie folgt beschrieben. Das TCP-Protokoll tauscht Daten zwischen zwei Geräten aus. Die Verbindung bleibt aufgebaut bis eine Seite diese absichtlich schließt oder die Daten vollständig übertragen werden. Dieses Prinzip ist wichtig, da ein Server niemals die Verbindung alleine aufbauen kann. Um zu wissen, wer seine Daten empfangen möchte, muss der Client die Verbindung zuerst einrichten.

Der Ablauf dieses Verbindungsaufbaus, laut der *Handshake-Methode*, wird von Florian Deinhard [8] folgendermaßen beschrieben:

1. Der Client sendet eine Verbindungsanfrage an den Websocket-Server und zeigt somit sein Interesse an den Daten.
2. Der Server speichert die Verbindung und wenn die Verbindung erfolgreich aufgebaut wurde, wird eine Bestätigung an den Client gesendet.
3. Danach kann sowohl der Server als auch der Client jederzeit Nachrichten an den jeweils anderen senden, ohne dass jede Nachricht extra bestätigt werden muss.

Diese Technologie findet in vielen Systemen Anwendung. Nach Florian Deinhard [8] wird es für Echtzeit-Anwendungen, in denen eine schnelle Antwort erwartet wird, verwendet. Beispiele sind Online-Gaming, Chat-Applikationen, Sportwetten und Auktionen.

2.4 Datenbank

Für die Speicherung sämtlicher Daten wird PostgreSQL verwendet, wobei die Datenbankversionierung mit Liquibase sichergestellt wird.

2.4.1 PostgreSQL

PostgreSQL ist nach The PostgreSQL Global Development Group [48] ein kostenloses Open-Source Datenbankmanagementsystem, das relationale Daten und auch objektrelationale Konzepte, wie benutzerdefinierte Datentypen, unterstützt. Es ist einfach erweiterbar und flexibel skalierbar. Durch die ACID-Konformität und schnelle Lese- und Schreibgeschwindigkeiten sowie den Einsatz von MVCC hat es sich über viele Jahre bewährt.

MVCC (Multiversion Concurrency Control) ist ein Verfahren zur gleichzeitigen Verarbeitung von Transaktionen bei mehreren Benutzern. Anstatt Datensätze bei Schreibzugriffen zu sperren, erzeugt PostgreSQL neue Versionen eines Tupels (Zeile). Jede Transaktion sieht dabei eine konsistente Momentaufnahme (Snapshot) der Datenbank zum Zeitpunkt ihres Starts. Dadurch können Lese- und Schreiboperationen parallel ausgeführt werden, ohne dass Leser Schreibzugriffe blockieren oder umgekehrt. Jede Zeile enthält eine Transaktions-ID vom Ersteller (xmin) und des Löschers bzw. der ersetzenden Transaktion (xmax). Wenn eine Zeile aktualisiert wird, wird diese nicht überschrieben, sondern eine neue Version der Zeile erstellt. Da die alte Version der Zeile bestehen bleibt, führt PostgreSQL regelmäßig das VACUUM-Verfahren aus. Dabei werden nicht mehr benötigte Tupelversionen gelöscht.

Die Architektur von PostgreSQL folgt einem Client-Server-Modell, bei dem ein zentraler Datenbankserver die Verwaltung und Speicherung der Daten übernimmt, während Clients über definierte Schnittstellen, wie SQL, Treiber (ODBC, JDBC, ...) oder APIs, auf die Daten zugreifen.

Fortgeschrittene SQL-Funktionalitäten wie Joins, Subqueries oder Stored Procedures werden ebenfalls unterstützt und sind wichtig, um große Datenmengen effizient zu speichern und abzufragen.

2.4.2 Liquibase

Liquibase ist nach Wikipedia [54] ein Open-Source-Tool zur Datenbankversionierung und -migration. Es wird genutzt, um Änderungen an der Datenbankstruktur (Schema, Tabellen, Spalten, Indizes, usw.) kontrolliert, wiederholbar und nachvollziehbar durchzuführen. PostgreSQL

und viele weitere relationale Datenbanksysteme werden unterstützt. In einer Spring Boot Anwendung wird Liquibase beim Applikationsstart mitausgeführt. Das bietet den Vorteil, dass die Datenbankversion immer mit dem Code kompatibel ist.

Änderungen werden in Changelogs definiert, die eine Liste von Changesets enthalten. Diese repräsentieren die konkreten Änderungen und enthalten eine eindeutige ID, einen Autor und die Änderungsanweisungen wie createTable oder addColumn. Durch den Autor-Tag kann nachverfolgt werden, wer welche Änderungen vorgenommen hat. Die Liquibase-Changelogs können in Formaten wie XML, YAML, JSON oder SQL geschrieben sein, wobei in dieser Arbeit das XML-Format verwendet wird. Ausgeführte Changesets werden in der Datenbank in der Tabelle „DATABASECHANGELOG“ gespeichert. So wird sichergestellt, dass jede Änderung nur einmal ausgeführt wird.

2.5 Microsoft Entra ID (Azure Active Directory)

Azure Active Directory (*Azure AD*), seit 2023 unter dem Namen *Microsoft Entra ID* bekannt [26], ist ein cloudbasierter Dienst zur Identitäts- und Zugriffsverwaltung (Identity and Access Management, IAM) von Microsoft [31]. Mit einem Microsoft-365-Online-Abonnement steht Microsoft Entra ID automatisch zur Verfügung [31].

Der Dienst ermöglicht es nach Microsoft [31] Organisationen und Firmen zentral steuern zu können, wer Zugriff auf welche Ressourcen hat. Dieser ermöglicht es Benutzerkonten, Gruppen, Geräte und Anwendungen eindeutig zu identifizieren und zu verwalten. Mithilfe von modernen Autorisierungsprotokollen, wie OAuth 2.0, wird sichergestellt, dass Benutzerkonten nur auf Ressourcen zugreifen können, für die sie Berechtigungen zugewiesen bekommen haben [27].

Jedes Benutzerkonto besitzt nach Microsoft [17] eine eindeutige interne Kennung, die sogenannte *User Object ID* (auch Azure User ID genannt). Diese ID ist eine GUID, die den Benutzer innerhalb eines Microsoft-Entra-ID-Mandanten eindeutig identifiziert.

Ein Microsoft-Entra-ID-Mandant (Tenant) wird nach Microsoft [21] beim Einrichten eines Microsoft-365- oder Azure-Abonnements automatisch erstellt und bildet den Organisationskontext. In diesem werden Identitäten, Gruppen und Zugriffsrichtlinien verwaltet.

2.5.1 Microsoft Graph API

Microsoft Graph ermöglicht den Zugriff auf sämtliche Daten, die in Microsoft 365 gespeichert sind. Die Microsoft Graph API ist nach Microsoft [32] eine sichere und einheitliche Schnittstelle,

über die sich diese Daten innerhalb der Organisation abrufen und verwenden lassen. So kann auf Benutzer und Gruppen, Teams-Daten sowie auf Aufgaben, Dateien, E-Mails und Kalender zugegriffen werden. Der entscheidende Vorteil der Microsoft Graph-API gegenüber separaten Abfragen von einzelnen Diensten wie Outlook oder OneDrive ist, dass alle Anfragen über den zentralen Endpunkt *https://graph.microsoft.com/v1.0/* gesendet werden.

Die Authentifizierung für Zugriffe auf die Microsoft Graph API erfolgt über Microsoft Entra ID. Eine Anwendung muss sich gegenüber Microsoft Entra ID authentifizieren, um einen Zugriffstoken zu erhalten.

Davor muss eine App registriert werden, wobei eine Client-ID, eine Tenant-ID und ein Client Secret erzeugt werden, die für die Authentifizierung notwendig sind. Nach Microsoft [24] ist die Client-ID eine eindeutige Kennung der App und die Tenant-ID identifiziert die Azure-Umgebung. Das Client Secret ist das Passwort für das Backend und ermöglicht es, dass sich die App programmgesteuert bei Microsoft Entra ID authentifiziert, ohne dass ein Benutzerkonto aktiv eingeloggt sein muss. Dieses Vorgehen wird als *Client Credentials Flow* bezeichnet.

2.6 Microsoft Teams Bot

Ein Microsoft Teams Bot ist eine automatisierte Anwendung, die in Microsoft Teams über Nachrichten, Befehle oder *Adaptive Cards* interagiert [19]. Adaptive Cards sind nach Microsoft [29] ein UI-Format von Microsoft, das in einer einheitlichen, plattformunabhängigen JSON-Struktur definiert ist. Dieses Format ermöglicht es, reichhaltige Inhalte wie formatierte Texte, Bilder oder Buttons anzuzeigen. Sie werden in den jeweiligen Anwendungen - wie Microsoft Teams oder Skype - nativ gerendert und passen sich automatisch dem Design der Host-Anwendung an.

Ein Bot folgt laut Microsoft [19] vordefinierten Regeln und festen Abläufen. Eingehende Nachrichten werden ausgewertet und mit bestimmten Triggern oder Befehlen – wie bestimmten Wörtern oder Slash-Commands wie /help – verglichen. Wenn ein Trigger erkannt wird, führt der Bot die dazugehörige Antwort oder Funktion aus. Optional kann ein Microsoft Teams Bot mit künstlicher Intelligenz erweitert werden, damit flexible Dialoge geführt werden können.

Technologisch basiert der Microsoft Teams Bot auf dem SDK vom Microsoft Bot Framework, das in Node.js über das Paket *botbuilder* bereitgestellt wird [20]. Dieses SDK bietet die zentrale Bot-Logik.

Um die Entwicklung für Teams-spezifische Anwendungen zu erleichtern, wird das Framework durch das *TeamsFx-SDK* erweitert. Dieses reduziert den Implementierungsaufwand für typische Teams-Funktionalitäten, wie das Senden und Empfangen von Nachrichten. Es stellt Benutzerinformationen bereit und erleichtert die Authentifizierung [28].

Microsoft Teams unterscheidet laut Microsoft [19] verschiedene Bot-Typen nach ihrer Funktion:

Konversationsbot reagiert auf Benutzereingaben und kann passende Antworten senden.

Benachrichtigungsbot wird zur Benachrichtigung in Teams-Kanälen, Gruppenchats oder persönlichen Chats verwendet.

Workflow-Bot automatisiert Aufgaben, die sich wiederholen.

Befehlsbot sendet vordefinierte Antworten auf einfache Befehle.

Für das Hosting werden *Azure Functions* genutzt, wodurch der Bot serverless betrieben und über HTTP ausgelöst werden kann [33]. „Serverless“ bedeutet, dass Azure sich um den Server, die Betriebssysteme und die Infrastruktur kümmert. Es ist lediglich der auszuführende Code bereitzustellen. Die Functions sind event-getriggert, laufen also, wenn ein bestimmtes Ereignis passiert, und existieren nur für die Dauer der Ausführung. Dadurch dass sie HTTP-triggered sind, kann der Bot auf Nachrichten von Teams und auch auf Requests von einem externen Backend reagieren. Die HTTP-Endpunkte des Bots können mit Sicherheitsmechanismen wie OAuth 2.0 geschützt werden [18]. Für dieses Projekt wird jedoch *HTTP Basic Authentication* verwendet, da keine rollenbasierten Zugriffe oder andere erweiterte Features von OAuth 2.0 benötigt werden und die Kommunikation ausschließlich intern erfolgt.

Basic Authentication ist nach Cristian Sifuentes [5] ein standardisiertes HTTP-Authentifizierungsverfahren, bei dem Benutzername und Passwort in jedem Request über den **HTTP-Header Authorization** übertragen werden. Der Header besteht aus dem Präfix *Basic* und der Kombination von Benutzername und Passwort und ist Base64-kodiert. Es wird häufig für interne APIs oder einfache Integration genutzt.

2.6.1 Microsoft 365 Developer Sandbox

Die Microsoft 365 Developer Sandbox ist nach Microsoft [22] eine speziell für Entwickler bereitgestellte Testumgebung vom Microsoft 365 Developer Program. Sie bietet eine isolierte Microsoft 365-Entwicklungsumgebung mit Microsoft Teams, SharePoint, Outlook und weite-

ren Diensten. Diese Sandbox wird für Tests, Experimente oder Lernzwecke verwendet, um Microsoft-365-Lösungen in einer sicheren Umgebung auszuprobieren, bevor diese in echten Produktivsystemen eingesetzt werden. In dieser Arbeit wird sie verwendet, um den Teams Bot zu testen.

3 Catrin

Catrin ist nach Neumüller und Cloudflight [36] eine digitale Assistentin am Eingang und im Flur des Büros sowie in der Eventlocation im Dachgeschoss. Sie stellt Mitarbeiterinnen und Mitarbeitern Informationen, wie beispielsweise den zugewiesenen Sitzplatz, zur Verfügung. Um personalisierte Informationen bereitstellen zu können, werden Mitarbeiterinnen und Mitarbeiter mithilfe der Gesichtserkennung identifiziert. Auch die Sitzplatzzuweisungen von Kolleginnen und Kollegen können eingesehen werden. Mithilfe einer weiteren Funktion können andere Mitarbeitende über Microsoft Teams benachrichtigt werden.

Mitarbeiterinnen und Mitarbeiter werden erst angezeigt, wenn diese zugestimmt haben. Bei keiner Zustimmung kann weder die Person noch ihr Sitzplatz angezeigt werden und keine Benachrichtigung über Teams erfolgen. Für die Gesichtserkennung müssen einige Bilder der Personen hochgeladen werden.

In einigen Büros zeigt Catrin das Tagesmenü einiger Restaurants in der Nähe an **(Lunch Service Spring Boot)**.

Die Architektur wird nach Neumüller und Cloudflight [36] basierend auf dem Architekturdiagramm (siehe Abbildung 2) in diesem Kapitel detaillierter ausgeführt.

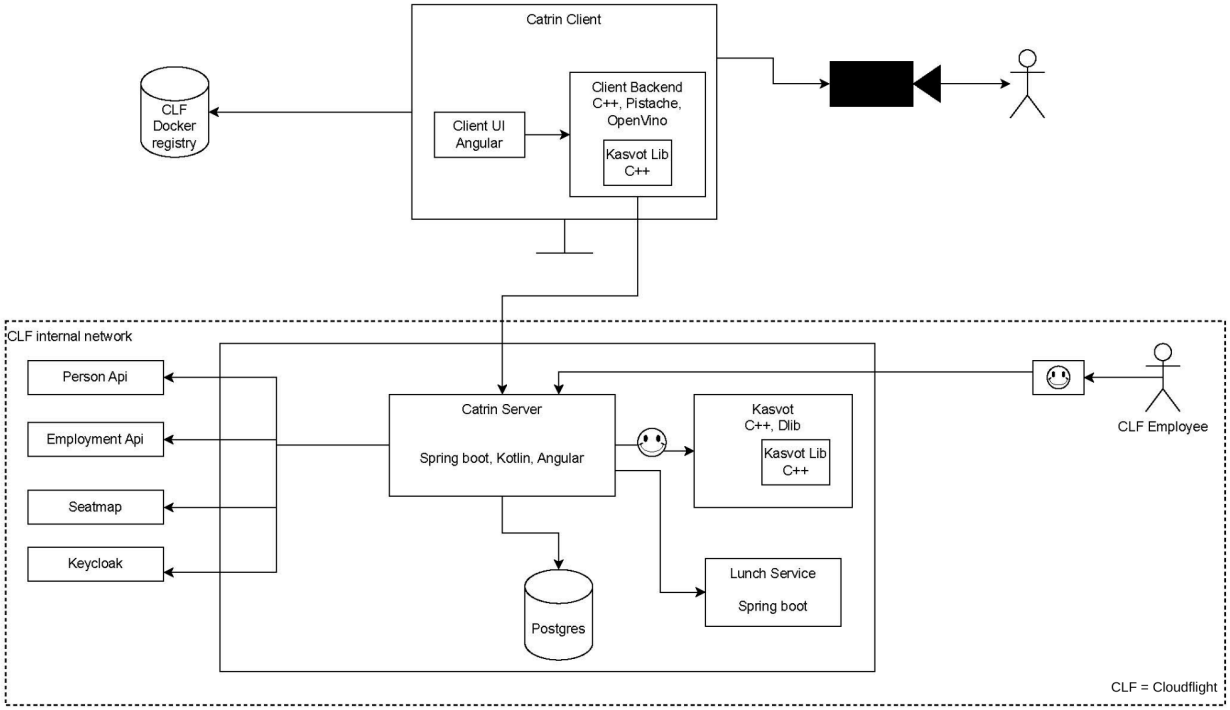


Abbildung 2: Architektur Diagramm [36]

3.1 Catrin Client

Der Catrin Client stellt die Benutzeransicht dar und besteht aus dem Client Backend und dem Client UI in Angular. Dieser Teil ist hauptsächlich für die Anzeige und die Interaktion mit dem User zuständig.

3.1.1 CLF Docker registry

Die Catrin-Geräte betreiben einen einzelnen Docker-Container, der die Benutzeroberfläche und das Client Backend enthält. Die Geräte starten täglich um 4 Uhr neu, laden dabei automatisch das aktuelle Docker-Image und setzen den Container neu auf. Dadurch werden alle Daten, die im Arbeitsspeicher gespeichert sind, gelöscht.

3.1.2 Client UI Angular

Das Frontend ist in vier große Teile unterteilt: Models, Seiten, Komponenten und Services. Die Models definieren Datentypen, die Seiten binden Komponenten ein, die Daten anzeigen und die Services stellen Daten laut Datentypen bereit. Alle Bestandteile sind eng miteinander verbunden.

