Admin-Portal

Studien- und Bachelorarbeit

Herbstsemester 2022

Autoren:	Manuel Weber
	Yael Schärer
	Rolf Oberhänsli
Betreuer:	Prof. Dr. Markus Stolze
Projektpartner:	2BIT GmbH
	Hochbordstrasse 9
	8600 Dübendorf
Experte:	Markus Flückiger
Gegenleser:	Prof. Dr. Frieder Loch

Inhaltsverzeichnis

Teil I

Abstract, Management Summary & Danksagung

Abstract

Ausgangslage

Das Unternehmen 2BIT betreibt eine Software-as-a-Service Lösung für Aufgaben im Bereich Personalwesen mit dem Namen 2getHR. Diese erlaubt gängige Prozesse wie Zeiterfassung, Ferien- und Spesenmanagement und On-Boarding automatisiert und benutzerfreundlich durchzuführen. Mit zunehmender Beliebtheit der Applikation entsteht für 2BIT ein erhöhter Supportaufwand. Um den Supportprozess zu unterstützen, wurde ein rudimentäres Admin-Portal entwickelt, welches es erlaubte, einige supportrelevante Daten einzusehen und bestimmte Daten zu löschen. Das bestehende Admin-Portal deckte jedoch nicht alle notwendigen Anforderungen ab. Zudem wurde nur eine Auswahl der supportrelevanten Daten abgebildet. In diesem Projekt wurde das bestehende Admin-Portal ausgebaut und benutzerfreundlicher gestaltet, um den Supportprozess zu vereinfachen und effizienter zu gestalten. Das neue Tool erlaubt es alle supportrelevanten Daten einzusehen, zwischen den Objekten zu navigieren, diese zu löschen und klar definierte Änderungen an Daten vorzunehmen.

Ansatz

Die Funktionalität und die Bedienbarkeit des neuen Admin-Portals wurden auf die Endbenutzer-Basis ausgelegt. In einem ersten Schritt wurde eine Analyse der Anforderungen des Supports durchgeführt, um ein Bild der bestehenden Situation zu erhalten. Basierend auf dieser Analyse wurden Anforderungen und Wireframes ausgearbeitet und gemeinsam mit dem Product Owner und dem Supportpersonal von 2BIT auf Bedienbarkeit und Relevanz geprüft. In Form einer Priorisierung wurde bestimmt, in welcher Reihenfolge die Anforderungen umgesetzt werden sollten. Mit der Analyse als Grundlage wurde eine Applikation konzipiert, welche die Bedürfnisse des Supports bestmöglich abdeckt. Während der Konstruktion wurde die Applikation wiederholt mit 2BIT getestet und die Priorisierung angepasst. Das Resultat ist eine Applikation, welche auf die Bedürfnisse von 2BIT abgestimmt ist.

Fazit

Automatisierte und manuelle Tests haben gezeigt, dass das neue Admin-Portal von 2BIT im produktiven Betrieb genutzt werden kann. Die bestehende Funktionalität des alten Tools wurde überarbeitet, fehlende Funktionen ergänzt sowie die Benutzbarkeit erhöht. Ein Ausblick auf weitere Features wurde ausgearbeitet. Diese können von 2BIT zu einem späteren Zeitpunkt umgesetzt werden. Da das Tool im produktiven Betrieb eingesetzt werden soll, wurde besondere Aufmerksamkeit auf die Stabilität und Korrektheit der Features gelegt. Dies wurde mit Tests verifiziert. Um eine möglichst reibungslose Übergabe des Projektes an 2BIT zu ermöglichen, wurde die Applikation nach Vorgaben von 2BIT dokumentiert und der Code gemäss Coding Guidelines und Clean Code geschrieben.

In der folgenden Dokumentation wird ersichtlich, wie die Planung und Umsetzung des Projektes gehandhabt wurde und wie die enge Zusammenarbeit mit 2BIT das Resultat beeinflusst hat.

Management Summary

Ausgangslage

2getHR ist eine Software-as-a-Service Lösung der Firma 2BIT GmbH. Die Applikation unterstützt und automatisiert Prozesse im Personalwesen auf eine einfache und verständliche Art.

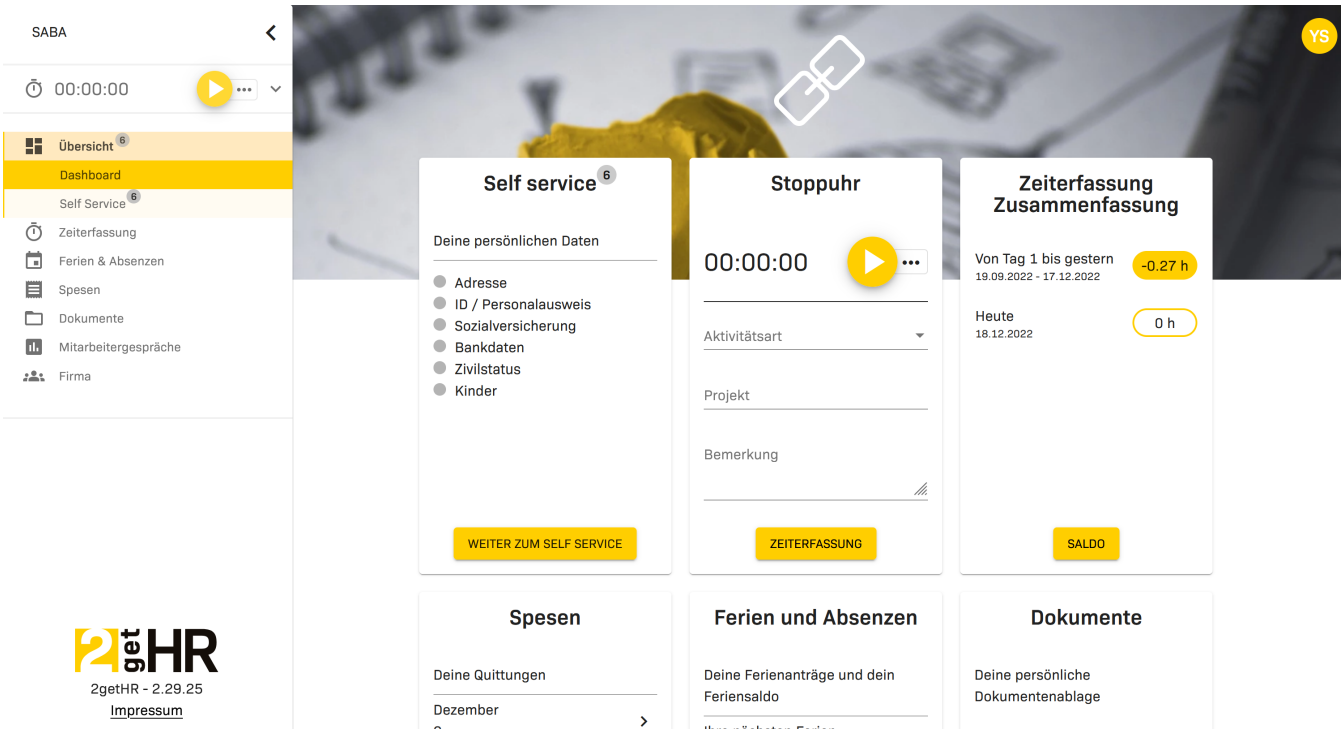


Abbildung 1: 2getHR

Mit der wachsenden Kundenbasis von 2getHR wird die Belastung des Supportteams immer grösser. Um die Mitarbeitenden des Supports von 2getHR zu unterstützen, wurde ein rudimentäres Admin-Portal implementiert. Das Admin-Portal ist ein Interface, das supportrelevante Daten von 2getHR zur Verfügung stellt. Das ursprüngliche Admin-Portal genügt den Ansprüchen des Supportpersonals nicht mehr. Aus diesem Grund wurde von 2BIT in Zusammenarbeit mit dem Projektteam ein Projekt zur Realisierung eines neuen Admin-Portals organisiert. Im Verlauf dieses Projektes sollte ein optimiertes Admin-Portal entstehen, das besser auf die Bedürfnisse des Supportteams angepasst und einfacher zu bedienen ist.

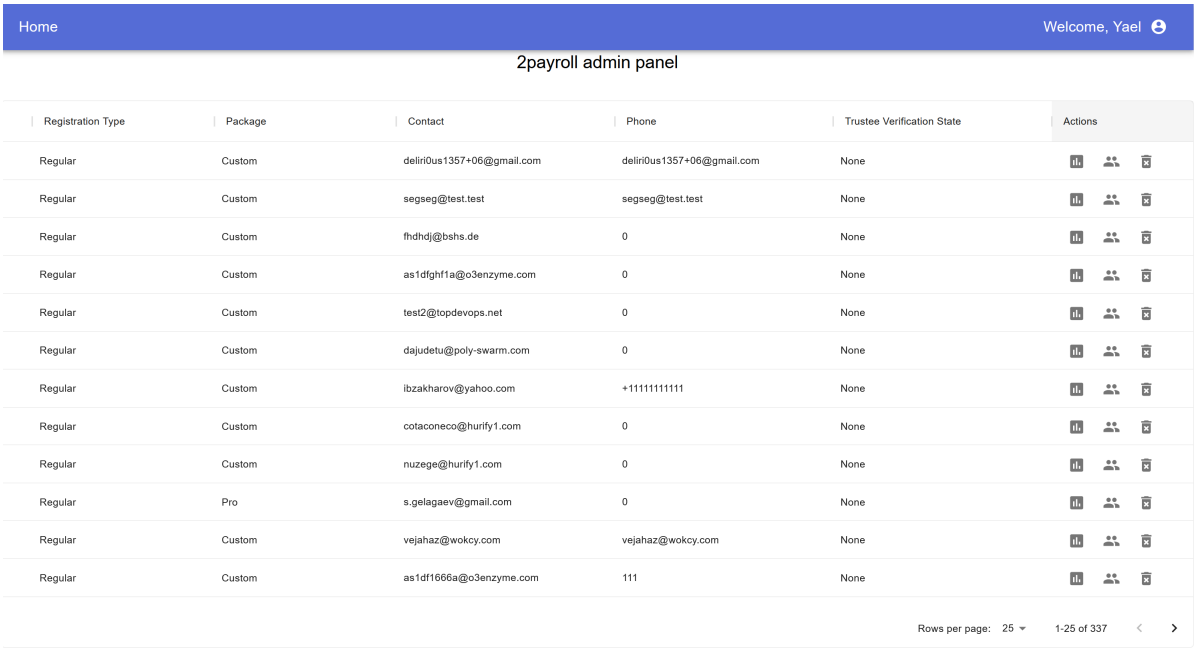


Abbildung 2: Bestehendes Admin-Portal

Vorgehen

In einem ersten Schritt wurde die Projektplanung durchgeführt. Man hat sich dazu entschieden das Projekt nach dem RUP (Rational Unified Process) zu strukturieren. Innerhalb der einzelnen Phasen von RUP wurden die Arbeiten in Sprints nach Scrum organisiert und umgesetzt.

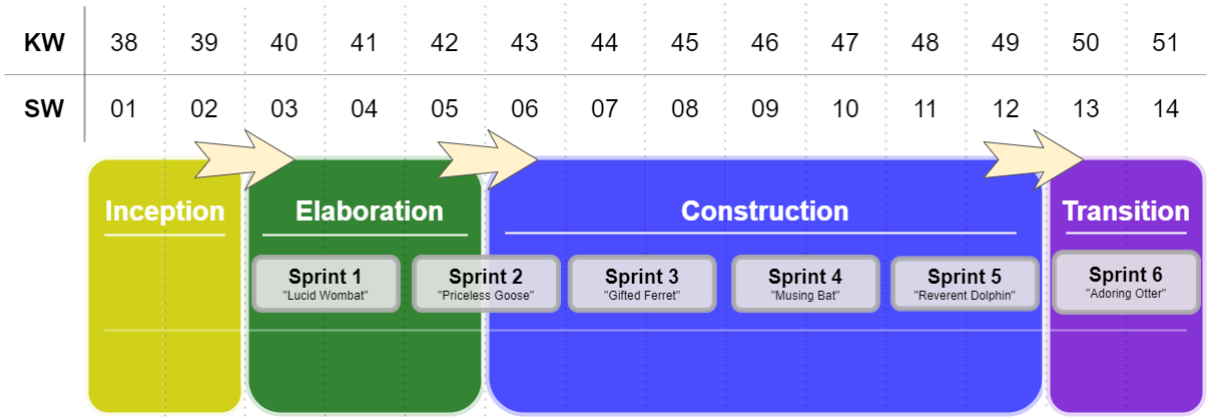


Abbildung 3: Projektplan

Zusammen mit dem Supportpersonal und dem Product Owner wurde in einem nächsten Schritt eine Anforderungsanalyse durchgeführt. Basierend auf dieser wurden Anwendungsfälle ausgearbeitet und gemeinsam mit 2BIT priorisiert. Zudem wurden die nicht funktionalen Anforderungen untersucht und festgehalten. Da 2getHR eine Software im Personalwesen ist und entsprechend viele sensible Benutzerdaten beinhaltet, wurde ein besonderes Augenmerk auf Sicherheit gelegt. Die Architektur der Applikation wurde an das bestehende System angelehnt, damit die Integration in die produktive Umgebung möglichst reibungslos abläuft.

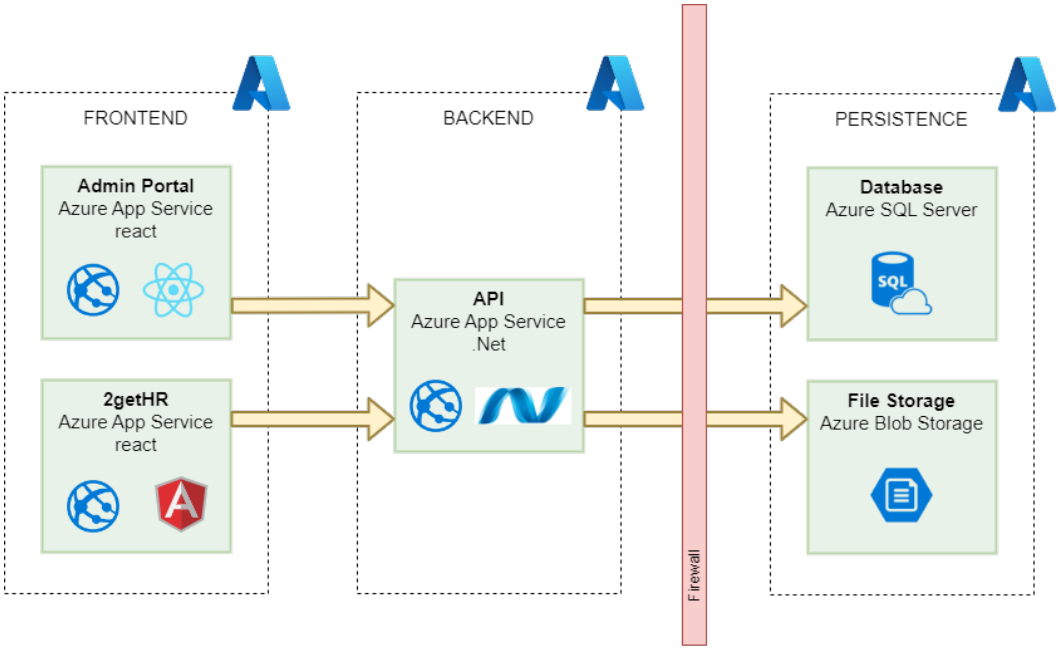


Abbildung 4: Azure Architektur

Nach der Ausarbeitung eines Prototypen wurde in der Konstruktionsphase die Software realisiert. Um eine hohe Qualität des Tools zu erreichen und die Eignung im produktiven Supportprozess sicherzustellen, wurden umfassende Tests ausgearbeitet. Einerseits wurden automatisierte Unit-, Integrations- und End2End-Tests erstellt und andererseits manuelle System- und Usability Tests durchgeführt. Die Tests haben ein zeitnahes Feedback zur Applikation geliefert und allfällige Unstimmigkeiten konnten frühzeitig identifiziert werden.
In den letzten zwei Wochen des Projektes wurden die erarbeiteten Features kontrolliert und verbessert. Die Übergabe des Admin-Portals an 2BIT wurde durchgeführt und eine Empfehlung zur Wartung und Weiterentwicklung abgegeben. Die Applikation wurde in die produktive Umgebung übernommen und wird in Zukunft von 2BIT unterhalten.

Ergebnis

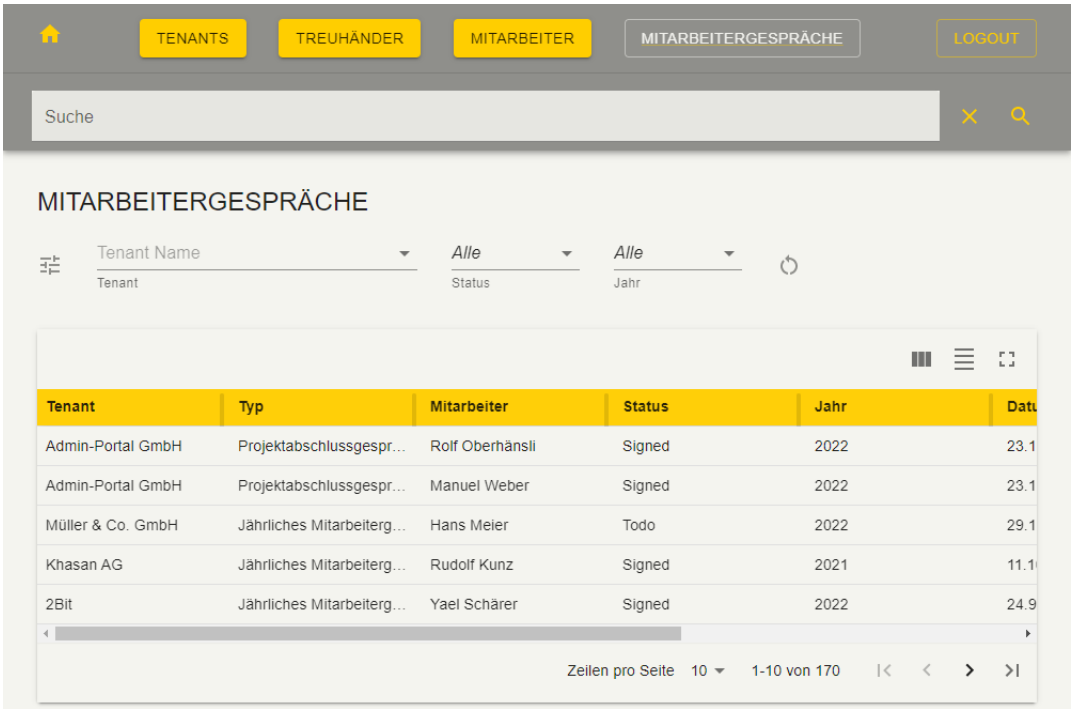


Abbildung 5: Neues Admin-Portal - Listenansicht

Das Ergebnis dieser Bachelor- und Studienarbeit ist ein umfassendes Admin-Portal, das produktionsfähig und auf die Bedürfnisse des Supports angepasst ist. Die Software ist durch automatisierte und manuelle Tests qualifiziert und wurde gemäss dem Prinzip des Clean-Code entwickelt.
Die Benutzerfreundlichkeit des neuen Admin-Portals wurde gemäss den durchgeführten Usability Tests gegenüber der bestehenden Applikation verbessert. Zudem werden im neuen Admin-Portal alle supportrelevanten Daten und die wichtigsten Funktionen angeboten. Die Applikation erlaubt es dem Supportpersonal Informationen für die Ticketbearbeitung einfach zu finden und schnell zwischen den zusammenhängenden Entitäten zu navigieren. Zudem können Mitarbeitende des Supports Daten löschen und klar definierte Datenfelder editieren.

Abbildung 6: Neues Admin-Portal - Detailansicht

Ausblick

In einem ersten Schritt muss das neue Admin-Portal im produktiven Supportprozess von 2getHR etabliert werden. Nachdem sich das Supportpersonal in die neue Applikation eingearbeitet hat, empfiehlt das Projektteam eine Analyse der Situation, um weitere Optimierungen für das Admin-Portal zu identifizieren. Während des Projektverlaufs wurden zusätzliche Anforderungen identifiziert und ausgearbeitet, die im Rahmen dieser Bachelor- und Studienarbeit aus zeitlichen Gründen nicht umgesetzt werden konnten. Aus Sicht des Projektsteams hat das Admin-Portal das Potenzial weiter optimiert und ausgebaut zu werden, um den Supportprozess von 2getHR noch effizienter zu gestalten. Die ausgearbeiteten Anforderungen wurden an 2BIT übergeben und es steht 2BIT frei diese bei Bedarf umzusetzen. Da ein Projektteammitglied bei 2BIT arbeitet, bleibt das während des Projektverlaufs gesammelte Wissen 2BIT erhalten.

Danksagung

Wir möchten uns bei folgenden Personen für die Unterstützung in unserem Projekt bedanken.

Prof. Dr. Markus Stolze Wir bedanken uns herzlich bei Professor Stolze für die kompetente und besonders qualitative Betreuung unserer Arbeit.

Prof. Dr. Frieder Loch Wir bedanken uns bei Professor Loch für die enthusiastische Teilnahme unserer Präsentationen und die aufschlussreichen Inputs, die er uns gegeben hat.

Markus Flückiger Wir bedanken uns bei Herr Flückiger für die Teilnahme an unseren Präsentationen und die hilfreichen Feedbacks, die er uns geben konnte.

Michael Gfeller Wir bedanken uns bei Herr Gfeller für die wertvollen Reviews.

Romain Poulin Wir bedanken uns bei Herr Poulin für die gewinnbringenden Inputs und die Unterstützung während des Projektverlaufs.

Valentina Knecht Wir bedanken uns bei Frau Knecht für die bedeutungsvollen Reviews und Rückmeldungen.

Raphael Ritter Wir bedanken uns bei Herr Ritter für die geschätzten Hilfestellungen und die Initialisierung des Projektes.

2BIT Wir bedanken uns bei allen Mitarbeitenden von 2BIT, mit denen wir eng zusammen arbeiten durften und ohne deren Feedback das Projekt nicht die gezielte Qualität erreicht hätte.

Prof. Dr. Mirko Stocker Wir bedanken uns bei Professor Stocker für das Expertenwissen, das er mit uns geteilt hat.

Teil II

Einleitung

Kapitel 1

Einleitung

1.1 Dokumentstruktur

Dieses Dokument ist grob in vier Teile gegliedert. Der erste Teil ist eine kurze Einleitung in das Projekt. Danach finden sich die Produkt- und Projektdokumentation. Am Ende wird das Ergebnis des Projektes ausgeführt und ein Fazit gezogen. Die Produktdokumentation zeigt alle applikationsrelevanten Analysen und Informationen auf. Zuerst wird die Ausgangslage des Projektes erläutert. Darauf folgt eine ausführliche Anforderungsanalyse, in der die funktionalen und nicht funktionalen Anforderungen spezifiziert werden. Anschliessend werden Themen wie die angewendete Architektur und Qualitätsmassnahmen erläutert. In der Projektdokumentation ist dokumentiert, wie das Projekt aus einer Management-Perspektive geplant und umgesetzt wurde. Schlussendlich wird das Ergebnis der Arbeit genauer analysiert und ein Fazit über den Verlauf des Projektes und das Endprodukts gezogen. Auch wird ein Ausblick auf ein weiterführendes Vorgehen gegeben.

1.2 Ausgangslage

Der Softwarehersteller 2BIT GmbH hat eine Software-as-a-Service-Lösung mit dem Namen 2getHR für die Unterstützung von Prozessen im Personalwesens entwickelt und vertreibt diese. Die Software bietet die Möglichkeit Prozesse wie Zeiterfassung, Ferien- und Spesenmanagement und On-Boarding automatisiert, einfach und zentral über ein Tool zu managen. Das Produkt ist nun seit einiger Zeit im Einsatz bei Kunden von 2BIT und geniesst eine steigende Beliebtheit. Der stetig wachsende Kundenstamm und der daraus resultierende erhöhte Supportaufwand verlangt Lösungen zur Unterstützung des Supportprozesses. 2BIT hat dazu vor einem Jahr ein rudimentäres Admin-Portal entwickelt, welches das Supportpersonal bei der Bearbeitung von Supportfällen unterstützt. Dieses Portal hat die Aufgabe dem Support schnell und einfach Informationen über die Kunden von 2getHR zu liefern und das Mutieren von bestimmten Daten zu ermöglichen.

1.3 Problemdefinition

Das aktuelle Admin-Portal entspricht nicht mehr den Anforderungen des Supportpersonals von 2BIT. Das rudimentär gehaltene Tool kann nur wenige Daten anzeigen und stellt nicht alle benötigten Funktionalitäten zur Verfügung. Das Supportpersonal, das selbst nicht stark technik-affin ist, muss zu oft auf die Hilfe der Entwickler zurückgreifen, damit zusätzliche Informationen aus der Datenbank herausgelesen werden und kann Supportfälle somit nicht selbstständig lösen. Dies bedeutet einen grossen Mehraufwand für alle involvierten Parteien und verzögert den Abschluss von Supporttickets, was für 2BIT nicht tolerabel ist. Zu den mangelnden Funktionalitäten und Daten ist das Portal unübersichtlich gestaltet und umständlich bedienbar. Eine bessere Lösung muss implementiert werden. Die genaue Aufgabenstellung kann im Anhang eingesehen werden.

1.4 Vision

Das Projektteam hat sich zum Ziel gesetzt basierend auf einer Analyse der Supportanforderungen eine neue Applikation auszuarbeiten, die das bestehende Admin-Portal verbessert und erweitert. Es sollen alle supportrelevanten Daten sowie Funktionalitäten vorhanden sein, um einen rund um Supportdienst leisten zu können. Mit dem neuen Admin-Portal soll eine nicht technisch ausgebildete Person Supporttickets eigenständig und abschliessend lösen können. Entwickler sollen nur zugezogen werden müssen, wenn das Ticket sich auf technische Implementationen bezieht. Auch soll der Zeitaufwand für Supportfälle so stark wie möglich reduziert werden.
Dazu muss die bestehende Lösung vollständig überarbeitet und verbessert werden. Die Benutzerfreundlichkeit sowie das Design müssen ansprechend und intuitiv gestaltet sein, damit der

bestehende und der zukünftige Support keine Schwierigkeiten beim Um- beziehungsweise Einstieg hat.

1.5 Herangehensweise

Das Projekt wurde gemäss dem Rational Unified Process (RUP) in vier Phasen unterteilt: Inception, Elaboration, Konstruktion und Übergabe. Innerhalb dieser Phasen wurde nach SCRUM in zweiwöchigen Sprints geplant und gearbeitet. In der Inception-Phase wurde das Projektmanagement aufgesetzt und die genaue Aufgabenstellung ausgearbeitet. Während der Elaborationsphase wurde eine intensive Studie der bestehenden Situation und der Anforderungen eng zusammen mit 2BIT durchgeführt und dokumentiert. Somit wurde sichergestellt, dass das Problem genau verstanden wurde. Daraus resultierten formalisierte Anforderungen, funktional und nicht funktional, und ein Prototyp der umzusetzenden Architektur. Auch wurde die Machbarkeit und der Umfang des Projektes überprüft und angepasst. In der Konstruktionsphase wurde die angestrebte Lösung iterativ umgesetzt und intensiv getestet. Um die Eignung des fertigen Produktes für einen produktiven Supportbetrieb zu kontrollieren, wurden fortlaufend Usability Tests mit der Benutzerbasis durchgeführt und Anforderungen durch Systemtests untersucht. In der letzten Woche des Projektes wurde eine Übergabe an 2BIT durchgeführt, wobei der Stand des Produktes präsentiert wurde. Schlussendlich konnte die Applikation erfolgreich in die produktive Umgebung von 2BIT migriert werden.

Teil III

Produkt Dokumentation

Kapitel 2

Analyse Ausgangslage

Das in diesem Projekt geplante Admin-Portal ist eine ergänzende Applikation zum bestehenden Tool 2getHR von 2BIT. Die Applikation wird das Bearbeiten von administrativen und Support Aufgaben rund um 2getHR erlauben. Es besteht bereits ein Admin-Portal mit grundlegender Funktionalität. In diesem Projekt wird auf dem bestehenden Admin-Portal aufgebaut. Im folgenden Kapitel werden das bestehende Admin-Portal, die Applikation 2getHR und alle weiteren Komponenten vorgestellt.

2.1 2getHR

2getHR ist eine Web-Applikation für Human Resource Management. Die Software erlaubt Onboarding, Timetracking, Absenzen-Management, Durchführung von Mitarbeitergespräche, Management von Dokumenten und mehr. Die Applikation ist bereits produktiv in Einsatz bei 2BIT und einigen Kunden der Firma. Mit der wachsenden User-Base muss nun der Supportprozess optimiert werden. Das bestehende Admin-Portal ist ein wichtiger Teil dieses Prozesses und soll nun stark optimiert werden.
2getHR besteht aus einer Angular-Progressive Web Application, einem .NET-Backend welches in einer Azure Web Application Service gehostet ist, einer Azure SQL Server Datenbank und einer Azure Storage Subscription für die Persistenz von Dokumenten. Die Applikation ist komplett in der Cloud gehostet. Die bestehende Azure Architektur bleibt erhalten und wird im Verlauf des Projektes nicht geändert.

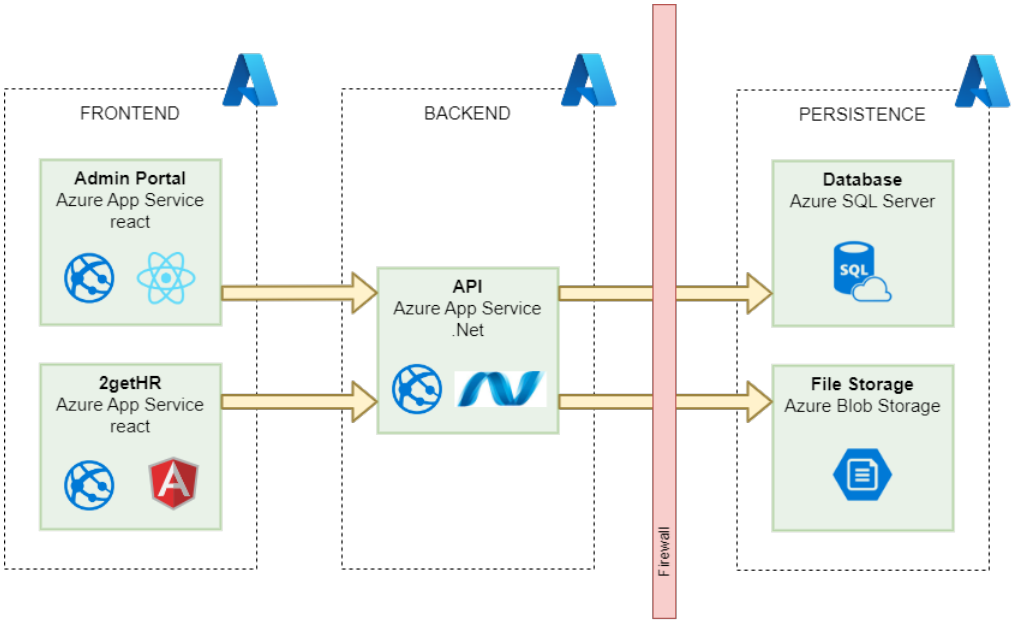


Abbildung 2.1: 2getHR Componenten

Frontend

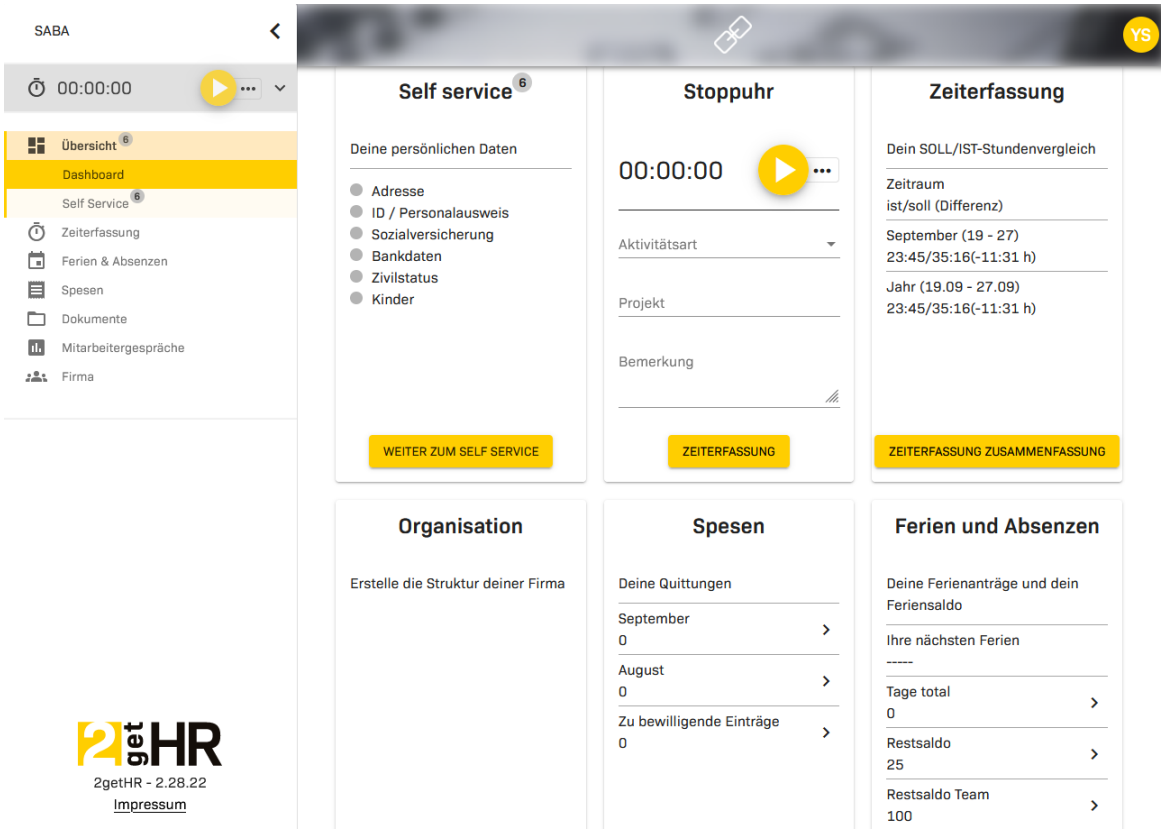


Abbildung 2.2: 2getHR GUI

Das Frontent ist eine Angular-SPA, die mit Typescript geschrieben ist. Die Progressive Web Applikation legt Wert auf Usability mit responsive Design, Push Nachrichten und integrierter Kamera Funktionalität.