Das Model definiert Datentypen, wie zum Beispiel einen Mitarbeitenden oder auch einen Standort. Es werden auch verschiedene Varianten von einem Mitarbeitenden gespeichert, wie inklusive einem Bild, oder nur das Mitarbeiter-Bild beziehungsweise das Mitarbeiterin-Bild. Bei einem regulären Mitarbeitenden werden nicht nur Daten, wie Name oder Bild gespeichert, sondern es wird bereits eine Logik implementiert. Es gibt Methoden, um ein neues Bild zu speichern oder um den ersten Buchstaben des Vor- und Nachnamens zu erhalten. Diese Funktionalitäten klingen recht simpel, aber da sie oft in der Applikation verwendet werden, ist es sinnvoll, sie nur einmal zentral zu implementieren, da so Code-Duplikationen verhindert werden können.

Bei den einzelnen Seiten gibt es sechs Hauptseiten:

- After-Work-Beer
- Dashboard
- Mitarbeiterinnen und Mitarbeiter
- Externe Seiten
- Mittagessen
- Karte mit den Sitzplätzen

Die Startseite, die auch alle anderen Seiten einbindet, ist die Dashboard-Seite. Ganz oben wird zuerst abgefragt ob gerade eine Aktivität läuft und wenn ja, wird ein Banner eingeblendet. Der Sinn hinter der After-Work-Beer Funktion, oder generell den Aktivitäten liegt darin, dass wenn sich mehrere Mitarbeiterinnen und Mitarbeiter zu einem Feierabend Bier treffen, jeder darüber Bescheid weiß und auch jeder daran teilnehmen kann. Das fördert den Zusammenhalt zwischen den Kolleginnen und Kollegen. Auf der linken Seite des Dashboards steht zuerst ein Begrüßungstext und der Name des Standorts und danach wird die Webcam eingebunden. Es gibt eine Gesichtserkennung, die über die Kasvot Komponente aus dem Backend stattfindet. Auf der rechten Seite finden der interaktive Teil und die weitere Navigation statt. Es werden auf die Catrin-Seiten „Employee“, „Lunch-Service“ und „Floorplan“ und auf die externen Seiten verwiesen. Eine weitere Funktionalität ist die Anzeige der aktuellen Geburtstage der aktiven Mitarbeiterinnen und Mitarbeiter. Ganz unten kann außerdem eine neue Aktivität (wie Feierabend Bier) von allen gestartet werden.

Auf der Seite der Mitarbeiterinnen und Mitarbeiter gibt es eine Mitarbeiterübersicht mit unterschiedlichen Filtermöglichkeiten. Es kann nach Anfangsbuchstaben des Vornamens oder des Nachnamens gefiltert werden. Wird ein Mitarbeitender ausgewählt wird die Komponente der Detailansicht geöffnet. Ganz oben kann die Seite auch wieder geschlossen werden und wieder zurück zur Dashboard Seite navigiert werden.

Die Seite mit den externen Links besteht einfach aus Verlinkungen. Es können Restaurant-Seiten oder zum Beispiel eine Informationsseite für öffentliche Verkehrsmittel eingebunden werden. Wobei es für Restaurants auch die Seite Mittagessen gibt, auf der speziell täglich mehrere aktuelle Lunch-Menüs von umliegenden Restaurants angezeigt werden.

Ein interaktiver Lageplan wird auf der „Floorplan“ Seite angezeigt. Er besteht aus einer grafischen Karte mit den Umrissen des Büros und bestehenden Sitzplätzen. Wenn ein Sitzplatz bereits belegt ist, wird der dort sitzende Mitarbeitende mit einem roten Ring angezeigt und bei einem freien Platz wird ein grüner Ring angezeigt. Es ist auch möglich den Sitzplatz für den aktuellen Tag zu wechseln. Dafür muss der freie Sitzplatz ausgewählt und die Buchung bestätigt werden. Wird auf einen besetzten Sitzplatz geklickt, dann wird die Detailansicht der Mitarbeiterin oder des Mitarbeiters angezeigt.

Die Services sind die Verbindung zum Server. Es werden dynamisch Daten geladen, die durch die bereits erwähnten Komponenten und Seiten angezeigt werden. Es gibt Services für alle API-Endpoints aus dem Service, ein Beispiel ist die Webcam, die durch den passenden Service auf die Gesichtserkennung zugreifen kann.

3.1.3 Client Backend

Das Client Backend ist für die Kommunikation mit dem Catrin Server sowie für die Echtzeit-Gesichtserkennung verantwortlich. Beim Start eines Catrin Geräts werden alle biometrischen Fingerprints vom Catrin Server abgerufen und lokal im Arbeitsspeicher gespeichert. Die Clients greifen über einen tokenbasierten Zugriff auf die Server-Endpunkte zu. Die Anfragen an den Server werden über HTTPS übertragen. Dadurch sind die Daten verschlüsselt und werden sicher zwischen Client und Server ausgetauscht. Administrative UI-Endpunkte sind ausschließlich aus dem internen Netzwerk erreichbar.

Biometrische Fingerprints sind nach Schroff, Kalenichenko und Philbin [42] numerische Merkmalsvektoren, die aus einem Gesicht berechnet werden, um Gesichter effizienter zu vergleichen oder zu identifizieren. Sie werden berechnet, indem ein erkanntes und vorverarbeitetes Gesichtsbild mithilfe von einem tiefen neuronalen Netz in einen kompakten Merkmalsvektor (Face Embedding) umgewandelt wird. In diesem Merkmalsraum liegen Gesichter von der selben Person nah beieinander. Gesichter von verschiedenen Personen haben größere Distanzen. Der Vergleich erfolgt über Distanzmaße, wie die euklidische Distanz, und ermöglicht so die Identifikation von Personen.

Die Kasvot Komponente ist ein internes Geichtserkennungs-System und die technisch anspruchsvollste sowie sicherheitsrelevanteste Komponente im Catrin-System. Sie basiert auf C++ und ist für die Fingerprint-Berechnung der Gesichtserkennung zuständig.

Die Kasvot-Komponente im Client Backend verarbeitet die von der Benutzeroberfläche gelieferten Kamerabilder, die ausschließlich im Arbeitsspeicher gehalten werden. Aus datenschutzrechtlichen Gründen werden sie nicht persistiert oder weitergeleitet. Aus jedem Bild wird das größte, also das am nähesten an die Kamera gehaltene Gesicht, identifiziert und daraus ein biometrischer Fingerprint berechnet. Der kNN-Algorithmus vergleicht den aktuellen Fingerprint mit den lokal gespeicherten Fingerprints und identifiziert so die passende Person. Der ganze Prozess findet auf einem Catrin Gerät statt und stellt sicher, dass keine Bilder an den Server übertragen werden. Es wird auch keine Erkennungshistorie aufgezeichnet.

Der k-Nearest-Neigbours-Algorithmus (kNN) ist nach GeeksForGeeks [11] ein Klassifizierungsalgorithmus, um neue Datenpunkte zu klassifizieren und vorherzusagen, indem diese mit den ähnlichsten Trainingsdaten verglichen werden. Dafür wird zuerst das optimale k gewählt. Das ist die optimale Anzahl der nächsten Nachbarn, die die geringste Fehlerquote liefern. Danach wird die Distanz zwischen dem neuen Punkt und allen Trainingspunkten berechnet und die k-Nachbarn mit der kleinsten Distanz ausgewählt. Um die Distanz zu berechnen gibt es folgende vier Arten:

1. Euklidische Distanz
2. Manhattan-Distanz
3. Minkowski-Distanz
4. Gewichtete Distanz

Die Vorhersage basiert auf der Mehrheit der Klassen der k-Nachbarn oder auf dem Durchschnitt der Klassen der k-Nachbarn.

3.2 Catrin Server

Der Catrin Server ist das zentrale Backend und für die Datenerhaltung sämtlicher Daten verantwortlich. Er versorgt den Client mit den Daten und macht keine Gesichtserkennung.

Die Client UI ist eine Art Admin-Dashboard, bei der sämtliche Daten in einer einfachen Anzeige verwaltet werden können. Hier werden auch die Bilder für die Gesichtserkennung von Mitarbeiterinnen und Mitarbeitern hochgeladen.

3.2.1 Angular

Die grafische Admin Oberfläche für das Catrin Backend wurde in Angular programmiert. Die gespeicherten Daten können aktualisiert oder neu eingetragen werden, es kann zum Beispiel ein neuer Standort angelegt oder der Nachname eines Mitarbeitenden geändert werden. Anwendung findet das vor allem bei der Erstellung von Testdaten, für die Entwicklung eines neuen Features oder für die lokale Testung.

3.2.2 Spring Boot, Kotlin

Das Spring Boot-Backend, implementiert in Kotlin, dient als zentrale Schnittstelle des Catrin Systems und ist im internen Cloudflight Netzwerk (CLF internal network) deployed. Es verbindet die Kasvot-/C++-Komponente zur Gesichtserkennung mit mehreren firmeninternen APIs und den Catrin Clients, die Daten abrufen und anzeigen.

Ein geplanter Cron Job wird täglich um 01:00 Uhr (UTC) ausgeführt, ruft die aktuellen Daten aus der **Person API**, der **Employment API** und der **Seatmap** (siehe Abschnitt 3.2.4) ab und speichert die Daten in der Datenbank. Wenn eine Mitarbeiterin oder ein Mitarbeiter nicht mehr von der Person API zurückgemeldet wird, werden all dessen Fingerprints und Informationen automatisch aus der Catrin-Datenbank gelöscht.

Um für den Client die Fingerprints der einzelnen Mitarbeiterinnen und Mitarbeiter bereitzustellen, kommuniziert das Backend mit dem Kasvot-System (siehe Abschnitt 3.2.3). Die Bilder, die im Frontend (siehe Abschnitt 3.2.1) von den Mitarbeiterinnen und Mitarbeitern hochgeladen werden, werden vom Spring Boot-Server an das Kasvot-System weitergeleitet. Dort werden aus den Bildern biometrische Fingerprints berechnet, die an das Spring Boot-Backend zurückgegeben werden und ebenfalls in der Datenbank zu der dazugehörigen Person gespeichert werden.

Der Server stellt über Spring Boot eine klar definierte REST-API bereit. Beim Start eines Catrin Clients ruft dieser die entsprechenden Endpunkte auf, um alle aktuellen Fingerprints, Sitzplatzinformationen und Mitarbeiterdaten zu laden und gegebenfalls anzuzeigen. Jeder Client authentifiziert sich mit einem eigenen Token, der vom Server geprüft wird.

3.2.3 Kasvot, C++, Dlib

Wie in Abschnitt 3.1.3 beschrieben ist die Kasvot Komponente in C++ implementiert und für die Fingerprint-Berechnung zuständig.

Sie verwendet die Bibliothek *dlib*, die nach King [15] ein modernes C++-Toolkit ist und eine Sammlung von Machine-Learning-Algorithmen und Werkzeugen bereitstellt. Der Schwerpunkt liegt auf Bildverarbeitung, da Funktionen zur **Gesichtserkennung** (Face Detection), **Gesichtsmerkmalserkennung** (Landmarking) und **Merkmalextraktion** (Feature Extraction) bereitgestellt werden. Sie stellt vortrainierte Modelle zur Berechnung von Gesichts-Fingerprints bereit. Die Bibliothek ist Open Source und wird von Davis E. King seit 2002 entwickelt.

Im Catrin Server ist die Kasvot Komponente für die Verarbeitung der Bilder, die vom Spring Boot-Backend weitergeleitet werden, zuständig. Aus diesen Bildern werden biometrische Fingerprints berechnet und an das Backend zurückgegeben.

3.2.4 Seatmap

Die Seatmap ist eine externe API von Cloudflight, die in regelmäßigen Abständen, etwa wöchentlich, 2-wöchentlich, usw., eine zufällige Zuordnung der Sitzplätze zu den Mitarbeitenden liefert. Ein Cron-Job im Catrin Server Backend ruft täglich am Morgen die bestehenden Sitzplatzdaten über die Seatmap-API ab. Dabei werden alle lokal bei Catrin gespeicherten Zuordnungen vollständig überschrieben.

3.3 Postgres Datenbank

Die PostgreSQL (siehe Abschnitt 2.4.1) Datenbank (im Diagramm 2 mit der Abkürzung Postgres bezeichnet) ist für die gesamte Datenverwaltung verantwortlich. Die gespeicherten Informationen werden vom Server abgerufen.

3.3.1 Entitäten

Im nachstehenden werden alle Entitäten und deren Attribute erklärt, wobei die Primärschlüssel in blau und die Fremdschlüssel in gelb dargestellt werden.

Jede Entität beginnt mit *clf_*, was ein Kürzel für *Cloudflight* ist.

Entität: `clf_catrin`

Diese Entität beschreibt einzelne Catrin-Instanzen, die jeweils einem Standort zugeordnet sind.

Gespeichert werden Informationen wie:

id - eindeutige Kennung einer Catrin-Instanz

name - Bezeichnung einer Catrin-Instanz

api_key - der Key, der verwendet wird um einen Catrin Client gegenüber Server- oder Teams Bot-Endpunkten zu verifizieren

last_seen - der Zeitpunkt, der letzten Aktivität der jeweiligen Catrin-Instanz

confidence - der Zuverlässigkeitswert der Gesichtserkennung

location_id - Fremdschlüssel zu einem Standort, an dem sich der jeweilige Catrin Client befindet
Integritätsregel: In der Spalte `location_id` darf nur ein Wert eingetragen werden, der in `clf_location.id` existiert.

Name	Typ	Nullable
id	uuid	false
name	varchar(255)	false
api_key	varchar(255)	true
last_seen	timestamp	true
confidence	integer	false
location_id	uuid	false

Entität: `clf_location`

Diese Entität bildet die verschiedenen Standorte des Unternehmens ab. Neben dem Namen (`name`) und einer Beschreibung (`description`) werden optionale Zusatzinformationen gespeichert:

id - eindeutige Kennung eines Standortes

name - Bezeichnung eines Standortes

description - Beschreibung eines Standortes

core_id - Standort-ID in der Core App
Die Core App ist ein zentrales Tool für Employee Relationship Management (ERM), also für den Austausch und die Koordination zwischen Arbeitgeber und Mitarbeitenden.

after_work_beer - Zeitpunkt der Ausführung einer Aktivität, wie beispielsweise „After Work Beer“

after_work_beer_initiator_id - Initiator des After-Work-Beers; Fremdschlüssel auf einen Mitarbeiter der eine Aktivität ausgelöst hat

teams_channel_id - ID des Teams-Kanals, an den Aktivitäts-Benachrichtigungen gesendet werden

current_activity_id - ein Standort kann nur eine aktuelle Aktivität haben

Name	Typ	Nullable
id	uuid	false
name	varchar(255)	false
description	varchar(255)	false
core_id	integer	true
lunch_service_name	varchar(255)	true
after_work_beer	timestamp	true
after_work_beer_initiator_id	uuid	true
seatmap_id	integer	true
static_seatmap	boolean	false
teams_channel_id	varchar(255)	true
current_activity_id	uuid	true

Entität: `clf_face`

In dieser Tabelle werden biometrische Daten der Mitarbeitenden mit den jeweiligen Mitarbeitenden verknüpft. Zweck ist die automatisierte Erkennung von Mitarbeitenden, z.B. beim Betreten des Büros oder bei der Arbeitsplatzsuche.

id - eindeutige Kennung eines Gesichts

face_image - das hochgeladene Bild

face_image_content_type - der Bildtyp des Fotos

vector - Vektordaten, der Fingerprint, für die Gesichtserkennung

vector_status - Status der Vektordaten

employee_id - Fremdschlüssel auf einen Mitarbeiter, der zu dem zugehörigen Foto gehört

display_image - das Profilbild, das für den Mitarbeitenden angezeigt wird. Es kann nur ein Profilbild pro Mitarbeiter auf „true“ gesetzt sein.

Name	Typ	Nullable
id	uuid	false
face_image	bytea	false
face_image_content_type	varchar(255)	false
vector	clob	true
core_image	boolean	false
vector_status	varchar(255)	true
employee_id	uuid	false
display_image	boolean	false

Entität: `clf_employee`

Diese Tabelle enthält alle Informationen zu den Mitarbeitenden des Unternehmens. Neben den Stammdaten:

id - eindeutige Kennung eines Mitarbeitendes

first_name - Vorname

last_name - Nachname

birthday - Geburtstag

Werden auch organisatorische und technische Attribute gespeichert:

core_id - Mitarbeiter-ID in der Core App

location_id - Fremdschlüssel auf den Standort des Mitarbeitenden

azure_user_id - eindeutige Kennung eines Benutzers in Microsoft Entra ID, für Benachrichtigungen oder die Verfügbarkeitsabfrage

availability - Verfügbarkeitsstatus, erlaubte Werte: `homeoffice`, `available`, `away`

Zudem enthält die Tabelle Opt-in-Felder, wobei ausgewählt werden kann, ob:

opt_in_face - Mitarbeitende bei der Gesichtskennung identifiziert werden wollen

opt_in_list - Mitarbeitende auf der Catrin angezeigt werden wollen

Name	Typ	Nullable
id	uuid	false
opt_in_face	boolean	false
first_name	varchar(255)	false
last_name	varchar(255)	false
core_id	integer	false
location_id	uuid	false
birthday	varchar(64)	true
opt_in_list	boolean	false
azure_user_id	uuid	true
availability	varchar(20)	true

Entität: `clf_seat_assignment`

In dieser Tabelle wird festgehalten, welcher Sitzplatz zu welchem Zeitpunkt einer Mitarbeiterin oder einem Mitarbeiter zugewiesen ist. Damit kann Catrin jederzeit anzeigen, wo jemand sitzt oder ob ein Platz frei ist.

id - eindeutige Kennnung der Sitzplatzzuweisung

seat - die Bezeichnung eines Sitzplatzes

employee_id - Fremdschlüssel auf einen Mitarbeiter

location_id - Fremdschlüssel auf einen Standort

from_timestamp, **to_timestamp** - das Start- und Enddatum für die Sitzplatzbuchung

Name	Typ	Nullable
id	uuid	false
seat	varchar(255)	false
employee_id	uuid	true
location_id	uuid	false
from_timestamp	timestamp	false
to_timestamp	timestamp	false

Entität: `shedlock`

Diese Tabelle verhindert konkurrierende Zugriffe auf kritische Prozesse (z.B. Hintergrundjobs), ähnlich einem systemweiten Lock.