Backend

Das Backend beinhaltet folgende Komponenten für das Backend.

- .NET Core / AspNetCore
- Entity Framework
- Azure WebJobs
- WebPush für Push Notifications
- SendGrid für Email Notifications
- Azure SQL Server Datenbank
- Azure Storage für Dokumentenspeicherung

2getHR API

Eine Swagger-Dokumentation des 2getHR APIs kann unter https://i-2payroll-api.azurewebsites.net/swagger/index.html# eingesehen werden.

2.2 Admin Portal

Frontend

Das Admin-Portal ist eine React Applikation mit Yarn Package Manager. Sie zeigt als Homepage eine Liste von Tenants von 2getHR mit Support relevanten Informationen an.

Abbildung 2.3: 2getHR Admin Portal GUI Homepage

In der Tabelle alles Tenants werden zur Zeit die folgenden Spalten angezeigt.

- ID
- Name
- Last Login
- Days Since Last Login
- Amount Of Users
- Registration Date
- Registration Type
- Package
- Contact
- Phone
- Trustee Verification State
- Actions: Appraisals, Employees, Delete

Für jeden Tenant können drei Aktionen im Portal ausgeführt werden: Tenant löschen (Delete), Mitarbeiter anzeigen (Employees) und Mitarbeitergespräche (Appraisals) anzeigen.

Page Mitarbeiter

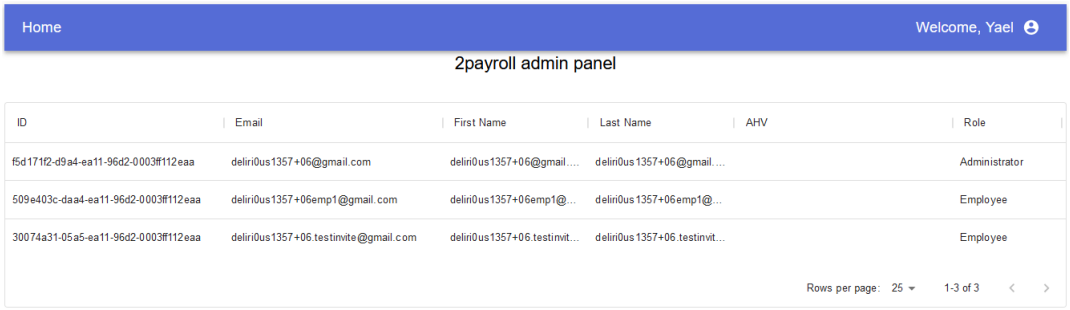


Abbildung 2.4: 2getHR Admin Portal GUI Mitarbeiter

Über die Aktion Mitarbeiter kann eine Liste aller Mitarbeiter des Tenants mit minimalen Details angezeigt werden. Diese Liste enthält die folgenden Spalten (genauere Informationen können in Kapitel Domain Analyse 4 eingesehen werden):

- ID
- Email
- First Name
- Last Name
- AHV
- Role

Page Mitarbeitergespräche

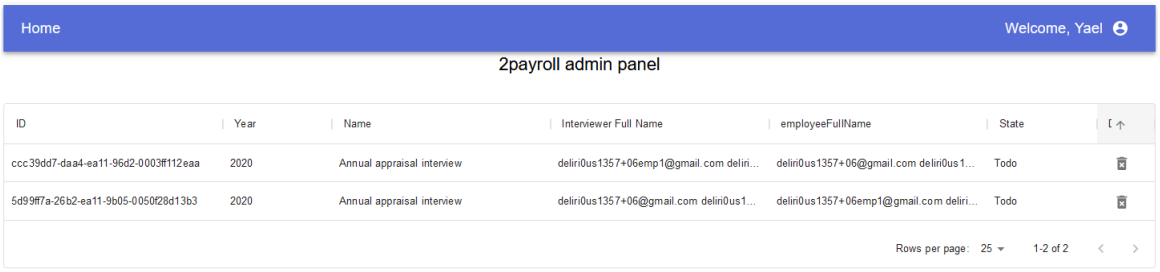


Abbildung 2.5: 2getHR Admin Portal GUI Mitarbeitergespräch

Mit der Aktion Mitarbeitergespräche anzeigen wird eine Liste aller Mitarbeitergespräche des Tenants mit minimalen Informationen angezeigt. Anders als die Mitarbeiter können Mitarbeitergespräche wie Tenants gelöscht werden. Die Liste enthält folgende Felder (genauere Informationen können in Kapitel Domain Analyse 4 eingesehen werden):

- ID
- Year
- Name

- Interviewer Full Name
- emplyoyeeFullName
- State
- Aktion: Delete

Komponentendiagramm des Admin-Portals Frontend

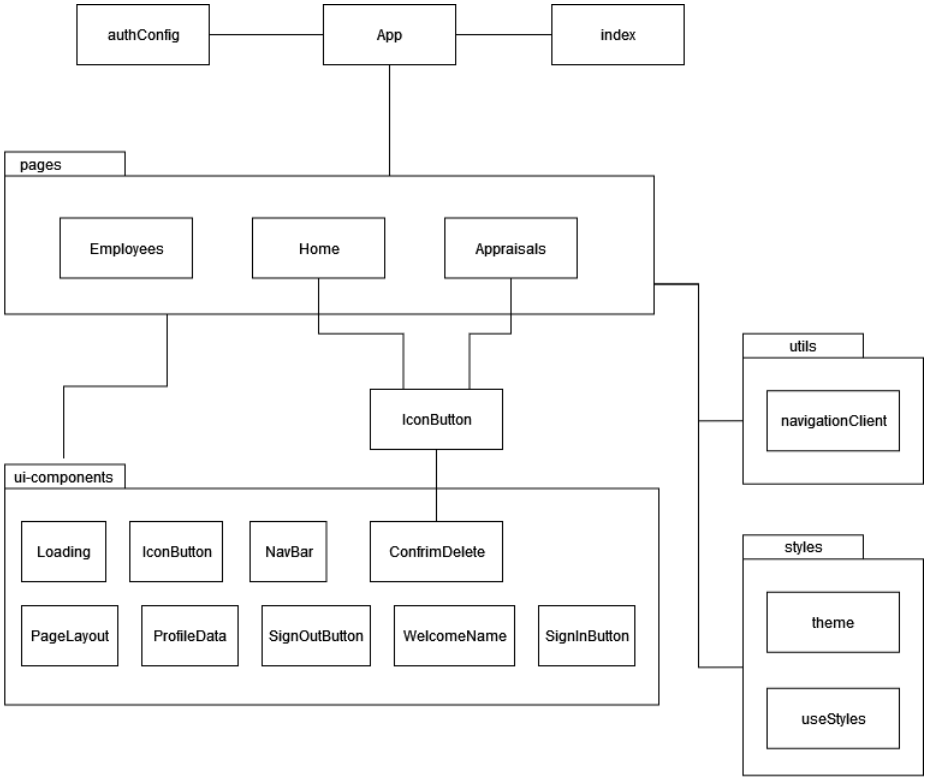


Abbildung 2.6: 2getHR Admin Portal Current Class Overview

Backend

Klassendiagram

Das Admin-Portal Frontend greift auf ein eigenes API im Backend zu, das AdminAPI. Dieses .NET Projekt definiert die Routen, mit denen das Admin Portal Frontend Daten von 2getHR abfragen kann. Dazu greift das API auf ein Projekt 'Core' zu, das mit 2getHR geteilt wird.

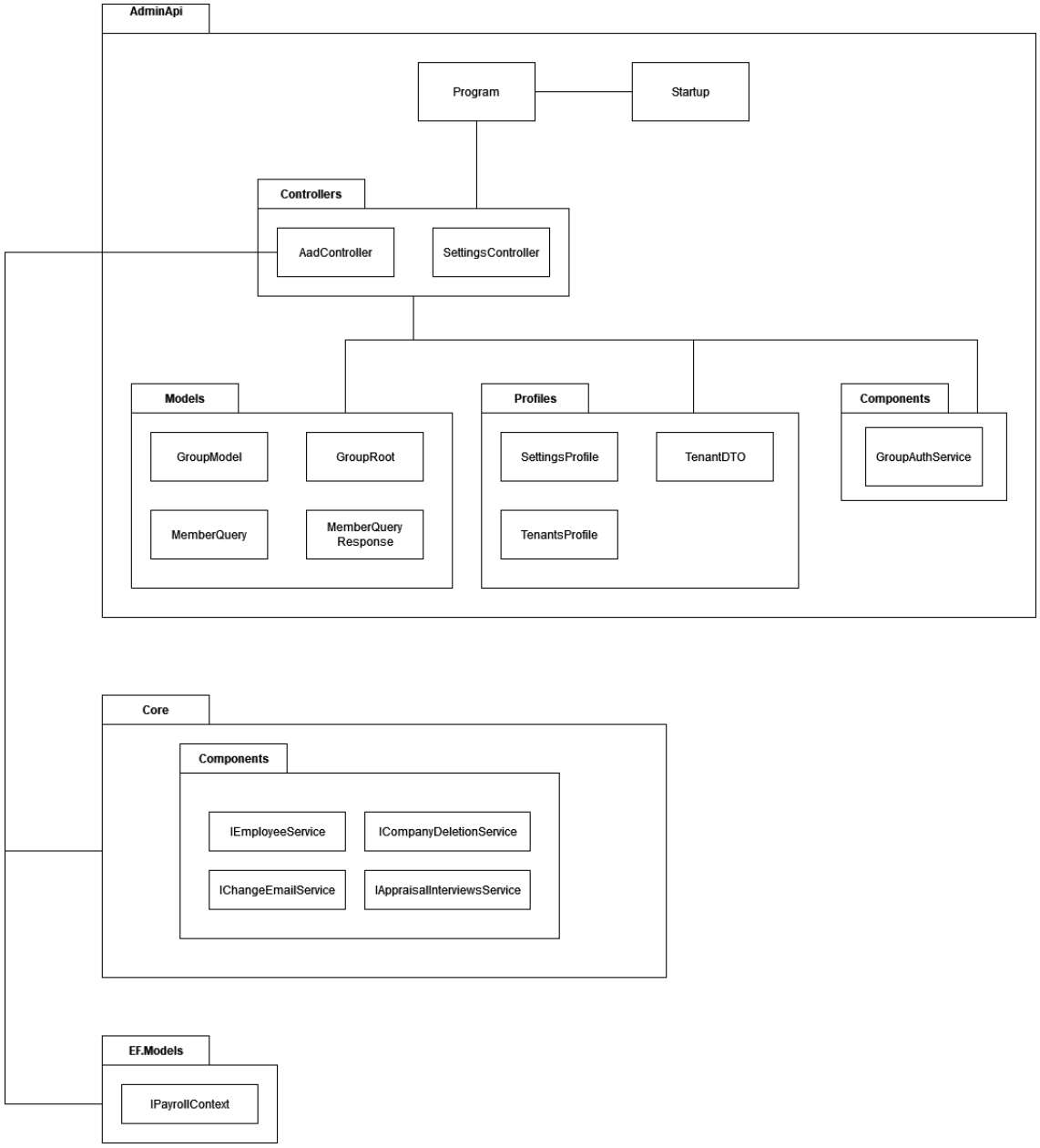


Abbildung 2.7: 2getHR Admin-Portal Klassendiagramm der bestehenden Lösung

API

Die Swagger Dokumentation des bestehenden Admin-Portal APIs wird aus Sicherheitsgründen nicht publiziert. Sie kann lokal unter `localhost:5005/swagger` eingesehen werden. Die bestehende API wird während des Projekts erweitert, um alle benötigten Daten und Funktionalitäten gemäss dem Kapitel Use Cases 3.2 zur Verfügung zu stellen. Abbildung 2.8 zeigt alle bestehenden Routen des Admin-APIs.

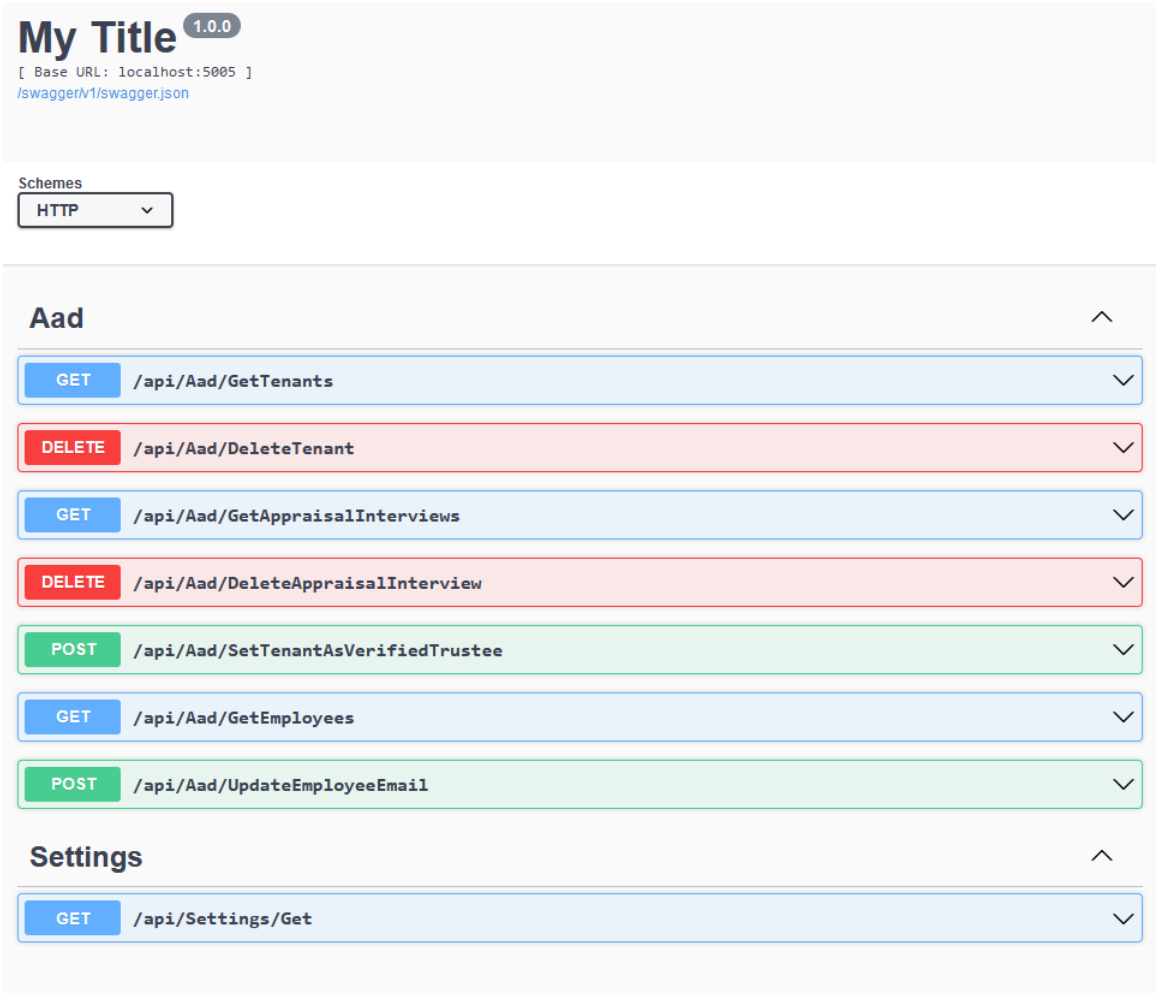


Abbildung 2.8: 2getHR Admin-Portal API Swagger-Dokumentation Ausgangslage

DTO's

Abbildungen 2.9, 2.10 und 2.11 zeigen die Ausgangslage der DTO's der Hauptentitäten Tenant, Mitarbeiter und Mitarbeitergespräch. Das Format basiert auf der Norm von Swagger. Eine genauere Analyse der nötigen Erweiterungen wurde im Kapitel API vorgenommen (siehe Kapitel 5.8).

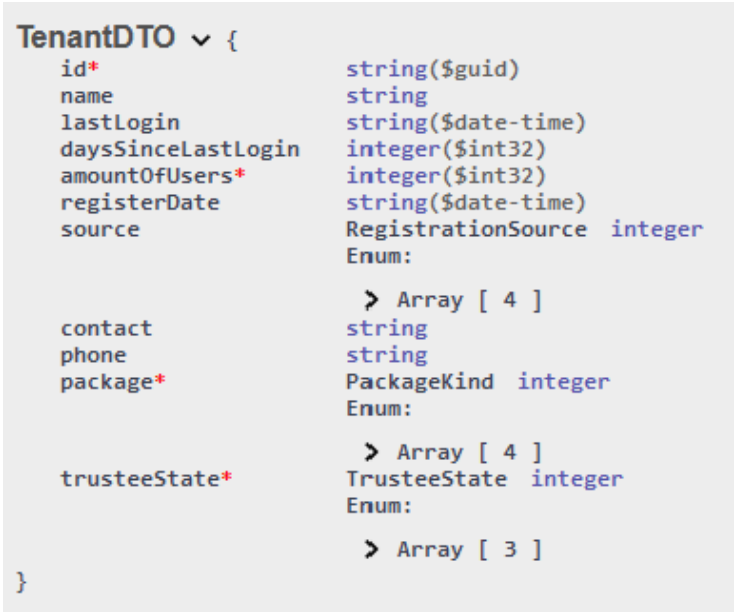


Abbildung 2.9: 2getHR Admin-Portal TenantDTO Ausgangslage

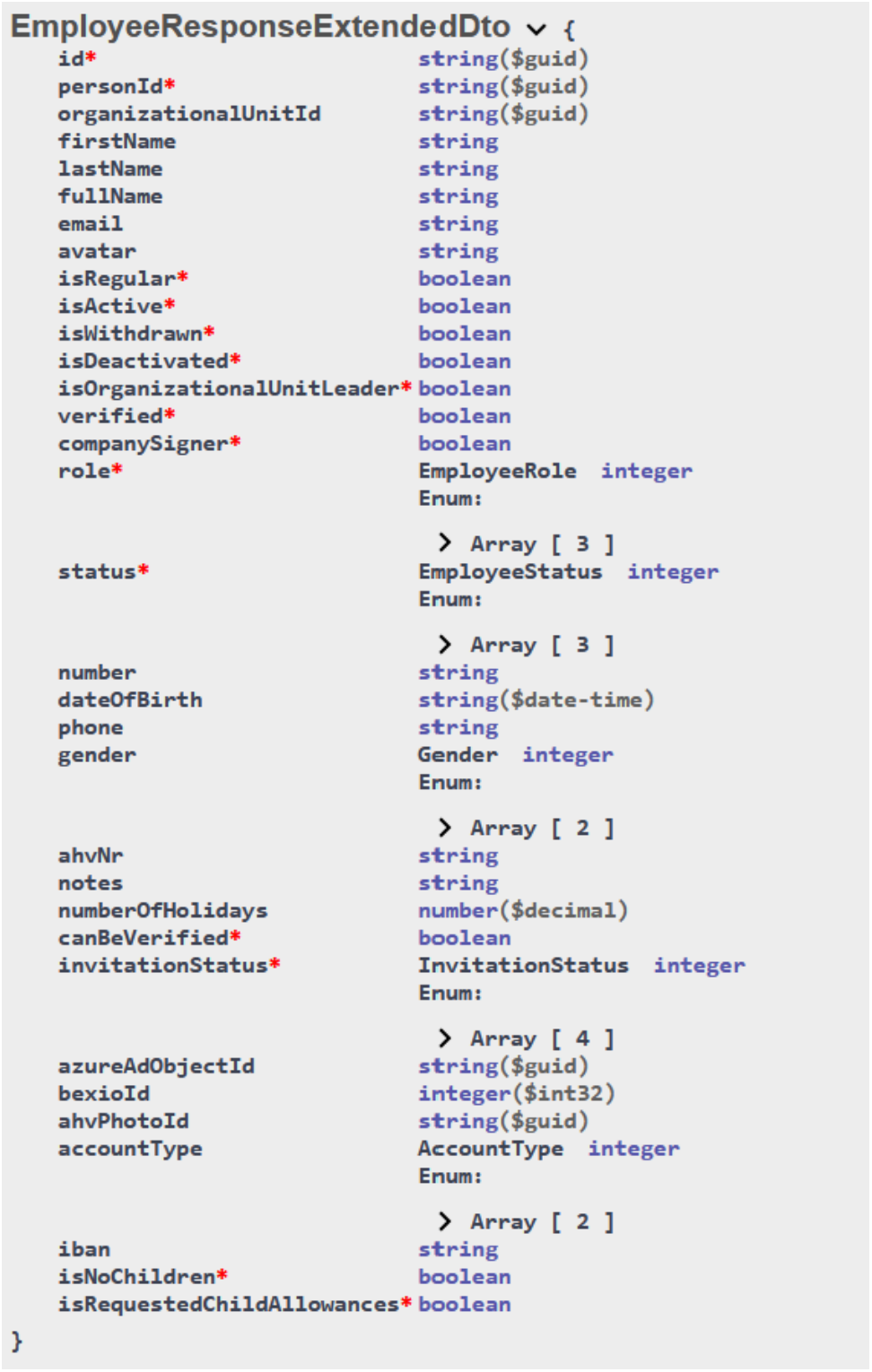


Abbildung 2.10: 2getHR Admin-Portal EmployeeResponseExtendedDTO Ausgangslage

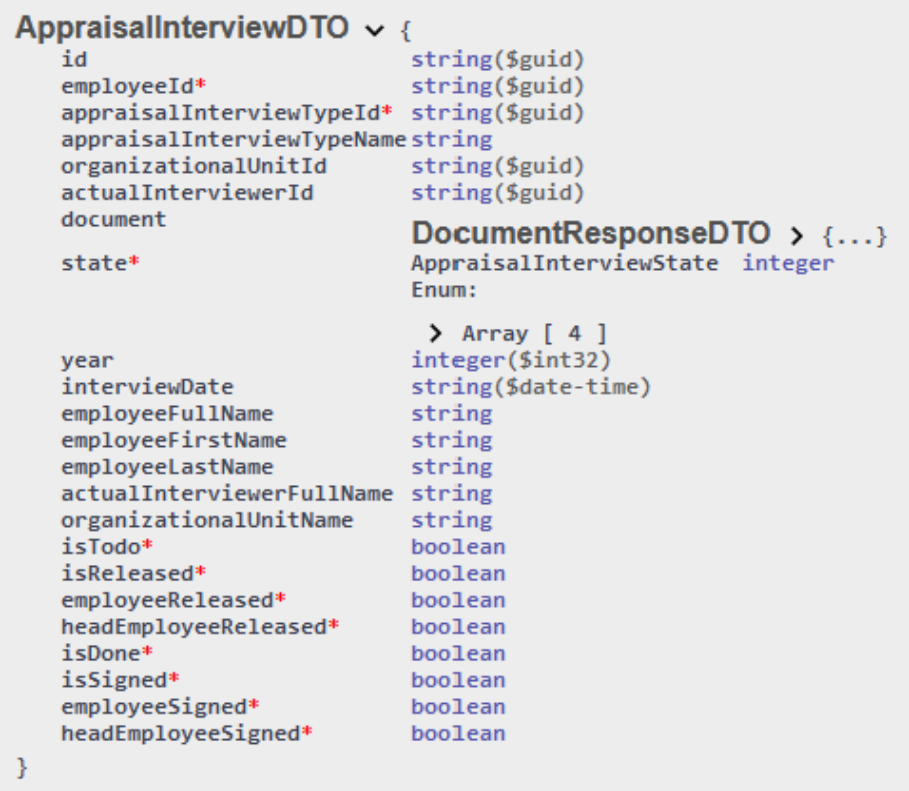


Abbildung 2.11: 2getHR Admin-Portal AppraisalInterviewDTO Ausgangslage

2.3 Flectra - Ticketing

2BIT betriebt eine Flectra Helpdesk Applikation, die unter anderem für das Management von Support Tickets benutzt wird. Die Applikation bietet eine external API an, über die per Query Objekte aus der Datenbank gelesen werden können. Die Daten sind in einer PostGres Datenbank, die eventuell auch direkt aufgerufen werden kann. Eine mögliche Art die Tickets einzubinden zeigt dieses Video https://www.youtube.com/watch?v=TKISSjScEDQ&list=RDCMUCVKlUZP7HAhdQgs-9iTJklQ&index=21.

2.4 SendGrid - E-Mail Versand

2getHR bietet einer Firma die Funktionalität Mitarbeiter per Mail in einen Tenant einzuladen. Diese Mails werden vom Service SendGrid versendet. Ein oft auftretender Supportfall ist, dass noch nicht aktive E-Mail Adressen für Mitarbeiter verwendet werden und diese in SendGrid blockiert werden. Die Unblocking-Funktionalität könnte nun ins Admin-Portal verlagert werden, um den Support zu erleichtern. Zugriff auf SendGrid-Objekte ist per API möglich. Die Dokumentation für das SendGrid-API kann hier eingesehen werden https://docs.sendgrid.com/api-reference/invalid-e-mails-api/retrieve-all-invalid-emails.

2.5 Benutzer

Das Admin-Portal wird ausschliesslich von Mitarbeitenden der Firma 2BIT benutzt und hat dementsprechend eine äusserst kleine und spezifische Userbase. Die User lassen sich in zwei Kategorien unterteilen: 2getHR-Developers und Support-Personal.

Developers

Developers sind ausgebildete Informatiker, die an der 2getHR-Software arbeiten und technischen Support für diese leisten. Diese User kennen das Source-System sehr gut und verstehen die technischen Zusammenhänge innerhalb von 2getHR ohne zusätzliche Informationen. Das Admin-Portal soll dementsprechend auch technische Informationen wie Objekt-IDs anzeigen.

Support-Personal

Das Support-Personal bietet Support für 2getHR und arbeitet nicht an der Weiterentwicklung der Software 2getHR mit. Sie haben keine Informatik-Ausbildung absolviert und verstehen die technischen Zusammenhänge von 2getHR nicht ohne zusätzliche Informationen. Ein Grundverständnis von Technik kann jedoch angenommen werden, da diese Mitarbeiter in 2BIT oft mit Software in Kontakt treten.
Das Admin-Portal soll diesen Usern Funktionalität liefern, die fachlichen Support ohne Hilfe von Developern erlaubt. Auch soll das Admin-Portal Informationen zur Verfügung stellen, die an Developer weitergereicht werden können, falls der Supportfall das Wissen vom Support-Personal übersteigt.

Support-Analyse

Tickets

Die Tickets des vergangenen Jahres, die in im Ticket-System Flectra aufgenommen und verarbeitet wurden, wurden analysiert und gängige Supportfälle daraus abgeleitet. Darauf basierend wurden die Usability Tests ausgearbeitet. Eine Liste der gängigsten Supporttickets und die Schritte zur Lösung derer sind in der folgenden Tabelle zu finden.

Ticket Kategorie	Lösungsansatz
Es treten Probleme mit der Verbindung zu Bexio auf. Daten wurden nicht richtig synchronisiert oder Mitarbeiter können sich nicht einloggen.	Hier wird als erstes kontrolliert, ob der Mitarbeiter richtig eingeladen wurde, sprich auf 2getHR überhaupt zu finden ist. Danach wird kontrolliert, ob Bexio richtig verbunden ist und die richtigen Synchronisierungsdaten hinterlegt sind. Das Supportpersonal navigiert zuerst zum relevanten Tenant und dann zu dessen Mitarbeiterliste. Hier wird der benannte Mitarbeiter gesucht. Mehr Informationen konnten aus dem bestehenden Admin-Portal nicht ausgelesen werden und das Ticket ging an einen Entwickler von 2getHR weiter. Dieser hat dann alle relevanten Informationen in der Datenbank zusammengetragen.
Neue Kunden haben sich als Treuhänder registriert und müssen sich jetzt von 2BIT verifizieren lassen.	Tickets dieser Art müssen von Entwicklern gelöst werden, da Änderungen direkt in der Datenbank vorgenommen werden müssen.

Kunden haben ein Custom Abo gekauft und die Konfiguration für das Abo muss erstellt und die eingekauften Module freigeschaltet werden.	Tickets dieser Art müssen von Entwicklern gelöst werden, da die entsprechenden Änderungen direkt in der Datenbank vorgenommen werden müssen.
Kunden haben Probleme mit der E-Mail-Einladung für Mitarbeitende. Diese wurde zu früh versendet (zu einem Zeitpunkt, wo die E-Mail-Adresse des Mitarbeitenden noch nicht aktiviert wurde).	Das Supportpersonal muss auf SendGrid abklären, ob die Einladung versendet werden konnte oder nicht. Wenn die Einladung nicht versendet werden konnte, muss die E-Mail-Adresse auf SendGrid freigegeben werden und der Kunde wird gebeten, den Versand der Einladung erneut auszulösen.
Der Kunde hat Probleme mit Mitarbeitergesprächen. Er kann den gewünschten Interviewer nicht setzen.	Das Supportpersonal sucht im Admin-Portal den entsprechenden Tenant und navigiert zur Liste dessen Mitarbeiter. Danach wird die Liste nach Mitarbeiter-Name oder E-Mailadresse gefiltert. Das Supportpersonal kann so die Rolle des Mitarbeiters auslesen. Zusätzlich muss die Information bezüglich Abteilungszugehörigkeit und -leitung über die Entwickler abgeklärt werden.
Der Kunde hat Probleme mit Mitarbeiterrollen.	Der Support kann auf dem bestehenden Admin-Portal die Mitarbeiterrolle analog zum vorherigen Ticket einsehen. Allfällige Änderungen müssen von Entwicklern direkt in der Datenbank vorgenommen werden.
Der Kunde hat in der Probezeit verschiedene 2getHR-Funktionen ausprobiert und das Supportpersonal soll die bisherige Aktivität löschen, damit Kunde frisch anfangen kann.	Der Support kann über das bestehende Admin-Portal den Tenant und die Mitarbeitergespräche löschen. Dazu werden die entsprechenden Mitarbeitergespräche mit Hilfe von der Filter- und Sortierfunktionalität auf der Tenant-Liste gesucht. Der Support löscht die entsprechenden Objekte - er muss sich aber in der Listenansicht versichern, dass das richtige Objekt ausgewählt ist.
Der Kunde will Mitarbeiter löschen und versteht nicht, wieso dies nicht möglich ist.	Supportpersonal schickt dem Kunden eine Anleitung bezüglich des Löschens von Mitarbeitern. Der Support kann selbst die Gründe, wieso ein Mitarbeiter nicht gelöscht werden kann, nicht einsehen. Wird dies nötig muss ein Entwickler hinzugezogen werden.

Der Kunde hat Probleme bezüglich der Berechnung von Arbeits- und Ferienzeit von Mitarbeitern.	Dieses Problem tritt oft bei Mitarbeitern auf, die Teilzeit arbeiten. Das Supportpersonal muss beim Kunden nachfragen, ob dies der Fall ist. Der Support muss mit dem Kunden abklären, wie die Einstellungen gesetzt sind und wo genau das Problem aufgetreten ist. Ist das Problem technischer Natur, muss ein Entwickler hinzugezogen werden. Liegt das Problem in den Einstellungen des Kunden und lässt es sich nicht über eine E-Mail-Korrespondenz identifizieren, bittet das Supportpersonal den Kunden den Treuhänderstatus für 2BIT zu setzen, damit der Supporter Zugriff auf den Kunden-Tenant erhält und die Einstellungen des Kunden somit einsehen kann.
Oft bitten Kunden um eine Erklärung von Funktionalitäten oder bestimmten Einstellungen in 2getHR.	Meist kann der Support solche Tickets ohne Einsicht der 2getHR-Daten lösen, jedoch kommt es auch vor, dass gewisse relevante Daten eingesehen werden müssen. Nicht alle diese Daten sind im alten Admin-Portal vorhanden und müssen somit über Entwickler (Datenbank) abgefragt werden.