Name	Typ	Nullable
name	varchar(64)	false
lock_until	timestamp	true
locked_at	timestamp	true
locked_by	varchar(255)	true

Entität: `clf_notificationlock`

Diese Tabelle steuert Teams-Benachrichtigungen an einen Benutzer, um sicherzustellen, dass ein Benutzer nicht von einer Catrin-Instanz mehrfach gespamt wird. Gespeichert werden:

locked_at - Zeitpunkt der Benachrichtigung

locked_until - Zeitpunkt, an dem die Sperre aufgelöst wird

catrin_id - auf welcher Catrin-Instanz die Nachricht ausgelöst wurde; Fremdschlüssel auf clf_catrin

Name	Typ	Nullable
name	varchar(255)	false
relation_id	uuid	false
locked_at	timestamp	false
locked_until	timestamp	false
catrin_id	uuid	false

Entität: `clf_floorplan`

Diese Tabelle speichert grafische Floorplans:

id - eindeutige Kennung eines Floorplans

name - Name des Floorplans

item_positions - Positionsdaten von Sitzen, Catrin-Instanzen, usw.

image - Bild des Floorplans

Die Tabelle wird mit `clf_location` verknüpft und dient der Visualisierung in der Benutzeroberfläche.

Name	Typ	Nullable
id	uuid	false
name	varchar(255)	false
item_positions	text[]	true
image	bytea	true

Entität: `clf_external_page`

Diese Tabelle enthält externe Links oder Seiten, die im Catrin-UI eingebettet werden können (z.B. Teletext oder Info-Seiten zum Mittagsmenü). Gespeichert werden:

id - eindeutige Kennung der externen Seite

name - Name der Seite

url - URL der Seite

location_id - Zugehörigkeit zu einem Standort

zoom - Zoomfaktor

display_order - Anzeigereihenfolge

Name	Typ	Nullable
id	uuid	false
name	varchar(255)	false
url	text	false
location_id	uuid	false
zoom	float	false
display_order	integer	false

Entität: `clf_activity`

Diese Tabelle verwaltet Aktivitäten (z.B. After-Work-Beer). Aktivitäten lösen Benachrichtigungen an einen Teams-Channel aus, inklusive

display_name - Titel der Nachricht

accouncement_message - die definierte Nachricht

image - Live-Foto der Beteiligten, die die Nachricht auslösen

teams_banner - Foto-Banner für die Nachricht

Name	Typ	Nullable
id	uuid	false
display_name	varchar(255)	false
announcement_message	varchar(255)	false
image	bytea	false
teams_banner	bytea	false

Entität: `clf_activity_assignment`

Diese Tabelle weist Aktivitäten bestimmten Standorten zu.

location - Standort

activity - Aktivität

cooldown_minutes - Anzeigezeit in Minuten, bevor die Home-Ansicht wieder erscheint

Name	Typ	Nullable
location	uuid	false
activity	uuid	false
cooldown_minutes	integer	false

Entität: `clf_script`

Diese Tabelle dient der technischen Verwaltung von Skripten, die automatisiert durch Catrin ausgeführt werden können (z.B. Statusabfragen oder Benachrichtigungen).

Name	Typ	Nullable
id	uuid	false
name	varchar(255)	false
creation_date	timestamp	false
script_content	text	false

Entität: `clf_script_execution_status`

Diese Tabelle dokumentiert, welche Skripte auf welcher Catrin-Instanz laufen und in welchem Zustand sie sich befinden. Zulässige Werte für `execution_state` sind: `RUNNING`, `FAILED`, `SUCCESS`, `RERUN_PENDING`, `NOT_SCHEDULED`.

Name	Typ	Nullable
script	uuid	false
catrin	uuid	true
execution_state	varchar(20)	false
last_update	timestamp	false
output	text	true

3.3.2 Beziehungsformulare

Die folgenden Beziehungsformulare beschreiben die Beziehungen zwischen den Entitäten.

EMPLOYEE_FACE

Relation	EMPLOYEE_FACE
Entitäten	Employee, Face
Beziehung	Employee besitzt Face
Kardinalität	(0..N)
Integritätsregeln	Ein Employee kann ein Face speichern, muss jedoch keines speichern. Faces müssen löschbar sein.

EMPLOYEE_SEATASSIGNMENT

Relation	EMPLOYEE_SEATASSIGNMENT
Entitäten	Employee, SeatAssignment
Beziehung	Employee hat einen Sitz
Kardinalität	(0..1)
Integritätsregeln	Ein Employee hat standardmäßig einen Sitz gespeichert, kann jedoch in Ausnahmefällen keinen besitzen.

SEATASSIGNMENT_LOCATION

Relation	SEATASSIGNMENT_LOCATION
Entitäten	SeatAssignment, Location
Beziehung	Location speichert SeatAssignments
Kardinalität	(0..N)
Integritätsregeln	Wenn ein Standort beispielsweise ausschließlich als Lager dient, sind keine SeatAssignments erforderlich.

EMPLOYEE_LOCATION

Relation	EMPLOYEE_LOCATION
Entitäten	Employee, Location
Beziehung	Employee ist einer Location zugeordnet
Kardinalität	(1..N)
Integritätsregeln	Ein Employee muss mindestens einer Location zugeordnet sein und kann mehreren Locations zugewiesen werden.

ACTIVITY_LOCATION

Relation	ACTIVITY_LOCATION
Entitäten	Activity, Location
Beziehung	Location hat Activities
Kardinalität	(0..N)
Integritätsregeln	Eine Activity benötigt mindestens eine Location. Sie kann mehreren Locations zugeordnet sein.

LOCATION_EXTERNALPAGE

Relation	LOCATION_EXTERNALPAGE
Entitäten	Location, ExternalPage
Beziehung	Location hat ExternalPages
Kardinalität	(0..N)
Integritätsregeln	Eine Location kann optional mehrere ExternalPages speichern.

LOCATION_FLOORPLAN

Relation	LOCATION_FLOORPLAN
Entitäten	Location, Floorplan
Beziehung	Location besitzt Floorplans
Kardinalität	(0..N)
Integritätsregeln	Eine Location kann mehrere Floorplans enthalten (z. B. unterschiedliche Etagen).

LOCATION_CATRIN

Relation	LOCATION_CATRIN
Entitäten	Location, Catrin
Beziehung	Location hat Catrins
Kardinalität	(0..N)
Integritätsregeln	Eine Location kann mehrere Catrins besitzen, beispielsweise verteilt auf verschiedene Etagen.

FLOORPLAN_CATRIN

Relation	FLOORPLAN_CATRIN
Entitäten	Floorplan, Catrin
Beziehung	Catrin zeigt einen Floorplan an
Kardinalität	(0..N)
Integritätsregeln	Pro Catrin wird genau ein Floorplan gespeichert, der zur Anzeige verwendet wird.

CATRIN_NOTIFICATIONLOCK

Relation	CATRIN_NOTIFICATIONLOCK
Entitäten	Catrin, NotificationLock
Beziehung	Catrin speichert NotificationLocks
Kardinalität	(0..N)
Integritätsregeln	Pro Catrin kann es mehrere NotificationLocks geben, um Benachrichtigungen temporär zu sperren.

3.3.3 ERD - Entity Relation Diagram

Das Entity Relation Diagram (siehe Abbildug 3) zeigt die Verbindungen zwischen den Entitäten. Dabei werden insbesondere die Beziehungen sowie die minimale und maximale Kardinalität svisualisiert.

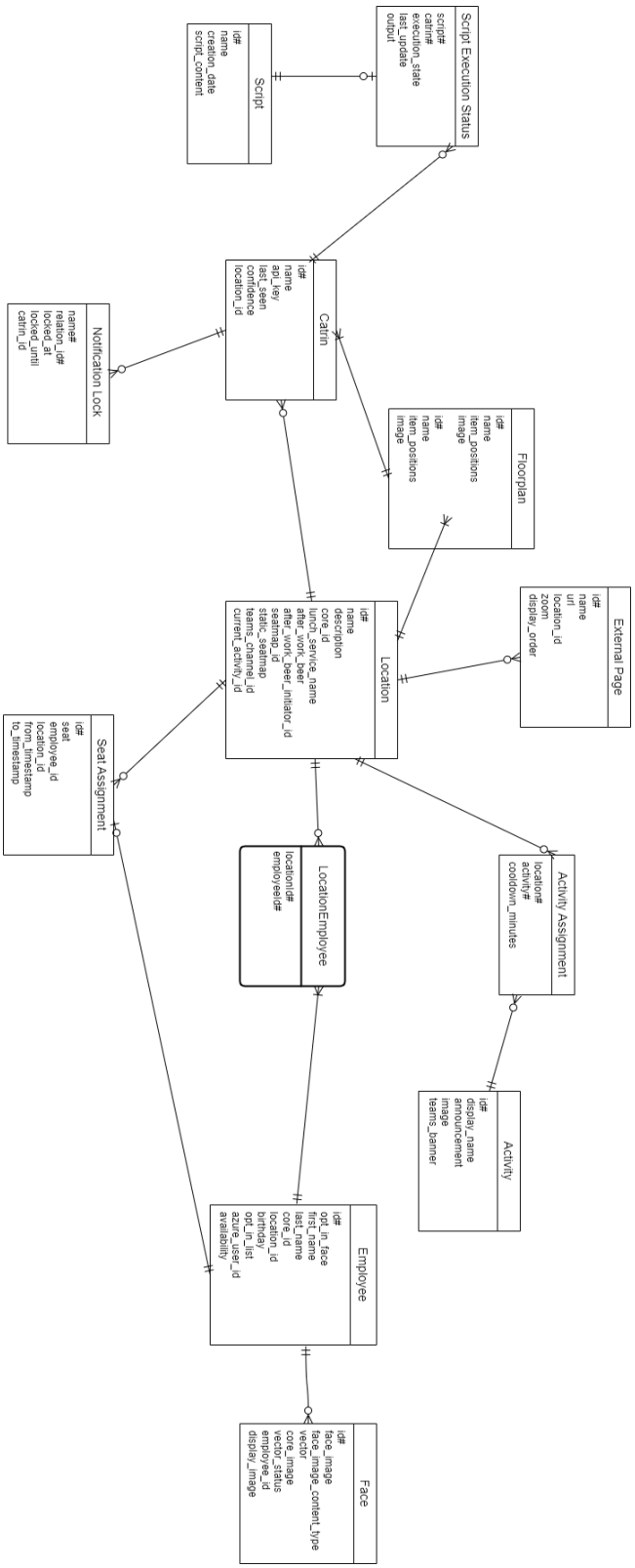


Abbildung 3: ERD Catrin

3.4 Technologien

Folgende Technologien werden dafür von Cloudflight verwendet und werden im Kapital Theoretische Grundlagen (siehe Abschnitt 2) genauer ausgeführt:

- Angular (2.2.1)
- Spring Boot (2.3.1)
- REST-API (2.3.2)
- PostgreSQL (2.4.1)

Technologien, die im Zuge der Erweiterung verwendet werden:

- Microsoft Entra ID (Azure Active Directory) (2.5)
- Microsoft Graph API (2.5.1)
- WebSockets (2.3.3)
- Microsoft 365 Developer Sandbox (2.6.1)
- Microsoft Teams Bot (2.6)

4 Erweiterung Softwaresystem Catrin

In diesem Abschnitt werden die Anforderungen an die Erweiterung beschrieben. Die Anforderungen werden in funktionale, technische Anforderungen sowie in Anforderungen aus Benutzersicht unterteilt.

4.1 Funktionale Anforderungen

Die Erweiterung muss den Nutzerinnen und Nutzern ermöglichen, Sitzplätze für eine ganze Woche zu reserverien. Die bestehenden Webhooks zur Teams-Kommunikation müssen mit einem Teams Bot ersetzt werden und eine bidirektionale Kommunikation ermöglichen. Dadurch können Nachrichten beantwortet und die Antworten der benachrichtigenden Person angezeigt werden.

Neben der Reservierung und Kommunikation ist eine Anzeige geplant, die den Nutzerinnen und Nutzern ermöglicht, die Büro-Anwesenheit von Kolleginnen oder Kollegen zu sehen. Alle funktionalen Anforderungen basieren auf den in Kapitel 1 beschriebenen Use Cases.

4.2 Technische Anforderungen

Der bereits implementierte Code des Systems muss in der jeweils verwendeten Programmiersprache und dem dazugehörigen Framework erweitert werden. Hierbei wird Angular (siehe Abschnitt 2.2.1) für das Frontend verwendet. Für das Backend ist die Kombination aus Spring Boot und Kotlin (siehe Abschnitt 2.3.1) vorgesehen. Die Kommunikation zwischen Frontend und Backend erfolgt über REST-API (siehe Abschnitt 2.3.2).

Für die Erweiterung der Teams-Kommunikation ist angedacht, einen bestehenden Microsoft Teams Bot (siehe Abschnitt 2.6) zu integrieren und entsprechend anzupassen. Für das Testen der erweiterten Funktionalitäten bevor sie ins Produktivsystem integriert werden, eignet sich die Microsoft 365 Developer Sandbox (siehe Abschnitt 2.6.1). Für die Anzeige der Antworten von Teams-Usern auf Nachrichten, die über den Bot übermittelt wurden, sind Webhooks (siehe Abschnitt 2.3.3) zu verwenden.

Zur datenschutzkonformen Abfrage der Anwesenheit von Mitarbeiterinnen und Mitarbeitern wird die Microsoft Graph API (siehe Abschnitt 2.5.1) genutzt. Über diese Schnittstelle kann das System den von den Nutzerinnen und Nutzern selbst eingestellten Anwesenheitsstatus abfragen. Hierbei hat jede und jeder die Möglichkeit, den eigenen Status im Microsoft Teams Account zu setzen und zu bearbeiten.

Die bestehende Systemarchitektur aus Abschnitt 3 darf lediglich durch die Integration des Teams Bots sowie durch Endpunkte für die Kommunikation zwischen Client und Server erweitert und ansonsten nicht verändert werden. Die Verwendung aller weiteren firmeninternen APIs sowie dem Kasvot-System bleiben unverändert.

Alle Code-Erweiterungen müssen den firmeninternen Clean Code Richtlinien entsprechen. Diese umfassen die Vermeidung ungenutzter Variablen oder Imports, eine konsistente Benennung von Methoden und Variablen, klare Strukturierung von Klassen und Modulen sowie die lesbare und wartbare Gestaltung des Codes.

Alle Änderungen im Code werden über das firmeninterne GitLab verwaltet. Commit-Messages müssen dem standardisierten Schema nach Creative Commons [4] folgen. Hier werden Präfixe wie *feat*: für neue Funktionen oder *fix:* für Fehlerbehebungen vorgegeben, um die Art der Änderung zu kennzeichen. Dies stellt sicher, dass Änderungen klar kategorisiert und nachvollziehbar sind.

Für die Erweiterung und Anpassung der Datenbank müssen alle Änderungen in Liquibase-Dateien (siehe Abschnitt 2.4.2) dokumentiert und implementiert werden, um die Versionierung, Nachvollziehbarkeit und Reproduzierbarkeit der Datenbankstruktur sicherzustellen. Dies gewährleistet, dass bestehende sowie neue Datenbankversionen konsistent bereitgestellt und Änderungen kontrolliert nachverfolgt werden können.

4.3 Anforderungen aus Benutzersicht

Neben den funktionalen und technischen Ansichten, ist auch die Benutzersicht ein wichtiger Betrachtungspunkt. Dafür werden die Anforderungen aus den Augen einer Anwenderin oder eines Anwenders betrachtet. Die entwickelnde Person hat den Fokus oft mehr auf Funktionalitäten, während ein User mehr auf Benutzerfreundlichkeit und Visualisierung achtet. Am Ende ist aber der Anwendende derjenige, der täglich mit dem Catrin System in Berührung kommt und somit darf auf dessen Wünsche und Anregungen nicht verzichtet werden.