Tabelle 2.1: Analyse Support Tickets

Arbeitsweise des Supportpersonals

Das Supportpersonal wurde bei der Arbeit an Tickets analysiert und hat die Usability Tests sowohl auf dem neuen sowie dem alten Admin-Portal durchgeführt. Das Projekt-Team konnte somit den Support-Prozess aus der Beobachter-Perspektive analysieren. Nachfolgend sind die für das Projekt relevanten Erkenntnisse aufgeführt:

- Das Supportpersonal achtet sich stark darauf, die Mitarbeiter und Mitarbeitergespräche des korrekten Tenants vor sich zu haben. Im bestehenden Admin-Portal sind die Mitarbeiter und Mitarbeitergespräche nur über den Tenant erreichbar und geben dem Supportpersonal so die Sicherheit, dass das richtige Objekt geöffnet wurde.
- Objekte werden ausschliesslich über E-Mail-Adressen oder den Namen der Objekte identifiziert.
- Das Supportpersonal benutzt bei Mitarbeitergesprächen den Typ des Gesprächs als Bezeichnung.
- Objekte müssen im bestehenden Admin-Portal mühsam über das Filtern nach Tenants und anschliessendem Filtern nach Mitarbeiter oder Mitarbeitergespräche gesucht werden.
- Das Supportpersonal hat sich stark an die Listenansichten gewöhnt und bevorzugt diese zu anderen Darstellungen.
- Das Admin-Portal bietet nicht alle benötigten Informationen und Funktionalitäten an. Aus diesem Grund müssen oft Entwickler für das Lösen von Support-Fällen hinzugezogen werden.
- In der sehr breiten Liste der Tenants muss oft horizontal hin und her gescrollt werden, da Informationen nicht in der optimalen Reihengfolge dargestellt sind.
- Die Löschaktion für Objekte ist in der hintersten Spalte der Liste angeordnet. Das Supportpersonal bevorzugt dies, da so der ganze Eintrag einmal geprüft werden kann, bevor dieser gelöscht wird.

- Der Product Owner und das Supportpersonal klagen über das umständliche und unübersichtliche Filtern und Sortieren in den alten Listenansichten.

Kapitel 3

Anforderungen

In diesem Abschnitt sind die funktionalen und nicht funktionalen Anforderungen der Software definiert. Für eine einfachere Übersicht sind die Anforderungen mit folgendem Farbencode priorisiert.

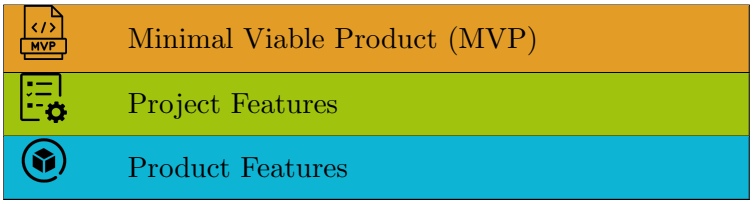


Tabelle 3.1: Use Case Color Code

MVP - Definiert die per Meilenstein `M5-Alpha` zu implementierenden Anforderungen.

Project Features - Anforderungen, die zusätzlich zu den MVP Features im Projekt umgesetzt werden, solange Zeit reicht.

Product Features - Anforderungen, die nach dem abgeschlossenen Projekt weiterführend von 2BIT implementiert werden können.

3.1 Akteure

Akteure	**Beschreibung**
Unauthenticated	User ist nicht authentifiziert im Admin Portal. Ein Login per Azure Active Directory ist nötig um Zugang zum Portal zu bekommen.
User	User ist autorisiert in der Applikation und kann die gesamten Funktionalitäten benutzen.

Tabelle 3.2: Beschreibung Akteure

Eine genaue Beschreibung der User der Webseite, sprich Anforderungen an die Applikation und Vorwissen, kann im Kapitel Benutzer 2.5 eingesehen werden.

3.2 Use Case Diagramm

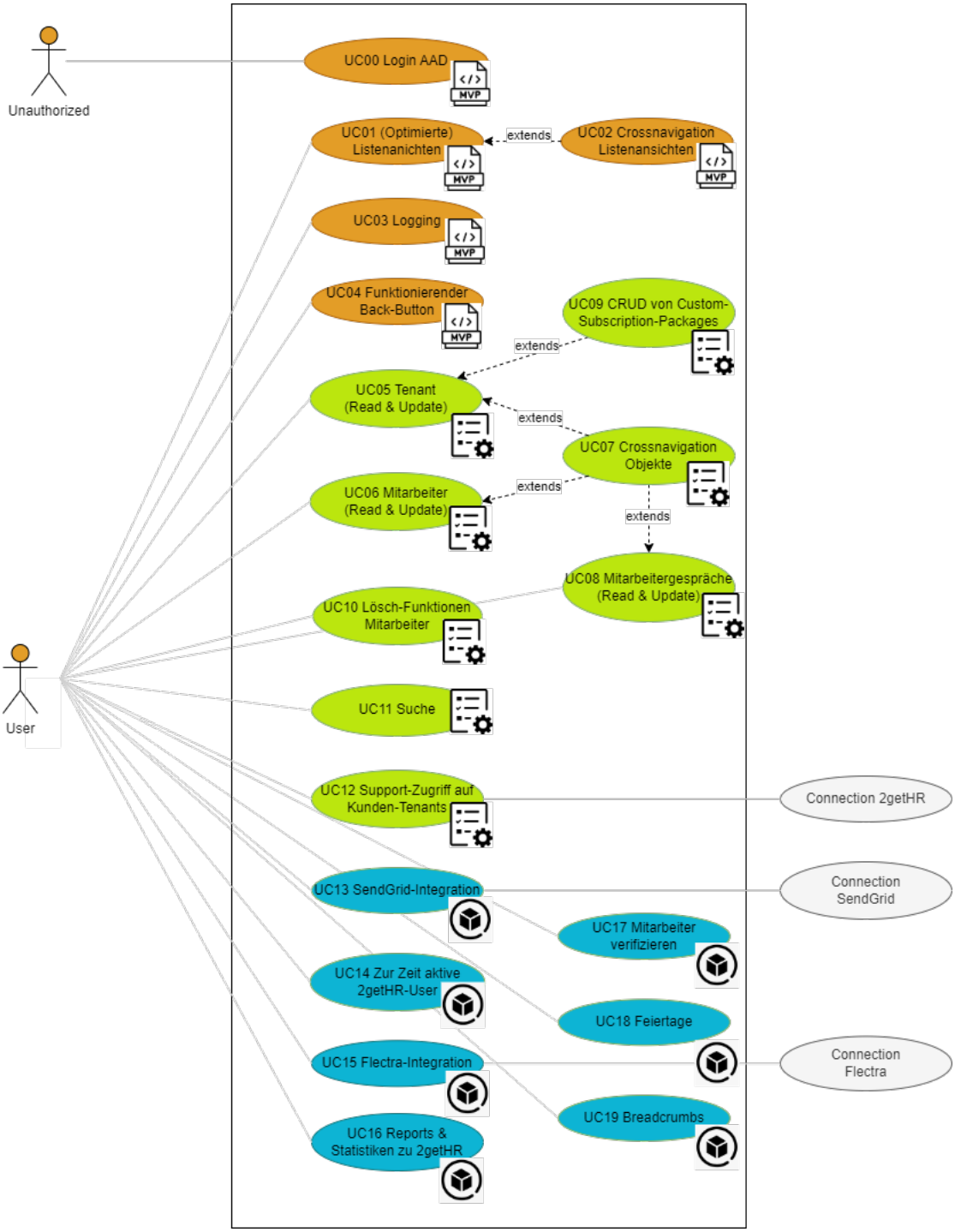


Abbildung 3.1: Use Case Diagramm

3.3 Funktionale Anforderungen

Die Reihenfolge (Nummerierung) der Use Cases stellt gleichzeitig die initiale Priorisierung dar. Die Priorität der Use Cases wurden bei jedem Usability-Test mit 2BIT besprochen und fall nötig neu vergeben. Spätere Änderungen wurden direkt auf dem Itemboard vom Admin Portal vorgenommen. Die

Anforderungen wurden gemäss der Vorlage von Craig Larman [uc] formal ausformuliert.

UC00 Login AAD

Unauthenticated möchte sich in das Admin Portal einloggen.

Akteur	Unauthenticated
Beschreibung	Der Unauthenticated möchte sich in das Admin-Portal einloggen. Nach erfolgreichem Login wird der Unauthenticated zu einem User.
Wichtigstes Erfolgsszenario	Unauthenticated kann sich erfolgreich im Admin-Portal einloggen und wird somit zu User.
Alternatives Erfolgsszenario	Unauthenticated hat keinen Zugriff auf das Admin-Portal, kann sich somit nicht einloggen und bleibt Unauthenticated.
Notwendige UC	-
Detailspezifikation	• Login funktioniert nur mit einem Azure Active Directory Account • Zugriff auf das Admin Portal wird über Azure Active Directory gesteuert

Tabelle 3.3: Beschreibung Use Case 00: Login AAD

UC01 (Optimierte) Listenansichten

User möchte je eine Listenansicht von allen Mitarbeitern, allen Tenants und allen Treuhändern sehen.

Akteur	User
Beschreibung	Der User möchte Listenansichten von allen Mitarbeitern, allen Tenants und allen Treuhänder sehen. Die Listen sind filter- und sortierbar und paginiert.
Wichtigstes Erfolgsszenario	Der User sieht eine Liste der gewünschten Objekte. Der User kann die Liste nach Wunsch sortieren und filtern. Der User kann die Listenansicht konfigurieren.
Notwendige UC	UC00
Detailspezifikation allgemein	• Es steht je eine Listenansicht für Tenants, Mitarbeiter und Treuhänder zur Verfügung. • Die Listenansichten sind nach vordefinierten Spalten filterbar. • Es ist möglich eine Listenansicht nach mehreren Kriterien gleichzeitig zu filtern. • Die Listenansichten sind nach allen Spalten sortierbar. Es ist nicht möglich nach mehreren Spalten gleichzeitig zu sortieren. • Die angezeigten Spalten einer Listenansicht sind konfigurierbar. Folgende Konfigurationsmöglichkeiten bestehen: – Angezeigte Spalten – Reihenfolge der angezeigten Spalten • Die Konfiguration der Listenansicht ist pro User persistierbar. • Die Listenansichten werden paginiert angezeigt. – Standardmässig werden 20 Elemente pro Seite angezeigt. – Die Anzahl Elemente pro Seite ist konfigurierbar.
Detailspezifikation Listenansicht Tenants	Die Listenansicht der Tenants stellt die folgenden Spalten zur Verfügung: • Name • Last Login • Days since last login (all users) • Number of active users • Registration Date • Registration Type • Package Type • Contact • Trustee Verification State • PaymentState • Active

Detailspezifikation Listenansicht Mitarbeiter	Die Listenansicht der Mitarbeiter stellt die folgenden Spalten zur Verfügung: • MitarbeiterID • FirstName • LastName • E-Mail • Rolle • InvitationState • isActive • LastActivityDate • Pensum • isOrganizationalUnitLeader • organizationalUnitID
Detailspezifikation Listenansicht Treuhänder	Die Listenansicht der Treuhänder stellt die folgenden Spalten zur Verfügung: • Name • Last Login • Days since last login (all users) • Number of active users • Registration Date • Registration Type • Package Type • Contact • Trustee Verification State • PaymentState • Active

Tabelle 3.4: Beschreibung Use Case 01: (Optimierte) Listenansichten

UC02 Crossnavigation Listenansichten

User möchte zwischen den verschiedenen Listenansichten hin und her navigieren.

Akteur	User
Beschreibung	Der User möchte zwischen den Listenansichten der Objekte hin und her navigieren.
Wichtigstes Erfolgsszenario	User kann mit nur einem Klick von einer Listenansicht zur anderen Listenansicht wechseln.
Andere Erfolgsszenarien	Wenn der User in der gleichen Session in eine bereits besuchte Listenansicht navigiert, bleibt die vorherige Sortierung, Filterung und Spaltenkonfiguration eingestellt.
Notwendige UC	UC01
Detailspezifikation	• Die verschiedenen Listenansichten werden in einem Menü angezeigt. • Es ist mit einem Klick möglich von einer Listenansicht zur Anderen zu navigieren. • Die Sortierung, Filterung und Spaltenkonfiguration einer Listenansicht bleibt für diese Listenansicht während einer Session bestehen.

Tabelle 3.5: Beschreibung Use Case 02: Crossnavigation Listenansichten

UC03 Aktionsjournal

User möchte eine im Admin-Portal ausgeführte Aktion und Fehler nachvollziehen.

Akteur	User
Beschreibung	Der User möchte im Admin-Portal ausgeführte Aktionen oder aufgetretene Fehler nachvollziehen.
Voraussetzungen	Nur eingeloggte User können Aktionen im Admin-Portal ausführen.
Wichtigstes Erfolgsszenario	Der User kann die Details zu einer im Admin-Portal ausgeführten Aktion nachvollziehen.
Notwendige UC	UC00
Detailspezifikation	• Es werden nur Update und Delete-Useraktionen geloggt. • Pro Useraktion werden folgende Informationen geloggt: – Zeitpunkt – User – Betroffenes Objekt (ID) – Aktion (Delete oder Update) – Falls Update: ∗ Betroffenes Datenfeld ∗ Alter Wert ∗ Neuer Wert – Die Logs sind nicht direkt über das Admin-Portal einsehbar, sondern werden auf der Azure-Cloud gespeichert.

Tabelle 3.6: Beschreibung Use Case 03: Aktionsjournal

UC04 Anzeigegedächtnis

User möchte mit dem Back-Button des Browsers zu vorherigen Ansichten navigieren.

Akteur	User
Beschreibung	Der User möchte die bereits besuchten Ansichten in umgekehrter Reihenfolge besuchen und möchte dazu den Back-Button des Browsers verwenden. Alle Einstellungen gemäss Filter und Tabellenkonfigurationen sollen dabei bestehen bleiben. Der Back-Button des Browsers verhält sich wie auf einer Webseite (obwohl wir uns in einer SPA befinden).

Wichtigstes Erfolgsszenario	Der User klickt auf den Back-Button des Browsers und landet auf der Ansicht, die er zuvor geöffnet hatte. Dabei bleibt die vorherige Sortierung, Filterung und Spaltenkonfiguration eingestellt.
Notwendige UC	-

Tabelle 3.7: Beschreibung Use Case 04: Anzeigegedächtnis

UC05 Tenant (Read & Update & Delete)

User möchte die Details eines Tenants einsehen und bearbeiten.

Akteur	User
Beschreibung	Der User möchte alle Support-relevanten Daten eines Tenants sehen und bearbeiten.
Wichtigstes Erfolgsszenario	Der User kann Informationen zu einem Tenant einsehen.
Andere Erfolgsszenarien	Der User kann Informationen zu einem Tenant verändern.
Notwendige UC	UC01
Detailspezifikation	• Folgende Datenfelder werden für einen Tenant angezeigt: – TenantId – Name – Last Login – Days since last login (all users) – Amount of active users – Registration Date – Registration Type – Package Type – Contact – Phone – Trustee Verification State – PaymentState – Address – Active – Standard holidays Amount – Standard weekly Hours • Auf dem Tenant sind die folgenden Datenfelder bearbeitbar: – Trustee Verification State • Folgende Aktionen stehen für einen Tenant bereits zur Verfügung: – Löschen

Tabelle 3.8: Beschreibung Use Case 05: Tenant (Read & Update & Delete)

UC06 Mitarbeiter (Read & Update)

User möchte die Details eines Mitarbeiters einsehen und bearbeiten.

Akteur	User
Beschreibung	Der User möchte alle Support-relevanten Daten eines Mitarbeiters einsehen und bearbeiten.
Wichtigstes Erfolgsszenario	Der User kann Informationen zu einem Mitarbeiter einsehen.
Andere Erfolgsszenarien	Der User kann Informationen zu einem Mitarbeiter verändern.
Notwendige UC	UC01
Detailspezifikation	• Folgende Datenfelder werden für einen Mitarbeiter angezeigt: – MitarbeiterID – FirstName – LastName – E-Mail – Rolle – LanguageCode – ToSAcceptedDate – BexioID (vorhanden oder nicht) – DateOfBirth – Gender – Address – AHV – Number – Phone – EmployeeID – OrganizationalUnit – AccountType – InvitationState – InvitationDate – isActive – LastActivityDate – Pensum – isOrganizationalUnitLeader – organizationalUnitID • Folgende Datenfelder sind für einen Mitarbeiter bearbeitbar: – E-Mail – Rolle

Tabelle 3.9: Beschreibung Use Case 06: Mitarbeiter (Read & Update)

UC07 Crossnavigation Objekte

User möchte zugehörige Objekte zu einem Objekt einsehen.

Akteur	User
Beschreibung	Der User möchte zu einem Objekt zugehörige Objekte einsehen. Als Beispiel möchte der User alle Mitarbeiter eines Tenant sehen.
Voraussetzungen	Der User ist in einer Detailansicht, in der die Crossnavigation der Objekte zur Verfügung steht.
Wichtigstes Erfolgsszenario	Der User kann Informationen bezüglich zugehöriger Objekte eines Objekts einsehen.
Andere Erfolgsszenarien	Der User kann in zugehörige Objekte eines Objekts navigieren.
Notwendige UC	UC05, UC06
Detailspezifikation allgemein	• Mit einem Klick auf die Anzeige der zugehörigen Objekte kann der User zur Detailansicht des Objekts navigieren.
Detailspezifikation Tenant	In der Detailansicht für einen Tenant werden die folgenden Informationen angezeigt: • Liste aller Mitarbeiter des Tenants. Die verfügbaren Spalten sind im UC01 (siehe Tabelle 3.4) definiert. • Liste aller Mitarbeitergespräche des Tenants. Folgende Spalten stehen für die Mitarbeitergespräche zur Verfügung: – EmployeeName – State – Year – InterviewDate – ActualInterviewerName • Falls der Tenant ein Treuhänder ist: Liste aller Tenants für die der Tenant Treuhänder ist. Die verfügbaren Spalten sind in im UC01 (siehe Tabelle 3.4) definiert. • Ein Datenfeld, welches den Treuhänder des Tenants enthält, sofern für den Tenant ein Treuhänder eingetragen ist
Detailspezifikation Mitarbeiter	In der Detailansicht für einen Mitarbeiter werden die folgenden Informationen angezeigt: • Tenant zu welchem der Mitarbeiter gehört.

Tabelle 3.10: Beschreibung Use Case 07: Crossnavigation Objekte

UC08 Mitarbeitergespräche (Read & Delete)

User möchte die Details eines Mitarbeitergespräches einsehen und bearbeiten.

Akteur	User
Beschreibung	Der User möchte alle Support-relevanten Daten eines Mitarbeitergesprächs einsehen und bearbeiten.
Wichtigstes Erfolgsszenario	Der User kann Informationen zu einem Mitarbeitergespräch einsehen.
Notwendige UC	UC07
Detailspezifikation	• Folgende Datenfelder werden für ein Mitarbeitergespräch angezeigt: – ID – EmployeeID – EmployeeName – State – Year – InterviewDate – ActualIntervieweID – ActualInterviewerName – EmployeeSignedDate – HeadEmployeeSignedDate – EmployeeReleasedDate – HeadEmployeeReleasedDate – AppraisalInterviewTypeId – DocumentId • Auf dem Mitarbeitergespräch sind keine Datenfelder bearbeitbar. • Folgende Aktionen stehen für ein Mitarbeitergespräch bereits zur Verfügung: – Löschen

Tabelle 3.11: Beschreibung Use Case 08: Mitarbeitergespräche (Read & Delete)

UC09 CRUD von Custom-Subscription-Packages

User möchte Custom-Subscription-Packages für Tenants erstellen, einsehen, bearbeiten und löschen.

Akteur	User
Beschreibung	Der User möchte Custom-Subscription-Packages für Tenants erstellen, einsehen, bearbeiten und löschen.
Wichtigstes Erfolgsszenario	Der User kann für einen Tenant ein Custom-Subscription-Package erstellen.
Andere Erfolgsszenarien	• Der User kann die Details eines bestehenden Custom-Subscription-Package eines Tenants einsehen. • Der User kann ein bestehendes Custom-Subscription-Package eines Tenants bearbeiten. • Der User kann ein bestehendes Custom-Subscription- Package eines Tenants löschen?
Notwendige UC	UC05
Detailspezifikation	Der Use Case wurde während des Projektverlaufs nicht umgesetzt. Deshalb wurde er als potentielle Optimierung an 2BIT übergeben.

Tabelle 3.12: Beschreibung Use Case 09: CRUD von Custom-Subscription-Packages

UC10 Lösch-Funktionen Mitarbeiter

User will prüfen, ob ein Mitarbeiter gelöscht werden kann und falls ja, einen Mitarbeiter löschen.

Akteur	User
Beschreibung	Der User möchte überprüfen, ob ein Mitarbeiter eines Tenants gelöscht werden kann. Hintergrund: Ein Mitarbeiter kann nicht gelöscht werden, wenn bestimmte Eigenschaften oder Referenzen auf dem Mitarbeiter vorhanden sind. Für die Überprüfung, ob ein Mitarbeiter gelöscht werden kann, existiert Logik in 2getHR. Der User möchte einen Mitarbeiter löschen. Dies sind zwei separate Funktionen.
Voraussetzungen	Der User befindet sich entweder in der Detailansicht eines Mitarbeiters oder in der Detailansicht eines Tenants.
Nachbedingungen	Nur der Mitarbeiter (und nicht die Person) wurde gelöscht.
Wichtigstes Erfolgsszenario	Der User kann überprüfen, ob ein Mitarbeiter eines Tenants gelöscht werden kann. Wenn Mitarbeiter nicht gelöscht werden kann, erhält der User die Information, weshalb der Mitarbeiter nicht gelöscht werden kann. Zudem wird dem User angezeigt, was gemacht werden muss, damit der Mitarbeiter gelöscht werden kann.
Andere Erfolgsszenarien	Der User kann einen Mitarbeiter eines Tenants löschen.
Notwendige UC	UC05, UC06
Detailspezifikation	Der Use Case wurde während des Projektverlaufs nicht umgesetzt. Deshalb wurde er als potentielle Optimierung an 2BIT übergeben.

Tabelle 3.13: Beschreibung Use Case 10: Lösch-Funktionen Mitarbeiter

UC11 Suche

User will alle im Admin-Portal verfügbaren Daten über ein Suchfeld durchsuchen. Die Resultate werden kategorisiert nach Objekten angezeigt.

Akteur	User
Beschreibung	Der User will alle im Admin-Portal verfügbaren Daten durchsuchen. Die Resultate werden kategorisiert nach Objekten angezeigt.
Wichtigstes Erfolgsszenario	Der User sucht nach einem Begriff. Die Applikation zeigt eine kategorisierte Auflistung aller Suchergebnisse (Objekte) an.
Notwendige UC	UC05, UC06, UC08
Detailspezifikation	• Die Felder Tenant Name, Tenant Kontakt, Mitarbeiter Name und Mitarbeiter-E-Mail werden durchsucht. • Wird ein Element mit passenden Feldern gefunden, wird dieses zurückgegeben. • Angehängte Objekte zu den passenden Objekten, wie zum Beispiel Tenant mit einem Mitarbeiter der per Name gefunden wurde, werden ebenfalls zurückgegeben. • Die Suche wird per Buttonklick gesteuert. • Das Suchresultat wird in Tenant, Mitarbeiter und Mitarbeitergespräch Kategorien aufgeteilt. • Mit Klick auf ein Suchresultat wird die entsprechende Detailview geöffnet. • Anpassung Usability-Test Beta: Suche wird automatisch ausgelöst.

Tabelle 3.14: Beschreibung Use Case 11: Suche

UC12 Support-Zugriff auf Kunden-Tenants

User möchte sich in einen Kunden-Tenant einloggen.

Akteur	User
Beschreibung	User möchte sich in das 2getHR von einem Kunden einloggen können. Dies ist hilfreich für Abklärungen in Support-Fällen. Insbesondere wenn die Beschreibung des Kunden ungenau/unzureichend ist.
Annahme	Kontaktierung des Kunden über die E-Mailadresse des Admins des betroffenen Tenants ist erlaubt.
Voraussetzungen	Der User bestimmt, dass für den Support-Request Zugriff auf den 2getHR-Tenant benötigt wird. Kontaktperson des Kunden ist einverstanden, dass der User sich in den Tenant des Kunden einloggen kann.
Nachbedingungen	Kunde wird angezeigt, dass 2BIT als Treuhänder eingetragen ist.
Wichtigstes Erfolgsszenario	Der User erhält eine Supportanfrage von einem Kunden-Tenant. Der User muss sich für den Support-Request in den Kunden-Tenant einloggen können. User beantragt den Zugriff auf den Kunden-Tenant über das Admin-Portal. Die Kontaktperson des Tenant bestätigt die Anfrage. Das Admin-Portal zeigt an, dass beim Kunden-Tenant 2BIT als Treuhänder eingetragen ist. Zudem wird ein Link auf den Kunden-Tenant im Admin-Portal angezeigt. Sobald der Kunde oder der User den Treuhänderstatus von 2BIT beendet, wird 2BIT als Treuhänder beim Tenant ausgetragen und der User hat keinen Zugriff mehr auf den Kunden-Tenant.
Andere Erfolgsszenarien	User beantragt den Treuhänderstatus für 2BIT beim Tenant über das Admin-Portal. Tenant lehnt Anfrage ab. Der User wird informiert, dass die Treuhänder-Anfrage vom Kunden abgelehnt wurde. 2BIT wird nicht als Treuhänder des Tenants hinzugefügt.
Fehlerergebnis	
Notwendige UC	UC05
Detailspezifikation	Der Use Case wurde während des Projektverlaufs nicht umgesetzt. Deshalb wurde er als potentielle Optimierung an 2BIT übergeben.

Tabelle 3.15: Beschreibung Use Case 12: Support-Zugriff auf Kunden-Tenants

UC13 SendGrid-Integration

User will den Status einer E-Mail-Adresse bei SendGrid einsehen und bei Bedarf entblocken.

Akteur	User
Beschreibung	SendGrid blockiert den Versand von E-Mails an E-Mail-Adressen, falls E-Mails unzustellbar sind (Hintergrund: Spam-Rating). Der User will den Status einer E-Mail-Adresse in SendGrid einsehen und falls nötig die E-Mail-Adresse freigeben.
Annahme	SendGrid bietet ein API für das Einsehen des Status und das Entblocken von E-Mail-Adressen an.
Voraussetzungen	User versteht die Rolle von SendGrid bezüglich E-Mailadressen in 2getHR. Es ist eine gültige E-Mailadressen für den Mitarbeiter hinterlegt.
Nachbedingungen	Die E-Mailadresse ist in SendGrid nicht mehr blockiert.
Wichtigstes Erfolgsszenario	Der User kann den Status von E-Mail-Adressen auf SendGrid im Admin-Portal einsehen. Ist die E-Mail blockiert, kann der User die E-Mailadresse per Klick entblocken.
Notwendige UC	UC05, UC06

Tabelle 3.16: Beschreibung Use Case 13: SendGrid-Integration

UC14 Zur Zeit aktive 2getHR-User

User will die Anzahl der zur Zeit aktiven Benutzer von 2getHR sehen.

Akteur	User
Beschreibung	Der User will die Anzahl der aktiven Benutzer von 2getHR sehen.
Annahme	Es ist definiert, aufgrund welchen Kriterien beurteilt wird, ob ein 2getHR-User 'aktiv', resp. 'eingeloggt' ist.
Wichtigstes Erfolgsszenario	Der User sieht die Anzahl aktiver 2getHR-User.
Andere Erfolgsszenarien	User kann abschätzen wie stark die Applikation aktuell genutzt wird.
Notwendige UC	-

Tabelle 3.17: Beschreibung Use Case 14: Zur Zeit aktive 2getHR-User

UC15 Flectra-Integration

User möchte alle Flectra-Tickets eines Tenants einsehen.

Akteur	User
Beschreibung	Der User möchte alle Flectra Tickets pro Tenant sehen. Zudem möchte der User bei einem Klick auf das Ticket zu Flectra weitergeleitet werden.
Vorbedingung	Der User ist in Flectra eingeloggt. Für den Tenant existieren Tickets in Flectra.
Wichtigstes Erfolgsszenario	Der User sieht alle Flectra-Tickets eines Tenants. Beim Klick auf ein Ticket wird der User auf das Ticket in Flectra weitergeleitet.
Notwendiger UC	UC05
Annahme	1. Verbindung zu Flectra ist möglich. 2. Fectra bietet ein API für Tickets an oder ein direkter Zugriff auf die Datenbank ist möglich.

Tabelle 3.18: Beschreibung Use Case 15: Flectra-Integration

UC16 Reports & Statistiken zu 2getHR

User möchte Reports und Statistiken zu 2getHR einsehen.

Akteur	User
Beschreibung	Der User möchte Reports und Statistiken zu 2getHR im Admin-Portal einsehen.
Notwendige UC	-

Tabelle 3.19: Beschreibung Use Case 16: Reports & Statistiken zu 2getHR

Zusätzliche Funktionale Anforderungen

Wärend der Implementation des Projektes wurden folgende Zusätzliche Anforderungen durch das Entwicklungsteam oder 2BIT angesprochen. Diese wurden nicht in der Lösung umgesetzt.

UC17 Mitarbeiter verifizieren

User möchte einen Mitarbeiter verifizieren.

Akteur	User
Beschreibung	Der User möchte einen Mitarbeiter verifizieren. Hierfür muss ersichtlich sein, ob der User einen gültigen Ausweis bei 2getHR hinterlegt hat und ein entsprechender Status gesetzt werden.
Notwendige UC	UC06

Tabelle 3.20: Beschreibung Use Case 17: User möchte einen Mitarbeiter verifizieren

UC18 Feiertage

User möchte die standardisierten Feiertage hinterlegen.

Akteur	User
Beschreibung	Der User möchte die standardisierten Feiertage der verschiedenen Kantone und der in 2getHR Kunden vertretenen Länder hinterlegen können.
Notwendige UC	-

Tabelle 3.21: Beschreibung Use Case 18: User möchte die standardisierten Feiertage hinterlegen

UC19 Breadcrumbs

User möchte via Breadcrumbs navigieren können.

Akteur	User
Beschreibung	Der User möchte die einfacher den Überblick behalten, indem Breadcrumbs in der Applikation angezeigt werden und über diese navigiert werden kann.
Notwendige UC	UC02

Tabelle 3.22: Beschreibung Use Case 19: User möchte via Breadcrumbs navigieren können

3.4 Umsetzung der Anforderungen im alten Admin-Portal

Use Case	Aktion	Klicks	Kommentar
UC00 - Login AAD	Einloggen	1	Kann übernommen werden.
UC01 - Listenansichten	Anzeigen	0/1	Tenants ist Homepage, Mitarbeiter und Mitarbeitergespräche müssen von Tenant aus erreicht werden.
UC01 - Listenansichten	Filtern	5	Muss stark verbessert werden.
UC01 - Listenansichten	Sortieren	2	
UC01 - Listenansichten	Persistenz		Nicht implementiert.
UC01 - Listenansichten	Pagination	1	
UC02 - Crossnavigation	Ansicht Wechseln	1/2	Momentan als Aktion vom Tenant aus implementiert. Einstellungen gehen bei einem Wechsel verloren. Muss stark verbessert werden.
UC03 - Logging	Logs einsehen		Nicht implementiert.
UC04 - Back-Button		1	Nur eingeschränkt brauchbar. Der Back-Button ruft die Homepage auf ohne Einstellungen zu speichern.
UC05 - Tenant	Read	0	Tenant-Details werden auf der Homepage angezeigt. Jedoch sind nicht alle gewünschten Informationen und keine separate Detailansicht vorhanden.
UC05 - Tenant	Update		Nicht implementiert.
UC06 - Person	Read	1	Personen können nur pro Tenant angezeigt werden. Es sind nicht alle gewünschten Details und keine separate Detailansicht vorhanden.
UC06 - Person	Update		Nicht implementiert.
UC07 - Crossnavigation Objekte			Nicht implementiert.
UC08 - Mitarbeitergespräche	Read	1	Mitarbeitergespräche können nur pro Tenant angezeigt werden. Es sind nicht alle gewünschten Details und keine separate Detailansicht vorhanden.
UC09 - CRUD Custom Package	Read		Nicht implementiert. Nur die Information, ob ein Custom Package verwendet wird, ist vorhanden.
UC09 - CRUD Custom Package	Create, Update, Delete		Nicht implementiert.
UC10 - Mitarbeiter Löschen			Nicht implementiert.
UC11 - Suche			Nicht implementiert.
UC12 - Support-Zugriff			Nicht implementiert.

UC13 - SendGrid			Nicht implementiert.
UC14 - Aktive User			Nicht implementiert.
UC15 - Felctra			Nicht implementiert.
UC16 - Reports			Nicht implementiert.

Tabelle 3.23: Use Case Vergleich zu bestehendem System

3.5 Nicht-Funktionale Anforderungen

Als Grundlage für die Identifizierung der nicht-funktionalen Anforderungen (NFRs) dient der ISO-Standard ISO25010 [iso]. Der ISO-Standard ISO25010 besteht aus 8 Kategorien, die je in weitere Unterkategorien unterteilt sind. Im Identifizierungsprozess der NFRs wurden alle Kategorien inkl. Unterkategorien auf die Relevanz in Bezug auf das Admin-Portal geprüft. Dabei wurden die in diesem Kapitel aufgeführten, für das Admin-Portal relevanten, NFRs identifiziert. Eine Auflistung der Analyseresultate kann am Ende des Kapitels 3.41 eingesehen werden. Desweiteren wurde die OWASP Top Ten [OWA] für die Identifizierung der NFRs geprüft.

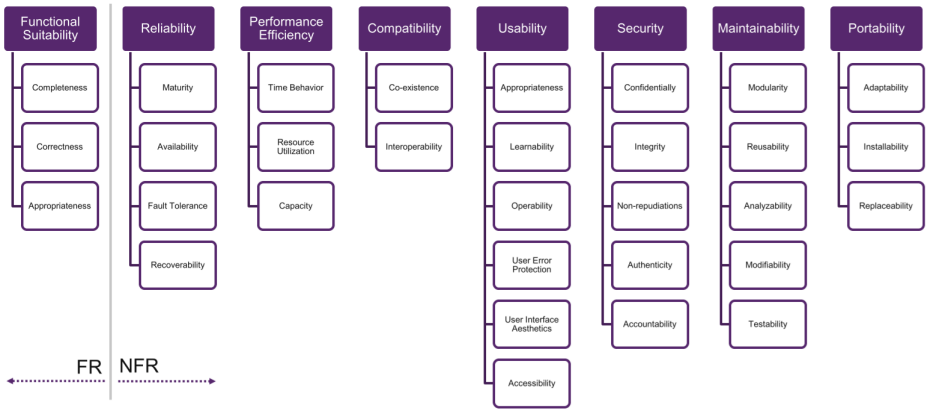


Abbildung 3.2: Kategorien inkl. Unterkategorien gemäss ISO25010-Standard

Die Priorisierung der identifizierten NFRs erfolgt anhand der folgenden Skala:

1. Hohe Priorität (MUSS)
2. Mittlere Priorität (SOLLTE)
3. Tiefe Priorität (KÖNNTE)

ID	NFR01
Kategorie	Usability
Unterkategorie	Operability
Szenario	Das Admin-Portal soll nach dem Umbau in diesem Projekt mindestens gleich gut bedienbar sein, als die bestehende Lösung. Dabei soll die Applikation sowohl von Developern mit technischem Hintergrund sowie von Support-Mitarbeitenden ohne technische Ausbildung bedient werden können (siehe Kapitel Benutzer 2.5).
Auslöser	Neue/zusätzliche Supporter werden von 2BIT eingesetzt.
Reaktion	Die neuen/zusätzlichen Mitarbeitenden verstehen den Aufbau und die Logik des Admin-Portals. Voraussetzung ist, dass sie die Begrifflichkeiten von 2getHR kennen.
Messkriterium	Es werden Usability-Tests durchgeführt. Während den Usability-Tests wird die Bedienbarkeit von Szenarien, die bereits in der bisherigen Lösung vorhanden sind, zwischen der bestehenden und der neuen Lösung verglichen und von den Teilnehmenden der Usability-Tests bezüglich Bedienbarkeit beurteilt. Dabei wird in $\geq$ 80% der Szenarien die neue Lösung als mindestens gleich gut bedienbar bewertet. Zudem können die Teilnehmenden der Usability-Tests im Durchschnitt mindestens 70% aller Szenarios in der neuen Lösung ohne Hilfe/Schulung ausführen.
Priorität	1

Tabelle 3.24: Beschreibung NFR01: Operability (Usability)

ID	NFR02
Kategorie	Compatibility
Unterkategorie	Co-Existence
Szenario	Das Admin-Portal läuft parallel zu der 2getHR-Applikation und greift auf die gleichen Daten zu. Dabei soll das Admin-Portal die Performance von 2getHR nicht massgeblich einschränken.
Auslöser	Es arbeiten sowohl auf dem Admin-Portal sowie auf 2getHR Personen gleichzeitig.
Reaktion	Die Performance von 2getHR ist gleich gut wie wenn niemand auf dem Admin-Portal arbeitet.
Messkriterium	Es werden Performance-Messungen in 2getHR durchgeführt, wenn niemand auf dem Admin-Portal arbeitet. Danach wird die Last von 10 Mitarbeitenden auf dem Admin-Portal simuliert und die gleichen Performance-Messungen werden nochmals in 2getHR durchgeführt. Die Performance, wenn auf dem Admin-Portal gearbeitet wird, ist maximal 10% schlechter, als wenn niemand auf dem Admin-Portal arbeitet.
Priorität	2

Tabelle 3.25: Beschreibung NFR02: Co-Existence (Compatibility)

ID	NFR03
Kategorie	Compatibility
Unterkategorie	Interoperability
Szenario	Die Daten im Admin-Portal sind konsistent mit den Daten in 2getHR.
Auslöser	Daten werden in 2getHR verändert (z.B. Namensänderung eines Tenants).
Reaktion	Die Änderungen werden im Admin-Portal spätestens nach einer der folgenden Aktionen angezeigt. • Applikation neu laden • View wechseln über Menü Buttons oder per Klick auf Listen- oder Sucheintrag • Filter auf Tabelle verändern oder zurücksetzen • Sortierung auf Tabelle verändern • Paginierung auf Tabelle verändern • Delete oder Update Aktion ausführen • Listendaten in der Detailansicht bei Listenwechsel
Messkriterium	Es werden Daten in 2getHR verändert. Danach werden im Admin-Portal folgende Aktionen ausgeführt: • View wechseln • Aktion ausführen • Applikation neu laden Hinweis: Vor jeder der drei Aktionen werden Daten in 2getHR geändert. Und nach jeder der drei Aktionen wird verifiziert, dass die im 2getHR vorgenommenen Änderungen im Admin-Portal ersichtlich sind.
Priorität	1

Tabelle 3.26: Beschreibung NFR03: Interoperability (Compatibility)

ID	NFR04
Kategorie	Security
Unterkategorie	Confidentiality
Szenario	Die Daten im Admin-Portal sind vor unautorisiertem Zugang gesichert. Nur Personen die gemäss dem Azure Active Directory Zugriff auf das Admin-Portal haben, können auf das Admin-Portal zugreifen.
Auslöser	Eine unautorisierte Person versucht auf das Admin-Portal zuzugreifen.
Reaktion	Das Admin-Portal leitet den User auf die Login-Seite des Azure Active Directory weiter. Es ist für den unautorisierten User nicht möglich auf das Admin-Portal zuzugreifen.
Messkriterium	Ein nicht autorisierter User versucht auf das Admin-Portal zuzugreifen. Es wird verifiziert, dass der User auf die Login-Seite des Azure Active Directory umgeleitet wird und keinen Zugriff auf das Admin-Portal hat.
Priorität	1

Tabelle 3.27: Beschreibung NFR04: Confidentiality (Security)

ID	NFR05
Kategorie	Security
Unterkategorie	Confidentiality
Szenario	Die Daten sind in Transit vor unauthorisiertem Zugang gesichert.
Auslöser	Eine dritte Partei greift die Verbindung mit einem MitM-Angriff per selbst-signierten Zertifikat an.
Reaktion	Die Applikation (oder der Browser) verweigert die Verbindung mit dem nicht-vertrauenswürdigen Zertifikat.
Messkriterium	Es wird ein MitM-Angriff mit einem selbst-signierten Zertifikat auf das Admin-Portal gestartet. Die Applikation (oder der Browser) zeigt eine Fehlermeldung bezüglich dem ungültigen Zertifikat an.
Priorität	2

Tabelle 3.28: Beschreibung NFR05: Confidentiality (Security)

ID	NFR06
Kategorie	Security
Unterkategorie	Confidentiality
Szenario	In 2getHR existieren sensible Kundendaten. Diese Daten dürfen im Admin-Portal nicht ersichtlich sein. Es wird per Whitelists definiert welche Daten im Admin-Portal pro Objekt angezeigt werden dürfen. Die Whitelist kann im Anhang im excel Dokument 'Whitelist_2BIT' eingesehen werden.
Auslöser	Ein User sieht sich Daten im Admin-Portal an.
Reaktion	Dem User werden nur Daten, die gemäss Whitelists im Admin-Portal bewilligt sind, angezeigt.
Messkriterium	Für alle Datenfelder, die im Admin-Portal angezeigt werden, wird verifiziert, dass sich diese auf der entsprechenden Whitelist befinden.
Priorität	1

Tabelle 3.29: Beschreibung NFR06: Confidentiality (Security)

ID	NFR07
Kategorie	Security
Unterkategorie	Confidentiality
Szenario	In der 2getHR-Applikation kann Userinput eingegeben werden. Daten, die im Admin-Portal verarbeitet und angezeigt werden, werden vor der Verarbeitung und Anzeige sanitized.
Auslöser	In 2getHR wird nicht gereinigter, gefährlicher Userinput eingegeben.
Reaktion	Im Admin-Portal werden die Daten sanitized bevor sie verarbeitet und angezeigt werden.
Messkriterium	Es wird simuliert, dass JavaScript-Code über 2getHR eingegeben wurde. Danach wird verifiziert, dass der JavaScript-Code im Admin-Portal sanitized wird, sprich, dass er nicht ausgeführt wird.
Priorität	2

Tabelle 3.30: Beschreibung NFR07: Confidentiality (Security)

ID	NFR08
Kategorie	Maintainability
Unterkategorie	Modifiability / Modularity
Szenario	Das Admin-Portal ist so aufgebaut, dass neue Funktionen zur Verfügung gestellt werden können, ohne dass der Code in nicht von den neuen Funktionen tangierten Komponenten angepasst werden muss.
Auslöser	Das Admin-Portal soll durch eine neue Funktionalität ergänzt werden.
Reaktion	Die neue Funktionalität wird in die bestehende Applikation eingebaut, ohne dass grosse Änderungen in nicht tangierten Komponenten notwendig sind.
Messkriterium	Nach dem Alpha-Release werden bis zum Beta-Release zusätzliche Funktionen im Admin-Portal eingebaut. Dabei wird verifiziert, dass die Applikation genügend modular aufgebaut ist, so dass kein oder nur wenig Code in nicht tangierten Komponenten angepasst werden muss.
Priorität	2

Tabelle 3.31: Beschreibung NFR08: Modifiability / Modularity (Maintainability)

ID	NFR09
Kategorie	Usability
Unterkategorie	User Error Protection
Szenario	Die Benutzereingaben im Admin-Portal werden validiert und sanitized. Falls während der Validierung Fehler entdeckt werden, wird dem User angezeigt, wie der Fehler behoben werden kann.
Auslöser	Der User gibt Daten im Admin-Portal ein. Mögliche Inputfelder sind editierbare Datenfelder.
Reaktion	Die Applikation validiert und sanitized die Eingaben und akzeptiert nur valide Daten. Falls die Daten als nicht valide befunden werden, wird dem User eine entsprechende Fehlermeldung angezeigt.
Messkriterium	Es werden ungültige Daten eingeben. Die Daten werden sanitized. Falls Input einen Fehler beinhaltet, wird verifiziert, dass das Admin-Portal eine Fehlermeldung ausgibt, die aussagt, wieso die Daten nicht valide sind.
Priorität	2

Tabelle 3.32: Beschreibung NFR09: User Error Protection (Usability)

ID	NFR10
Kategorie	Performance Efficiency
Unterkategorie	Time Behaviour
Szenario	Das Admin-Portal soll von Usern als performant beurteilt werden.
Auslöser	Ein User arbeitet auf dem Admin-Portal.
Reaktion	Die Applikation reagiert performant auf User-Interaktionen und zeigt Daten und Views schnell an.
Messkriterium	Es werden Performance-Messungen im Admin-Portal durchgeführt. Dabei werden die beiden folgenden Kriterien verifiziert: • Aktionen, die nicht vom Backend abhängig sind, werden zu 95% in ≤ 1 Sekunde ausgeführt. • Aktionen, die vom Backend abhängig sind, werden zu 95% in ≤ 3 Sekunden ausgeführt. • Für Aktionen, die durchschnittlich länger als 0.3 Sekunden dauern, wird ein Loading-Indikator angezeigt. • Für Aktionen, die durchschnittlich länger als 4 Sekunden dauern, wird eine Progress-Bar angezeigt.
Priorität	2

Tabelle 3.33: Beschreibung NFR10: Time Behaviour (Performance Efficiency)

ID	NFR11
Kategorie	Portability
Unterkategorie	Adaptability
Szenario	Das Admin-Portal funktioniert auf den von den 2BIT-Mitarbeitenden verwendeten Browsern (Chrome & MS Edge).
Auslöser	Das Admin-Portal wird in den spezifizierten Browsern geöffnet.
Reaktion	Das Admin-Portal funktioniert wie spezifiziert.
Messkriterium	Die Usability Tests werden mit den in diesem NFR spezifizierten Browsern durchgeführt. Zudem werden die End2End-Tests vor dem Alpha- und vor dem Beta-Release sowohl mit dem Chrome- sowie auch mit den Edge-Browser durchgeführt. Dabei wird verifiziert, dass die Applikation gemäss Spezifikation funktioniert.
Priorität	1

Tabelle 3.34: Beschreibung NFR11: Adaptability (Portability)

ID	NFR12
Kategorie	Security
Unterkategorie	Accountability
Szenario	Useraktionen und Fehler werden gemäss Spezifikation im Use Case 'Logging' (siehe Tabelle 3.6) geloggt.
Auslöser	Ein User performt eine Aktion, die gemäss Spezifikation geloggt werden muss.
Reaktion	Im Log ist ein Eintrag für die ausgeführte Aktion ersichtlich.
Messkriterium	Es werden Usability-Tests durchgeführt. Nach den Usability-Tests wird verifiziert, dass die Aktionen gemäss Spezifikation geloggt wurden.
Priorität	3

Tabelle 3.35: Beschreibung NFR12: Accountability (Security)

ID	NFR13
Kategorie	Maintainability
Unterkategorie	Testability
Szenario	Die Business Logik des Admin-Portals weist eine Test-Coverage von $\geq 90\%$ auf.
Auslöser	Die Unit- und Integrationstests werden ausgeführt.
Reaktion	Die Metrik bezüglich Test-Coverage ist $\geq 90\%$ für die Business Logik des Admin-Portals.
Messkriterium	Die Tests werden ausgeführt. Danach wird verifiziert, dass die Test-Coverage $\geq 90\%$ für die Business-Logik des Admin-Portals erreicht.
Priorität	2

Tabelle 3.36: Beschreibung NFR13: Testability (Maintainability)

ID	NFR14
Kategorie	Usability
Unterkategorie	Operability
Szenario	Falls ein Fehler im Admin-Portal auftritt, soll dem User eine relevante, informative und handlungsorientierte Fehlermeldung angezeigt werden. Hinweis: Dadurch, dass das Admin-Portal technisch-affine End-User hat (siehe Kapitel Benutzer 2.5), darf die Fehlermeldung auch technische Informationen enthalten. Alle Fehlermeldungen müssen jedoch auch von nicht-Informatikern verstanden werden.
Auslöser	Im Admin-Portal tritt ein Fehler auf.
Reaktion	Dem User wird eine relevante, informative und handlungsorientierte Fehlermeldung angezeigt.
Messkriterium	Für die Usability-Tests werden Szenarien entwickelt, die einen Fehler provozieren. Die Teilnehmenden der Usability-Tests können $\geq 80\%$ der Fragen bezüglich der Fehlermeldungen korrekt beantworten.
Priorität	3

Tabelle 3.37: Beschreibung NFR14: Usability (User Protection)

ID	NFR15
Kategorie	Maintainability
Unterkategorie	Modifiability
Szenario	Die im Admin-Portal angezeigten Texte werden in separaten Dateien gepflegt und über Label in die Applikation eingebunden. Damit soll einerseits eine Vorarbeit für eine allfällige Mehrsprachigkeit geleistet werden. Andererseits soll es möglichst einfach sein, bestehende Text anzupassen. Die Einfachheit ist daher gegeben, dass alle Text an einem Ort angepasst werden können.
Auslöser	Ein Text des Admin-Portals muss angepasst werden.
Reaktion	Der Text lässt sich in den Sprachdateien anpassen. Es ist somit nur an einem Ort eine Änderung notwendig.
Messkriterium	$\geq$90% der im Admin-Portal angezeigten Texte werden in einem zentralen File gepflegt und über ein Label eingebunden.
Priorität	3

Tabelle 3.38: Beschreibung NFR15: Modifiability (Maintainability)

ID	NFR16
Kategorie	Security
Unterkategorie	Confidentiality
Szenario	Das Admin-Portal verwendet aktuelle Versionen von den eingesetzten Libraries. Es muss nicht zwingend immer die aktuelle Version verwendet werden, jedoch dürfen nur Versionen verwendet werden, für welche die Sicherheit gegeben ist. Das bedeutet, es dürfen keine Versionen verwendet werden, die veraltet sind. Die Prüfung aller eingesetzten Libraries findet periodisch statt. Dabei wird die periodische Prüfung während des Projekts durch das Projektteam durchgeführt. Bei der Übergabe des Admin-Portals in den Betrieb wird vom Projektteam eine Wartungsempfehlung an 2BIT abgegeben.
Auslöser	Für eine verwendete Library steht eine neue Version zur Verfügung.
Reaktion	Die Aktualisierung der Library auf die neue Version wird geprüft. Falls man sich dazu entscheidet, dass ein Update der Library gemacht werden soll, wird die Library auf die neue Version aktualisiert und es wird verifiziert, dass die Applikation weiterhin wie spezifiziert funktioniert.
Messkriterium	Beim Beta-Meilenstein wird verifiziert, dass von allen eingesetzten Libraries eine aktuelle Version verwendet wird.
Priorität	3

Tabelle 3.39: Beschreibung NFR16: Security

ID	NFR17
Kategorie	Usability
Unterkategorie	Operability
Szenario	Daten werden gleichzeitig im 2getHR und im Admin-Portal bearbeitet.
Auslöser	Daten werden in 2getHR verändert, während ein User des Admin-Portals ein Tenant oder Mitarbeiter am aktualisieren ist.
Reaktion	Die Applikation macht einen Snapshot-Vergleich zwischen den Daten, die im Frontend vorliegen und den Daten in der Datenbank. Wurde das Objekt seit dem letzten Refresh der Daten im Admin-Portal verändert, wird eine Fehlermeldung angezeigt und der Updatevorgang wird nicht ausgeführt.
Messkriterium	Ein manueller Test wird für alle Update-Funktionalitäten durchgeführt, wobei ein Objekt in der Datenbank verändert wird. Beim Updaten des Objektes im Admin-Portal wird verifiziert, dass der Vorgang abgebrochen wird und dem User eine entsprechende Meldung angezeigt wird.
Priorität	3

Tabelle 3.40: Beschreibung NFR17: Multi-User

Analyse nicht spezifizierter ISO25010-Kategorien

Es wurde nicht für jede Unterkategorie aus dem ISO25010 [iso] ein NFR definiert. Tabelle 3.41 listet für jede Unterkategorie aus dem ISO25010 auf, ob ein NFR dafür definiert wurde. Falls kein NFR für eine Unterkategorie definiert wurde, wird in Tabelle 3.41 ausgeführt, wieso die Unterkategorie für das Admin-Portal nicht relevant ist.

Kategorie	**Subkategorie**	**Beschrieb**
Reliability	Maturity	Im NFR10 enthalten.
Reliability	Availability	Die Applikation läuft auf einem Azure App Service. Dieses wird von 2BIT verwaltet, welche die Availability Zone managen.
Reliability	Fault Tolerance	Die Applikation wird komplett mit Error Handling implementiert und getestet. Zusätzliche Fault Tolerance ist nur in Daten-ändernden User-Interaktionen relevant, was in diversen NFRs abgedeckt ist.
Reliability	Recoverablity	Die Applikation wird gemäss der Twelfe-Factor Methodologie gebaut (siehe auch Kapitel 5.7). Diese bestimmt stateless Prozesse, die ohne Probleme gestoppt und wieder gestartet werden können. Recoverability wird implizit in der Kontrolle der Methodologie abgedeckt.
Performance Efficiency	Time Behaviour	Spezifiziert im NFR10.
Performance Efficiency	Resource Utilization	Die einzige kritische Ressource vom Admin-Portal ist der Zugriff auf die 2getHR Datenbank. Die Kontrolle vom Performance-Einfluss auf diese Ressource und die 2getHR-Applikation wird bereits im NFR02 abgedeckt.
Performance Efficiency	Capacity	Die Userbasis der Applikation und die Komplexität der auszuführenden Funktionalitäten wird einen sehr niedrigen Treshhold nie überschreiten und wird deshalb nicht spezifisch untersucht werden.
Compatibility	Co-existence	Spezifiziert im NFR02.
Compatibility	Interoperability	Spezifiziert im NFR03.
Usability	Appropriateness	Die Applikation wird in intensiver Zusammenarbeit mit den Endusern (zur Zeit 2 Personen) erarbeitet. Mit den drei Usability-Tests wird diese Anforderung erreicht.
Usability	Learnability	Ist in NFR01 bereits abgedeckt. Das System soll einfach und intuitiv bedienbar sein, sprich auch schnell gelernt.
Usability	Operability	Spezifiziert im NFR01.
Usability	User Error Protection	Spezifiziert im NFR09.
Usability	User Interface Aesthetics	Die Applikation wird in intensiver Zusammenarbeit mit den Endusern (zur Zeit 2 Personen) erarbeitet. Mit den drei Usability-Tests wird diese Anforderung automatisch erreicht.

Usability	Accessibility	Die Applikation wird in intensiver Zusammenarbeit mit den Endusern (zur Zeit 2 Personen) erarbeitet. Mit den drei Usability-Tests wird diese Anforderung erreicht.
Security	Confidentiality	Spezifiziert in den NFR04 - NFR07.
Security	Integrity	In den NFR04 - NFR07 enthalten.
Security	Non-Repudiation	Ein User wird beim Login authentifiziert (inkl. 2FA). Zudem werden User-Aktionen im Admin-Portal geloggt (siehe NFR12 & UC03). Somit ist eine Non-Repudiation gegeben.
Security	Authenticity	Alle User des Admin-Portals müssen sich über mit 2FA über das Azure Active Directory authentisieren, bevor der Zugriff auf das Admin-Portal möglich ist.
Security	Accountability	Spezifiziert im NFR12.
Maintainability	Modularity	Spezifiziert im NFR08.
Maintainability	Reusability	Im NFR08 implizit enthalten. Die Architektur wird stark modular aufgebaut und kann dementsprechend einfach wiederverwendet werden.
Maintainability	Analyzability	Im NFR08 enthalten.
Maintainability	Modifiabillity	Spezifiziert im NFR15.
Maintainability	Testability	Spezifiziert im NFR13.
Portability	Adaptability	Spezifiziert im NFR11.
Portability	Installability	Die Applikation ist per Twelfe-Factor Methodologie cloudnative aufgebaut und wird durch diese kontrolliert (siehe auch Kapitel 5.7). Zudem werden CI/CD-Pipelines für das Deployment eingesetzt (siehe auch 6.1).
Portability	Replaceability	Im NFR08 enthalten.

Tabelle 3.41: Auslistung aller NFR Kategorien gemäss ISO 25010

3.6 Adressierung Nicht-Funktionaliter Anforderungen

Adressierung der NFR mit hoher Priorität per M4

Bis zum vierten Meilenstein sollen Massnahmen bezüglich den NFR mit Priorität 1 (hohe Priorität) eingeleitet sein (vergl. Kapitel 7.2). Folgende Tabelle 3.42 zeigt den Stand der NFR per M4: Architektur.

NFR	Stand
NFR01	Es wurden anhand der Wireframes Usability Tests mit einer Support-Mitarbeiterin sowie mit einem Developer durchgeführt (siehe Kapitel 2.5). Zudem wurden zwei weitere Usability Tests per Alpha-Release und Beta-Release geplant. In den geplanten Usability-Tests wird die Bedienbarkeit anhand des Stands der implementierten Lösung mit Support-Mitarbeitenden und Developern weiter sichergestellt. Die Usability-Tests basieren dabei auf realen Supporttickets von 2getHR. Der direkte Vergleich zwischen aktueller und neuer Lösung von den Teilnehmenden der Usability Tests ist für die Usability-Tests per Alpha- und Beta-Release eingeplant.
NFR03	Während der Implemenation wird sichergestellt, dass beim Neuladen der Applikation, nach jedem View-Wechsel und nach jeder ausgeführten Aktion die Daten vom Backend neu geladen werden. Da das Backend des Admin-Portals auf die gleiche Datenbank wie die 2getHR-Applikation zugreift, stellen diese Massnahmen sicher, dass im Admin-Portal nach jedem neu laden der Applikation, View-Wechsel und ausgeführter Aktion die aktuellen Informationen dargestellt werden.
NFR04	Die Anbindung an das Azure Active Directory von 2BIT konnte von der bestehenden Lösung übernommen werden. Diese wurde dahingehend optimiert, dass vor der Authentisierung und Autorisierung eines Users keine Daten eingesehen werden können (bei der bestehenden Implementation war die Applikation vor der Redirection zu der Microsoft-Anmeldeseite kurz zu sehen). Zudem wurde verifiziert, dass die Redirection auf die Login-Seite von Microsoft auch stattfindet, wenn versucht wird direkt über die URL auf eine bestimmte Ressource (z.B. Liste aller Mitarbeiter) zuzugreifen.
NFR06	Die Whitelist, die definiert welche Datenfelder im Admin-Portal angezeigt werden dürfen, wurde für alle Objekte in Zusammenarbeit mit 2BIT spezifiziert. Die DTO's der API-Endpoints wurden dementsprechend angepasst. Es können nun ausschliesslich Daten gemäss Whitelist über die Schnittstelle (Admin-API) zur Verfügung gestellt werden. Somit werden im Frontend sensible 2getHR Daten nicht verfügbar sein.
NFR11	Die Projektteam-Mitglieder entwickeln und testen das Admin-Portal in den von 2BIT-Mitarbeitenden genutzten Browsern. Zudem werden vor dem Alpha- und Beta-Release alle End2End-Tests sowohl im Chrome- sowie auch im Edge-Browser ausgeführt. Des Weiteren werden die Usability-Tests per Alpha- und Beta-Release von mindestens einer Person mit dem Browser Chrome und von mindestens einer Person mit dem Browser Edge durchgeführt.

Tabelle 3.42: Stand der NFR per M4: Architektur

Adressierung der NFR per Alpha Release

Tabelle 3.43 zeigt den Stand der NFR per Alpha Release.

NFR	Stand
NFR01	Auf dem Alpha Release wurden Usability Tests mit einer Support-Mitarbeiterin sowie mit einem Developer durchgeführt (siehe Kapitel 2.5). Die Bedienbarkeit der neuen Applikation wurde ausgiebig getestet und direkt mit der bestehenden Version verglichen. User haben einen merklich positiven Unterschied zur bestehenden Version festgestellt. Jedoch fehlt für den optimalen Gebrauch die erweiterte Suchfunktionalität. Diese wurde dementsprechend neu priorisiert und soll zeitgleich mit den Detail-Views für den Beta Release umgesetzt werden.
NFR02	Die Entwicklungsumgebung wurde auf einer Azure Dev Subscription eingerichtet und ist deshalb per Default in der Performance eingeschränkt. Das NFR wird per Final Release genauer untersucht.
NFR03	Jede Aktion in der Applikation, sprich neu laden, Filter setzten, Paginierung verändern und Ansichten wechseln stösst einen Refetch der Daten vom Backend an. Dies stellt sicher, dass die Daten immer aktuell sind. Einzige Ausnahme ist hier die Tabellen Konfiguration. Da diese nicht oft ausgeführt wird und eine direkte Anzeige des Loadingscreens auslöst, wurde entschieden auf einen Reload zu verzichten. So können User Änderungen an der Konfiguration direkt einsehen, ohne auf den Reload der Daten warten zu müssen.
NFR04	Ist der User nicht eingeloggt und ruft die Webseite auf, wird der User auf den Login Screen von Microsoft weitergeleitet. Ein Aufruf einer Adresse, die nicht Root ist, ist im Moment nicht möglich und verhindert so Zugriff auf ungeschützte Daten. Da ein Ausloggen der Seite einen Redirect auf Microsoft Auslog-Seite verursacht, wird auch automatisch der Store gelöscht. So werden auch bei einem erneuten Laden der Seite keine Daten angezeigt.
NFR05	Wird bei Beta-Release kontrolliert.
NFR06	Die DTOs der verschiedenen Objekte wurden gemäss Whitelist von 2BIT massgeschneidert und enthalten keine sensiblen Daten. Es werden also keine sensiblen Daten an das Frontend geschickt.
NFR07	Stand Codebasis kann kein Userinput gemacht werden. Wird beim Beta-Release genauer getestet.
NFR08	Kann erst in Beta-Release getestet werden.
NFR09	Kann erst in Beta-Release getestet werden da keine Usereingabe vorhanden ist.
NFR10	Bei allen API Calls wurden Loadingscreens implementiert. Ausserhalb dieser gibt es keine performanceintensiven Aktionen.
NFR11	Der Alpha-Release wurde im Usability-Test je einmal mit Chrome und MS Edge getestet und hat einwandfrei funktioniert.
NFR12	Kann erst in Beta-Release getestet werden, da keine zu loggenden Useraktionen möglich sind.
NFR13	Die Backend API wurde vollumfänglich getestet. Frontend wurde bis jetzt noch nicht getestet.
NFR14	Dem User wird bei einem Fehler eine Komponente angezeigt. Dabei kann eine detailierte Fehlermeldung per Knopfdruck angezeigt werden.
NFR14	Nicht implementiert.
NFR14	Alle Libraries sind aktuell.

Tabelle 3.43: Stand der NFR per Alpha Release

Adressierung der NFR per Beta Release

Tabelle 3.44 zeigt den Stand der NFR per Beta Release.

NFR	Stand
NFR01	Auf dem Beta Release wurden Usability Tests mit einer Support-Mitarbeiterin sowie mit einem Developer durchgeführt (siehe Kapitel 2.5). Die Bedienbarkeit der neuen Applikation wurde ausgiebig getestet und direkt mit der bestehenden Version verglichen. User haben einen merklich positiven Unterschied zur bestehenden Version festgestellt. Besonders die erweiterte Suche wurde gelobt und die Crossnavigation zwischen den Detailansichten ausgiebig benutzt.
NFR02	Die Entwicklungsumgebung wurde auf einer Azure Dev Subscription eingerichtet und ist deshalb per Default in der Performance eingeschränkt.
NFR03	Jede Aktion in der Applikation, sprich Views neu laden, Filter setzten, Paginierung verändern, Felder updaten, Objekte löschen und Ansichten Wechsel stösst einen Refetch der Daten vom Backend an. Dies stellt sicher, dass die Daten immer aktuell bleiben. Einzige Ausnahme ist hier die Tabellen Konfiguration. Da diese nicht oft ausgeführt wird und ein direktes laden einen Loadscreen auslöst, wurde entschieden hier auf einen Reload zu verzichten. So können User Änderungen an der Konfiguration direkt einsehen ohne auf den Reload der Daten warten zu müssen.
NFR04	Ist der User nicht eingeloggt und ruft die Webseite auf, wird der User auf den Login Screen von Microsoft weitergeleitet. Ein Aufruf einer Adresse, die nicht Root ist, wird durch Azure verhindert. Dies wird in der produktiven Umgebung geändert werden. Um die Sicherheit hier zu testen, wurde lokal ein Aufruf einer nicht root Seite ausprobiert und der User wurde auf die Login Seite von Microsoft weitergeleitet. Da ein Ausloggen der Seite einen Redirect auf die Microsoft Auslog-seite verursacht, wird auch automatisch der Store gelöscht. So werden auch bei einem erneuten Laden der Seite keine Daten angezeigt.
NFR05	Wird in Final Release getestet.
NFR06	Die DTOs der verschiedenen Objekte wurden gemäss Whitelist von 2BIT massgeschneidert und enthalten keine sensiblen Daten. Es werden also keine sensiblen Daten an das Frontend geschickt.
NFR07	Einziger Userinput, der gefährlich werden kann sind die Suchbegriffe und neue E-Mail-Adressen für Mitarbeiter. Beide werden im Backend durch LINQ nicht als ausführbare Query an die Datenbank weitergeleitet.
NFR08	In der Periode zwischen Alpha und Beta Release wurde neue Funktionalität eingebaut. Diese konnte ohne grosse Änderungen in nicht tangierten Komponenten hinzugefügt werden.
NFR09	Die Suche wird auf gefährliche Zeichen kontrolliert und in einen Fehlerzustand versetzt, falls problematischer Input vorhanden ist. Ebenso wird die Eingabe von E-Mail-Adressen für Mitarbeiter auf ein gültiges Format kontrolliert.

NFR10	Bei allen API Calls wurden Loadingscreens implementiert. Ausserhalb dieser gibt es keine performanceintensiven Aktionen. Auf der Entwicklungsumgebung muss der Server beim ersten Aufruf zuerst von Azure aufgefahren werden (da wir auf der Dev Subscription arbeiten). • Backend unabhängige Aktionen ohne Loading Indikator: – Konfiguration der Tabelle: <1 Sekunde – Ausblenden der Listen auf den Detail Seiten: <1 Sekunde – Reset der Suche: <1 Sekunde – Öffnen Löschdialog Tenant: <1 Sekunde – Öffnen Löschdialog Mitarbeitergespräch: <1 Sekunde • Backend abhängige Aktionen mit Loading Indikator: – Laden der Listenansichten: <2 Sekunden – Navigieren zwischen Listenansichten: <2 Sekunden – Navigieren mit Home Button: <2 Sekunden – Filter auf Listenansichten setzen: <2 Sekunden – Filter auf Listenansichten zurücksetzen: <2 Sekunden – Paginierungs Index auf Listenansicht ändern: <2 Sekunden – Paginierung der Seitengrösse auf Listenansicht ändern: <2 Sekunden – Listen auf Listenansichten sortieren: <2 Sekunden – Automatische Suche: <2 Sekunden – Manuelle Suche: <2 Sekunden – Navigation zu Detailansicht per Listenansicht: <2 Sekunden – Navigation zu Detailansicht per Suche: <2 Sekunden – Einblenden der Tabellen in Detailansicht: <2 Sekunden – Paginierungs Index auf Detailansicht ändern: <2 Sekunden – Paginierung der Seitengrösse auf Detailansicht ändern: <2 Sekunden – Listen auf Detailansicht sortieren: <2 Sekunden – Löschen Mitarbeitergespräch: <2 Sekunden – Löschen Tenant: Nicht vollständig testbar, da das Löschen der zugehörigen Azure Storage Blobs nicht möglich ist. Ohne diese Funktionalität <2 Sekunden – Check ob Mitarbeiter löschbar (öffnen Löschdialog): <2 Sekunden – Löschen Mitarbeiter: <2 Sekunden – Mitarbeiter-E-Mail updaten: Nicht vollständig testbar, da der E-Mail-Versand nicht durchgeführt wird. Ohne diese Funktionalität <2 Sekunden. – Mitarbeiter Rolle updaten : <2 Sekunden – Tenant Treuhänderstatus updaten: <2 Sekunden • Aktionen länger als 4 Sekunden: Keine

NFR11	Der Beta Release wurde im Usability-Test je einmal mit Chrome und MS Edge getestet und hat einwandfrei funktioniert.
NFR12	Die Useraktionen Mitarbeiter, Tenant und Mitarbeitergespräch löschen und das Updaten von Treuhänderstatus von Tenant und Mitarbeiter Rolle und E-Mail updaten werden alle als Logeintrag ans Backend gesendet. Da die Development Umgebung kein Azure Blob Storage konfiguriert hat, wird momentan auf dem Backend ein Eintrag auf den Standard-Output-Stream gemacht. Das Persistieren der Logeinträge wird nach Übergabe der Arbeit an 2BIT von Teammitglied Yael Schärer umgesetzt.
NFR13	Das Backend API wurde vollumfänglich getestet (siehe Kapitel 6.4). Das Frontend wurde mit Cypress Tests ausführlich getestet (siehe Kapitel 6.4).
NFR14	Dem User wird bei einem Fehler eine Komponente angezeigt. Dabei kann eine detailierte Fehlermeldung per Knopfdruck angezeigt werden.
NFR15	In Beta Release nicht implementiert. Wird aber im Final Release umgesetzt.
NFR16	Libraries werden in Final Release kontrolliert.
NFR17	Vorgang wurde für alle drei Update Funktionalitäten getestet und erfolgreich verifiziert, dass das Objekt nicht verändert wird und eine entsprechende Meldung angezeigt wird. Danach wurde ein Refresh der Daten produziert und der Vorgang wiederholt. Es wurde verifiziert, dass beim zweiten Updaten die Daten erfolgreich verändert wurden.