Zuerst hat das Entwicklungsteam jedoch mit der Betreuerin und dem Betreuer eine Themenstellung erstellt: Im Zuge der Diplomarbeit wird eine Kalenderintegration entwickelt, die es möglich macht, den eigenen Kalender, den von Kolleginnen und Kollegen und einen Team-Kalender zu sehen. So wird immer ein Überblick darüber ermöglicht, wer gerade im Home-Office ist oder frei hat sowie welche Events anstehen. Dazu wird die Sitzplatzzuweisung optimiert, in dem die Anwenderinnen und Anwender ein besseres UI/UX-Design beim Thema „Sitzplan selbst ändern“ und „Auswahl freier Plätze“ bekommen. Außerdem wird eine Out-of-Office-Anzeige basierend auf der Kalender-Integration angedacht. Als letzten Punkt werden die Mitarbeiterinnen und Mitarbeiter einen Sitzplatz gleich für eine ganze Woche reservieren können.

Damit jedoch auch auf die Wünsche von Mitarbeiterinnen und Mitarbeitern eingegangen werden kann, hat sich das Entwicklungsteam dazu entschieden einen Fragebogen an die zukünftigen Nutzerinnen und Nutzer der neuen Funktionalitäten zu senden. Daraufhin wurde die Aufgabenstellung gemeinsam mit der Firma noch einmal überarbeitet.

4.3.1 UI/UX Fragebogen

Wie bereits im vorherigen Absatz beschrieben, ist der Zweck dieses Fragebogens das Miteinbeziehen von Mitarbeiterinnen und Mitarbeiter in die Anforderungsanalyse. Bei der Erstellung einer Umfrage müssen aber spezielle Prinzipien eingehalten werden. Wie von Marktforschung.de [16] beschrieben, ist es sinnvoll, die Fragen in eine logische Reihenfolge zu bringen, damit vom Allgemeinen ins Konkrete gefragt wird. Die Fragen können offen, geschlossen oder halboffen sein. Geschlossene Fragen sind am einfachsten zum Auswerten, da sie eine begrenzte Anzahl an vorgegebenen Antworten beinhalten. Im Gegenteil dazu sind offene Fragen aufwendiger in der Auswertung, können aber zu ganz neuen Erkenntnissen führen, da hier nach einem freien Text gefragt wird. Halboffene Fragen sind ein Kompromiss zwischen diesen beiden Arten. Es werden zwar fixe Antwortmöglichkeiten angeboten, aber bei Bedarf kann immer noch eine selbstverfasste Antwort abgesendet werden.

Der Fragebogen vom Entwicklungsteam besteht aus sieben Fragen und wird mit Microsoft Forms erstellt. Der Titel lautet „Diplomarbeit zu Catrin“. Da Cloudflight ein zweisprachiges Unternehmen ist, wird der Fragebogen auf Deutsch und Englisch erstellt und die Sprache kann jederzeit umgestellt werden. Zu Beginn steht ein Einleitungstext, in dem die Sachlage erklärt und das Entwicklungsteam vorgestellt wird. Die erste Frage ist eine einfache geschlossene Frage, in der gefragt wird, wie oft die oder der Befragte das System Catrin nutzt. Hier gibt es sechs Antwortmöglichkeiten, damit nicht immer der Mittelwert verwendet werden kann. Die nächste Frage behandelt die Zeiten in denen Catrin verwendet wird, es ist eine Mehrfachantwort möglich

und als letzte Option gibt es ein freies Feld, in dem ein personalisierter Text geschrieben werden kann. Mit den nächsten beiden Fragen wird evaluiert, welche aktuell bestehenden Funktionen oft benutzt werden und ebenfalls, wie zufrieden die oder der Befragte mit Catrin ist, anhand einer Skala von eins bis sechs. Die letzten drei Fragen sind offen und fragen folgende Sachlagen ab: Was gefällt dir an Catrin? Was könnte man an Catrin verbessern? Welche neuen Funktionen wünscht du dir für Catrin? Mit diesen Fragen können Mitarbeiterinnen und Mitarbeiter ihre Wünsche oder Verbesserungen kundgeben und somit direkt die Planung beeinflussen.

Der Fragebogen wird Online gestellt und über die firmeninternen Betreuenden an die Zielgruppe gebracht. Jede und jeder kann anonym seine Meinung dazu bekannt geben. Das Entwicklungsteam hat sich aus folgenden Gründen für einen Online Fragebogen entschieden:

- Meinungen können anonym gegeben werden.
- Jede und jeder kann sich so viel Zeit wie nötig nehmen.
- Es hat jede und jeder die Möglichkeit sich einzubringen.

4.3.2 Auswertung Fragebogen

Sobald der Fragebogen von allen teilnehmenden Personen ausgefüllt ist, beginnt das Entwicklungsteam mit der Auswertung der Ergebnisse, um die Vorschläge auch aktiv umsetzen zu können. Insgesamt werden 23 gültige Antworten erfasst.

Die durchschnittlich benötigte Zeit zum Beantworten des Fragebogens beträgt elf Minuten, wobei zwei sehr hohe Ausreißer diese Zeit etwas verfälschen, ohne diese beträgt der Mittelwert etwa fünf Minuten. Die Ergebnisse zeigen, dass Catrin meist regelmäßig genutzt wird. Die Mehrheit der Befragten gibt an, die Anwendung einmal täglich zu verwenden. 15 Personen nutzen Catrin nicht jeden Tag, jedoch gibt keine Person an, die Anwendung öfter als fünfmal pro Tag zu verwenden.

Hinsichtlich der Verwendungszeit zeigt sich, dass Catrin hauptsächlich kurz vor Arbeitsbeginn, sowie nach Arbeitsende zum Einsatz kommt. Das passt auch sehr gut mit den beliebtesten Features zusammen, welche die Sitzplatzsuche (15 Stimmen) und die Feierabend Verabredungen (18 Stimmen) sind. Diese werden vor, beziehungsweise nach dem Arbeiten verwendet. Eine Nutzung während der regulären Arbeitszeit erfolgt seltener (9 Nennungen). Als drittes Feature wird aber auch das Suchen von Kolleginnen und Kollegen (12 Stimmen) oft angegeben.

Auf der Bewertungsskala ergibt sich ein Mittelwert von 4.17 von sechs möglichen Punkten, wobei am öftesten fünf Sterne vergeben werden (10-mal). Dreimal wird eine Höchstbewertung abgegeben, aber auch einmal nur ein Punkt.

In den offenen Antworten werden zahlreiche positive Aspekte genannt. Besonders hervorzuheben sind die einfache Bedienung, die Vermeidung langer Abstimmungsprozesse, die Möglichkeit zur KollegInnen-Suche, Verabredungen, der übersichtliche Sitzplan sowie spezielle Funktionen wie die Geburtstagsübersicht oder saisonale Features. Auch die Tatsache, dass keine Verpflichtung zur Nutzung besteht, wird positiv bewertet.

Für mögliche Verbesserungen werden unter anderem die Möglichkeit, Antworten direkt an Catrin zurückzusenden, Spitznamen hinzuzufügen, Nachrichten mit Fotos zu versenden, neue Restaurants zu integrieren sowie die bestehende Funktion der Gesichtserkennung optional deaktivieren zu können, genannt. Auch organisatorische Aspekte, wie das automatische Entfernen reservierter Plätze bei Homeoffice kommen mehrfach vor.

Darüber hinaus gibt es zahlreiche Wünsche von den Teilnehmenden für neue Funktionen, unter anderem ein Veranstaltungskalender, eine Out-of-Office-Erkennung, eine Wochenübersicht für Sitzplatz- und Homeoffice-Planung, eine Web-Version von Catrin, sowie zusätzliche Übersichten, beispielsweise zur Parkplatzreservierung oder zur Anzeige gemeinsamer Essenspläne.

4.3.3 Anforderungen nach Erhebung

Durch den Fragebogen und der originalen Themenstellung kann sich das Entwicklungsteam auf drei Hauptthemen fokussieren. Da der Wunsch zu einer Out-of-Office-Erkennung mehrfach genannt wird, wird dieses Feature bei der Planung aufgegriffen. Eine Erkennung durch die bereits implementierte Gesichtserkennung ist jedoch nicht möglich, da es gegen Datenschutzrichtlinien verstößt. Damit dieses Feature jedoch trotzdem umsetzbar ist, werden die Daten aus Teams ausgelesen und dann grafisch auf der Catrin angezeigt. Die Rückantwort auf eine Benachrichtigung, die auch als Wunsch geäußert wird, ist als zweiter Punkt angedacht und das dritte Feature ist die Erweiterung der Sitzplatzbuchung. Da sich eine Person die Möglichkeit, Sitzplätze umzubuchen, gewünscht hat, dies jedoch bereits möglich ist, wird eine neue Visualisierung für dieses Feature umgesetzt.

5 Implementierung

Die Implementierung der im vorherigen Abschnitt definierten funktionalen und technischen Anforderungen, sowie die zusätzlichen Wünsche aus Sicht der Anwenderinnen und Anwender werden folgendermaßen umgesetzt.

5.1 SCRUM

Die praktische Umsetzung des Scrum-Frameworks basiert auf der regelmäßigen Anwendung zentraler Scrum-Elemente. Ein wichtiger Bestandteil des Arbeitsprozesses ist daher die Umsetzung von Daily Stand-ups, in denen das Diplomarbeitsteam den **Product Owner (PO)** über den aktuellen Fortschritt und auftretende Probleme informiert und neue Aufgaben in Form von User Stories definiert. Diese 15-minütigen Meetings tragen dazu bei, den Arbeitsprozess fließend zu halten.

Die definierten User Stories werden kontinuierlich verfeinert und jeweils mit einem Fälligkeitsdatum versehen. Dadurch kann eine klare Priorisierung hinsichtlich der Bearbeitungsreihenfolge vorgenommen werden. Die Prioritäten und der Status der User Story sind auf dem Task-Board - im Diplomarbeitsprojekt konkret mithilfe des Tools Jira - visualisiert, das in die Spalten „To Do“, „In Progress“, „In Review“ und „Done“ unterteilt ist.

Die praktische Umsetzung der Diplomarbeit findet innerhalb von fünf 1-Wochen-Sprints statt, die je mit einem Review abgeschlossen werden. Sofern im Rahmen des Reviews festgestellt wird, dass eine User Story nicht alle Kriterien vollständig erfüllt, wird diese im Board wieder zurück zu „In Progress“ geschoben. Das passiert häufig bei den Merge-Requests. Die finalen Merges werden vom PO durchgeführt.

5.2 Usability (UI/UX Design)

Wie in den Anforderungen der Erhebung(siehe Abschnitt 4.3.3) bereits beschrieben, ist das Feature zur Sitzplatzbuchung nicht allen bekannt. Deswegen hat sich das Entwicklungsteam zwei

verschiedene grafische Ansätze überlegt, um dass Feature der Sitzplatzbuchung übersichtlicher zu gestalten und um es intuitiv bedienen zu können.

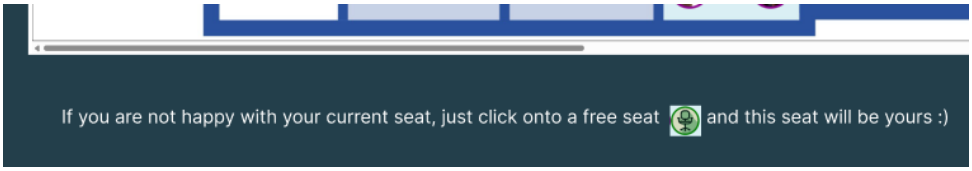


Abbildung 4: Ansatz 1: Möglichkeit 1

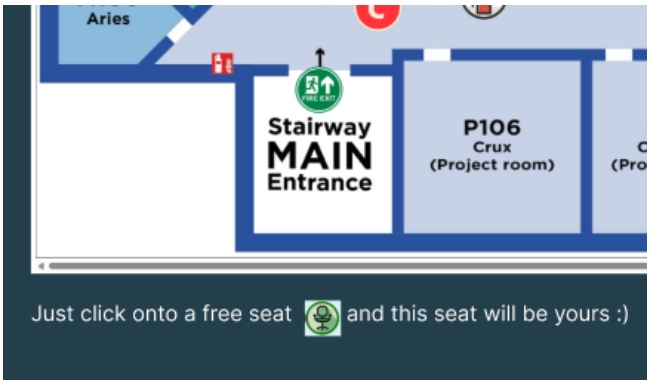


Abbildung 5: Ansatz 1: Möglichkeit 2

Der erste Ansatz ist eine einfache und textuelle Erklärung, wie die Funktion zu bedienen ist. Dieser Text wird unter der Karte mit den Sitzplätzen angezeigt. Die Abbildung 4 beschreibt die erste Möglichkeit, und die Abbildung 5 zeigt eine textuelle gekürzte Variante.

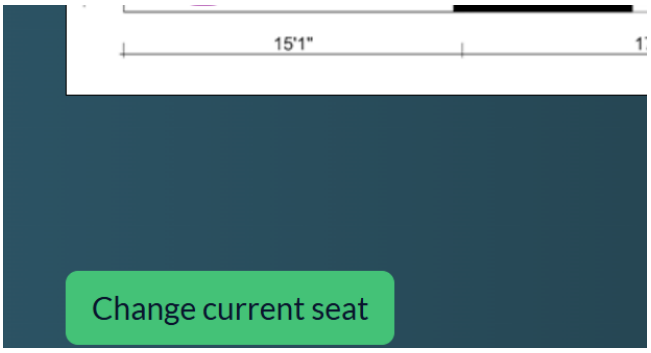


Abbildung 6: Ansatz 2: Button

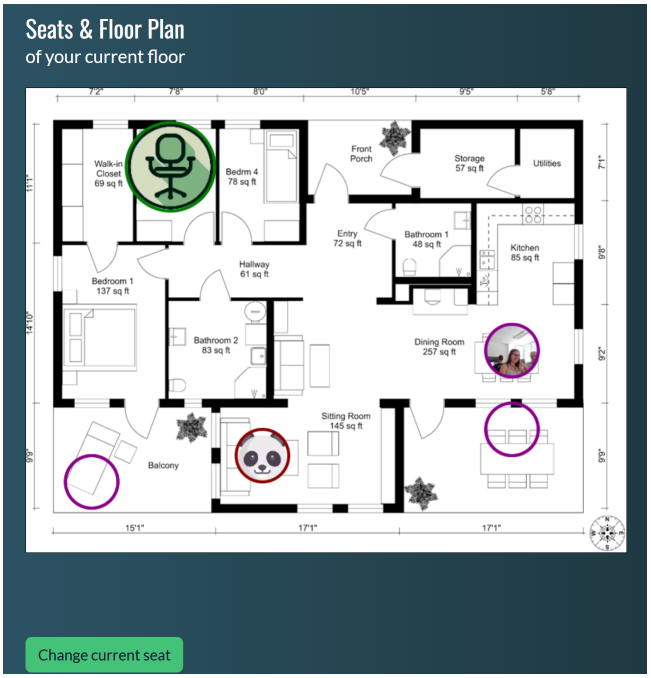


Abbildung 7: Ansatz 2: Pulsierender Sitzplatz

Der zweite Ansatz umfasst, wie in Abbildung 6 ersichtlich, einen Button, an der gleichen Position wie der Text im ersten Ansatz. Die Beschriftung des Buttons beschreibt kurz, dass der Sitzplatz gewechselt werden kann. Wenn der Button geklickt wird, werden alle freien Sitzplätze visuell hervorgehoben. Dafür werden alle Kreise mit den freien Sitzplätzen dynamisch mehrere Male größer und wieder kleiner gezeichnet. Dies ist in der Abbildung 7 erkennbar. Diese Funktionalität wird auch im bestehenden System bereits genutzt, um den Mitarbeiterinnen und Mitarbeiter ihren aktuell zugewiesenen Sitzplatz zu zeigen. Das bedeutet, dass die Art der Visualisierung bereits den Anwendenden vertraut ist.

Nach Besprechung mit dem Auftraggeber wird der zweite Ansatz umgesetzt, da der Button nicht nur ein Hinweis ist, sondern zusätzlich eine echte Funktionalität besitzt und somit auch eine Hilfe für diejenigen ist, denen das Feature schon bekannt ist.

5.3 Feature *Sitzplatzbuchung*

Das erste beschrieben Feature ist die Sitzplatzbuchung. Vor den Änderungen war es nur möglich einen Sitzplatz für den restlichen Tag zu buchen. In der Firma wurde aber der Wunsch geäußert, einen Sitzplatz gleich für mehrere Tage zu buchen. Ein Anwendungsfall wäre, dass eine Mitarbeiterin oder ein Mitarbeiter von Wien für eine Woche am Standort Linz arbeitet und nicht jeden Tag den Sitzplatz neu buchen möchte. In der Erweiterung kann dieser für ein, zwei, drei oder für die restliche Woche gebucht werden. In der durch das Entwicklungsteam durchgeführten

Umfrage wird überdies festgestellt, dass die ursprüngliche Funktion, einen Sitzplatz für nur einen Tag umzubuchen, nicht allen Mitarbeiterinnen und Mitarbeiter bekannt ist. Das Team bemüht sich daher um eine neue Visualisierung des Features. Die Daten kommen von der internen API SeatMap, diese wird neben dem Backend und dem Frontend zusätzlich erweitert.