Tabelle 3.44: Stand der NFR per Beta Release

Adressierung der NFR per Final Release

Tabelle 3.45 zeigt den Stand der NFR per finalem Release.

NFR	**Stand**
NFR01	Wie Beta Release.

NFR02

Wie bereits in der Adressierung der NFR per Alpha- und Beta-Release beschrieben, läuft die Entwicklungsumgebung auf einer Azure Dev Subscription und ist deshalb in der Performance eingeschränkt. Nichtsdestotrotz wurden Lasttests durchgeführt, um die Auswirkungen vom Admin-Portal auf die Performance von 2getHR abzuschätzen. Hierzu wurde in einem ersten Schritt die Ladezeit des Dashboards von 2getHR gemessen. Die folgende Tabelle zeigt die durchgeführten Messungen:

#	**Ladezeit in ms**
1	8284
2	14333
3	10860
4	6447
5	7181
6	6988
7	6262
8	6792
9	7541
10	7469
11	6919

Der Median der Messungen beträgt: **7181 ms**.

Anschliessend wurde mit LoadNinja [loa] die Last von 10 Usern auf dem Admin-Portal simuliert und die Ladezeit des Dashboards von 2getHR wurde erneut gemessen. Die Messwerte sind in folgender Tabelle zu finden:

#	**Ladezeit in ms**
1	9236
2	15343
3	12423
4	19275
5	23032
6	12245
7	13017
8	14071
9	14972

Der Median der Messungen beträgt: **14071 ms**.

Das Ergebnis des Lasttests ist somit, dass sich die Ladezeit des Dashboards von 2getHR während der generierten Last in der Entwicklungsumgebung ungefähr verdoppelt. Dieses Resultat ist jedoch nicht aussagekräftig für den Produktivbetrieb, da dort eine auf Performance optimierte Umgebung im Einsatz ist. Auf dem Admin-Portal ist gemäss Abklärungen mit 2BIT mit maximal 10 simultanen Usern zu rechnen. In 2getHR gibt es zurzeit ca. 2'700 aktive Benutzer. Somit entsprechen die 10 zusätzlichen Benutzer auf dem Admin-Portal einer Zunahme von 0.37%. Wenn davon ausgegangen wird, dass nur 25% der 2getHR-User gleichzeitig aktiv sind, entsprechen die 10 User einer Zunahme von 1.48%. Es ist somit davon auszugehen, dass die Performance-Einschränkung von maximal 10% durch das Admin-Portal gemäss diesem NFR im Produktivbetrieb nicht überschritten wird. Damit dies auch im Produktivbetrieb verifiziert werden kann, empfiehlt das Projektteam 2BIT nach dem Deployment des neuen Admin-Portals in der Produktivumgebung dort Lasttests inkl. Performance-Messungen durchzuführen.

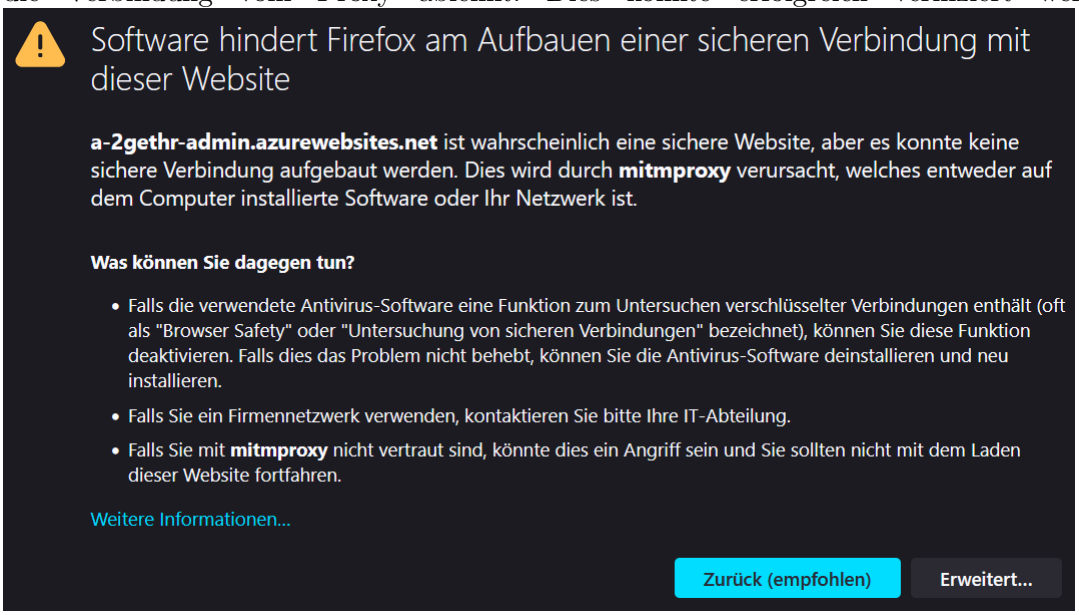
NFR03	Wie Beta Release.
NFR04	Wie Beta Release.
NFR05	Es wurde ein lokaler Proxy mit mitmproxy [mit] aufgesetzt. Auf dem Proxy wurde ein selbst-signiertes Zertifikat hinterlegt. Danach wurde überprüft, dass das Admin-Portal die Verbindung vom Proxy ablehnt. Dies konnte erfolgreich verifiziert werden: 
NFR06	Wie Beta-Release.
NFR07	Axios bietet integrierte sanitized Response für API Methoden. Gefährlicher Input wurde versucht in Tests einzubauen - Axios escapte den Input entsprechend.
NFR08	Wie Beta-Release.
NFR09	E-Mail-Adressen werden auf korrektes Format geprüft. Die Suche wird bei der Eingabe nicht validiert, um die Suchmöglichkeiten nicht einzuschränken. Der Suchstring wird von Axios sanitized, um ausführbaren Code zu verhindern und die im Backend benutzte LINQ Bibliothek escaped jeglichen Datenbank Input.
NFR10	Wie Beta-Release.
NFR11	Wie Beta-Release.
NFR12	Wie Beta-Release.
NFR13	Wie Beta-Release.
NFR14	Wie Beta-Release.
NFR15	Alle angezeigten Texte des Portals wurden in ein JSON-File ausgelagert. JSON wurde gewählt, da dies das Format ist, das in der 2getHR Frontend Applikation verwendet wird.
NFR16	Lirbaries wurden alle auf neuste Version updated.
NFR17	Wie Beta-Release.

Tabelle 3.45: Stand der NFR per Final Release

Kapitel 4

Domänenanalyse

4.1 Domänen-Diagramm

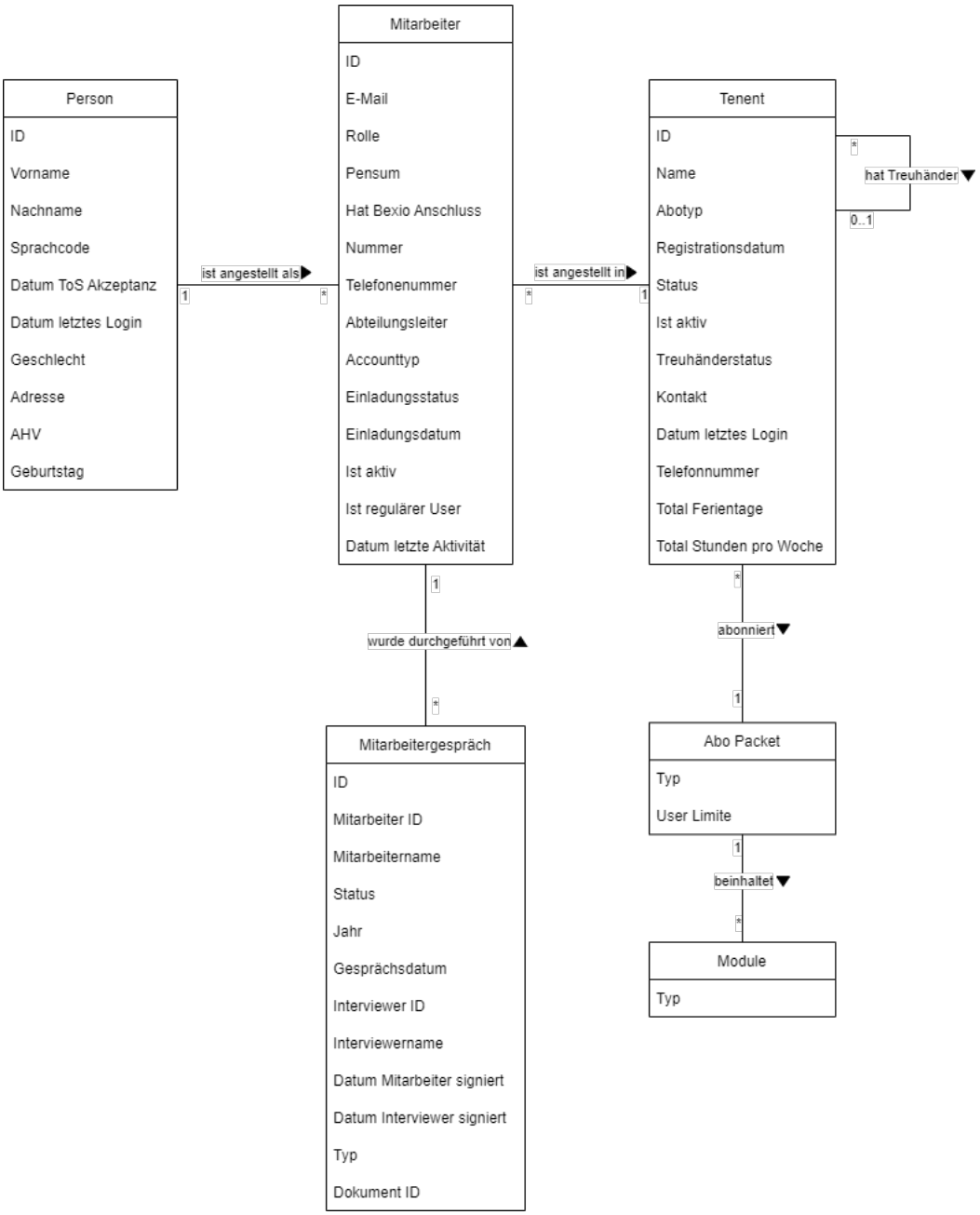


Abbildung 4.1: Domänen-Diagramm

4.2 Terminologie

Tenent

Ein Tenent ist eine bei 2getHR registrierte Firma. Eine Firma wird mit folgenden fürs Admin-Portal relevanten Informationen definiert.

ID - Beim Erstellen des Tenants zugewiesener Hash, mit der die Firma identifiziert wird.

Name - Name der Firma

Abotyp - Bei der Registration, wenn nicht anders spezifiziert ist, wird das Abonnement der Firma auf `FREE` gestellt. Dies kann jederzeit vom Support-Personal geändert werden. Die bestehenden Möglickheiten sind `FREE`, `PRO` und `CUSTOM`.

Registrationsdatum - Datum an dem die Firma registriert wurde.

Status - Der Status der Firma.

Ist aktiv - Zeigt, ob die Firma aktiv ist.

Treuhänderstatus - Zeigt, ob die Firma als Trustee registriert und verifiziert ist.

Kontakt - Kontakt, unter der die Firma erreicht werden kann.

Datum letztes Login - Datum des letzten Logins eines Mitarbeiters der Firma.

Telefonnummer - Telefon Nummer der Firma.

Total Ferientage - Anzahl der Standard Ferientage der Firma.

Total Stunden pro Woche - Anzahl der Stunden, die während der Woche standardmässig gearbeitet werden.

hat Treuhänder

Eine Firma kann bei 2getHR als Trustee, sprich Treuhänder, registriert werden. Ist eine Firma als solches definiert, kann sie andere, zugeteilte Firmen einsehen und verwalten. Ein Trustee ist selbst ein normaler Tenent bei 2getHR. Der Trustee Status muss vom Kunden (Tenant) des Treuhänders (Trustee) bestätigt werden. Der Status, ob ein Tenant ein Treuhänder ist, wird mit dem Feld Trustee State auf dem Tenant definiert.

Person

Eine natürliche Person, die sich bei 2getHR registriert hat. Eine Person kann Mitarbeiter in mehreren Tenants sein. Momentan wird dies nur für die Trustee Funktionalität verwendet.

ID Person - Beim Erstellen der Person zugewiesener Hash, mit der die Kombination von Person und Tenant identifiziert wird.

Vorname - Der Vorname des Mitarbeiters ist auf der Person definiert, um Duplikate zu vermeiden.

Nachname - Der Nachname des Mitarbeiters ist auf der Person definiert, um Duplikate zu vermeiden.

Sprachcode - 2getHR Anzeige Sprache für die Person.

Datum ToS Akzeptanz - Datum an dem die ToS akzeptiert wurden.

Datum letztes Login - Datum an dem die Person sich zuletzt in 2getHR eingeloggt hat.

Geschlecht - Geschlecht der Person.

Adresse - Adresse der Person.

AHV - AHV Nummer der Person.

Geburtsdatum - Geburtsdatum der Person.

Mitarbeiter

Ein Mitarbeiter ist ein Mitarbeiter einer Firma, sprich eines Tenants. Eine (natürliche) Person kann bei 2getHR bei mehreren Tenants als Mitarbeiter registriert sein.
Mitarbeiter werden per E-Mail-Einladung in einer Firma registriert.

ID Mitarbeiter - Beim Erstellen des Mitarbeiter zugewiesener Hash, mit der die Kombination von Person und Tenant identifiziert wird.

E-Mail - Supportanfragen werden anhand von E-Mail-Adressen auf die Personen gemappt. Die E-Mail-Adresse ist also ein besonders wichtiges Attribut für das Admin-Portal.

Rolle - Rolle des Mitarbeiters in der Firma. Dies kann die Werte Administrator, Mitarbeiter und Externer Mitarbeiter einnehmen.

Pensum - Eingestelltes Arbeitspensum des Mitarbeiters.

Hat Bexio Anschluss - Definiert, ob der Mitarbeiter eine Verbindung zu Bexio registriert hat. Gibt keine Identifizierenden Werte an, nur die Existenz der Verbindung.

Nummer - Personal Nummer des Mitarbeiters in der Firma.

Telefonnummer - Telefonnummer des Mitarbeiters.

Abteilungsleiter - Definiert in welcher Abteilung ein Mitarbeiter registriert ist.

Accounttyp - Definiert den Account Typen des Mitarbeiters. Kann folgende Werte einnehmen Post und Bank. Wird nicht angezeigt werden.

Einladungsstatus - Status der E-Mail Einladung für den Mitarbeiter. Dies definiert, ob die Einladung verschickt wurde und ob der Mitarbeiter sie akzeptiert hat.

Einladungsdatum - Datum des Versands der E-Mail Einladung.

Ist regulärer User - Definiert, ob Mitarbeiter einen Regulären Account besitzt oder ein Treuhänder Access benutzt wird.

Datum letzte Aktivität - Datum der zuletzt getätigten Aktion in 2getHR.

Mitarbeitergespräch

In 2getHR können Mitarbeitergespräche digital durchgeführt werden. Ein zuständiger Mitarbeiter kann ein Gespräch auslösen. Der Fragebogen des Gesprächs wird von beiden Teilnehmern separat ausgefüllt und freigegeben. Ist der Fragebogen von beiden freigegeben, kann das Gespräch durchgeführt werden. Schlussendlich müssen beide Teilnehmer das Gespräch digital signieren und somit abschliessen. Für allfällige Supportfälle muss das Admin-Portal Kenntnis über die Existenz aller Mitarbeitergespräche und deren Status haben. Der Inhalt des Gesprächs darf aus Datenschutzgründen im Admin-Portal nicht angezeigt werden.

ID - Beim Erstellen des Mitarbeitergesprächs zugewiesener Hash, mit dem das Gespräch identifiziert wird.

Mitarbeiter ID - ID des Mitarbeiters, über den das Gespräch gehalten wird.

Mitarbeitername - Name des Mitarbeiters, über den das Gespräch gehalten wird.

Jahr - Jahr des Mitarbeitergesprächs.

Gesprächsdatum - Datum an dem das Gespräch ausgeführt werden soll.

Status - Status des Gesprächs.

Interviewer ID - ID des Mitarbeiters, der das Gespräch leitet.

Interviewername - Name des Mitarbeiters, der das Gespräch leitet.

Datum Mitarbeiter signiert - Datum, an dem der Mitarbeiter den Fragebogen signiert hat.

Datum Interviewer signiert - Datum, an dem der leitende Mitarbeiter den Fragebogen signiert hat.

Typ - Definiert das Fragebogen Template, das für den Fragebogen benutzt wurde.

Dokument ID - Identifizierender Hash, der dem kompilierten Dokument des Gesprächs gegeben wird.

Abo Packet

Ein Abonnement für einen 2getHR-Tenant wird per Abo Packet gesteuert. Dieses definiert wie viele Mitarbeiter im Tenant registriert werden dürfen und welche Module für den Tenant freigeschaltet sind. Tenants teilen sich Package, wenn diese gleich eingestellt sind.

Typ - Typ des Abos. Bei der Registration, wenn nicht anders spezifiziert ist, wird das Abonnement der Firma auf `FREE` gestellt. Dies kann jederzeit vom Support-Personal geändert werden. Die bestehenden Möglichkeiten sind `FREE`, `PRO` und `CUSTOM`.

User Limite - Anzahl zur Verfügung stehende Mitarbeiter. Wird diese Anzahl überschritten, ändert sich der Abopreis. Hinweis: Bis zu 5 User werden pauschal verrechnet.

Module

Module sind 2getHR Features, die für Subscription Packages individuell freigeschaltet werden können. Folgende Module existieren in 2getHR:

1. Expenses
2. Payslips
3. Holiday
4. PersonalDossier
5. Rapports
6. ReceiptDateExtraction
7. AhvSearch
8. MBO
9. TimeTracking
10. SelfService
11. Bexio
12. Webhooks
13. PublicAPI

Kapitel 5

Architektur

5.1 C4 - Modell

Level 1 - Context

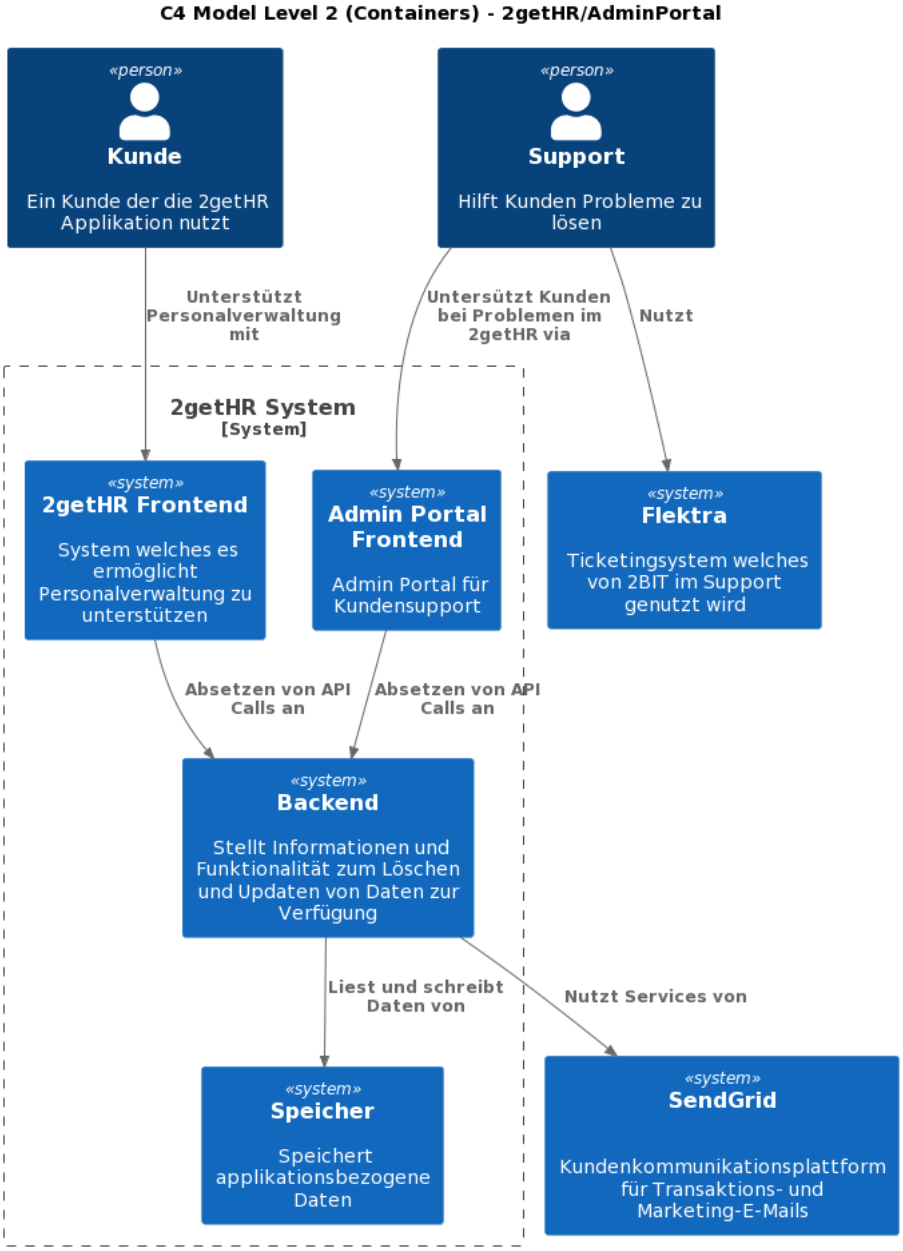


Abbildung 5.1: C4 - Modell Level 1 - Context

Das 2getHR System wird von zwei Arten von Usern genutzt: Dem Support-Personal und den Kunden der 2getHR-Applikation. Die Kunden nutzen das 2getHR-System, um die Prozesse der Personalverwaltung ihrer Firmen zu unterstützen. Der Support unterstützt seinerseits die Kunden bei Problemen im 2getHR-System. Die Komponenten innerhalb des 2getHR-Systems werden im Weiteren genauer aufgezeigt.

Level 2 - Container

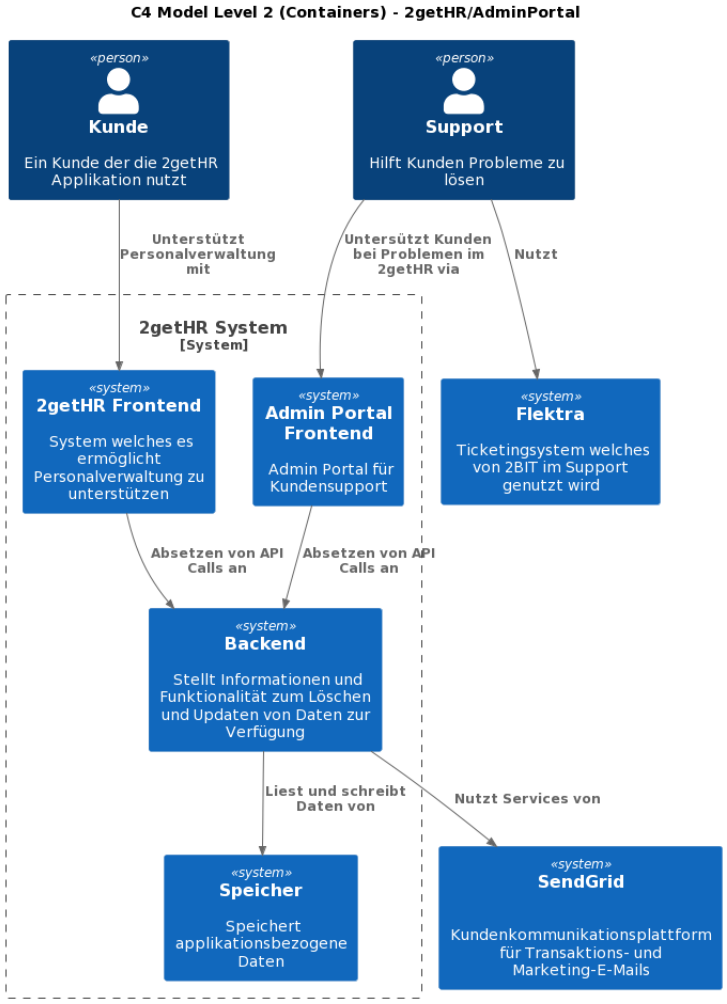


Abbildung 5.2: C4 - Modell Level 2 - Container

Für den Kunden steht im 2getHR-System ein Angular-Frontend zur Verfügung, das 2getHR-Frontend. In diesem werden dem Kunden alle Funktionalitäten bereitgestellt, die dieser mit seinem Abo eingekauft hat. Der Support hat die Möglichkeit über die React Single Page Applikation 'Admin Portal' auf Support-relevante Informationen zuzugreifen und Update- und Löschvorgänge sowie Verifizierungen durchzuführen. Beide Frontends greifen auf ein ASP.Net Core Backend zu. Backend und die beiden Frontends laufen je in einem Azure App Service auf der 2BIT Azure Subscription. Für die Persistierung von Daten wird Speicher eingesetzt, welcher in der Azure-Umgebung von 2BIT läuft.

Level 3 - Components

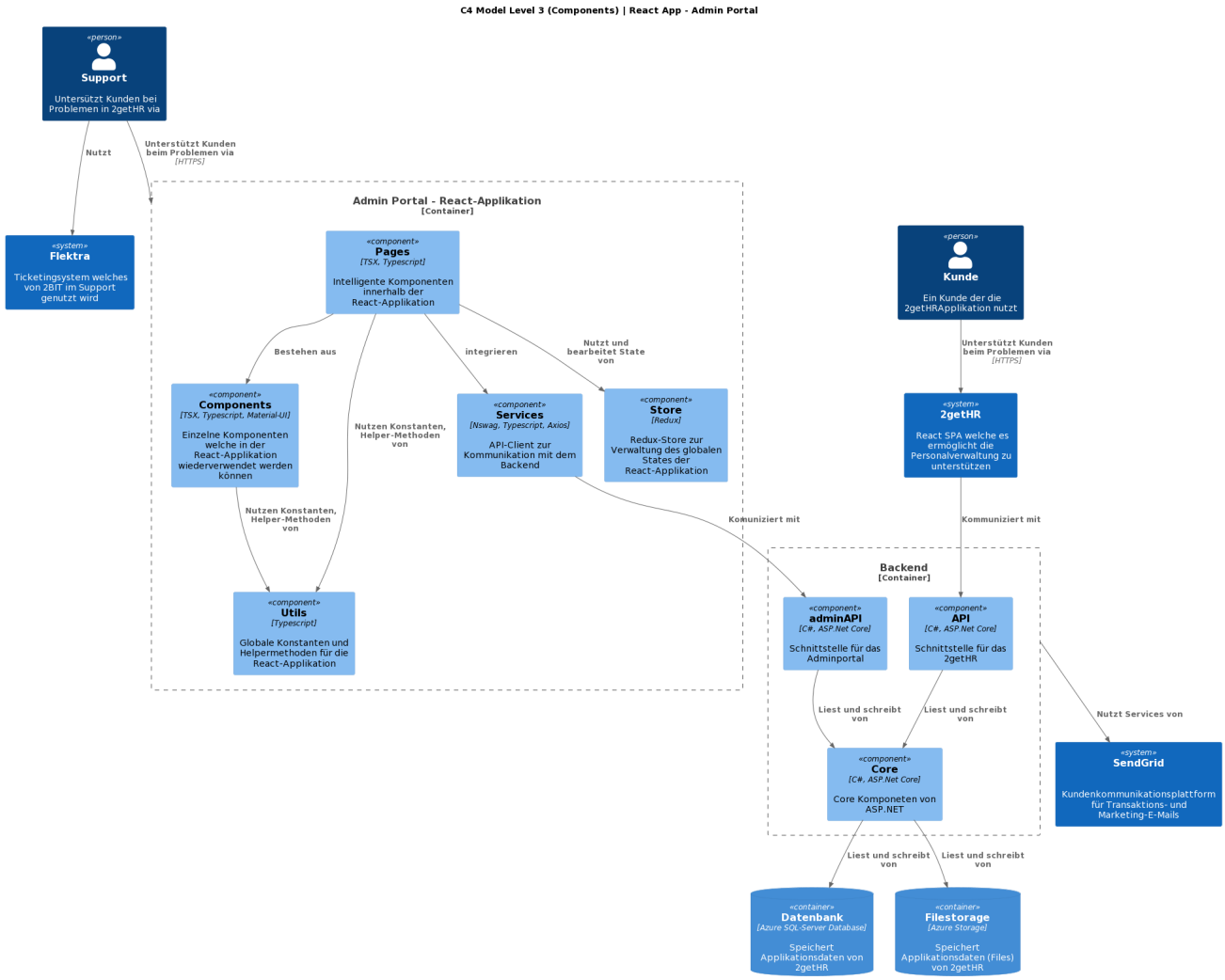


Abbildung 5.3: C4 - Modell Level 3 - Components

Die einzelnen Komponenten werden im Kapitel 5.2 im Detail erläutert.

5.2 Ordnerstruktur Frontend

Ziel

Die React SPA 'Admin-Portal' wird mit dem Ziel einer simplen und übersichtlichen Struktur entwickelt. Mit Blick auf die Weiterentwicklung und Wartung der Applikation nach Abschluss des Projektes soll es für neue Entwickler so einfach wie möglich sein, sich in der Applikationsarchitektur zurecht zu finden.

Folgende Punkte haben zu diesem Entscheid geführt:

- Eine klare Struktur senkt die Komplexität der Applikation.
- Die Gruppierung von wiederverwendbarer, zusammengehöriger bzw. gleichartiger Logik und Objekten führt zu einer besseren Übersicht.
- Alles was mit einer Komponente zu tun hat, soll auch dort zu finden sein.

Im folgenden werden die einzelnen Komponenten erläutert und der Entstehungsprozess beleuchtet.

Entscheid	**Begründung**
Pages	Pages sind intelligente Komponenten der Applikation. Sie verwalten alle Aktionen die Verbindungen zu anderen Komponenten beinhalten, wie zum Beispiel die Verwaltung des Stores. Die Pages bestehen aus einer Aggregation von dummen Unterkomponenten, denen alle nötigen Informationen mitgegeben werden und die von der Parent Page verwaltet werden. Dies, um die Komplexität der Applikation zu senken und eine saubere Trennung der Zuständigkeiten zu erreichen. Pages selbst enthalten nur minimale UI Beschreibung wo nötig.
(Custom) Components	Dumme Komponenten, die nur Informationen entgegennehmen, Aktionen und Informationen zurückgeben und das UI beschreiben. Einzige Ausnahme ist hier die 'SearchBar' Komponente, welche Logik für Navigation und API-Calls enthält. Diese wurde nicht in den Pages Ordner verschoben, da sie eine zu kleine Komponente ist. Trotzdem enthält sie komplexe Logik, um den Suchvorgang abzubilden. Pro Komponente wird ein Unterverzeichnis erstellt, welches folgende Files beinhaltet. • Component.ts File welches die eigentliche Komponente enthält. • Component.styles.ts File welches Styles für die Komponente enthält, falls benötigt.
Store	Das Store Verzeichnis enthält alle Redux-spezifischen Komponenten. Sprich den Store selbst, die Slices und die Konfiguration werden hier abgespeichert.
Utils	Im Utils-Ordner werden globale Funktionen sowie Helper-Methoden in eigenen Unterverzeichnissen verwaltet.

Services	Zu den Services gehört aktuell der API-Client. Dieser soll nur von den Pages genutzt werden. Dadurch soll einerseits eine klare Richtlinie geschaffen werden für einen saubereren Programmierstil und andererseits soll der API-Client isoliert bleiben.

Tabelle 5.1: Design-Entscheide

5.3 Technologien

Nachfolgend werden die eingesetzten Technologien aufgelistet. Eine genauere Analyse weshalb diese ausgesucht wurden, finden sich im folgenden Kapitel.