5.3.1 Design

Für das Design wird ein neues Dialogfenster ausgearbeitet. Als Zusatz zum alten Design gibt es nun eine Auswahlmöglichkeit, um die Dauer der Buchung festzulegen. Eine Option ist, ein Dropdown für die Zeitwerte zu verwenden. Das verstößt jedoch gegen das Prinzip, so wenig Klicks wie möglich zu benötigen, um das Feature zu verwenden. Welches besonders bei einer Touch-Bedienung, wie es hier der Fall ist, von Bedeutung ist. Deswegen werden mehr Buttons erstellt und es wird eine Standardauswahl bereits markiert sein. Die Vorauswahl ist die Dauer von einem Tag, da dies der am meisten verwendete Use-Case ist. Das finale Design ist in Abbildung 8 zu sehen. Auf die richtige Formatierung und Einfärbung kann verzichtet werden, da diese bereits im alten Fenster definiert ist und genauso übernommen werden kann. Der letzte Button im Design zeigt den Text

Until next shuffle, das wird später auf „Until next week“ geändert. Diese Texte beschreiben zwar sinngemäß das gleiche, jedoch kann nicht davon ausgegangen werden, dass alle Anwenderinnen und Anwender Bescheid wissen, wann neu geshuffelt wird. Wie bereits im Abschnitt 3.2.4 beschrieben, werden in der SeatMap regelmäßig die Plätze neu verteilt und danach werden alle Sitzplatzbuchungen ungültig.

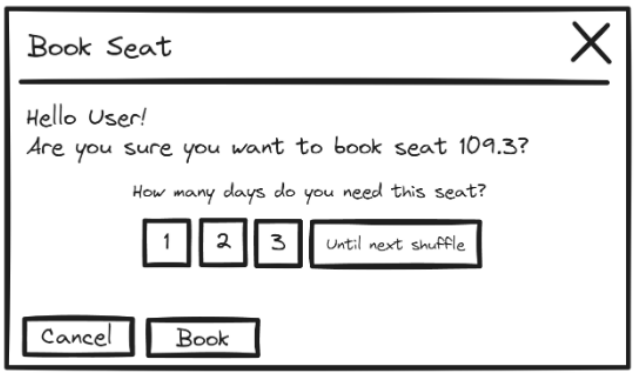


Abbildung 8: Design Dialogfenster Buchung

5.3.2 Frontend

Das oben erklärte Design wird auf dem Catrin Client, auf der Seite Floorplan, im bestehenden Dialog implementiert. Dieser Dialog ist in der Abbildung 9 zu sehen. Der zweite Bestandteil

im Frontend bildet die Anzeigelogik für die Zeitdaten, inklusive der Kommunikation mit dem Backend.

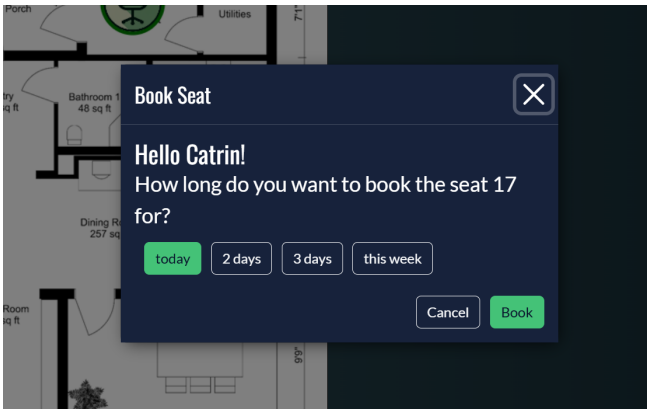


Abbildung 9: Neuer Dialog für die Sitzplatzbuchung

Die Auswahlmöglichkeiten sind dynamisch an den Wochentag angepasst. An einem Freitag werden zum Beispiel nicht mehr alle vier Felder angezeigt. Es wird nur mehr die Option, für einen Tag zu buchen, zum Auswählen angezeigt werden. Diese Logik ist in den Buttons selbst implementiert, die nur dann angezeigt werden, wenn sie noch sinnvoll gebucht werden können. Die Logik ist aber nur für den Betrieb unter der Woche ausgelegt und nicht für das Wochenende. Dafür besteht aber zu diesem Zeitpunkt noch kein Anwendungsfall.

Nach der Bestätigung, wird das API-Service aus dem Backend aufgerufen. Dafür wird die Mitarbeiterin oder der Mitarbeiter, der Sitzplatz, der Standort und die Dauer als Nummer an Tagen übergeben.

5.3.3 Backend

Im Backend ist ein neuer Endpunkt implementiert, der es erlaubt, einen Sitzplatz für mehr als einen Tag zuzuweisen. Anfangs konnte das Frontend die Start- und Endzeitpunkte (from und to) übergeben, die das Backend auf ihre Gültigkeit überprüfte. Dies führte zu Konflikten, da ein Benutzer im Frontend die Buchungsdauer auswählen konnte und die Zeitpunkte im Frontend berechnet werden mussten. Dies gilt es jedoch zu vermeiden. Die Lösung hierfür ist die Anpassung des Endpunkts. Das Frontend übermittelt die gewünschte Buchungsdauer und das Backend berechnet daraus die entsprechenden from- und to-Zeitpunkte.

Dabei muss die Buchungsdauer zwischen ein und fünf Tagen liegen. In der Methode „assignSeat()“, welche durch den Aufruf des Endpoints ausgeführt wird, wird die Mitarbeiterin oder der Mitarbeiter vom vorherigen Sitzplatz entfernt und dem neuen Sitzplatz zugewiesen. Diese Buchung wird in der Datenbank in der Tabelle *Seat Assignment* gespeichert.

Um sicherzustellen, dass manuell hinzugefügte Buchungen nicht von der Seatmap (siehe Abschnitt 3.2.4) überschrieben werden, wird bei der Synchronisation durch den täglich am Morgen ausgeführten Cron-Job der to-Zeitpunkt überprüft. Nur Sitzplätze, welche frei sind oder deren Buchung abgelaufen ist, weil der to-Zeitpunkt in der Vergangenheit liegt, werden überschrieben.

5.4 Feature *Bidirektionale Kommunikation via Teams*

Im Feature Teams-Kommunikation, wird der bestehende Teams-Bot, der nur wenige Funktionen hat, durch einen neuen ersetzt. Dadurch sind die bereits implementierten Funktionen, um eine Aktivität zu starten und einen Mitarbeitenden zu benachrichtigen, angepasst worden. Der neue Teams-Bot hat die gewünschte Funktionalität um auf eine Nachricht, die von Catrin gesendet wird, wieder antworten zu können. Um diese Chatfunktion produktiv nutzen zu können, wird auf die Verwendung von WebSockets umgestellt.

5.4.1 Frontend

Das Frontend muss auf die Nachrichten, welche vom WebSocket kommen, hören und diese grafisch, laut Figma-Design, darstellen. Es gibt ein Service (WebSocketService), welches sich um das Abfangen der Nachrichten kümmert und bei Erhalt einer Nachricht diese an die Chat-Dialog-Komponente weitergibt. Das WebSocketService wird in der Dashboard-Komponente aufgerufen.

Die **Dashboard-Komponente** definiert die Startseite in der Catrin-Applikation, in welcher alle allgemein gebrauchten Services italianisiert werden. Somit wird auch das WebSocketService inklusive der Methode, in der die WebSocket-Verbindung aufgebaut wird, an dieser Stelle aufgerufen.

In dem **WebSocketService** wird zuerst eine Verbindung mit dem WebSocket aufgebaut und anschließend ein „Listener“ implementiert, der auf erhaltene Nachrichten hört. Bevor die Verbindung angefragt wird, wird geprüft ob bereits eine bestehende Session gespeichert ist. Dies ist essentiell, um keine Verbindung zu überschreiben, die bereits erfolgreich hergestellt wurde.

Wird eine Nachricht an Catrin versendet, werden die Daten, bestehend aus dem User und einer Nachricht, ausgelesen. Es wird geprüft, ob es bereits vorhergegangene Nachrichten von diesem User gibt. Ist dies der Fall, werden diese Nachrichten zusätzlich angezeigt, um einen möglichen

Kontext verstehen zu können. Danach wird ein Chat-Fenster mit dieser einen oder mehreren Nachrichten geöffnet. Dazu wird die Komponente Chat-Dialog aufgerufen und der User und die Nachrichten übergeben.

Die **Chat-Dialog-Komponente** wird mit einer Liste von Nachrichten und einem User aufgerufen. An dieser Stelle wird das eigentliche Chat-Fenster geöffnet. Wie in Abbildung 6 zu sehen, wird in der Kopfzeile der Name des Versenders angezeigt und alle Nachrichten sind in Sprechblasen darunter sichtbar. Der Dialog kann mit dem Close-Button in der Fußzeile oder mit dem X in der Kopfzeile geschlossen werden. Anderenfalls wird er nach 30 Sekunden automatisch beendet. Diese Funktion schützt die Privatsphäre, indem verhindert wird, dass die Nachricht lange auf Catrin stehen bleibt und möglicherweise von anderen Personen gelesen wird.

Zur besseren Veranschaulichung wird im nachfolgendem Absatz, sowie in Abbildung 10, ein beispielhafter Ablauf aus Sicht der Mitarbeiterinnen und Mitarbeiter simuliert, um die praktische Verwendung des Tools zu zeigen:

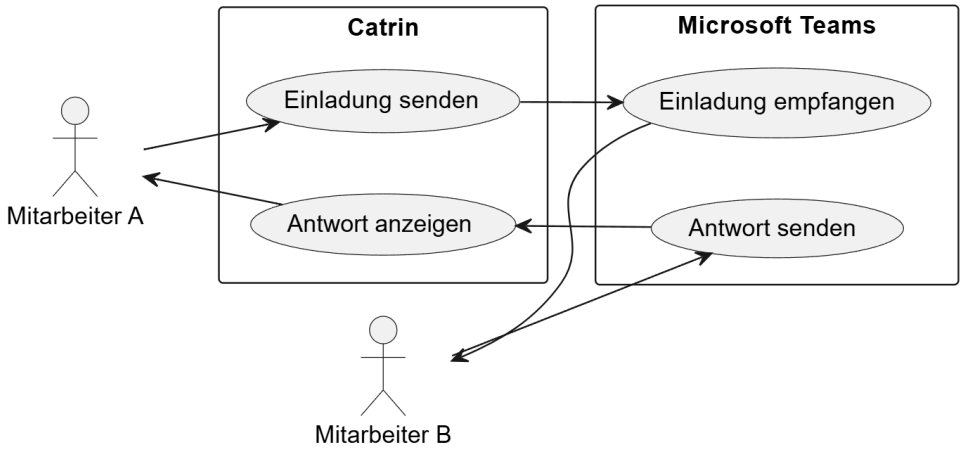


Abbildung 10: Use-Case-Diagram

Ausgangslage:

- Mitarbeiter A: befindet sich beim Eingang vor Catrin und möchte Mitarbeiter B zu einem gemeinsamen Mittagessen einladen
- Mitarbeiter B: sitzt auf seinem Arbeitsplatz

Ablauf:

- Mitarbeiter A: versendet eine Einladung über Catrin an Mitarbeiter B
- Mitarbeiter B: bekommt die Einladung auf Teams und antwortet (siehe Abbildung 11)

Abbildung 11: Screenshot Teams

- Mitarbeiter A: sieht die Antwort auf Catrin und weiß, dass er nur kurz auf Mitarbeiter B warten muss (siehe Abbildung 12)

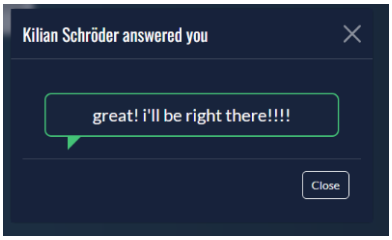


Abbildung 12: Screenshot Catrin

5.4.2 Backend

Im Rahmen der Erweiterung des bestehenden Systems wird die bisherige Webhook-basierte Kommunikation mit Microsoft Teams durch einen Teams Bot ersetzt. Die dafür notwendigen Anpassungen am Backend lagen bereits in Form eines bestehenden Merge Requests vor, werden geprüft und integriert.

Die beiden bestehenden Endpunkte um einen User zu benachrichtigen und eine Nachricht an einen Kanal zu senden, dienen als Schnittstelle zwischen Frontend und Teams Bot. Sie empfangen Nachrichten vom Frontend und leiten diese an die entsprechenden Endpunkte des Teams Bots weiter. Die Endpunkte sind dahingehend erweitert, dass jede Nachricht mit Datum und Uhrzeit in der Datenbank gespeichert wird, um die Kommunikation nachvollziehbar zu dokumentieren.

Da der Teams Bot mit einer Authentifizierung erweitert ist (siehe Abschnitt 5.4.4), werden auch die Endpunkt-Aufrufe des Backends an den Teams Bot angepasst. Beim Aufruf der Azure Functions des Teams Bots */sendMessageToUser* und */sendMessageToChannel* wird ein HTTP-Header *Authorization: Basic <Base64-String>* mitgesendet (siehe Listing 1).

Listing 1: HTTP Header in Spring Boot

```
1
2               webClient.post()
3                   .uri(uri)
4                   .headers { it.addAll(headers) } //username+password
5                   .contentType(MediaType.APPLICATION_JSON)
6                   .bodyValue(body) //message to user
7                   .retrieve()
8                   .bodyToMono(String::class.java)
9                   .block()
```

Für die bidirektionale Kommunikation ist der neue Endpunkt */sendMessageToUser* implementiert, der die vom Teams Bot empfangenen Benutzerantworten verarbeitet. Die zurückgesendeten Nachrichten werden überprüft, ob die zuvor gesendete Nachricht noch nicht länger als eine festgelegte Zeitspanne zurückliegt. Hier sind derzeit zwei Minuten eingestellt, was jedoch konfigurierbar ist. Ausschließlich Antworten, die innerhalb dieses Zeitlimits eingehen, werden an den WebSocketHandler weitergeleitet.

5.4.3 Websockets

Im Backend ist der WebSocket implementiert, der die Nachrichten ans Frontend schickt. Dafür muss zuerst eine Konfigurationsdatei in Kotlin definiert sein. Die WebSocket-Unterstützung wird aktiviert und ein eigener WebSocket-Endpunkt definiert. Bei dieser Arbeit ist der Endpunkt unter „/chat“ erreichbar. Dieser Endpunkt wird im Frontend bei der Verbindung aufgerufen. Die Logik zu dem Websocket ist im WebSocketHandler implementiert. Darin werden alle aktiven WebSocket-Verbindungen mit einem Token als Schlüssel verwaltet und eine eingehende Verbindungsanforderung verarbeitet. Sofern eine Nachricht über das Backend an den WebSocket weitergeleitet wird, muss dieser mithilfe des mitgesendeten Tokens die richtige Session erkennen und die Nachricht (inklusive User) an dieses Frontend weiterleiten.

5.4.4 Chatbot

Um den bestehenden Microsoft Teams Bot lokal ausführen und testen zu können, wird eine Microsoft 365 Developer Sandbox benötigt (siehe Abschnitt 2.6.1).

Beim Erstellen dieser Sandbox ist der Fehler *You do not currently qualify for a sandbox subscription* aufgetreten. Die Ursache für diesen Fehler ist, der neu erstellte Microsoft-Account. Microsoft erkennt daher den Account nicht als aktiven Entwickleraccount an. Da keine eigene

Sandbox erstellt werden kann, wird eine bereits existierende Sandbox mit entsprechenden Adminstratorrechten von Cloudflight verwendet.

Beim Versuch, den bestehenden Microsoft Teams Bot lokal mit Node.js auszuführen, ist der Fehler aufgetreten, dass die *conversation.id* den Wert *null* oder *undefined* hatte. Dieser Fehler verhindert, dass der Bot Nachrichten an Benutzer oder Channels senden kann.

Die *conversation.id* identifiziert eine aktive Konversation im 1:1 Chat (*29:abc123*) oder in einem Channel (*19:abc123@thread.tacv2*) und wird von Microsoft Teams erzeugt [23].

Die Ursache für die fehlende *conversation.id* liegt daran, dass der Bot noch nicht in einem Microsoft Teams-Channel installiert ist. Dies erfolgt über das Admin Center, welches via https://admin.teams.microsoft.com/ erreichbar ist. Der Bot muss zuerst unter „Manage apps“ installiert werden (siehe Abbildung 13).

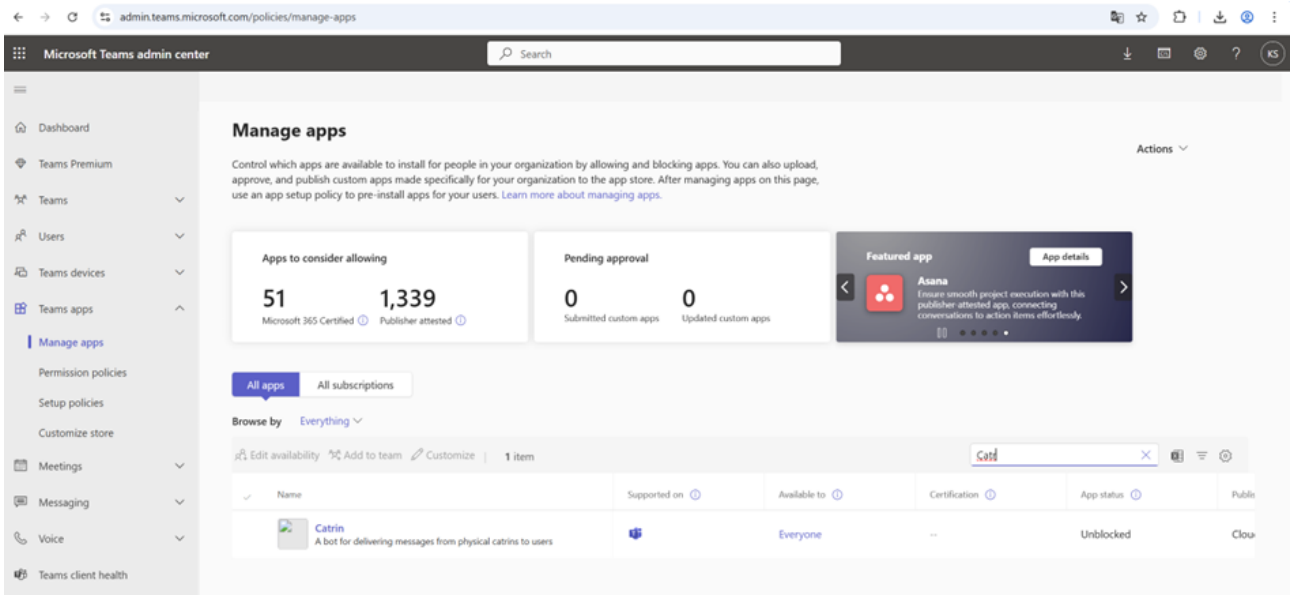


Abbildung 13: Hochladen des Teams Bot im Microsoft Admin Center

Dafür wird der Teams Bot mithilfe dem *Microsoft Teams Toolkit* für Visual Studio Code in das Paket *appPackage.local.zip* komprimiert. Nachdem der Bot als App hochgeladen ist, kann er, ebenfalls über das Admin Center, zu einem Teams-Channel hinzugefügt werden.

Der Teams Bot ist so erweitert und überarbeitet, dass auch Antworten zurückgesendet werden können. Mithilfe des TeamsFx SDK wird ein Command-Handler registriert (siehe Listing 2), der alle Nachrichten abfängt und an die Funktion **handleMessage** weiterleitet. Da alle Nachrichten unabhängig vom Text verarbeitet werden müssen, wird der Ausdruck `.*` verwendet.

Listing 2: Message Handler Azure Function

```
1
```

```
notificationApp.command.registerCommand({
  triggerPatterns: ['.*'],
  handleCommandReceived: handleMessage
});
```

In dieser **handleMessage**-Funktion wird die eingehende Nachricht verarbeitet, indem die UserId, der Benutzername und der Nachrichtentext ausgelesen werden. Die empfangene Nachricht wird per HTTP an folgenden Endpunkt des Spring Boot-Backends weitergeleitet: */sendAnswerToFrontend*. Dieser verarbeitet die Antwort (siehe Abschnitt 5.4.2).

Eine weitere Ergänzung des bestehenden Codes ist die Authentifizierung für beide Endpunkte */sendMessageToUser* und */sendMessageToChannel*, die vom Backend aufgerufen werden und öffentlich über eine URL erreichbar sind. Um sicherzustellen, dass nur das Spring Boot-Backend Nachrichten an den Bot senden kann, wird eine Authentifizierung benötigt. Da Azure Functions kein *Express.js* verwenden, steht keine Middleware wie *express-basic-auth* zur Verfügung. Die Authentifizierung erfolgt daher manuell durch das Auslesen des Authorization-Headers.

5.5 Feature *Verfügbarkeit von Mitarbeitern anzeigen*

Ein weiterer Wunsch der Mitarbeiterinnen und Mitarbeiter ist die Anzeige von einem Verfügbarkeitsstatus auf Catrin. Um unter anderem festzustellen, ob jemand gerade im Büro oder im Homeoffice ist. Diese Information kann auf Teams mithilfe der Microsoft Graph API (siehe Abschnitt 2.5.1) ausgelesen werden. Es gibt drei grundlegende Unterscheidungen:

Verfügbar – „AVAILABLE“ (Die Information wird nur pro Tag und nicht für den aktuellen Zeitpunkt angezeigt. Wenn die Person bereits Mittag nachhause geht, wird trotzdem der Wert verfügbar angezeigt. Es wird auf Ereignisse wie ein Urlaub oder ein Krankenstand geachtet. Das ist von Teams so vorgegeben.)

Nicht da – „AWAY“

Homeoffice – „HOMEOFFICE“

Diese Information ist für jede Mitarbeiterin und jeden Mitarbeiter grafisch auf Catrin einsehbar. Damit wird sofort klar, wer an diesem Tag im Büro ist und sinnvollerweise über Catrin benachrichtigt werden kann. Es hat wenig Sinn, jemanden im Homeoffice zu einem gemeinsamen Mittagessen einzuladen.

5.5.1 Frontend

Im Frontend wird der Verfügbarkeitsstatus vom Backend abgefragt und schließlich angezeigt. Die zwei Bestandteile im Frontend sind dabei das Model und die Anzeige.

Model

Die Daten über den Status sind im Employee-Model gespeichert. Deswegen gibt es keinen neuen Aufruf an das Backend. Das angepasste Model speichert ein neues Feld mit dem Status. Da der Status nur spezielle Werte annehmen darf („Available“ – verfügbar, „Away“ – nicht verfügbar und „Homeoffice“), werden keine Strings gespeichert, sondern es gibt ein Enum, um die Daten nicht immer validieren zu müssen. Somit können nur die drei richtigen Werte gespeichert werden. An allen Stellen, an denen auf eine Mitarbeiterin oder auf einen Mitarbeiter referenziert wird, wird bei der Implementierung des Features geprüft, ob das neue Feld ebenfalls gebraucht wird und auch initialisiert werden muss. An zwei dieser Stellen wird es tatsächlich gebraucht. In der SeatMap und in der Detailansicht einer Mitarbeiterin oder eines Mitarbeiters.

Anzeige

Die erste Variante, wie der Verfügbarkeitsstatus angezeigt wird, befindet sich in der SeatMap und ist in der Abbildung 14 zu sehen. In der Karte wird um einen Sitzplatz ein farbiger Ring gezeichnet. Seit den neuen Änderungen, kann dieser Ring vier verschiedene Farben annehmen:

- **Grün:** Dieser Sitzplatz ist gerade nicht besetzt und kann gebucht werden.
- **Magenta:** Dieser Sitzplatz wird von jemanden besetzt und derjenige ist gerade im Büro.
- **Rot:** Der Sitzplatz ist besetzt, aber der Mitarbeitende möchte nicht offen anzeigen, ob er da ist. (Die Funktion verhindert auch das Anzeigen eines Profilbildes. Jede und jeder kann diese Funktion selbstständig aktivieren.)
- **Grau:** Dieser Sitzplatz ist besetzt, aber die Person ist nicht im Büro.

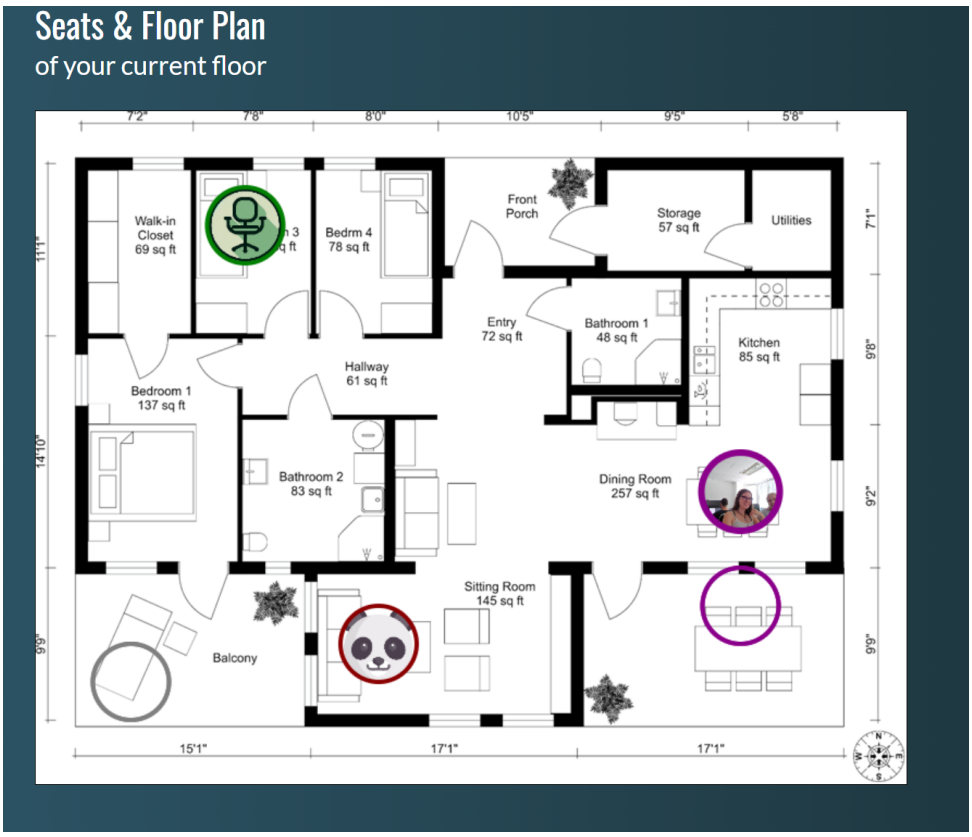


Abbildung 14: SeatMap mit grafischem Verfügbarkeitsstatus

In der Detailansicht, beziehungsweise der Listenelementansicht eines Mitarbeitenden wird ebenfalls der Verfügbarkeitsstatus verwendet. Im HTML werden diese Daten grafisch dargestellt, dies ist in Abbildung 15 zu sehen. Wenn jemand im Homeoffice ist, wird die Person immer noch grün angezeigt, aber mit einem Haus, beschrieben in Abbildung 16. Und wenn jemand nicht verfügbar ist, wird die Hintergrundfarbe zu grau. Das wird in der Styledatei, die im Listing 3 zu sehen ist, festgelegt.

Listing 3: Hintergrundfarbe Personenansicht

```
div.header {
    ...
    &.absent {
        div.presence-indicator-background {
            background-color: $color-checkedin-absent;
        }
        img.employee-img {
            border-color: $color-checkedin-absent;
        }
    }
}
```

Die Logik dahinter legt fest, dass es für jede angezeigte Person die Variable „absent“ gibt, welche richtig oder falsch sein kann. Ist die mitarbeitende Person „absent“, dann wird der Hintergrund

laut „$color-checkedin-absent“ grau angezeigt und auch der Ring rund um das Profilbild der Person wird grau. Ist die Variable richtig, ist die Ansicht wie früher, alle Farben sind grün.

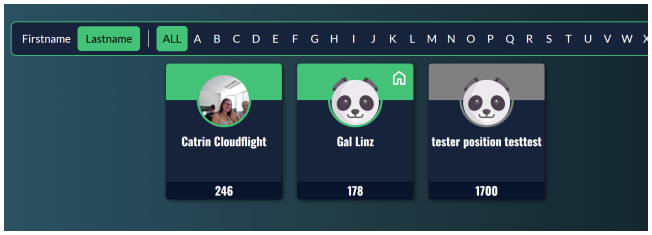


Abbildung 15: Mitarbeitende-Listenansicht

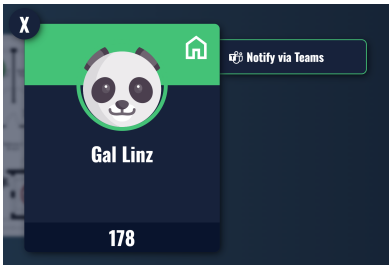


Abbildung 16: Detailansicht

5.5.2 Backend

Das Backend ruft die Verfügbarkeit der Mitarbeiterinnen und Mitarbeiter über deren Teams-Accounts ab, wie diese in Microsoft Teams eingestellt sind. Dafür wird die Microsoft Graph API verwendet.

Microsoft Graph API

Um auf die Microsoft Graph API zugreifen zu können, muss eine App in *Microsoft Entra ID* (siehe Abschnitt 2.5) registriert werden. Dies erfolgt über das Microsoft Azure Portal, erreichbar unter *https://portal.azure.com/*. Damit die App auf Anwesenheitsinformationen zugreifen kann, muss ihr die Berechtigung *Presence.Read.All* zugewiesen werden (siehe Abbildung 17).

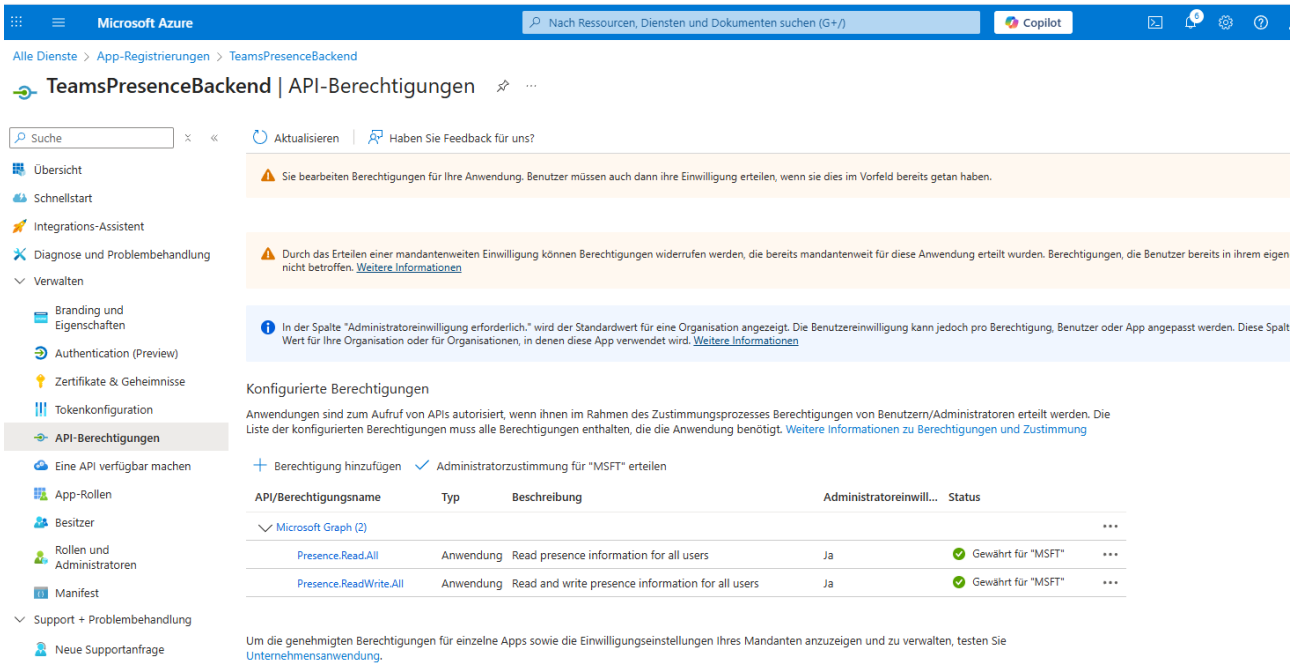


Abbildung 17: Microsoft Entra ID Azure Portal API Rechte

Bei der App-Registrierung wird eine Client-ID, eine Tenant-ID und ein Client Secret erzeugt, die für die Authentifizierung notwendig sind (siehe Abschnitt 2.5.1).

Backend

Das Backend nutzt den *Client Credentials Flow*, um einen Zugriffstoken von Microsoft Entra ID zu erhalten. Dieser Token authentifiziert das Backend gegenüber der Microsoft Graph API.

Das Backend sendet eine Token-Anfrage an folgenden Endpunkt: */{tenant-id}/oauth2/v2.0/token*. In der Anfrage werden die Client-ID, die Tenant-ID, das Client Secret und der Scope *https://graph.microsoft.com/.default* übergeben. Der Scope definiert, welche Berechtigungen der App zugewiesen sind. Microsoft Entra ID prüft die Anmeldedaten und stellt bei erfolgreicher Authentifizierung (siehe Abbildung 18) einen Zugriffstoken aus. Dieser Token muss in allen API-Aufrufen an die Microsoft Graph API im Header übergeben werden.