Feld	**Library/Framework**	**Beschreibung**
Developement Frontend	React / Typescript	• Die React Library wird für die Implementation des Frontends eingesetzt. • Typescript ist die verwendete Programmiersprache.
Developement Frontend	Redux	• Redux wird genutzt um den State der Applikation global zu verwalten und einzubinden.
Autorisierung Frontend	Azure/MSAL	• Microsoft Authentication Library wird für Login Funktionalitäten eingesetzt.
API-Client Frontend	NSwag mit Axios	• Nswag wird eingesetzt für das Generieren des API-Clients. • Der API-Client basiert auf Axios.
Developement Backend	ASP.Net / C#	• Das Backend basiert auf ASP.NET. • Die verwendete Programmiersprache ist C#.
Testing Backend	XUnit	Wird für das Unit-Testing eingesetzt.
Testing Frontend	Cypress	Wird für das E2E-Testing eingesetzt.
UI Library	Material Design	Wird für das Design der Applikation eingesetzt.
Package Manager	Yarn	Wird für das Paket-Management eingesetzt.
Tabellen-Komponente in der React App	Material React Table	Wird für die Tabellenkomponenten in der React-Applikation eingesetzt.

Tabelle 5.2: Eingesetzte Technologien

Technologie-Entscheide

Folgend wird erläutert weshalb die gewählten Technologien von vorherigem Kapitel eingesetzt wurden.

Library/Framework	Beschreibung
React	• Warum React? – Gut geeignet für die Grösse des Projektes – Verbreitete Technologie, die aktiv weiterentwickelt wird – Bestehendes Admin-Portal wurde mit React entwickelt – Know-How ist im Projektteam vorhanden – Gute Dokumentation – MIT-License • Geprüfte Alternativen – Angular (zu viel Overhead für die Grösse der Applikation) – Vue.js (React ist geeigneter für reine Entwickler-Teams (ohne Designer) aufgrund der grösseren Flexibilität. Zudem ist die React-Community grösser als die Vue.js-Community [Rea]). – HTML CSS JS (Overhead, da alles von Grund auf neu gebaut werden müsste)
Redux	• Warum Redux? – Popularität in der React Domäne – Ohne zentralem State-Management wird der Zustand der Applikation verstreut gehalten – Redux liefert einen Single-Point-of-Truth – Know-How ist im Projektteam vorhanden – MIT-License • Geprüfte Alternativen – Mobx (Popularität, Kein Know-How vorhanden) – Recoil (Nicht so gut für die definierte Architektur geeignet wie Redux, kein Know-How vorhanden)
Azure/MSAL	• Warum Azure/MSAL? – Bestehende Lösung wird übernommen. – Offizielle Library von Azure für Azure Active Directory Authentication. – MIT-Lizenz
NSwag	• Warum NSwag? – Bestehende Lösung wird übernommen. – Qualitatives Tool für Generieren von API-Clients. – Empfohlen durch Experten – MIT-Lizenz

Axios	• Warum Axios? – Bestehende Lösung wird übernommen. – Beliebtes Tool für TypeScript Clients. – MIT-Lizenz
.NET / C#	• Warum .NET / C#? – Design Contraint, da das bestehende API ausgebaut wird. – Das bestehende Backend ist mit .NET und C# entwickelt. – Das bestehende Backend wird auch von der 2getHR-Applikation verwendet. – Apache-Lizenz 2.0
XUnit	• Warum XUnit? – Bestehendes Backend wird mit XUnit getestet. – Apache-Lizenz 2.0
Cypress	• Warum Cypress? – Design Contraint, 2BIT verlangt Cypress e2e Tests für Produktive Software. – Dokumentation und Ressourcen – Aktive Community – Know-How ist im Projektteam vorhanden – MIT-License • Geprüfte Alternativen – Framework Playwright (Zu inaktive Community) – Library Puppeteer (Tooling gegenüber Cypress, Reliability)
Material UI	• Warum Material UI? – Design Contraint, Software sollte ins Design Konzept von 2BIT passen, was auf Material UI basiert. – Dokumentation und Ressourcen – Community – Einfache Handhabung – Know-How ist im Projektteam vorhanden – Gewünschtes Design – MIT-License – Erfüllt 2BIT Design Konzept (wird auch in der 2getHR-Applikation eingesetzt) • Geprüfte Alternativen – React Bootstrap (Design ist nicht wie gewünscht) – Reactstrap (Design ist nicht wie gewünscht)

Typescript	• Warum Typescript? – Typisiert – Bessere Wartbarkeit – Fehler werden einfacher ersichtlich – Apache-Lizenz
Material React Table	• Warum Material React Table? – Ausgezeichnete Dokumentation – Alle benötigten Funktionalitäten out of the box – MIT-License • Geprüfte Alternativen – React Data Grid – Material UI Table – React-Table
Yarn	• Warum Yarn? – Ausgezeichnete Dokumentation – Alle benötigten Funktionalitäten out of the box – MIT-License • Geprüfte Alternativen – React Data Grid – Material UI Table – React-Table

Tabelle 5.3: Technologie-Entscheide

5.4 Design-Entscheide

Entscheid	**Begründung**
Positionierung von Filter für Listenansichten	Für die Positionierung von Filter bei den Listenansichten wurden zwei Varianten analysiert: 1. Filter direkt in den Spalten-Headern der Tabelle einsetzen. 2. Filter oberhalb der Tabellen als eigenes Panel anzeigen. Da es in der Listenansicht nur Enum-Filter für ausgewählte Felder gibt, wurde die Variante 2 umgesetzt. Dies soll die Liste übersichtlich halten und die filterbaren Kolumnen reduzieren, was weniger repetitiven Code im API verursacht. Die im Vergleich zur bestehenden Lösung fehlenden Filter werden mehrheitlich mit der Suche ersetzt oder sind für den Supportprozess nicht nötig.

(Soft-)Undo	Im Team wurde anfänglich entschieden ein (Soft-)Undo zu implementieren. Diese Funktionalität wäre auf die aktuelle Session eingeschränkt gewesen. Dies, um dem User mehr Komfort und Sicherheit für Datenänderungen zu geben und Userfehler möglichst zu verhindern, die mühsam wiederhergestellt werden müssten. Aufgrund des Umfangs der Entwicklung und fehlender zeitlicher Ressourcen, wird diese Funktionalität nicht umgesetzt.
Table Implementation	Für die Tabelle wurden per Default das 'dense' Layout gewählt, um unnötigen Platzverlust zu vermeiden. Es wurde von Sticky Header abgesehen, da diese Einstellung ungewolltes Verhalten auslöst, sprich einen seitlichen Overflow der Tabelle produziert. Damit die Tabelle möglichst komplett auf dem Bildschirm angezeigt werden kann und Header und Footer möglichst sichtbar bleiben, wurde eine Default Pagesize von 10 Einträgen gewählt. Dies kann vom User jedoch angepasst werden und wird im lokalem Speicher Session übergreifend hinterlegt.
Darstellung Datenfelder	E-Mail und Kontakt wurden prominent in den Detailansichten und in den Standardkonfigurationen der Listenansichten positioniert. Reihenfolge von den Datenfeldern wurde vom Supportpersonal kontrolliert und entsprechend angepasst.
Zugang Listenkonfiguration	Positionierung des Buttons zur Listenkonfiguration (Spaltenvisibilität und Ordnung) wurde auf Wunsch von Supportpersonal am oberen Rand der Liste positioniert.
Kontrast	Farbgebung wurde angepasst, insbesondere auf der Detailview, da Gelb auf Weiss nicht optimal ist.
Unterscheidung Tenant und Treuhänder	In der Detailansicht wurde ein Unterschied zwischen Tenant und Treuhänder gemacht, um unnötige Verwirrung zu vermeiden und diese Information prominent anzugeigen. Auch änderte sich eine der integrierten Listenansichten, entsprechend wurde entweder die Kunden des Treuhänders angezeigt oder die Liste der Treuhänder.
Mitarbeiter-Listenansicht	Auf der Mitarbeiter-Listenansicht wurde die Sortierung der Filter angepasst. Der Tenantfilter wurde besser platziert. Es wurde kein neuer Filter "Mitarbeiter-Nameëingestzt, da diese Funktionalität über die Suche abgedeckt wird.
Symbolik und Tooltipps	Symbole wurden so gewählt, dass diese nicht falsch interpretiert werden können. Zudem wurden Tooltips angebracht, um bei Unsicherheit zu helfen.
Detail View	Es wird keine Suche innerhalb der Detailview angeboten, da alle Informationen übersichtlich auf einen Blick darstellt werden. Falls ein Datenfeld nicht sofort gefunden wird, hat der Nutzer über die Suchfunktion im Browser (CRTL + F) die Möglichkeit Begriffe zu finden. Die Felder ToS und Mitarbeitertyp werden weggelassen aufgrund dessen, dass sie nicht benötigt werden. Standardmässig werden die Gelinkten Objekte ausgeklappt dargestellt. Der Mechanismus des ein- resp. Ausklappens wird ganz entfernt, da dieser zur Verwirrung bei den Usern führte.

5.5 Technische Entscheide

Entscheid	Begründung
Multiuser Handling	Das Multiuser-Handling wird in Frontend kontrolliert. Dies um die Handhabung zu erleichtern und das API isoliert zu halten. Beim Aufruf der Update-Funktionalität wird im Frontend ein erneuter Fetch der Daten gemacht und diese werden mit den bestehenden abgeglichen. Stimmen die Daten überein wird der Update API-Call abgesetzt. Sind die Daten unterschiedlich wird dem User eine entsprechende Nachricht angezeigt und der Vorgang abgebrochen (siehe auch Tabelle 3.40).
Delete	Es wird von einem logischen Löschen der Daten abgesehen und ein physisches Löschen umgesetzt. 2getHR bietet keine Funktionalität für logisches Löschen und es müssten grosse Änderungen gemacht werden, um dies zu implementieren. Zudem werden von 2getHR sensible Userdaten gemanagt, die nicht aufbewahrt werden dürfen, wenn der User sein Einverständnis zurückzieht. Ein logisches Löschen würde dies also nicht korrekt umsetzen. Zudem wurde von 2BIT explizit gewünscht von einem logischen Löschen abzusehen.
Log wird nach erfolgreicher Aktion abgesetzt	Log-Einträge werden erst nach erfolgreichem Update oder Löschen abgesetzt. Dies um die Last im Backend zu minimieren und um sicherzustellen, dass nur erfolgreiche Aktionen geloggt werden.

5.6 Wireframes

Die Wireframes wurden als High Fidelity Prototyp erstellt, mit bereits stark ausgearbeitetem Design. Dies, um ein möglichst nahes Benutzererlebnis für die ersten Usability-Tests zu erstellen und mit der bestehenden Lösung konkurrenzieren zu können. Ein reduziertes Wireframe, welches nur die Funktionalität abgebildet hätte, wären nicht sehr nützlich gewesen für die ersten Usability-Tests.
Folgend sind die wichtigsten Elemente der Wireframes aufgezeigt. Für eine ausführlichere Einsicht kann im Anhang das Figma File eingesehen werden.

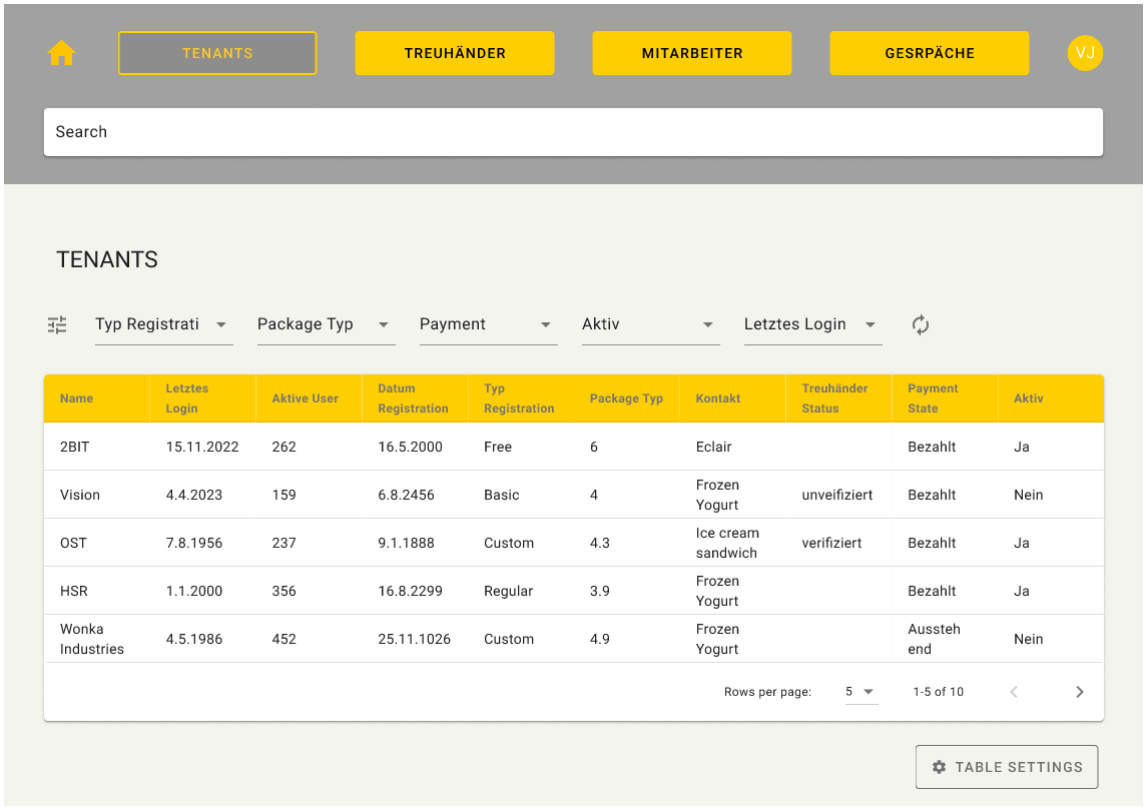


Abbildung 5.4: Beispiel Listenansicht

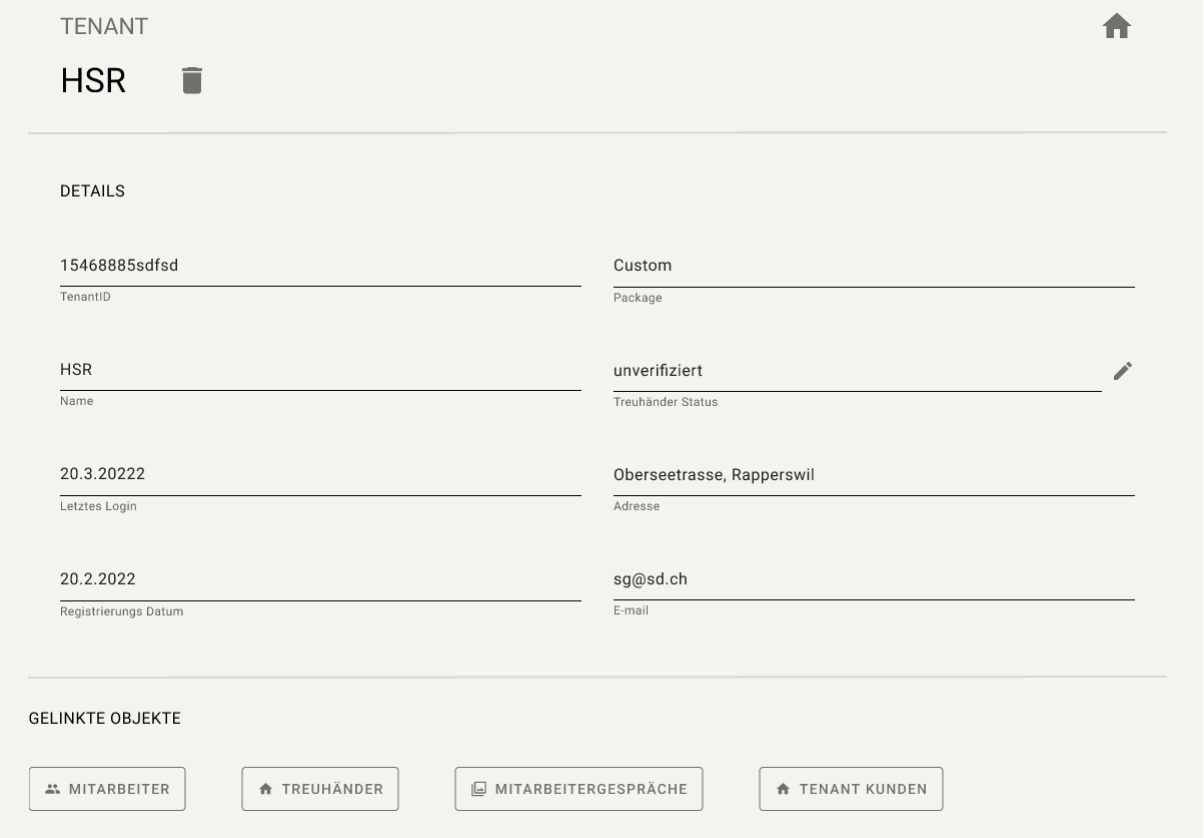


Abbildung 5.5: Beispiel Detailansicht

5.7 Twelfe-Factor Methode

Die Twelfe-Factor Methode [12f] wurde eingesetzt in der Analyse der bestehenden Umgebung und der Planung der Applikation um die Admin-Portal Applikation cloudnative zu gestalten. Folgende Methoden wurden aus der Analyse ausgearbeitet und in der Applikation eingesetzt.

I. Codebase

Eine im Versionsmanagementsystem verwaltete Codebase, viele Deployments. [12f]

Die bereits existierende DevOps-Einrichtung von 2BIT auf Azure DevOps wird verwendet für das Projekt. Die 2getHR-Codebasis und das Admin-Portal sind dabei im selben Repository gespeichert. Da 2getHR und eine Basisversion des Admin-Portals bereits produktiv im 2BIT System laufen, wird dieses Projekt während des Projektverlaufs nicht in das produktive System deployed. Stattdessen wird das Projekt nach erfolgreichem Abschluss an 2BIT übergeben und gemeinsam mit 2BIT in die produktive Umgebung integriert. Desweiteren wird der Betrieb und die Weiterentwicklung nach erfolgreichem Abschluss an 2BIT übergeben. Pipelines für das Deployment in eine Testing- und
Integrations-Umgebung sind bereits vorhanden und werden von 2BIT gewartet. Eine eigene Entwicklungsumgebung wurde für dieses Projekt zur Verfügung gestellt. Die Deployments werden über Pipeline-Variablen an die verschiedenen Umgebungen gerichtet und sind so komplett unabhängig von der Codebase.

✔ **Evaluation:** Erfüllt

II. Dependencies/Abhängigkeiten

Abhängigkeiten explizit deklarieren und isolieren. [12f]

Abhängigkeiten im React Frontend werden per Yarn gehandhabt, um eine optimale Isolation des Dependency Managements zu erreichen. Die Abhängigkeiten werden im File yarn.lock gelistet und können ohne Probleme gemanagte werden. Für .NET-Entwicklungen im Backend wird der Package Manager nuget eingesetzt. Dieses erlaubt ein einfaches Packagemanagement im Visual Studio eigenen Graphical Interface.

✔ **Evaluation:** Erfüllt

III. Config/Konfiguration

Die Konfiguration in Umgebungsvariablen ablegen. [12f]

Konfigurationen für Deployment wird mit Pipeline-Umgebungsvariablen gehandhabt. Für die restlichen Konfigurationen werden Umgebungsvariablen in den Azure App Service Umgebungen benutzt.

✔ **Evaluation:** Erfüllt

IV. Backing Services / Unterstützende Dienste

Unterstützende Dienste als angehängte Ressourcen behandeln. [12f]

Das Backend API des Admin-Portals greift auf ein isoliertes 2getHR .NET Projekt 'Core' zu, in dem alle notwendigen Services definiert sind. Diese angehängte Ressource wird in einem eigenen React Modul isoliert und ist per Interface-Aufruf einfach ersetzbar. So kann das Backend ohne Änderungen im Frontend ersetzt oder angepasst werden. Der Aufruf der Datenbank wird im Core gehandhabt und ist bereits

isoliert umgesetzt. Das Admin API Backend greift ausschliesslich über ein Context Interface darauf zu.

✔ **Evaluation:** Erfüllt

V. Build, Release, Run

Build- und Run-Phase strikt trennen. [12f]

Build, Testing und Metriken werden in einer separaten Pipeline je für Backend und Frontend ausgeführt. Release und Deployment basiert auf den Build Artefakten der vorangegeangenen Pipeline und wird nur angestossen, wenn diese fehlerfrei durchläuft. Weitere Details zur Pipeline werden in Kapitel 6.1 ausgeführt.

✔ textbfEvaluation: Erfüllt

VI. Processes / Prozesse

Die App als einen oder mehrere Prozesse ausführen. [12f]

Die Applikation besteht aus zwei unabhängigen Prozessen, React Frontend und .NET API Backend. Beide Prozesse sind stateless und können ohne Datenverlust terminiert und neu gestartet werden. Daten werden in der 2getHR Datenbank persistiert. Zusätzlich dazu läuft das 2getHR Frontend ebenfalls in einem eigenen Prozess, der jedoch nicht Teil des Projektes ist.

✔ **Evaluation:** Erfüllt

VII. Port Binding / Bindung an Ports

Dienste durch das Binden von Ports exportieren. [12f]

Die beiden Framworks .NET und React benutzen Port-Binding.

✔ **Evaluation:** Erfüllt

VIII. Nebenläufigkeit

Mit dem Prozess-Modell skalieren. [12f]

Alle drei Prozesse des 2getHR Universums, 2getHR Frontend, Admin-Portal Frontend und Backend laufen unabhängig voneinander. Das Scaling des Backends wird von 2getHR gesteuert und wird von 2BIT gemanagte. Das SPA Frontend vom Admin-Portal hat keine Anforderungen an Scaling, da keine Performance-intensiven Aktionen ausgeführt werden und die Userbase in den nächsten Jahren nicht über 10 Personen heraus wachsen wird.

✔ **Evaluation:** Erfüllt soweit Teil des Projektes

IX. Disposability / Einweggebrauch

Robuster mit schnellem Start und problemlosen Stopp. [12f]

Das Admin Portal Frontend ist stateless und kann ohne Mehraufwand gestartet und gestoppt werden. Das AdminAPI, das als Backend der Applikation dient, ist mit dem 2getHR Backend verbunden und wird von 2BIT vorgegeben. Das API ist ebenfalls stateless und alle Änderungen, die im Rahmen dieses Projektes vorgenommen wurden, sind cloudnative gerecht implementiert.

✔ **Evaluation:** Erfüllt

X. Dev/Prod Parity / Dev-Prod-Vergleichbarkeit

Entwicklung, Staging und Produktion so ähnlich wie möglich halten. [12f]

Die Umgebungen werden von 2BIT gemanagte. Die Backend Umgebung und das Admin-Portal laufen auf komplett identischen Azure App Services. Für dieses Projekt wurde die Dev-Umgebung mit den Dev-Daten kopiert und separiert. Werden Änderungen von 2BIT vorgenommen, so wird uns dies kommuniziert und eine konsistente Umgebung kann wieder hergestellt werden. So ist sichergestellt, dass die Reintegration in die 2BIT Dev-, Test- und Prod-Umgebungen ohne Probleme durchgeführt werden kann.

✔ **Evaluation:** Erfüllt

XI. Logs

Logs als Strom von Ereignissen behandeln. [12f]

Es wurden keine klassischen Logs implementiert. Die kann auf Kundenwunsch geändert werden. Das Aktionsjournal wird per API-Call an das Backend geloggt. Dieses wird auf den Standard-Out Stream geschrieben.

✔ **Evaluation:** Erfüllt

XII. Admin Processes / Admin-Prozesse

Admin/Management-Aufgaben als einmalige Vorgänge behandeln. [12f]

Einmalige Administrationsprozesse werden für 2getHR in der Dev- oder Test-Umgebung ausgeführt und getestet, bevor sie übernommen werden. Diese sind identisch zu der Prod-Umgebung. Für das Admin-Portal werden diese Prozesse auf der zur Verfügung gestellten separaten Umgebung ausgeführt, da diese nicht produktiv benutzt wird aber identisch zu den 2getHR Umgebungen gehalten wird. Jeglicher Code der Prozesse wird im Azure DevOps Repository versioniert. Datenbank spezifische Migrationen und Änderungen werden im Projekt EF.Migrations versioniert.

✔ **Evaluation:** Erfüllt

5.8 API - Definition

Benötigte Anpassungen

MVP

Basierend auf der Analyse des bestehenden APIs (siehe Kapitel 2.2) wurden folgende nötigen Änderungen identifiziert.
Für das MVP sind folgende Funktionalitäten nicht vorhanden:

- Route für alle Employees (zurzeit nur pro Tenant möglich)
- Route für alle Treuhänder (zurzeit nur alle Tenants abrufbar, braucht auf Treuhänderstatus filterbarer Endpunkt)
- Route für alle Mitarbeitergespräche (zurzeit nur pro Tenant möglich)
- Endpunkte sind zurzeit nicht paginiert abrufbar.
- Endpunkte zurzeit nicht gefiltert abrufbar.
- Endpunkte zurzeit nicht sortiert abrufbar.

Der bestehende API-Client AadController wird für bessere Übersicht aufgetrennt in Tenants, Employees und AppraisalInterviews Controller. In den entsprechenden Controllern müssen folgende neue Routen implementiert werden:

- /api/Tenants/GetTenantsPaginated
  ?id: string
  &desc: bool
  &pageIndex: number
  &pageSize: number
  &trusteeState=EnumValue
  &packageType=EnumValue
  &paymentState=EnumValue
  &active=EnumValue
- /api/Employees/GetEmployeePaginated
  ?id: string
  &desc: bool
  &pageIndex: number
  &pageSize: number
  &role=EnumValue
  &invitationState=EnumValue
  &isActive=EnumValue

- /api/AppraisalInterviews/GetAppraisalInterviewPaginated
  ?id: string
  &desc: bool
  &pageIndex: number
  &pageSize: number
  &state=EnumValue
  &year=EnumValue

Diese neuen Routen sollen die entsprechenden Listen gemäss den Parametern 'id' und 'desc' sortieren und entsprechend den Parametern 'pageIndex' und 'pageSize' die aktuellen Einträge der Paginierung zurückgeben. Damit das Total der Einträge korrekt gesetzt werden kann, wird nicht nur die Liste der Einträge berechnet, sondern auch das Total aller gemäss den gesetzten Filtern verbliebenen Einträge.

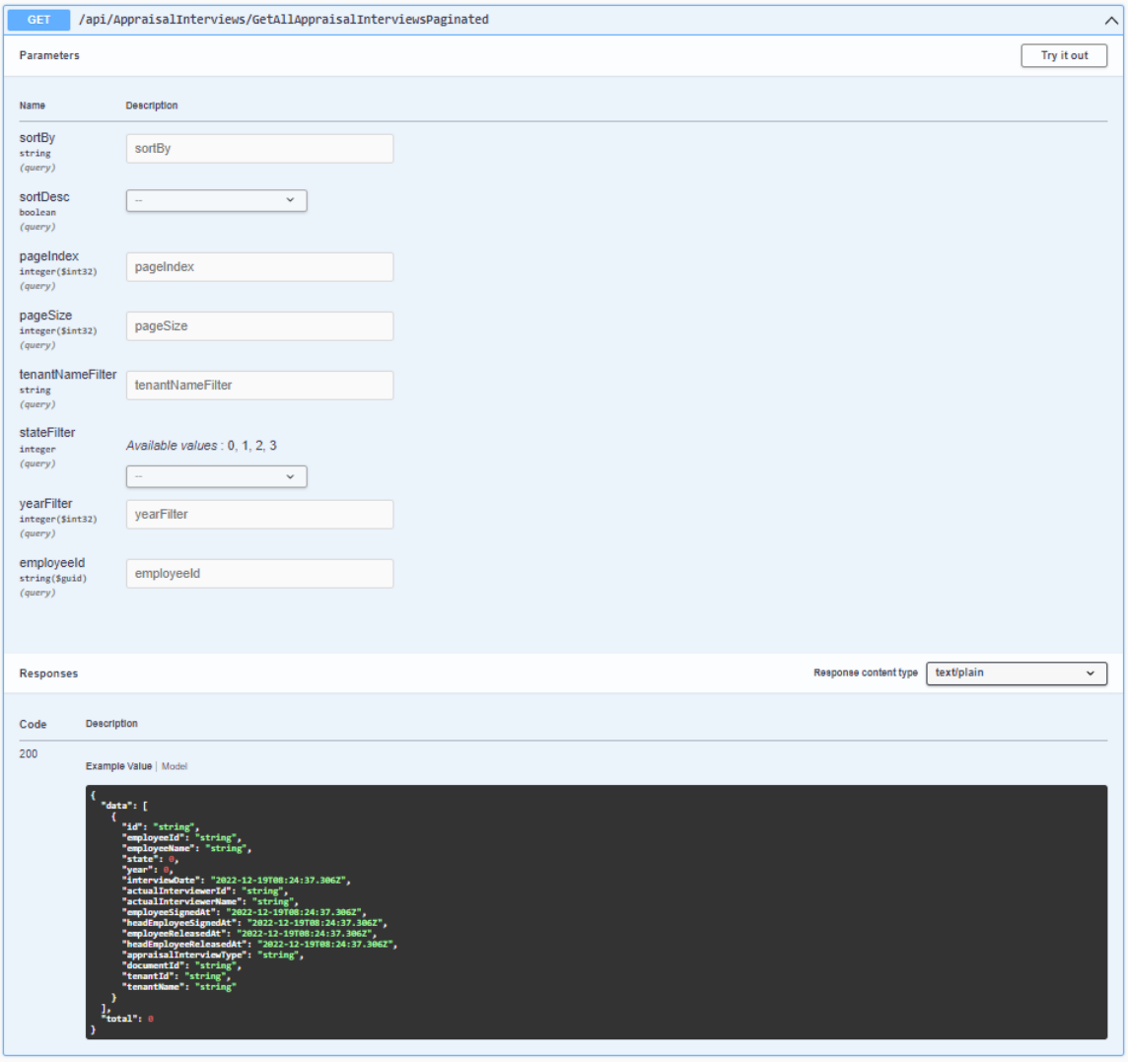


Abbildung 5.6: Swagger Beispiel der neuen API-Routen

Der fertige Call besitzt nun zusätzliche Query Parameter, die in der Implementation von Alpha und Beta Release hinzugefügt wurden.

Erweiterung DTOs

Da bei der Implementation vom MVP die Whitelist von 2BIT bereits bekannt war, wurde entschieden direkt alle in Listenansicht und Detailview benötigten Datenfelder in die DTOs einzubauen, um einen unnötigen Mehraufwand zu vermeiden. Für die Vollständigkeit der von 2BIT gewünschten Daten, mussten die DTOs, wie in den folgenden Sektionen beschrieben, erweitert werden. Auch gibt es in den DTOs Felder, die gemäss der von 2BIT definierten Whitelist nicht angezeigt werden dürfen.

TenantDTO

Datenfeld	**Nicht auf Whitelist**	**Liste**	**Detail**
id	x		x
name	x	x	x
lastLogin	x	x	x
daysSinceLastLogin	x	x	x
amountOfUsers	x	x	x
registerDate	x	x	x
registerType		x	x
package	x	x	x
contact	x	x	x
phone	x		x
trusteeState	x	x	x
paymentState		x	x
adress			x
active		x	x
standardHolidayAmount			x
standardWeeklyHours			x

Tabelle 5.6: API TenantDTO Felder Analyse

EmployeeResponseExtendedDTO

Datenfeld	**Nicht auf Whitelist**	**Liste**	**Detail**
mitarbeiterId	x		x
id	x		x
fullName	x	x	x
email	x	x	x
role	x	x	x
languageCode			x
tosAcceptedDate			x
bexio	x		x
dateOfBirth	x		x
gender	x		x
adress			x
invitationDate			x
lastActivityDate		x	x
pensum		x	x
isWithdrawn	x		
isRegular	x		
iban	x		
isNoChildren	x		
isRequestedChildAllowances	x		
ahvPhotoId	x		
azureAdObjectId	x		
canBeVerified	x		
numberOfHoliday	x		
notes	x		
ahvNr	x		
CompanySigner	x		
verified	x		
isOrganizationalUnitLeader	x		
avatar	x		

Tabelle 5.7: API EmployeeResponseExtendedDTO Felder Analyse

AppraisalInterviewDTO

Datenfeld	Nicht auf Whitelist	Liste	Detail
employeeSignedDate			x
headEmployeeSignedDate			x
employeeReleasedDate			x
headEmployeeReleasedDate			x
documentId			x
organizationalUnitId	x		
document	x		
organizationalUnitName	x		

Tabelle 5.8: API AppraisalInterviewDTO Felder Analyse

Alpha und Beta Release

Für Beta und Alpha Release wurden die API-Endpunkte iterativ definiert und implementiert. Hier wurde entschieden keine dem MVP entsprechende Dokumentation zu führen. Die fertige API kann im folgenden Abschnitt eingesehen werden.

Neue Admin-API

Die Swagger Dokumentation der Lösung wird nicht publiziert und kann nur lokal unter `localhost:5005/swagger` eingesehen werden. Aus Sicherheitsgründen wird davon abgesehen das API in diesem Dokument genauer zu dokumentieren.
Alle benötigten Routen konnten erfolgreich umgesetzt werden. Der bestehende API-Client wurde in spezifischere Controller unterteilt, um das API übersichtlicher zu organisieren.