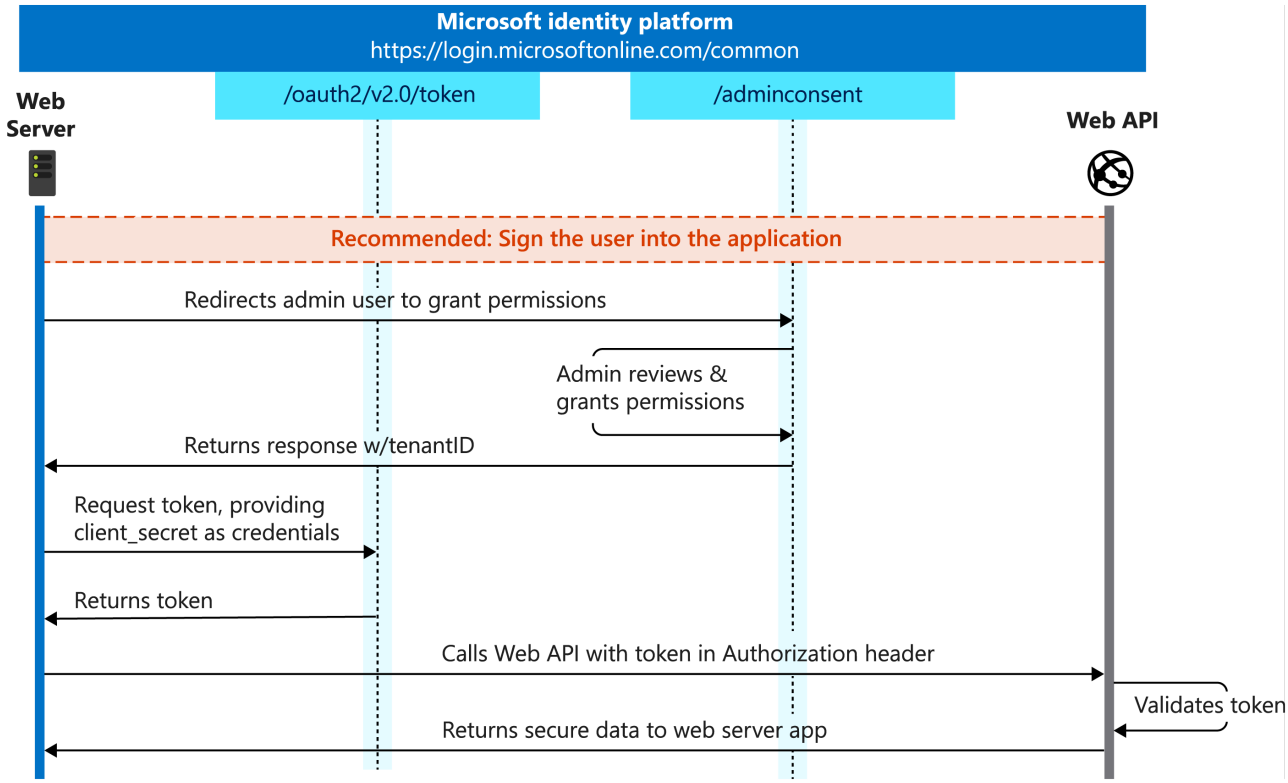


Abbildung 18: Diagramm Informations-Flow der Clientanmeldung ([25])

Über den Endpunkt */users/{user-id}/presence* der Microsoft Graph API kann die Verfügbarkeit von einem Benutzerkonto abgerufen werden. Die *User-ID* ist die *Azure User-ID*, die in der Datenbanktabelle *clf_employee* gespeichert wird.

Der Rückgabewert enthält, wie in Abbildung 19 gezeigt, unter anderem die Felder *availability* und *outOfOfficeSettings:isOutOfOffice*.

```
"availability": "Away",
"activity": "Away",
"sequenceNumber": "A0175871601",
"statusMessage": null,
"outOfOfficeSettings": {
    "message": null,
    "isOutOfOffice": false
}
```

Abbildung 19: Ausschnitt des Rückgabewerts des Endpunkts */users/{user-id}/presence*

Diese Werte werden in drei Zustände übersetzt, die im Frontend angezeigt werden:

away – wenn *availability* den Status *OutOfOffice* oder *OffWork* hat.

homeoffice – wenn *outOfOfficeSettings:isOutOfOffice* auf *true* gesetzt ist.

available – alle anderen Zustände, die nicht als abwesend oder Homeoffice markiert sind.

In der Datenbank wird hierfür in der Tabelle *clf_employee* eine zusätzliche Spalte *available* als ENUM angelegt, das die Werte *available*, *away* und *homeoffice* speichern kann. Dadurch kann das Backend jederzeit auf den aktuellen Status der Mitarbeiter zugreifen.

Um die Daten regelmäßig zu aktualisieren, wird ein geplanter Job (Cron-Job) (siehe Abbildung 4) in Spring Boot implementiert.

Listing 4: Annotation Cron Job

```
1 @Transactional
2 @Scheduled(cron = "0 */5 * * * *")
3 fun updateEmployeesAvailability(): String {
```

Mithilfe der Annotation *@Scheduled* wird die Synchronisation alle fünf Minuten ausgelöst. Die Implementierung ist transaktional, sodass die Datenbankoperationen für alle Mitarbeiter in einer Transaktion ausgeführt werden. So werden keine inkonsistenten Zustände gespeichert, falls ein Fehler während der Aktualisierung auftritt.

6 Evaluierung und Ergebnis

Für das Ergebnis dieser Arbeit werden die UI/UX-Anforderungen, die aus der Benutzerumfrage 4.3.2 hervorgegangen sind, sowie die in Kapitel 1 beschriebenen Use Cases, berücksichtigt.

6.1 Evaluierung UI/UX aus Benutzersicht

Im UI/UX-Bereich wurden die angesprochenen Punkte aus der Anfangsumfrage (siehe Abschnitt 4.3.2) umgesetzt. Zur wichtigsten Anpassung des Benutzerinterfaces zählt der neue Button, um freie Sitzplätze anzuzeigen (siehe Abschnitt 5.2). Dieser zeigt die Existenz des Sitzplatzbuchungsfeature. Außerdem bringt er darüber hinaus auch einen Nutzen für Mitarbeitende, die bereits über das Feature Bescheid wissen. Da das Feature im Zuge der Diplomarbeit um eine längere Sitzplatzbuchung über mehrere Tage erweitert wurde, weist das Interface nun auch auswählbare Zeiten auf (siehe Abschnitt 5.3). Dies wurde zwar nicht explizit von den Teilnehmerinnen und Teilnehmern der Umfrage vorgeschlagen, jedoch ist es eine Verbesserung der Anwendung.

Ein weiterer Punkt aus der Umfrage ist die visuelle Anzeige, ob eine Mitarbeiterin oder ein Mitarbeiter im Homeoffice ist. Dies wurde erfolgreich auf dem System Catrin umgesetzt. Besonders im Fokus war eine verständliche und einheitliche grafische Darstellung, die für Mitarbeitende leicht bedienbar ist (siehe Abschnitt 5.5).

Das Benutzererlebnis wurde besonders im Feature Bidirektionale Kommunikation (siehe Abschnitt 5.4) verbessert. Mit Hilfe dieser Erweiterung kann eine direkte Antwort auf das System Catrin geschickt werden und somit muss niemand mehr unwissend vor Catrin warten und sich fragen, ob die Nachricht gelesen wurde.

6.2 Evaluierung Software

Wie in Kapitel 5 beschrieben, konnten alle zu Beginn definierten Use-Cases aus Kapitel 1 erfolgreich implementiert werden. Die erweiterte Software ist bei Cloudflight bereits im produktiven Einsatz.

Das Problem von ständiger Wartezeit vor einem Catrin-System ist durch das Feature Bidirektionale Kommunikation (siehe Abschnitt 5.4) sowie die visuelle Darstellung des aktuellen Verfügbarkeitsstatus (siehe Abschnitt 5.5) eliminiert. Dies bringt einen hohen Mehrwert für Mitarbeiterinnen und Mitarbeiter und macht den Arbeitsalltag effizienter. Anfangs war geplant, die Verfügbarkeit von Kolleginnen und Kollegen mit dem Outlook-Kalender abzurufen. Aufgrund des Datenschutzes kommt die Microsoft Graph API (siehe Abschnitt 2.5.1) zum Einsatz, die den Status aus Teams ausliest. Dieser kann selbst eingerichtet und festgelegt werden.

Das Feature für die Sitzplatzbuchung über einen längeren Zeitraum (siehe Abschnitt 5.3) erleichtert das Umbuchen eines Sitzplatzes. Vor allem die Organisation bei Team-Arbeiten wird optimiert, da Sitzplätze nicht mehr täglich neu gebucht oder angepasst werden müssen. Dadurch wird der Zeitaufwand für Mitarbeitende reduziert und es kann nicht mehr darauf vergessen werden, den Sitzplatz zu verlängern.

Mittels Code-Reviews und Unit-Tests wurde sichergestellt, dass der Code trotz Erweiterungen sauber strukturiert, wartbar und nachvollziehbar bleibt. Gleichzeitig wurde darauf geachtet, dass alle neuen Funktionen einsatzbereit und den zuvor abgestimmten Anforderungen entsprechend implementiert sind. Es wurde darauf geachtet, dass der zuvor bestehende, nicht von der Erweiterung betroffene, Systemumfang weiterhin fehlerfrei lauffähig ist und das Design für die Mitarbeitenden geläufig bleibt.

Bei einer Beobachtung konnte erkannt werden, dass die neue Buchungslogik bei dem Feature der erweiterten Sitzplatzbuchung grundsätzlich funktioniert, die manuellen Buchungen jedoch von der Seatmap überschrieben werden. Dieses Problem wurde entsprechend adaptiert und behoben.

Durch den gezielten Fokus auf UI/UX wird eine hohe Benutzerfreundlichkeit erreicht, wodurch Mitarbeitende rasch mit der Anwendung vertraut werden. Das Design der Erweiterung passt sich der vorherigen Anwendung an.

7 Erkenntnisse

Durch diese Arbeit konnten nicht nur technische Fähigkeiten erweitert, sondern auch andere wichtige Kompetenzen erlernt werden. Dazu zählen die Relevanz von Kommunikation, Feedback vom Auftraggeber und die ausführliche Beschreibung von User Stories.

7.1 Kommunikation

Eine zentrale Erkenntnis der Erweiterung des Catrin Systems ist, wie wichtig die klare und explizite Kommunikation mit dem Auftraggeber / PO ist. Eine unzureichende Abstimmung hat zu erheblichem Mehraufwand geführt. Dies war bei dem Endpunkt, der in Abschnitt 5.3.3 beschrieben wird, der Fall. Die Parameter mussten im Nachhinein von duration auf to- und from-Timestamps im Client UI sowie im Server Backend angepasst werden. Ähnlich war dies auch bei der Authentifizierung gegenüber dem Teams Bot. Erst später stellte sich heraus, dass die Requests an den Bot ebenfalls mit dem Catrin Client Token (API Key) abgesichert sein müssen. Die nachträgliche Integration der Tokenprüfung führte zu zusätzlicher Logik im Endpoint und Anpassungen im Teams Bot.

7.2 Feedback

Ein weiteres wesentliches Learning ist die Relevanz von regelmäßigem Zwischenfeedback während der Enwicklung. Im Rahmen dieser Diplomarbeit erfolgte das Feedback durch Code Reviews und Merge Requests des entwickelten Codes. Der PO überprüfte die eingereichten Änderungen und gab Rückmeldung, welche Teile angepasst oder ergänzt werden müssen. Dadurch konnte sichergestellt werden, dass alle Anforderungen erfüllt werden.

7.3 User Stories

Im Zuge der Diplomarbeit wurden am Anfang mehrere User Stories für die Features definiert, die in dem Kapitel 1 beschrieben sind. Bei der Sitzplatzbuchung ist die Angabe noch sehr genau definiert und wurde auch genau befolgt. Bei der Teams Kommunikation sind die User

Stories nur mehr vage beschrieben. Wie oben bereits erwähnt, ist zusätzlich die Kommunikation vernachlässigt worden. Daher musste im Nachhinein noch einiges geändert werden. Das Team konnte dadurch aber die Relevanz von genauen Angaben und auch der Kommunikation erkennen und lernen, wie solch ein Fehler zukünftig vermieden werden kann.

8 Ausblick

Abschließend werden mögliche Weiterentwicklungen beschrieben. Das System Catrin ist keine abgeschlossene Applikation, es wird täglich verwendet und so ergeben sich immer wieder neue Erweiterungsmöglichkeiten. Auch das Entwicklungsteam konnte nicht alle möglichen Erweiterungen umsetzen, sondern hat sich nur auf eine begrenzte Auswahl fokussiert. Im folgenden Kapitel werden einige Vorschläge über weitere Erweiterungen erläutert.

Eine denkbare Erweiterung wäre eine zusätzliche Buchung von Parkplätzen. Dieser Wunsch ist in der Umfrage (siehe Abschnitt 4.3.2) genannt worden. Dazu würde ein Plan mit den Parkplätzen pro Standort benötigt werden. Zu entscheiden ist, wo dieses Feature angeboten wird, im Bürogebäude oder auf einem eigenen Gerät in der Nähe der Parkplätze. Dieses müsste zusätzliche Sicherheitsvorkehrungen erfüllen, damit es nicht gestohlen werden kann. Eine andere Option wäre, dieses Feature nur auf einer Online-Applikation anzubieten. Es ist durchaus sinnvoll die Buchungsoptionen etwas anzupassen und nicht direkt von der Sitzplatzbuchung zu übernehmen. Einen Parkplatz, der nur ab dem aktuellen Tag zu buchen ist, ist möglicherweise nicht zielführend. Es müsste die Möglichkeit geben, einen Parkplatz für den nächsten, oder einen bestimmten Tag zu buchen. Bevor dieses Feature implementiert wird, müsste unbedingt eine Nutzwertanalyse durchgeführt werden, um die möglichen Anwenderinnen und Anwender nach ihrer Meinung befragen zu können.

Im bestehenden System gibt es bereits eine Geburtstagsanzeige. Dort werden diejenigen Personen angezeigt, die am aktuellen Tag oder in naher Zukunft Geburtstag haben. Als Erweiterung kann dieses Feature in zukünftigen Projekte mit der Benachrichtigungs-Komponente verbunden werden. Eine Möglichkeit ist einen Button mit der Aufschrift

Gratulieren zu implementieren. Zu beachten ist der mögliche Nachteil, dass voraussichtlich weniger Personen ihre Glückwünsche persönlich übermitteln werden. Ein Kompromiss ist dieses Feature nach Belieben bei sich selbst aktivieren oder deaktivieren zu können.

Durch die Integration des Verfügbarkeitsstatus, ergibt sich die Möglichkeit ein weiteres Feature in diesem Bereich sinnvoll einzusetzen. Bislang kann der Status nur in Teams geändert werden und vor der Implementierung dieses Status auf dem System Catrin wurde die Funktion in Teams nur selten genutzt. Um das Feature zu verbessern, könnte die Option, den eigenen Status direkt

auf Catrin zu setzten oder zu verändern, implementiert werden. Diese Erweiterung würde das vom Entwicklungsteam implementierte Feature noch relevanter machen.

Zukünftige Arbeiten könnten sich auf einen Teams-Kalender auf Catrin konzentrieren, der entweder als Wochen- oder Monatsübersicht dargestellt werden kann. Ziel davon ist interne Veranstaltungen im Unternehmen besser zu kommunizieren. Einträge können zum Beispiel die wöchentlichen Standort Treffen oder andere Aktivitäten wie das Sommerfest oder Ausflüge sein. Eine weitere Möglichkeit sind offene Projektmeetings oder Daily Standup Meetings (siehe Abschnitt 2.1.3). Bei spannenden Projekten, die nicht vertrauliche Informationen beinhalten, kann das überaus interessant für andere Mitarbeiterinnen und Mitarbeiter sein. Technisch muss dafür ein lokaler Teams Kalender erstellt werden, der für alle Mitarbeitende freigegeben wird und der auf Catrin mithilfe der Microsoft Graph API (siehe Abschnitt 5.5.2) angezeigt wird.

Im weiteren Verlauf könnte das System zusätzlich um die Möglichkeit, Sitzplätze für die Zukunft zu buchen, ergänzt werden. Bis jetzt ist nur eine Buchung ab dem aktuellen Tag möglich. In diesem Feature könnte zum Beispiel am Montag ein Sitzplatz für Mittwoch bis Freitag gebucht werden. Die Frage ist, ob das überhaupt Anwendung finden würde. Es würde zuerst wieder eine Anwenderbefragung stattfinden müssen. Zusätzlich wären viele Änderungen am bestehenden System notwendig. Die Abfrage für die Anzeige und die Buchung würden komplexer sein müssen. Wenn ein neuer Sitz gebucht wird, muss nicht nur die Verfügbarkeit des aktuellen Tages, sondern auch die gesamte Verfügbarkeit in der ausgewählten Zeitspanne festgestellt werden. Es dürfen nur die theoretisch möglichen Zeitspannen angezeigt werden.

Insgesamt zeigt sich, dass das System noch lange nicht vollständig abgeschlossen ist und stets neue Erweiterungen sinnvoll werden können. Besonders mit den neu implementierten Features werden möglicherweise wieder neue Wünsche von den Mitarbeiterinnen und Mitarbeiter geäußert. Vom Entwicklungsteam wird empfohlen in regelmäßigen Abständen Benutzerumfragen zur Zufriedenheit mit Catrin durchzuführen und die Ergebnisse mit alten Daten zu vergleichen und bei Bedarf neue Features einzuführen oder alte Funktionalitäten zu verbessern.

Literatur

[1] Atlassian. *Die drei Säulen von Scrum: Transparenz, Überprüfung und Anpassung*. letzter Zugriff am 21.02.2026. URL: https://www.atlassian.com/de/agile/project-management/3-pillars-scrum.

[2] Broadcom Inc. *Kotlin Support*. letzter Zugriff am 04.11.2025. URL: https://docs.spring.io/spring-boot/reference/features/kotlin.html.

[3] Cloudflight GmbH. *Make a Digital Difference | Cloudflight*. letzter Zugriff am 16.11.2025. URL: https://www.cloudflight.io/de/.

[4] Creative Commons. *Conventional Commits*. letzter Zugriff am 05.02.2026. URL: https://www.conventionalcommits.org/en/v1.0.0/.

[5] Cristian Sifuentes. *Understanding Authentication Types: The Complete Developer's Guide*. letzter Zugriff am 15.01.2026. URL: https://dev.to/cristiansifuentes/understanding-authentication-types-the-complete-developers-guide-31kg.

[6] Domain Factory GmbH. *Kotlin vs. Java – was eignet sich wofür?* letzter Zugriff am 04.11.2025. URL: https://www.df.eu/blog/kotlin-vs-java-was-eignet-sich-wofur/.

[7] Universität Duisburg-Essen. *Visualisieren – Diagramme, Infografiken, Tabellen*. letzter Zugriff am 22.01.2026. URL: https://www.uni-due.de/imperia/md/images/iw/de/studium/5_le5_visualisieren_web.pdf.