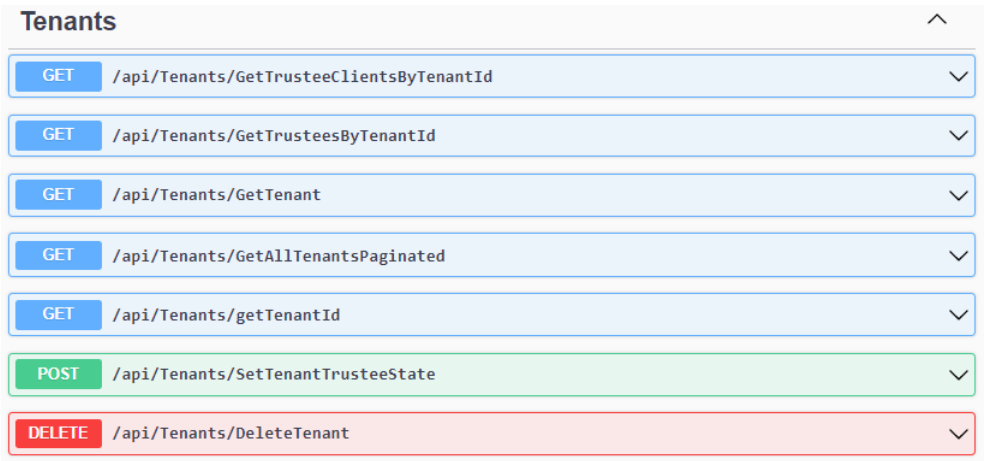


Abbildung 5.7: Admin-Portal Swagger-Dokumentation Tenant

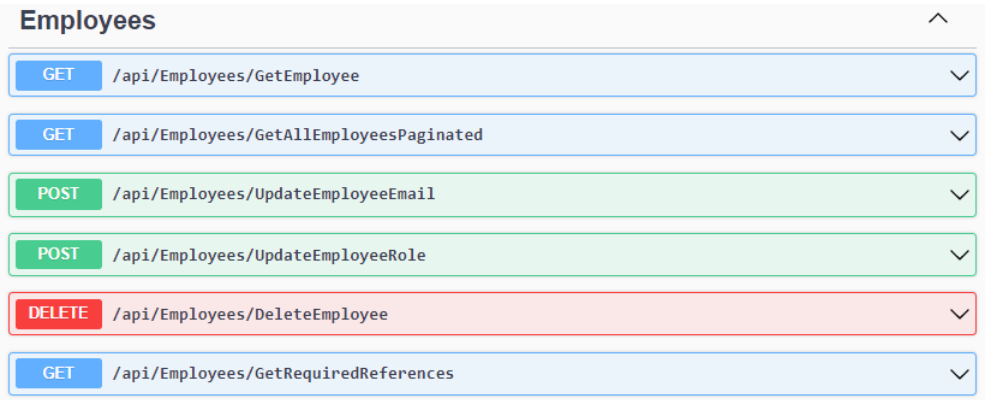


Abbildung 5.8: Admin-Portal Swagger-Dokumentation Mitarbeiter

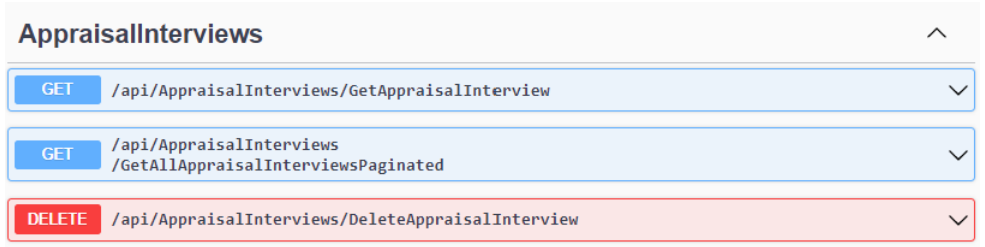


Abbildung 5.9: Admin-Portal Swagger-Dokumentation Mitarbeitergespräch

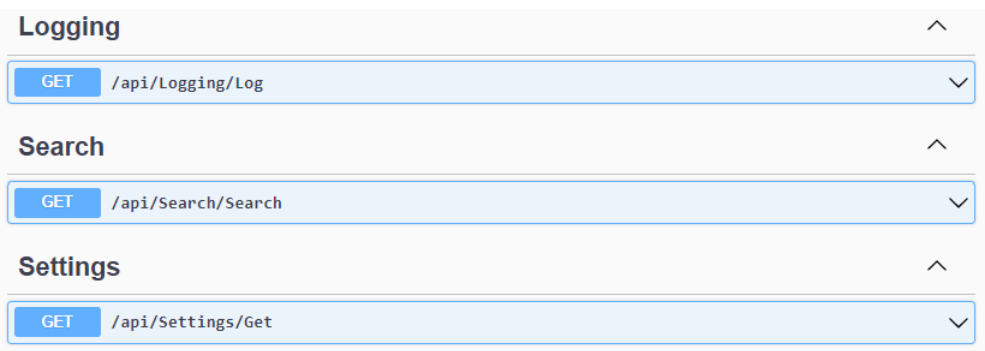


Abbildung 5.10: Admin-Portal Swagger-Dokumentation restliche Calls

Ausstehende Arbeiten

Für einige Funktionalitäten konnte auf bereits existierende Service-Methoden von 2getHR zurückgegriffen werden. Jedoch sind nicht alle diese Funktionalitäten in der Entwicklungsumgebung des Projektes vollständig durchführbar. Alle Punkte wurden während der Übergabe an 2BIT entsprechend kommuniziert und nach erfolgreichem Merge in den Dev-Branch von 2getHR korrigiert und kontrolliert.

- Tenant löschen - Das Löschen des Tenants bringt einige zusätzliche Nebeneffekte mit sich. Wird ein Tenant gelöscht, müssen alle seine zugehörigen Daten auf dem Azure Blob Storage gelöscht werden. Ein Beispiel ist der Avatar. Die Verbindung zum Azure Blob Storage wurde für dieses Projekt nicht konfiguriert und eingerichtet, da kein Bedarf ausserhalb dieser Funktionalität bestand. Solange der Code für das Löschen von Tenants auf dem Development-Branch des Admin-Portals geführt wird, ist

das Löschen des Tenants im Backend auskommentiert. Dies um eine möglichst getreue Abhandlung des Requests im Frontend zu simulieren und das Verhalten entsprechend ausbauen zu können.

- E-Mail-Adresse von Mitarbeitern ändern - Wird eine E-Mail-Adresse nicht von der zugehörigen Person sondern von einer Drittperson geändert, wird per Default eine E-Mail-Notifikation an den betroffenen Mitarbeiter versendet. Das Abfangen und Simulieren vom E-Mail-Versand wurde in der Entwicklungsumgebung des Projektes nicht konfiguriert, da ausserhalb dieser Funktion kein Bedürfnis dazu bestand. Hier wurde das Abfangen des API-Errors vom Backend im Frontend auskommentiert und mit einem entsprechenden TODO für die Integration in die 2getHR Umgebung versehen.
- Logging - Die Logs der Useraktionen werden im Backend momentan auf den Standard-Output-Stream geschrieben. Eine Implementation der Persistierung der Logs auf Azure Blob Storage ist geplant sobald das neue Admin-Portal in die Entwicklungsumgebung vom 2getHR migriert wurde, wo die Storage Connection konfiguriert ist.

Kapitel 6

Qualitätsmassnahmen

Coding Guidelines

Die Coding Guidelines für React wurden für das Frontend befolgt. Für das Backend wurden die Coding Guidelines von 2getHR übernommen, um einen möglichst uniformen Code zu garantieren.

Object Deconstruction

In React ist es üblich Object Deconstruction zu nutzen. Es bedeutet, dass Props im Aufruf der Klasse nicht als Sammelelement übergeben werden, sondern aufgesplittet werden. Dies wurde bewusst nicht eingesetzt, da oft grosse Props Objekte implementiert wurden. Dies, um eine grössere Generalisierung der Komponenten und somit eine bessere Wartbarkeit zu erreichen. Das aufsplitten der Props würde zu einer unhandlich grossen Liste in den Komponenten Argumenten führen, ohne einen grösseren Mehrwert im Code.

Clean Code

Für die Erstellung von Code halten sich die Developer an die gängigen Richtlinien von Clean Code [Mar09]. Die Zielsetzung ist eine Code-Basis, welche sich als Dokumentation präsentiert.

Folglich eine Auflistung der wichtigsten zu beachtenden Kriterien für Clean Code:

- Die Strukturierung des Codes soll einheitlich sein.
- Funktionen und Klassen sollen so gross wie nötig und klein wie möglich sein.
- Test-, Variabel- und Funktionsnamen müssen aussagekräftig sein.
- Kommentare sollen nur in kritischen Fällen vermerkt werden.
- Keine Codesmells.

6.1 Versionskontrolle

Struktur Repository

Zur Verwaltung der Produkt- und Projekt-Dokumentation wird die Infrastruktur der OST (konkret gitlab.ost.ch) verwendet. Die Dokumentation wird als Mono-Repo geführt. Der Source-Code von den Applikationen 2getHR und Admin-Portal wird in der Infrastruktur von 2BIT verwaltet. 2BIT setzt Azure DevOps ein und die Codebasis wird in einem Mono-Repo verwaltet. Die Struktur des Source-Codes im Repo ist wie folgt aufgebaut:

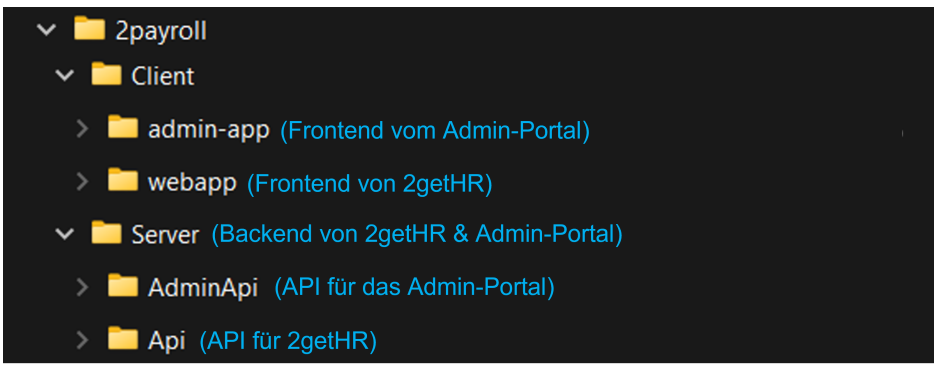


Abbildung 6.1: Struktur Source-Code-Repo

Hinweis: 2payroll ist der alte Name von der Applikation 2getHR und verbleibt als Name der Root vom Repo verwendet.

CI/CD

CI/CD-Pipeline der Dokumentation

Die CI/CD-Pipeline der Dokumentation auf GitLab besteht aus zwei Stages: build und diff.
In der build-Stage wird das PDF der Projekt- und Produkt-Dokumentation erstellt. Dabei wird auch das Glossar erstellt und in das PDF integriert. In der diff-Stage wird ein PDF-Dokument erstellt, das Unterschiede zwischen der aktuellen Version und der Version, die mit dem Tag 'latexdiff-base' in git markiert wurde, aufzeigt. Der Use-Case dieser Stage ist das Review einer überarbeiteten Dokumentation. Wenn die Unterschiede zwischen zwei Versionen hervorgehoben sind, kann sich der Reviewer auf geänderte Abschnitte konzentrieren.

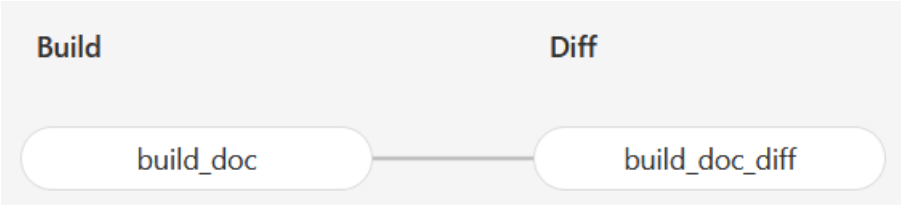


Abbildung 6.2: CI/CD-Pipeline der Dokumentation auf GitLab

CI/CD-Pipelines des Source-Codes

Die CI/CD-Logik für den Source-Code auf AzureDevOps ist auf verschiedene Pipelines verteilt. So erhalten das Admin Portal Frontend und das Backend je eine eigene Pipeline für Integration, die unabhängig angestossen werden kann. Für das Deployment gibt es zusätzlich eine Pipeline. Es existieren die folgenden vier Pipelines:

1. AP - AdminPortal
    - Diese Pipeline installiert die notwendigen Dependencies für das Admin-Portal. Danach wird das Admin-Portal gebuildet, getestet und es werden Code-Metrik-Tools durchgelaufen. Diese Schritte müssen alle erfolgreich durchlaugen werden damit die Pipeline erfolgreich abschliesst.
    - Diese Pipeline wird bei jedem Commit auf einen Feature-Branch getriggert.
2. AP - Backend
    - Diese Pipeline installiert die notwendigen Dependencies für das Backend und danach wird dieses gebuildet und getestet. Auch hier müssen alle Schritte erfolgreich durchlaufen.
    - Diese Pipeline wird bei jedem Commit auf einen Feature-Branch getriggert.
3. AP - Misc
    - Diese Pipeline beinhaltet die gleichen Schritte wie AdminPortal und Backend Pipelines und durchläuft Dependency Management, Building und Testing zusätzlich für das 2getHR Frontend. Somit werden alle Projekte noch einmal gründlich getestet bevor ein Deployment stattfindet.
    - Nach erfolgreichem Durchlauf der Builds und Tests werden alle Build-Artefakte der Teil-Projekte als Vorbereitung für ein Deployment in den (Web-)Server-Ordner der Pipeline Umgebung kopiert.
    - Diese Artefakte des Admin-Portals, des 2getHR-Frontend und des Backends werden danach für die Deployment pipeline ('AP - adminportal') publiziert.

- Die Pipeline wird nur bei einem Commit auf den Master-Branch dieses Projekts (AP/dev) getriggert.
- Diese Pipeline wurde von 2BIT übernommen um eine Integration des Projektes in das 2getHR System zu erleichtern.

4. dev - adminportal

   - Diese Pipeline führt das Deployment der von 'AP - Misc' bereitgestellten Artefakte durch. Diese Artefakte werden an die folgenden Orte deployt:
     - Admin-Portal: `https://a-2gethr-admin.azurewebsites.net/`
     - Admin-API: `https://a-2gethr-api.azurewebsites.net/`
     - 2getHR-Frontend: `https://a-2gethr.azurewebsites.net`
     - Gibt es Datenbank Migrationen wird hier der Schritt ef.migrations durchgeführt. Diese Migrationen werden auf dem DB-Server: `dev-2BIT.database.windows.net` in der DB: `test.a-2getHR` durchgeführt.
   - Diese Pipeline wird nur bei einem Commit auf den Master-Branch dieses Projekts (AP/dev) getriggert.
   - Zudem wird diese Pipeline nur getriggert, wenn die Pipeline 'AP - Misc' erfolgreich durchgelaufen ist.

Da in diesem Projekt das Admin-Portal und bei Bedarf das Backend (Admin-Api) angepasst werden, sollen diese beiden Pipelines auch bei Commits auf Feature-Branches getriggert werden. Das 2getHR-Frontend wird im Projektverlauf nicht verändert. Somit ist die Continuous Integration dieses Projektes nur notwendig, falls ein Deployment erstellt werden soll. Per Definition wird ein Deployment nur getriggert, wenn ein Commit auf den Master-Branch dieses Projekts (AP/dev) durchgeführt wird.

Workflow für die Arbeit am Source-Code

Jeder Task, an dem gearbeitet wird, ist als Issue auf Azure DevOps vorhanden. Issues sind auf Branches gemappt, wobei kleine Issues in einem Branch zusammengeführt werden können. Immer wenn ein Issue abgeschlossen ist, wird der relevante Feature-Branch mit dem Master-Branch zusammengeführt. Dazu wird ein Pull-Request auf den Master-Branch des Projekts (AP/dev) erstellt. Im Zuge dessen wird ein Reviewer hinzugezogen. Der Reviewer muss die Änderungen auf dem Feature-Branch nachvollziehen und verifizieren bzw. falsifizieren und kontrollieren, ob die DoD (siehe Kapitel 6.3) eingehalten wurde. Bevor ein Entwickler einen Review eines Pull-Requests anfordert, wird sichergestellt, dass der Feature-Branch die neuesten Änderungen aus dem Master-Branch enthält, um die Wahrscheinlichkeit eines Merge-Konflikts zu vermindern. Wenn dies der Fall ist, darf eine Review für den Pull-Request angefragt werden. Wenn das Review erfolgreich ist und das DoD eingehalten wurde, bestätigt der Reviewer den Pull-Request und der Feature-Branch wird mit dem Master-Branch des Projekts zusammengeführt. Der Merge mit dem Master-Branch des Projekts löst ein neues Deployment aus. Der Feature-Branch wird gelöscht, sobald der Merge mit dem Master-Branch durchgeführt wurde.

Der Workflow für die Arbeit am Source-Code ist in Abbildung 6.1 abgebildet:

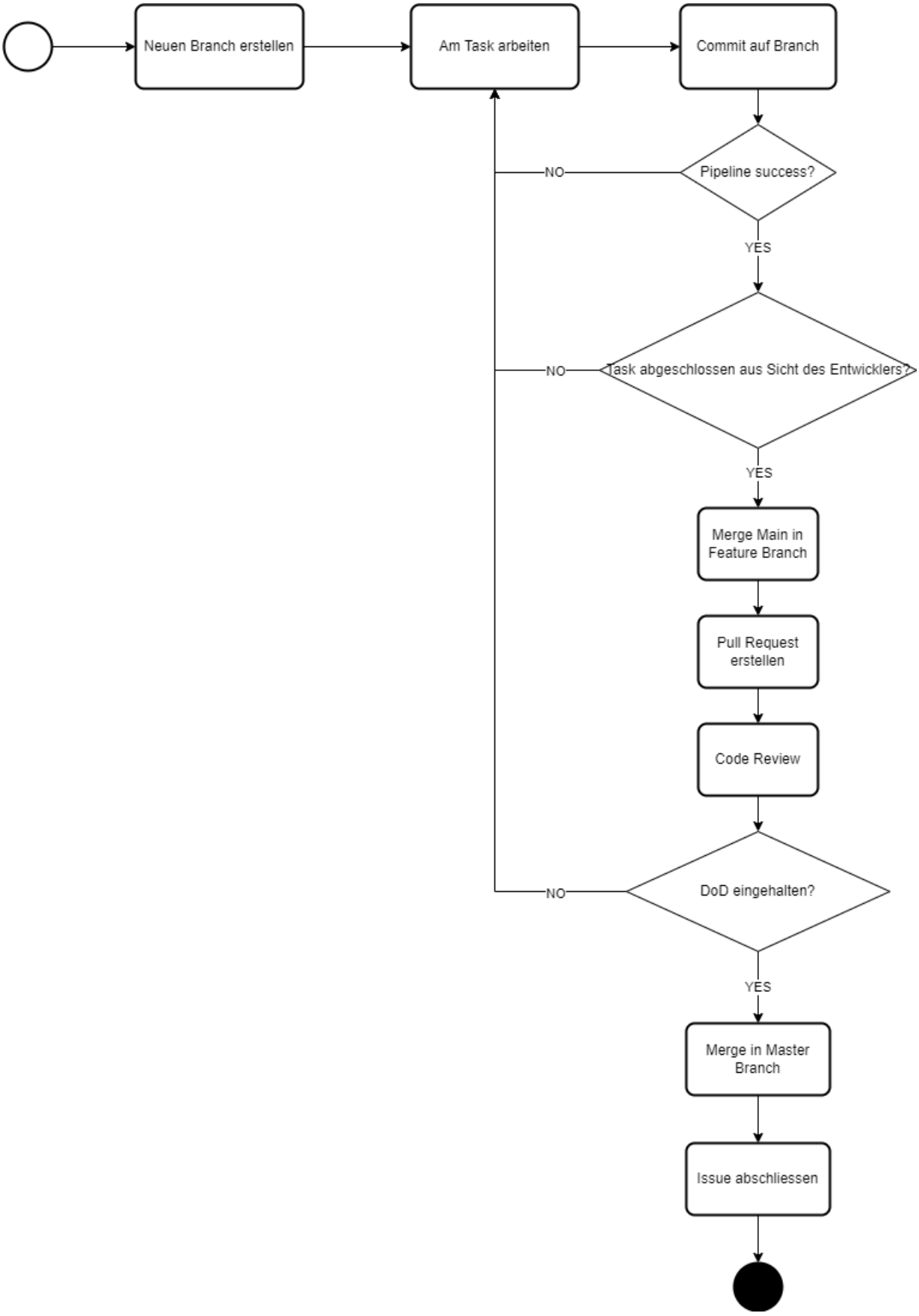


Abbildung 6.3: Workflow für die Arbeit am Source-Code

6.2 Code-Reviews

Für Tasks, in denen Code geschrieben wurde, muss eine Code Review durchgeführt werden. Erst nach abgeschossener Review kann ein Task als abgeschlossen markiert werden (siehe auch Kapitel 6.3). Reviews finden typischerweise zu zweit statt. Code-Abschnitte, die vom Projektteam als systemkritisch betrachtet werden, müssen zu dritt gereviewt werden. Code-Reviews werden nicht explizit protokolliert, stattdessen wird im Task festgehalten, dass ein Review stattgefunden hat. Code-Anpassungen, die aus dem Code-Review hervorgehen, sind implizit in den Commits der Versionskontrolle (siehe auch Kapitel 7.10) dokumentiert.

6.3 Definition of Done

Die Definition of Done definiert welche Kriterien erfüllt sein müssen, damit ein Task oder eine User Story als abgeschlossen gilt. Die Definition of Done muss in jedem Review eines Tasks durchgearbeitet werden. Ein Task muss alle Punkte der Liste erfüllen, damit der Code in den Master-Branch (AP/dev) gemergt wird.

- Der Code kompiliert ohne Fehler.
- Falls Warnungen vorhanden sind, müssen diese während dem Review begründet akzeptiert werden. Eine allfällige Begründung wird im Issue dokumentiert.
- Die CI/CD-Pipeline läuft ohne Fehler durch.
- Alle Tests laufen erfolgreich durch.
- Es sind genügend Tests vorhanden, so dass die in Kapitel 6.4 definierte Test-Coverage erreicht wird.
- Die in Kapitel 6 definierten Coding Guidelines und Standards sind erfüllt.
- Der Task (inkl. Code) wurde von mindestens einem Teampartner gereviewed.
- Eine User Story gilt als Done, wenn alle daraus abgeleiteten Tasks erledigt sind.

6.4 Metriken und Code Analysis

Als Metrik wird Code Coverage eingesetzt und analysiert. Zusätzlich wird im Backend die Cyclomatic Complexity konsultiert.

Frontend

Die Tests für das Frontend wurden in Cypress [Cyp] geschrieben. Die Cypress-Tests mocken das gesamte Backend sowie die Integration des Azure Active Directory. Mit den Cypress-Tests wurden die gängigsten Use Cases abgedeckt, um sicherzustellen dass die Applikation weiterhin funktioniert, bevor ein Feature Branch gemergt wird.

Das Frontend hat 1'927 Zeilen ausführbarer Code und 40 implementierte Funktions-Komponenten. Die folgende Abbildung zeigt die Code-Coverage der Cypress-Tests.

Abbildung 6.4: Frontend Test-Coverage

Eine genaue Auflistung der Codelines im Frontend findet sich in der folgenden Abbildung.

language	files	code	comment	blank	total
JSON	27	71,115	0	26	71,141
HTML	274	57,240	23	3,974	61,237
XML	1	4,526	0	1	4,527
TypeScript	51	3,829	69	532	4,430
TypeScript React	42	3,175	15	306	3,496
JavaScript	5	252	25	31	308
CSS	2	204	9	14	227
Markdown	1	71	0	31	102
SCSS	3	57	5	16	78
JSON with Comments	3	42	2	2	46
Properties	1	2	0	0	2

Abbildung 6.5: Frontend Test-Coverage

Backend

Testing im Backend wurde an die bestehende Testinfrastruktur vom 2getHR Backend angepasst. Die Businesslogik ist zu 90% zu testen gemäss NFR13 reftab:NFR13. Es wurden Unittests geschrieben, da test-relevante Logik nur in den Service Klassen isoliert aufgerufen wird. Ein Report über die Tests kann im Anhang (im Dokument coverage_api.html) eingesehen werden. Die Businesslogik konnte erfolgreich zu 96% getestet werden. Die Coverage wurde mit coverlet ausgelesen und mit pycobertura formatiert. Folgend findet sich ein Ausschnitt des Reports über die kritischen Services.

Filename	Stmts	Miss	Cover	Missing	Has Business Logic	Anz Klassen
dev\AdminPortal\Code\2payroll\Server\AdminApi\Services\AppraisalInterviewAdminService.cs	87	4	95.40%	125-126, 137-138	yes	
dev\AdminPortal\Code\2payroll\Server\AdminApi\Services\EmployeeAdminService.cs	112	20	82.14%	160-161, 187-188, 60-68, 42-57	yes	
dev\AdminPortal\Code\2payroll\Server\AdminApi\Services\PaginationAdminService.cs	3	0	100.00%		yes	
dev\AdminPortal\Code\2payroll\Server\AdminApi\Services\TenantAdminService.cs	138	18	86.96%	203-204, 224-228, 233-234, 61-70, 79-83	yes	
...	...	...	...	...	...	
TOTAL Businesslogik Tests	426	42	96.77%			40

Abbildung 6.6: API-Services Test-Coverage

Ein genauer Report der Cyclomatic Complexity und weiterer gängiger Metriken kann im Anhang im Excel 'metrics-backend' eingesehen werden. Besonderes Augenmerk muss hier auf die Controllers und Services gelegt werden, da dort die Businesslogik implementiert ist. Die Sortierungsmethoden wiegen in den drei Services Tenant (26), Employee (24) und AppraisalInterview (18) mit einer grossen Cyclomatic Complexity auf. Dies da ein Switch-Case implementiert werden musste, um die Sortierung der Tabellen zu lösen. Der dem Aufruf der Methode mitgegebene Sortierungsstring beinhaltet den Namen des zu sortierenden Feldes aus dem Frontend. Diese muss mit allen möglichen Sortierfeldern verglichen werden, um die richtige Sortierung der Query auf die Datenbank hinzuzufügen. Eine Generalisierung dieses Vorgangs wurde ausprobiert, jedoch gab es keine Lösung, die eine bessere Wartbarkeit des Codes zur Folge hatte und den Switch-Case vollständig ausbauen konnte. Als solches wurde die hohe Komplexität hier als unumgehbar eingestuft. Alle restlichen Methoden weisen einen akzeptablen Komplexitätsgrad von ≤ 10 auf.
Alle implementierten Klassen konnten einen Maintainability Index im grünen Bereich erreichen.

Code Metrics Results

Filter: Min: Max:

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling
AdminApi (Debug)	86	434	2	192
AdminApi	80	18	2	98
AdminApi.Components	80	6	1	15
AdminApi.Controllers	79	25	2	40
AppraisalInterviewsController	82	4	2	13
EmployeesController	81	8	2	19
LoggingController	83	1	2	6
SearchController	70	2	2	12
SettingsController	77	2	2	8
TenantsController	82	8	2	20
AdminApi.DTOs	98	185	1	23
AdminApi.Models	100	48	1	3
AdminApi.Profiles	77	10	2	41
AdminApi.Services	82	142	1	51
AppraisalInterviewAdminService	60	37	1	33
EmployeeAdminService	61	42	1	38
IAppraisalInterviewAdminService	100	3	0	7
IEmployeeAdminService	100	5	0	8
ITenantAdminService	100	7	0	11
PaginationAdminService<T>	90	1	1	2
TenantAdminService	63	47	1	40

Abbildung 6.7: Backend Metriken

Lighthouse

Das Admin-Portal wurde mit Hilfe des Lighthouse Tools auf Best Practices, Search Engine Optimization, Bedienbarkeit und die Leistung geprüft. Es wurden alle Tests erfüllt. Im Anhang können ausführliche PDF Reports für die Listenansicht (Lighthouse_Report_Listview) sowie für die Detailansicht (Lighthouse_Report_Detailview) eingesehen werden.

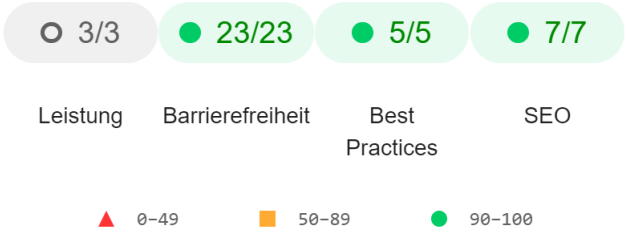


Abbildung 6.8: Lighthouse-Bericht auf Tenant-Listenansicht

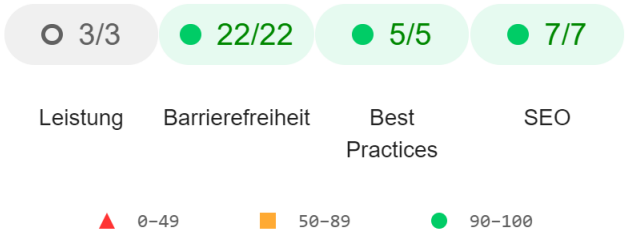


Abbildung 6.9: Lighthouse-Bericht auf Tenant-Detailansicht

6.5 Test Konzept

Tests sind ein Mittel um sicherzustellen, dass die geforderten Qualitätsziele und Anforderungen einhalten werden. Das Testkonzept definiert wie die funktionalen und nicht-funktionalen Anforderungen messbar geprüft werden. Dies soll sicherstellen, dass die definierten Anforderungen erfüllt sind, indem diese aus verschiedenen Blickwinkeln getestet werden. Folgend sind die übergreiffenden Kriterien, die dabei besonders beachtet werden.

- Functional Suitability
- Performance Efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability

Unit-Tests

Unit Tests werden eingesetzt, um die funktionale Korrektheit zu verifizieren. Sie werden manuell während der Entwicklung sowie automatisiert in der Pipeline für Business-Lgoik durchgeführt und kontrolliert.

Integration-Tests

Um das korrekte Zusammenspiel von Komponenten zu garantieren, werden im Frontend sowie Backend Integrations-Tests durchgeführt.

End2End-Tests

End to End Tests werden durchgeführt, um das Zusammenspiel zwischen Front- und Backend zu überprüfen und User-Interaktionen zu simulieren.

Lasttests

Lasttests werden eingesetzt, um die Performance der Applikation zu überprüfen und allfällige Probleme zu identifizieren. Die Lasttests werden in der Adressierung der NFR des Final Releases durchgeführt. Sie sind nur eingeschränkt umsetzbar, da das Projekt in einer Entwicklungsumgebung durchgeführt wird, in welcher die Performance eingeschränkt ist. Das Resultat kann in der Tabelle 3.45 eingesehen werden.

Systemtests

Systemtests werden vor dem Alpha-Release sowie vor dem Beta-Release manuell durchgeführt. Es soll verifiziert werden, dass das gesamte System zu diesem Zeitpunkt korrekt funktioniert. Folgend sind die durchzuführenden Systemtests dokumentiert 6.1.
Die durchgeführten Tests können im Anhang 11.1 eingesehen werden.

Systemtests

Nr.	Link	Vorgehen	Erwartetes Resultat
1	UC01	1. Loginversuch mit einem existierenden User und anschliessendes Ausloggen 2. Loginversuch mit einem nicht-existierenden User 3. Ein nicht-autorisierter User greift über eine spezifische URL (z.B. Liste aller Mitarbeiter) auf das Admin-Portal zu.	1. Login funktioniert erfolgreich. Der User wird eingeloggt und auf die Startseite weitergeleitet. Nach dem Logout wird der User wieder auf die Login Page verwiesen. 2. Der User wird nicht eingeloggt und es wird eine generische Fehlermeldung angezeigt. 3. Der User wird auf die Login-Seite verwiesen ohne sensible Daten oder Funktionalitäten zu sehen.

2	UC02	1. Vorbedingung: Der User ist eingeloggt. 2. Der User wechselt zwischen Listenlistenansichten (Tenants, Personen und Treuhänder). 3. Der User navigiert durch die verschiedenen Seiten der Listenansicht (Paginierung). 4. Der User setzt Filter und sortiert nach einer Spalte in jeder Litenansicht. 5. Der User wechselt die Listenansicht. 6. Der User konfiguriert eine Listenansicht. 7. Der User führt einen Refresh durch. 8. Der User loggt sich aus und wieder ein.	1. Der User befindet sich auf der Startseite (Listenansicht der Tenants) 2. Die Listen werden korrekt dargestellt. 3. Die Liste enthält nicht mehr als die konfigurierte Anzahl Einträge. Die Navigationselemente der Paginierung werden angezeigt. Wenn der User eine Seite weiterblättert, werden die nächsten Einträge angezeigt. 4. Filter und Sortierung werden angewendet und zwischengespeichert. 5. Das Umschalten zwischen den Listen funktioniert und die Listen werden korrekt gerendert. Dabei bleiben die Filter und die Sortierung pro Listenansicht erhalten. Session-Filter werden automatisch auf den entsprechenden Litenansichten wiederverwendet. 6. Die Listenkonfiguration wird angewendet und persistiert. 7. Die sichtbaren Spalten und Spaltenreihenfolge sind entsprechend der Einstellungen des Users wieder vorhanden. 8. User wird auf die Login-Page verwiesen und nach erfolgreichem Login sieht er wieder die Tenantliste mit entsprechender Listenkonfiguration.
3	UC03	1. Vorbedingung: Es wurden Delete oder Updates durchgeführt und sind in der Applikation möglich. 2. Userlogin wird durchgeführt. 3. User navigiert zur Detailansicht eines Testtenant und führt eine Update-Aktion aus. 4. Der User wechselt zu Azure und sieht die Log-Files ein.	1. User ist ausgeloggt. 2. Listenansicht (Home) wird dargestellt. 3. User-Aktion funktioniert. 4. Log-Files enthalten alle relevanten Einträge mit den definierten Daten.