[8] Florian Deinhard. *Was ist WebSockets?* Letzter Zugriff am 11.02.2026. 15.04.2024. URL: https://www.it-schulungen.com/wir-ueber-uns/wissensblog/was-ist-websockets.html.

[9] GDSYS. *TCP-Verbindung*. Letzter Zugriff am 11.02.2026. URL: https://www.gdsys.com/de/service/glossar/tcp-verbindung.

[10] Geeks for Geeks. *REST API Introduction*. letzter Zugriff am 05.11.2025. URL: https://www.geeksforgeeks.org/node-js/rest-api-introduction/.

[11] GeeksForGeeks. *K-Nearest Neighbor(KNN) Algorithm*. letzter Zugriff am 05.02.2026. 2025. URL: https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/.

[12] Martin Hahn. *Webdesign – Das Handbuch zur Webgestaltung*. Letzter Zugriff am 28.01.2026. URL: https://openbook.rheinwerk-verlag.de/webdesign/.

[13] IBM. *Was ist REST-API?* letzter Zugriff am 26.01.2026. URL: https://www.ibm.com/de-de/think/topics/rest-apis.

[14] IONOS Digital Guide. *HTTP-Request erklärt*. Letzter Zugriff am 11.02.2026. 14.05.2020. URL: https://www.ionos.at/digitalguide/hosting/hosting-technik/http-request-erklaert/.

[15] Davis E. King. *dlib C++ Library*. letzter Zugriff am 05.02.2026. 2002. URL: http://dlib.net.

[16] Marktforschung.de. *Umfrage konzipieren: Wie erstelle ich einen guten Fragebogen?* Letzter Zugriff am 05.02.2026. o. J. URL: https://www.marktforschung.de/diy-research/leitfaden-umfrage-erstellen-alle-wichtigen-schritte/umfrage-konzipieren-wie-erstelle-ich-einen-guten-fragebogen/.

[17] Microsoft. *Access token claims reference - Payload claims*. letzter Zugriff am 25.01.2026. URL: https://learn.microsoft.com/en-us/entra/identity-platform/access-token-claims-reference#payload-claims.

[18] Microsoft. *Add authentication to your Teams bot*. letzter Zugriff am 16.11.2025. URL: https://learn.microsoft.com/en-gb/microsoftteams/platform/bots/how-to/authentication/add-authentication.

[19] Microsoft. *Bot-übersicht*. letzter Zugriff am 16.11.2025. URL: https://learn.microsoft.com/en-gb/microsoftteams/platform/bots/overview.

[20] Microsoft. *Create a bot with the Bot Framework SDK*. letzter Zugriff am 16.11.2025. URL: https://learn.microsoft.com/en-gb/azure/bot-service/bot-service-quickstart-create-bot.

[21] Microsoft. *Einführung in Microsoft Entra Mandanten*. letzter Zugriff am 25.01.2026. URL: https://learn.microsoft.com/de-de/microsoft-365/education/deploy/intro-azure-active-directory.

[22] Microsoft. *Einrichten eines Microsoft 365-Entwickler-Sandboxabonnements*. letzter Zugriff am 24.01.2026. URL: https://learn.microsoft.com/de-de/office/developer-program/microsoft-365-developer-program-get-started.

[23] Microsoft. *ID fields in the Bot Framework - Conversation ID*. letzter Zugriff am 15.01.2026. URL: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-resources-identifiers-guide?view=azure-bot-service-4.0#conversation-id.

[24] Microsoft. *Microsoft Identity Platform und der Fluss von OAuth 2.0-Clientanmeldeinformationen-Abrufen von Token*. letzter Zugriff am 25.01.2026. URL: https://learn.microsoft.com/de-de/entra/identity-platform/v2-oauth2-client-creds-grant-flow#get-a-token.

[25] Microsoft. *Microsoft Identity Platform und der Fluss von OAuth 2.0-Clientanmeldeinformationen-Protokolldiagramm*. letzter Zugriff am 25.01.2026. URL: https://learn.microsoft.com/de-de/entra/identity-platform/v2-oauth2-client-creds-grant-flow#protocol-diagram.

[26] Microsoft. *Neuer Name für Azure Active Directory*. letzter Zugriff am 25.01.2026. 2025a. URL: https://learn.microsoft.com/de-de/entra/fundamentals/new-name.

[27] Microsoft. *Scopes and permissions in the Microsoft identity platform*. letzter Zugriff am 26.01.2026. URL: https://learn.microsoft.com/en-us/entra/identity-platform/scopes-oidc.

[28] Microsoft. *TeamsFx SDK*. letzter Zugriff am 16.11.2025. URL: https://learn.microsoft.com/de-de/microsoftteams/platform/toolkit/teamsfx-sdk.

[29] Microsoft. *Übersicht über adaptive Karten*. letzter Zugriff am 16.11.2025. URL: https://learn.microsoft.com/en-gb/adaptive-cards/.

[30] Microsoft. *Was ist Java Spring Boot?* letzter Zugriff am 03.11.2025. URL: https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-java-spring-boot.

[31] Microsoft. *Was ist Microsoft Entra?* letzter Zugriff am 25.01.2026. 2025b. URL: https://learn.microsoft.com/de-de/entra/fundamentals/what-is-entra.

[32] Microsoft. *Was ist Microsoft Graph?* letzter Zugriff am 12.11.2025. URL: https://learn.microsoft.com/de-de/training/modules/msgraph-intro-overview/2-what-is-microsoft-graph.

[33] Microsoft. *What is Azure Functions?* letzter Zugriff am 16.11.2025. URL: https://learn.microsoft.com/en-gb/azure/azure-functions/functions-overview.

[34] A. Moiseev und Y. Fain. *Angular Development with TypeScript*. letzter Zugriff am 22.01.2026. Manning, 2018. ISBN: 9781638355250. URL: https://books.google.at/books?id=1TgzEAAAQBAJ.

[35] Kawa Nazemi u. a. „Datenvisualisierung“. In: letzter Zugriff am 28.01.2026. Jan. 2021, S. 477. ISBN: 978-3-11-065365-6. DOI: 10.1515/9783110657807-026. URL: https://www.researchgate.net/publication/348603669_Datenvisualisierung.

[36] Stefan Neumüller und Cloudflight. „Catrin Dokumentation“. Technische Dokumentation. 2026.

[37] B. Preim und R. Dachselt. *Interaktive Systeme: Band 1: Grundlagen, Graphical User Interfaces, Informationsvisualisierung*. eXamen.press. letzter Zugriff am 28.01.2026. Springer Berlin Heidelberg, 2010. ISBN: 9783642054020. URL: https://books.google.at/books?id=4asoBAAAQBAJ.

[38] R. Fielding, J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. letzter Zugriff am 26.01.2026. URL: https://datatracker.ietf.org/doc/html/rfc7231#section-4.2.2.

[39] Red Hat. *Was ist eine REST API? Einführung in RESTful Schnittstellen*. letzter Zugriff am 05.11.2025. URL: https://www.redhat.com/de/topics/api/what-is-a-rest-api.

[40] Kenneth S. Rubin. *Essential Scrum: Umfassendes Scrum-Wissen aus der Praxis*. letzter Zugriff am 21.01.2026. 2014. URL: https://books.google.at/books?id=SRySAwAAQBAJ&dq=Scrum&lr=&hl=de&source=gbs_navlinks_s.

[41] Ryte Wiki. *WebSocket*. Letzter Zugriff am 11.02.2026. URL: https://de.ryte.com/wiki/Websocket/.

[42] Florian Schroff, Dmitry Kalenichenko und James Philbin. „FaceNet: A unified embedding for face recognition and clustering“. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. letzter Zugriff am 05.02.2026. IEEE, Juni 2015, S. 815–823. DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[43] Spring Boot. „Spring Boot 3“. In: (2024).

[44] Torsten Stapelkamp. *Informationsvisualisierung*. Letzter Zugriff am 28.01.2026. URL: https://www.torstenstapelkamp.de/Informationsvisualisierung_Stapelkamp_.pdf.

[45] Statistik Austria. *Registrierte Behinderung*. Letzter Zugriff am 04.02.2026. URL: https://www.statistik.at/statistiken/bevoelkerung-und-soziales/behinderung-und-teilhabe/registrierte-behinderung.

[46] M. Steyer und V. Softic. *Angular JS: Moderne Webanwendungen und Single Page Applications mit JavaScript*. letzter Zugriff am 22.01.2026. O'Reilly Verlag, 2015. URL: https://books.google.at/books?id=OwXmBwAAQBAJ.

[47] Technikum Wien Academy. *Was ist Usability?* Letzter Zugriff am 04.02.2026. URL: https://academy.technikum-wien.at/ratgeber/was-ist-usability/.

[48] The PostgreSQL Global Development Group. *About PostgreSQL*. letzter Zugriff am 05.02.2026. URL: https://www.postgresql.org/about/.

[49] Elena Tillmanns. *Usability – was ist das und warum ist es wichtig?* Letzter Zugriff am 04.02.2026. URL: https://www.marketinginstitut.biz/blog/usability/.

[50] Usability.de. *Usability-Test: Nutzerforschung und UX-Testing*. Letzter Zugriff am 04.02.2026. URL: https://www.usability.de/leistungen/ux-testing-nutzerforschung/usability-test.html.

[51] UXMA. *Accessibility Whitepaper*. Letzter Zugriff am 04.02.2026. URL: https://uxma.com/de/accessibility-whitepaper.

[52] Webdesignratgeber.de. *Webdesign Grundlagen – Ihr Leitfaden zur digitalen Gestaltung*. Letzter Zugriff am 04.02.2026. URL: https://webdesignratgeber.de/webdesign-grundlagen-pdf-ihr-leitfaden-zur-digitalen-gestaltung/.

[53] Wikipedia. *Kotlin (Programmiersprache) — Wikipedia, die freie Enzyklopädie*. letzter Zugriff am 04.11.2025. 2025. URL: https://de.wikipedia.org/w/index.php?title=Kotlin_(Programmiersprache)&oldid=260015763.

[54] Wikipedia. *Liquibase*. letzter Zugriff am 05.02.2026. URL: https://en.wikipedia.org/wiki/Liquibase.

[55] Wikipedia. *Scrum*. letzter Zugriff am 21.02.2026. URL: https://de.wikipedia.org/w/index.php?title=Scrum.

[56] Wirtschaftskammer Österreich. *Informationen zum Barrierefreiheitsgesetz – 28.06.2025*. Letzter Zugriff am 04.02.2026. 2025. URL: https://www.wko.at/oe/netzwerke/informationen-barrierefreiheitsgesetz-28-06-2025.pdf.

[57] Yuyang W. (Compile N Run). *Spring Application Context*. letzter Zugriff am 27.01.2026. URL: https://www.compilenrun.com/docs/framework/spring/spring-fundamentals/spring-application-context/.

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang

A Aufgabenverteilung

Im folgenden Punkt ist festgehalten, welche Kapitel der Diplomarbeit von welcher Autorin verfasst wurden.

A.1 Inhaltsverzeichnis Fiona Houdek

A.2 Inhaltsverzeichnis Lea Joppich

B Umfrage

Diplomarbeit zu Catrin

* Required

Hallo zusammen! 👋Wir sind Fiona und Lea - Schülerinnen der HTL Perg und werden diesen Sommer als Praktikantinnen bei euch arbeiten. Im Rahmen unserer Diplomarbeit haben wir die spannende Aufgabe, eine Erweiterung von CATRIN zu entwickeln. Damit diese Erweiterung für euch möglichst nützlich ist, möchten wir eure Meinungen, Ideen und Wünsche einholen. Dafür möchten wir herausfinden, welche neuen Anwendungen und Features euch im Arbeitsalltag mit CATRIN besonders unterstützen würden. Gibt es Funktionen, die ihr euch schon lange wünscht? Oder Situationen, bei denen euch CATRIN noch nicht helfen kann? Die Ergebnisse dieser anonymisierten Umfrage sind für uns besonders wichtig, da sie direkt in die Planung und Umsetzung unserer Diplomarbeit einfließen. Wir wollen eine Lösung schaffen, die nicht nur technisch innovativ ist, sondern auch einen echten Mehrwert für euch bringt. Wir freuen uns über jede Idee und jedes Feedback – damit wir CATRIN noch besser machen! 😊Bei Fragen oder Anliegen zu dieser Umfrage könnt ihr euch jederzeit an 20210039@students.htl-perg.ac.at oder 20210004@students.htl-perg.ac.at wenden!

1. Wie oft nutzt du Catrin? *

- ◯ nie
- ◯ 1x im Monat
- ◯ 1x in der Woche
- ◯ 1x am Tag
- ◯ bis 5x am Tag
- ◯ häufiger

2. Wann verwendest du Catrin? *

- ☐ morgens vor der Arbeit
- ☐ mittags
- ☐ zwischendurch
- ☐ zu Feierabend
- ☐ Other

3. Wofür verwendest du Catrin? *

- ☐ eigene Sitzplatzsuche
- ☐ Kolleg:innen-Suche
- ☐ um Hilfe bitten
- ☐ Mittagessen-Verabredungen
- ☐ Feierabend-Verabredungen
- ☐ Geburtstagsfunktion
- ☐ Other

4. Wie zufrieden bist du mit Catrin? *

wenig zufrieden 😄 😄 😄 😄 😄 😄 total zufrieden

5. Was gefällt dir an Catrin? *

6. Was könnte man an Catrin verbessern? *

7. Welche neuen Funktionen wünschst du dir für Catrin? *

Danke, dass du uns mit deinen wertvollen Ideen und deiner Meinung unterstützt hast.

C Diplomarbeitsplakat

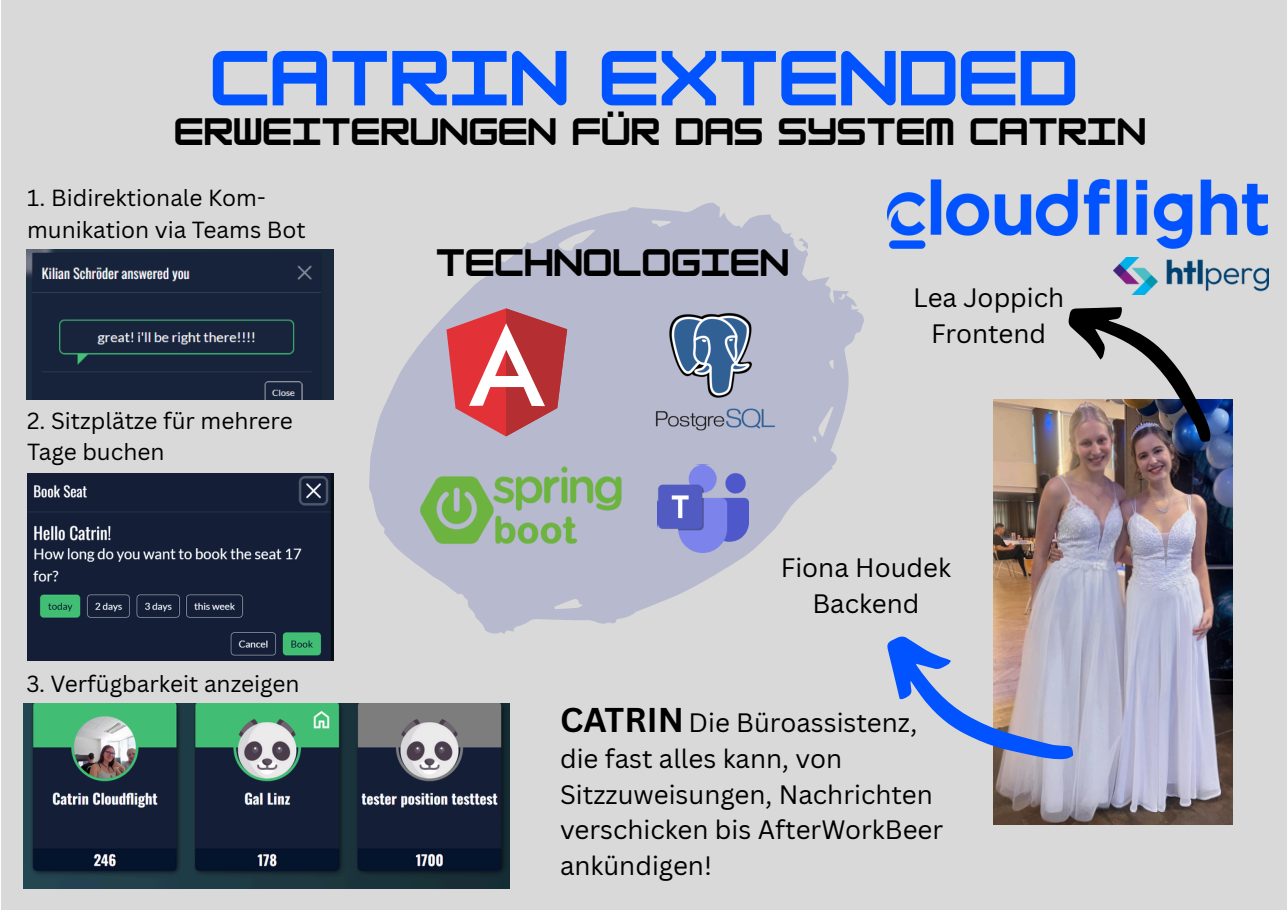


Abbildung 20: Diplomarbeitsplakat