4	UC04	1. Vorbedingung: User ist eingeloggt. 2. User navigiert von Listenansicht Tenant zu Listenansicht Mitarbeiter 3. User navigiert von Listenansicht Mitarbeiter zu Listenansicht Treuhänder 4. User betätigt Back-Button. 5. User betätigt Back-Button.	1. Tenant-Listenansicht wird dargestellt. 2. Mitarbeiter-Listenansicht wird dargestellt. 3. Treuhänder-Listenansicht wird dargestellt. 4. Mitarbeiter-Listenansicht wird dargestellt. 5. Tenant-Listenansicht wird dargestellt.
5	UC05	1. Vorbedingung: User ist eingeloggt. 2. User navigiert zur Tenant-Detailansicht. 3. User ändert Trustee Verification State von Tenant. 4. User löscht Tenant.	1. Tenant-Listenansicht wird dargestellt. 2. Tenant-Detailansicht wird dargestellt und zeigt Tenant spezifische Detailangaben wie in UC beschrieben. 3.8 3. Tenant-Information wird gemäss Veränderung angepasst und Tenant-Detail wird neu geladen. 4. Warnmeldung wird angezeigt. Nachdem 'LÖSCHEN' eingegeben wurde, wird Löschaktion ausgelöscht. Ist Löschen erfolgreich wird eine entsprechende Meldung angezeigt.
6	UC06	1. Vorbedingung: User ist eingeloggt und auf Mitarbeiter-Listenansicht. 2. User navigiert zur Mitarbeiter-Detailansicht. 3. User ändert Rolle von Mitarbeiter. 4. User ändert E-Mail Adresse von Mitarbeiter.	1. Mitarbeiter-Listenansicht wird dargestellt. 2. Mitarbeiter-Detailansicht wird dargestellt und zeigt Mitarbeiter spezifische Detailangaben wie in UC beschrieben. 3.9 3. Mitarbeiter-Information wird gemäss Veränderung angepasst und Mitarbeiter-Detail wird neu geladen. 4. Mitarbeiter-Information wird gemäss Veränderung angepasst und Mitarbeiter-Detail wird neu geladen.

7	UC07	1. Vorbedingung: User ist eingeloggt und auf Tenant-Listenansicht. 2. User navigiert zu Tenant-Detailansicht. 3. User navigiert zu Mitarbeiter des Tenants. 4. User navigiert zu Mitarbeiter-Detailansicht eines Mitarbeiters. 5. User navigiert zu Mitarbeitergespräche des Mitarbeiters. 6. User navigiert zu Mitarbeitergespräch-Detailansicht. 7. User navigiert zu Tenant-Detailansicht des Mitarbeitergesprächs. 8. User navigiert zu Mitarbeitergespräch des Tenants. 9. User navigiert zu Mitarbeitergespräch-Detailansicht. 10. User navigiert zu Mitarbeiter-Detailansicht. 11. User navigiert zu Tenant-Detailansicht des Mitarbeiters. 12. User navigiert zu Treuhänderliste des Tenants. 13. User navigiert zu Tenant-Detailansicht eines der Treuhänder.	1. Tenant-Listenansicht wird dargestellt. 2. Tenant-Detailansicht wird korrekt angezeigt. 3. Mitarbeiterliste wird korrekt angezeigt. 4. Mitarbeiter-Detailansicht wird korrekt angezeigt. 5. Mitarbeitergesprächsliste wird korrekt angezeigt. 6. Mitarbeitergespräch-Detailansicht wird korrekt angezeigt. 7. Tenant-Detailansicht wird korrekt angezeigt. 8. Mitarbeitergesprächsliste wird korrekt angezeigt. 9. Mitarbeitergespräch-Detail wird korrekt angezeigt. 10. Mitarbeiterliste wird korrekt angezeigt. 11. Tenant-Detailansicht wird korrekt angezeigt. 12. Treuhänderliste wird korrekt angezeigt. 13. Tenant-Detailansicht wird korrekt angezeigt.
8	UC08	1. Vorbedingung: User ist eingeloggt und auf Mitarbeitergespräch-Listenansicht. 2. User navigiert zur Mitarbeitergespräch-Detailansicht. 3. User löscht Mitarbeitergespräch.	1. Mitarbeitergespräch-Listenansicht wird dargestellt. 2. Mitarbeitergespräch-Detailansicht wird dargestellt und zeigt Mitarbeitergespräch spezifische Detailangaben wie in UC beschrieben. 3.11 3. Warnmeldung wird angezeigt. Nachdem 'LÖSCHEN' eingegeben wurde, wird Löschaktion ausgelöscht. Ist Löschen erfolgreich wird eine entsprechende Meldung angezeigt.

10	UC10	1. Vorbedingung: User ist eingeloggt und auf Mitarbeiter-Detailansicht. 2. User klickt auf Löschicon. 3. User gibt 'Löschen' in Textfeld ein und versucht Mitarbeiter zu löschen. 4. User löscht Mitarbeiter.	1. Mitarbeiter-Detailansicht wird dargestellt. 2. Hat Mitarbeiter Referenzen werden diese angezeigt und der restliche Löschdialog ausgeblendet. 3. Hat Mitarbeiter keine Referenzen wird dies angezeigt und der restliche Löschdialog normal angezeigt. 4. Mitarbeiter kann nicht gelöscht werden. 5. Mitarbeiter wird gelöscht und entsprechender Dialog angezeigt.
10	UC11	1. Vorbedingung: User ist eingeloggt und auf Tenant-Listenansicht. 2. User klickt auf Suchfeld. 3. User fängt an zu tippen und pausiert 1 Sekunde lang. 4. User öffnet Suchresultat und navigiert zu Mitarbeiter-Detailansicht. 5. User gibt erneut Input ins Suchfeld und navigiert zu anderer Mitarbeiter-Detailansicht. 6. User gibt erneut Input ins Suchfeld und navigiert zu Tenant-Detailansicht. 7. User gibt erneut Input ins Suchfeld und navigiert zu Mitarbeitergespräch-Detailansicht. 8. User gibt erneut Input ins Suchfeld und wartet auf Suchresultat. User klickt auf Reset Button. 9. User gibt erneut Input ins Suchfeld und klickt auf Suchen-Button. 10. User klickt neben Suchresultat Dropdown.	1. Tenant-Listenansicht wird angezeigt. 2. Suche wird automatisch ausgelöst und zeigt Suchresultat an. 3. Mitarbeiter-Detailansicht wird korrekt und mit richtigem Mitarbeiter angezeigt. 4. Mitarbeiter-Detailansicht wird korrekt und mit richtigem Mitarbeiter angezeigt. 5. Tenant-Detailansicht wird korrekt und mit richtigem Tenant angezeigt. 6. Mitarbeitergespräch-Detailansicht wird korrekt und mit richtigem Mitarbeitergespräch angezeigt. 7. Suchfeld Input wird gelöscht und Suchresultat wird nicht mehr angezeigt. 8. Suche wird durch klick ausgelöst bevor automatische Suche ausgelöst wird. 9. Suchresultat wird nicht mehr angezeigt, Suchfeld Input bleibt erhalten.

Tabelle 6.1: System Tests

Usability-Tests

Es werden Usability Tests durchgeführt, um die Intuitivität der Applikation zu messen, direktes Feedback vom Enduser zu erhalten und allfällige Probleme zu identifizieren. Endnutzer im Kontext des Adminportals sind die supportleistenden Personen bei 2BIT für die 2getHR-Applikation (siehe Kapitel Benutzer 2.5). Die Testfälle für die Usability-Tests werden ausgearbeitet und in der (Tabelle 6.3) dokumentiert. Nachdem ein Endnutzer die Usability-Tests durchgeführt hat, werden die Resultate in der Tabelle 11.3 festgehalen und aus dem Resultat des Usability-Tests abgeleitete Massnahmen definiert. Die Testfälle sind aufgrund Szenarien aus dem produktiven Betrieb (tatsächliche Supportfälle) aufgebaut.
Für die Alpha Usability-Tests wurden das Testkonzept überarbeitet. Das Protokoll zu den Wireframe Tests kann im Anhang eingesehen werden.
Ausgangslage für alle Usability-Tests ist ein existierender User, sowie die Anzeige des Loginfensters des Admin-Portals. Der Usability Test wird je einmal mit Chrome und Microsoft Edge durchgeführt, um alle in 2BIT gängigen Browser abzudecken. Ebenfalls werden die Szenarien auch auf der alten Version des Admin Portals durchgeführt. So kann die Verbesserung der Nutzbarkeit getestet und allfällige Korrekturen eingeleitet werden.
Folgend sind die Usability Tests für Wireframe und die überarbeiteten Tests für Alpha und Beta Release dokumentiert.
Die durchgeführten Tests können im Anhang 11.2 eingesehen werden.

Usabilitytest für Wireframe

Nr.	Aufgabe	Erwartetes Resultat
1	Sie haben eine Supportanfrage von einem Kunden per Mail erhalten. Ein Mitarbeiter des Kunden kann sich nicht einloggen. Das E-Mail enthält den Namen des Kunden, Manuel Weber, des Tenant, HSR, und des Mitarbeiters, Yael Schärer. Finden Sie heraus, was der Einladungsstatus des Mitarbeiters ist. Finden Sie heraus, ob Sie in SendGrid kontrollieren müssen, ob die E-Mail-Adresse blockiert ist oder nicht. (Abhängig vom Status der Einladung) Finden Sie heraus, wann die Einladung verschickt wurde.	1. Der User ist eingeloggt. 2. Der User findet den Mitarbeiter über die Mitarbeiter-Liste oder über das Suchfeld. 3. Der User findet das Datenfeld für den Einladungsstatus, äusgeschickt". 4. Der User navigiert zur Detailansicht der Person und findet das Datum der Einladung heraus, "12.15.2022". 5. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?

2	Sie haben eine Supportanfrage eines Kunden erhalten. Der Kunde hat Probleme mit der Konfiguration der Aktivitätsarten in 2getHR. Sie haben die E-Mail-Adresse und den Namen des Kunden erhalten, g@g.ch und Yael Schärer. Kontrollieren Sie das Datum der letzten Aktivität des Kunden.	1. Der User ist eingeloggt. 2. Der User findet die Person in der Mitarbeiter-Liste oder über das Suchfeld. 3. Der User findet die den Zeitpunkt der letzten Aktivität der Person, "01.10.2021". 4. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?
3	Sie haben eine Supportanfrage eines Kunden erhalten. Der Kunde klagt darüber, dass die Daten in seinem Bexio Account nicht übereinstimmen mit den 2getHR Daten. Sie erhalten den Namen des Mitarbeiters, Yael Schärer, und die E-Mail-Adresse, g@g.ch. Finden sie heraus was das Problem sein könnte.	1. Der User ist eingeloggt. 2. Der User findet die Person über das Suchfeld oder in der Mitarbeiter-Liste. 3. Der User findet heraus, das Bexio "nicht angebundenïst. 4. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?
4	Sie haben eine Supportanfrage von einem Kunden per Mail erhalten, können jedoch den Arbeitgeber (Tenant) nicht identifizieren. Das E-Mail enthält den Namen des Kunden (Yael Schärer) sowie seine E-Mail Adresse (yael.schaere@ost.ch). Finden sie heraus, zu welchem Tenant (Arbeitgeber) die Mitarbeiterin Yael Schärer gehört.	1. Der User ist eingeloggt. 2. Der User findet die Person via Suchfeld oder in der Mitarbeiterliste. 3. Der User öffnet das Mitarbeiter-Detail und klickt auf Tenants. 4. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?

5	Sie haben eine Supportanfrage eines Kunden erhalten. Der Kunde behauptet, dass niemand in seinem Unternehmen mehr Admin-Einstellungen tätigen kann. Finden Sie heraus wie viele Mitarbeiter der Tenant besitzt und wie viele davon die Rolle Admin im entsprechenden Tenant besitzen. Sie haben den Namen des Tenants, HSR, und den Namens eines aktuellen Admins des Tenants, Yael Schärer, erhalten. Wenn keine Person mehr die Admin Rolle besitzt, wünscht der Kunde den benannten Mitarbeiter wieder als Admin zu definieren.	1. Der User ist eingeloggt. 2. Der User sucht den Tenant via Suchfeld oder in der Tenant-Liste. 3. Der User öffnet das Tenant-Detail und findet heraus wie viele Mitarbeiter der Tenant registriert hat. 4. Der User navigiert zu den Mitarbeitenden des Tenants. 5. Der User findet heraus, welche Mitarbeiter zu diesem Zeitpunkt die Admin-Rolle besitzen. 6. Gibt es keine Mitarbeiter mit der Rolle Admin mehr, ändert der User die Rolle des benannten Mitarbeiters zu Admin. 7. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?
6	Sie erhalten eine Supportanfrage von einem Kunden. Der Kunde möchte sich als Treuhänder registrieren. Setzen sie den Treuhänderstatus des Tenants auf verifiziert. Sie haben den Namen des Tenants, HSR, des Tenants erhalten.	1. Der User ist eingeloggt. 2. Der User findet den Tenant über das Suchfeld oder in der Tenant-Liste. 3. Der User öffnet das Tenant-Detail. 4. Der User ändert den Treuhänderstatus des Tenants. 5. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?

7	Sie erhalten eine Supportanfrage von einem Tenant. Der Tenant hat in der Testphase die Funktionalität des Mitarbeitergesprächs ausprobiert. Der entstandene Fragebogen wird nun nicht mehr gebraucht. Der Kunde gibt Ihnen den Titel des Fragebogens, Herbst 2022 von Yael Schärer. Finden und löschen Sie den entsprechenden Fragebogen.	1. Der User ist eingeloggt. 2. Der User findet den Fragebogen über das Suchfeld oder in der Mitarbeitergesprächs-Liste. 3. Der User öffnet das Mitarbeitergespräch-Detail. 4. Der User löscht das Mitarbeitergespräch. 5. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?
8	Sie erhalten eine Supportanfrage von einem Kunden. Der Kunde hat 2getHR mit einem Test Tenant ausprobiert und will diesen nun löschen. Sie erhalten den Namen des Tenants, HSR.	1. Der User ist eingeloggt. 2. Der User findet den Tenant über das Suchfeld oder in der Tenant-Liste. 3. Der User öffnet das Tenant-Detail. 4. Der User löscht den Tenant. 5. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?
9	Sie erhalten eine Supportanfrage von einem Kunden. Der Kunde klagt darüber, dass ein Mitarbeiter, Yael Schärer, nicht gelöscht werden kann. Nachdem der Kunde über den Grund informiert wurde, wünscht er, dass der User den Mitarbeiter direkt löscht.	1. Der User ist eingeloggt. 2. Der User findet den Mitarbeiter über das Suchfeld oder in der Mitarbeiter-Liste. 3. Der User öffnet das Mitarbeiter-Detail. 4. Der User findet heraus, warum der Mitarbeiter nicht gelöscht werden konnte. 5. Falls möglich, löscht der User den Mitarbeiter. 6. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?

10	Sie erhalten eine Supportanfrage von einem Kunden. Der Kunde will für seinen Tenant eine Custom-Subscription bestellen. Sie erhalten den Namen des Tenants, die E-Mail und eine Liste der gewünschten Module sowie die gewünschte Anzahl User.	1. Wird definiert sobald implementiert.
11	Sie bemerken einen Trend in den Supportfällen, die Sie in der letzten Zeit erhalten haben. Sie mussten oft die Information XYZ für Mitarbeiter und ABC für Tenants kontrollieren. Personalisieren Sie die Listenansicht nach folgenden Vorgaben: 1. Genau drei Spalten werden in der Tenantliste angezeigt: X, Y, Z 2. Genau vier Spalten werden in der Mitarbeiteransicht angezeigt: W, X, Y, Z 3. Genau eine Spalte wird in der Treuhänderansicht angezeigt: X	1. Der User loggt sich ein, findet die Spaltenkonfiguration und kann die Konfiguration vornehmen.

Tabelle 6.2: Usability Test

Usabilitytest für Alpha-/Betarelease

Die Usabilitytests wurden stark überarbeitet nach dem Wireframe Test, Die Tests wurden stärker an die existierenden Supporttickets angelehnt und reduziert, um ein besseres Bild der Applikation zu erhalten.

Nr.	**Aufgabe**	**Erwartetes Resultat**
1	Sie haben ein Supportticket vom Tenant 'Mikhail company 5' erhalten, per E-Mail 'mikhail5@test.test'. Ein neuer Mitarbeiter seiner Firma, 'ext1 employee', kann sich nicht einloggen. Finden sie heraus was das Problem sein könnte und Beschreiben sie eine mögliche Lösung.	1. Der User ist eingeloggt. 2. Der User findet den Mitarbeiter über die Mitarbeiter-Liste oder über das Suchfeld. 3. Der User findet das Datenfeld für den Einladungsstatus, İnvitedünd verifiziert somit, dass die Einladung versendet wurde aber nicht akzeptiert. 4. Der User navigiert zur Detailansicht der Person und findet das Datum der Einladung heraus, "12.15.2022". 5. Wie sicher ist sich der User die richtigen Daten gefunden zu haben? 6. Mögliche Lösung: Einladung wurde verschickt und ist noch nicht abgelaufen: Akzeptieren.

2	Sie haben ein Supportticket vom Mitarbeiter 'luc fonjallaz' der Firma 'billing bing bing' über die Mailadresse 'fonjallaz97+billing@gmail.com' erhalten. Für den Mitarbeiter 'sneaky bastard' soll ein Mitarbeitergespräch durchgeführt werden. Der Kunde möchte selbst als Interviewer das Gespräch durchführen, kann aber nicht als Interviewer eingestellt werden. Finden sie heraus wo das Problem liegt und Beschreiben sie eine mögliche Lösung.	1. Der User ist eingeloggt. 2. Der User findet den Tenant in der Tenant-Liste oder über das Suchfeld. 3. Der User findet die zugehörigen Mitarbeiter des Tenants. 4. Der User findet heraus, dass der Mitarbeiter Luc Fonjallaz nicht als 'Ist Einheit Vorsteher' eingetragen ist und nicht Administrator ist und deshalb das Gespräch nicht erstellt werden kann. 5. Wie sicher ist sich der User die richtigen Daten gefunden zu haben? 6. Mögliche Lösung: Den Kunden als OrganizationalLeader registrieren.
3	Sie haben ein Supportticket vom Mitarbeiter 'Zeya Kermolin' mit der E-Mailadresse 'zeya@lagify.com' erhalten. Der Kunde möchte seinen Tenant als Treuhänder verifizieren lassen. Verifizieren sie den Tenant.	1. Der User ist eingeloggt. 2. Der User sieht den Tenantnamen 'Reto AG' in der Liste oder in der Detailansicht. 3. Der User verifiziert den Tenant. 4. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?
4	Sie haben ein Supportticket vom Tenant 'billing bing bing' erhalten. Ein Mitarbeiter des Kunden hat die falsche E-Mailadresse eingetragen. Ändern sie die E-Mailadresse vom Mitarbeiter 'testi tester' zu 'a@a.ch'. Dieser Mitarbeiter soll auch gleich Admin werden. Ändern sie die Rolle entsprechend.	1. Der User ist eingeloggt. 2. Der User sucht den Tenant via Suchfeld oder in der Tenant-Liste. 3. Der User findet den Mitarbeiter via Tenant Gelinkte Objekte oder Suchfeld. 4. Der User ändert die E-Mailadresse erfolgreich. 5. Der User ändert die Rolle erfolgreich.

5	Sie bemerken einen Trend in den Supportfällen, die Sie in der letzten Zeit erhalten haben. Sie mussten oft die Information 'Ist Einheit Vorsteher' und 'Einladungsstatus' für Mitarbeiter oft kontrollieren. Dafür haben sie die Felder 'Datum letzte Aktivität', 'Ist Aktiv' und 'Objekt ID' nie gebraucht. Konfigurieren sie die Tabelle Mitarbeiter entsprechend gemäss folgender Vorgabe: 1. Name 2. E-Mail 3. Einladungsstatus 4. Ist Einheit Vorsteher 5. Rolle	1. Der User loggt sich ein, findet die Spaltenkonfiguration und kann die Konfiguration vornehmen.
6	Sie haben ein Supportticket vom Tenant 'Home' per Emailadresse 'ibzakharov@yahoo.com' erhalten. Der Kunde hat alle Funktionalitäten von 2getHR ausprobiert und möchte nun neu anfangen. Der Kunde bittet sie darum alle Objekte zu löschen. Löschen sie zuerst Mitarbeitergespräche, dann Mitarbeiter und zuletzt Tenant.	1. Der User ist eingeloggt. 2. Der User findet den Fragebogen über das Suchfeld oder in der Liste und löscht den Fragebogen im Gespräch-Detail. 3. Der User findet die vier Mitarbeiter über das Suchfeld oder in der Liste und löscht sie in den Mitarbeiter-Details. (Die ersten zwei reichen aus) 4. User verifiziert warum Mitarbeiter nicht gelöscht werden kann. 5. Der User findet den Tenant über das Suchfeld oder in der Liste und löscht ihn im Tenant-Detail. 6. Wie sicher ist sich der User die richtigen Daten gefunden zu haben?

7	Ausprobieren: • Wieviele Mitarbeiter und Arbeitsstunden hat Tenant 'TEST@TEST.TEST' • Navigieren sie zum Mitarbeiter '123 123' des Tenants • Zu welcher Abteilung gehört dieser Mitarbeiter und ist er Bexio User • Navigieren sie zum Gespräch des Mitarbeiters • Was ist der Status des Gesprächs? Wann wurde dieser erreicht? • Navigieren sie zurück zum Tenant.	

Tabelle 6.3: Usability Test

6.6 Error-Handling

Sowohl im Frontend sowie auch im Backend ist ein Error-Handling eingebaut, um einen Absturz der Applikation zu verhindern. Das Backend fängt allfällige Fehler ab und meldet diese per HTTP-Status an das Frontend. Tritt im Frontend ein Fehler auf oder wird ein Fehler vom Backend empfangen, wird dieser abgefangen und verarbeitet, ohne dass das System abstürzt. In anderen Worten wird der Fehler so abgefangen, dass die Applikation weiter bedienbar ist. Für optimalen User-Komfort wird dem User eine Fehlermeldung angezeigt und mögliche Aktionen zur Verfügung gestellt.

Error Boundaries

React stellt sogenannte Error Boundaries zur Verfügung. Diese fangen Fehler ihrer Kind-Komponenten ab und ermöglichen ein alternatives UI zu rendern. Dies erlaubt ein einfaches Error Handling ohne grösseren Overhead. Fehler aus folgenden Quellen werden in den Error Boundaries jedoch nicht automatisch abgefangen und sind speziell zu beachten:

- Event-Handler
- Asynchroner Code
- Serverseitiges Rendering
- Fehler, die in der Fehlergrenze selbst und nicht in deren Kind-Komponenten auftreten

Um diese Defizite auszugleichen, wird in Eventhandlern und in den Fehlergrenzen mit Try-Catch-Blöcken gearbeitet, um Fehler abzufangen. Bei asynchronem Code (API-Client) werden die Rejected-Promises abgefangen und entsprechend behandelt. Serverseitiges Rendering wird nicht eingesetzt und ist somit nicht relevant für das Projekt.

Struktur Error Boundaries

Die Error Boundaries wurden innerhalb jeder Page, sowie um die App-Komponente platziert. Dies ermöglicht es Fehler einzelner Pages zu isolieren und ein Absturz der gesamten Applikation zu verhindern. Auf der Ebene einzelner Komponenten wurden keine Error Boundaries eingesetzt, da Komponenten auf der Page aggregiert werden. Fehler innerhalb der Unterkomponenten werden von der Page Error Boundary abgefangen und behandelt. Dies verhindert unnötig komplizierte Logik in den dummen Komponenten.

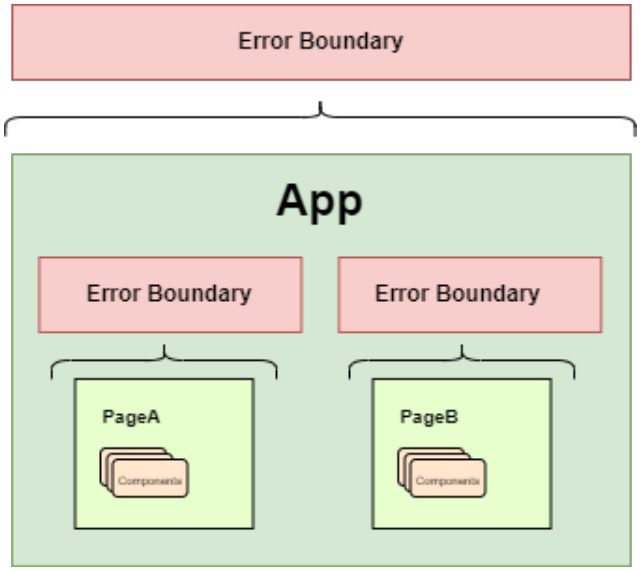


Abbildung 6.10: Aufbau Error Boundaries

Error Component

Um Nutzern Fehler darzustellen, wurde eine Error-Komponente erstellt. Diese meldet das Auftreten eines Fehlers und bietet Aktionsmöglichkeiten. Zum einen kann hier die Seite neu geladen werden, falls es ein transienter Fehler ist. Andererseits kann für technisch affine User eine genaue Fehlermeldung angezeigt werden.

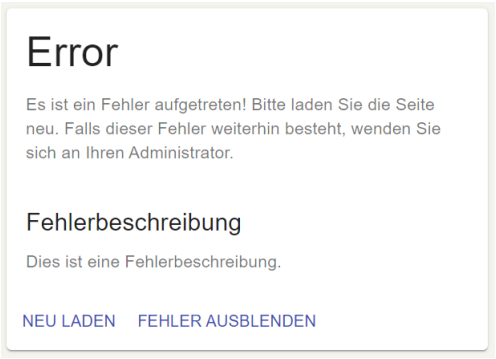


Abbildung 6.11: Beispiel - Error Component

6.7 Error-Handling

Sowohl im Frontend sowie auch im Backend ist ein Error-Handling eingebaut, um einen Absturz der Applikation zu verhindern. Das Backend fängt allfällige Fehler ab und meldet diese per HTTP-Status an das Frontend. Tritt im Frontend ein Fehler auf oder wird ein Fehler vom Backend empfangen, wird dieser abgefangen und verarbeitet, ohne dass das System abstürzt. In anderen Worten wird der Fehler so abgefangen, dass die Applikation weiter bedienbar ist. Für optimalen User-Komfort wird dem User eine Fehlermeldung angezeigt und mögliche Aktionen zur Verfügung gestellt.

Error Boundaries

React stellt sogenannte Error Boundaries zur Verfügung. Diese fangen Fehler ihrer Kind-Komponenten ab und ermöglichen ein alternatives UI zu rendern. Dies erlaubt ein einfaches Error Handling ohne grösseren Overhead. Fehler aus folgenden Quellen werden in den Error Boundaries jedoch nicht automatisch abgefangen und sind speziell zu beachten:

- Event-Handler
- Asynchroner Code
- Serverseitiges Rendering
- Fehler, die in der Fehlergrenze selbst und nicht in deren Kind-Komponenten auftreten

Um diese Defizite auszugleichen, wird in Eventhandlern und in den Fehlergrenzen mit Try-Catch-Blöcken gearbeitet, um Fehler abzufangen. Bei asynchronem Code (API-Client) werden die Rejected-Promises abgefangen und entsprechend behandelt. Serverseitiges Rendering wird nicht eingesetzt und ist somit nicht relevant für das Projekt.

Struktur Error Boundaries

Die Error Boundaries wurden innerhalb jeder Page, sowie um die App-Komponente platziert. Dies ermöglicht es Fehler einzelner Pages zu isolieren und ein Absturz der gesamten Applikation zu verhindern. Auf der Ebene einzelner Komponenten wurden keine Error Boundaries eingesetzt, da Komponenten auf der Page aggregiert werden. Fehler innerhalb der Unterkomponenten werden von der Page Error Boundary abgefangen und behandelt. Dies verhindert unnötig komplizierte Logik in den dummen Komponenten.

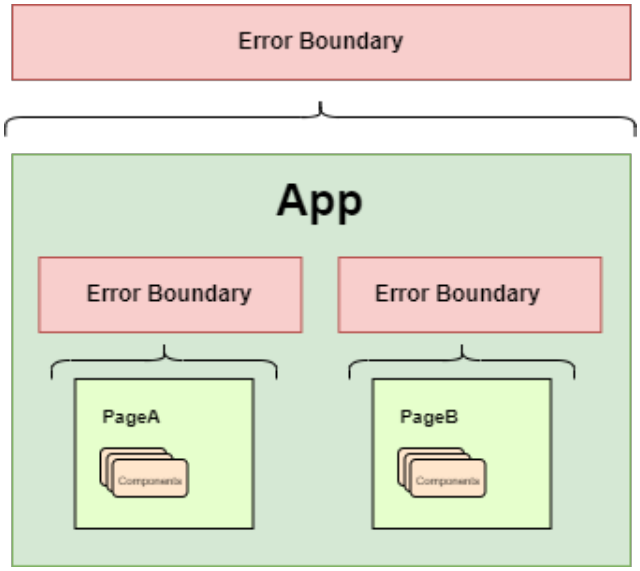


Abbildung 6.12: Aufbau Error Boundaries

Error Component

Um Nutzern Fehler darzustellen, wurde eine Error-Komponente erstellt. Diese meldet das Auftreten eines Fehlers und bietet Aktionsmöglichkeiten. Zum einen kann hier die Seite neu geladen werden, falls es ein transienter Fehler ist. Andererseits kann für technisch affine User eine genaue Fehlermeldung angezeigt werden.

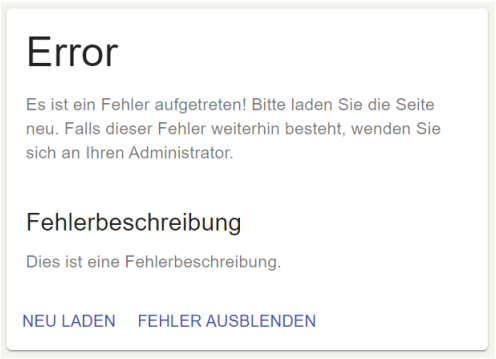


Abbildung 6.13: Beispiel - Error Component

Teil IV

Project Dokumentation

Kapitel 7

Projektplan

7.1 Entwicklungsprozess

Für die Planung des gesamten Projekts wird der Rational Unified Process [RUP] verwendet. Die effektive Arbeit wird jedoch mit Scrum [Scr] geplant - also in Sprints. RUP unterteilt das Projekt in die folgenden vier Lifecycle-Phasen.

Phasen

1. Inception
2. Elaboration
3. Construction
4. Transition

7.2 Meilensteine

Für das Projekt wurden die folgenden 7 Meilensteine definiert . Zudem sind in der Tabelle 7.6 auch Artefakte aufgeführt, die bis zum entsprechenden Meilenstein erarbeitet werden.
Hinweis: Die Liste der Artefakte kann, wo sinnvoll, im Verlauf des Projekt erweitert werden.

Meilenstein	**Artefakte**	**Datum**
M1: Projektplan	• Projektplan • Dokumentation unter Versionskontrolle (inkl. Pipeline) • Issue Management aufgesetzt • Zeiterfassung aufgesetzt	02.10.2022
M2: Anforderungen	• Funktionale Anforderungen definiert, priorisiert, Aufwand grob geschätzt und die Wichtigsten bereits verfeinert • Entwicklungsumgebung komplett aufgesetzt • Nicht-Funktionale Anforderungen definiert • Domain Model • Abnahme der funktionalen Requirments inkl. Priorisierung durch 2BIT	09.10.2022

M3: Ende der Elaborationsphase	• Überarbeiteter Projektplan • Aktualisiertes Risikomanagement (Minderung der grössten Risiken) • Testkonzept • Wireframes • Architektur-Entwurf • Interfaces zu allf. Umsystemen sind definiert • Prototyp der Applikation (End-to-End) • Aktualisierte Dokumentation	23.10.2022
M4: Architektur	• Definitive Architektur (für das Projekt) • Massnahmen bezüglich der wichtigsten nicht-funktionalen Anforderungen sind eingeleitet • Feedback aus den Usability Tests mit den Wireframes von 2BIT • Abgeleitete Massnahmen aus den Usability Tests mit 2BIT • Aktualisierte Dokumentation	13.11.2022
M5: Alpha	• Alpha-Release (MVP) für Review mit 2BIT • Feedback aus Usability Tests von 2BIT • Abgeleitete Massnahmen aus den Usability Tests mit 2BIT • Aktualisierte Dokumentation	27.11.2022
M6: Beta	• Beta-Release für Review mit 2BIT • Feedback aus Usability Tests von 2BIT (inkl. Usability Tests von einer Person, die zu diesem Zeitpunkt die Applikation zum ersten Mal sieht) • Abgeleitete Massnahmen aus den Usability Tests mit 2BIT • Entwurf des Abstracts • Aktualisierte Dokumentation	11.12.2022

M7: Finale Abgabe	• Finaler Release • Finale Dokumentation	23.12.2022

Tabelle 7.1: Liste der Meilensteine

7.3 Zeitplan

In der folgenden Grafik (7.1) ist der Projektplan mit den RUP-Phasen, den geplanten Sprints und Meilensteinen sowie den angedachten Reviews mit dem Betreuer und 2BIT (siehe Kapitel 7.4) visualisiert.

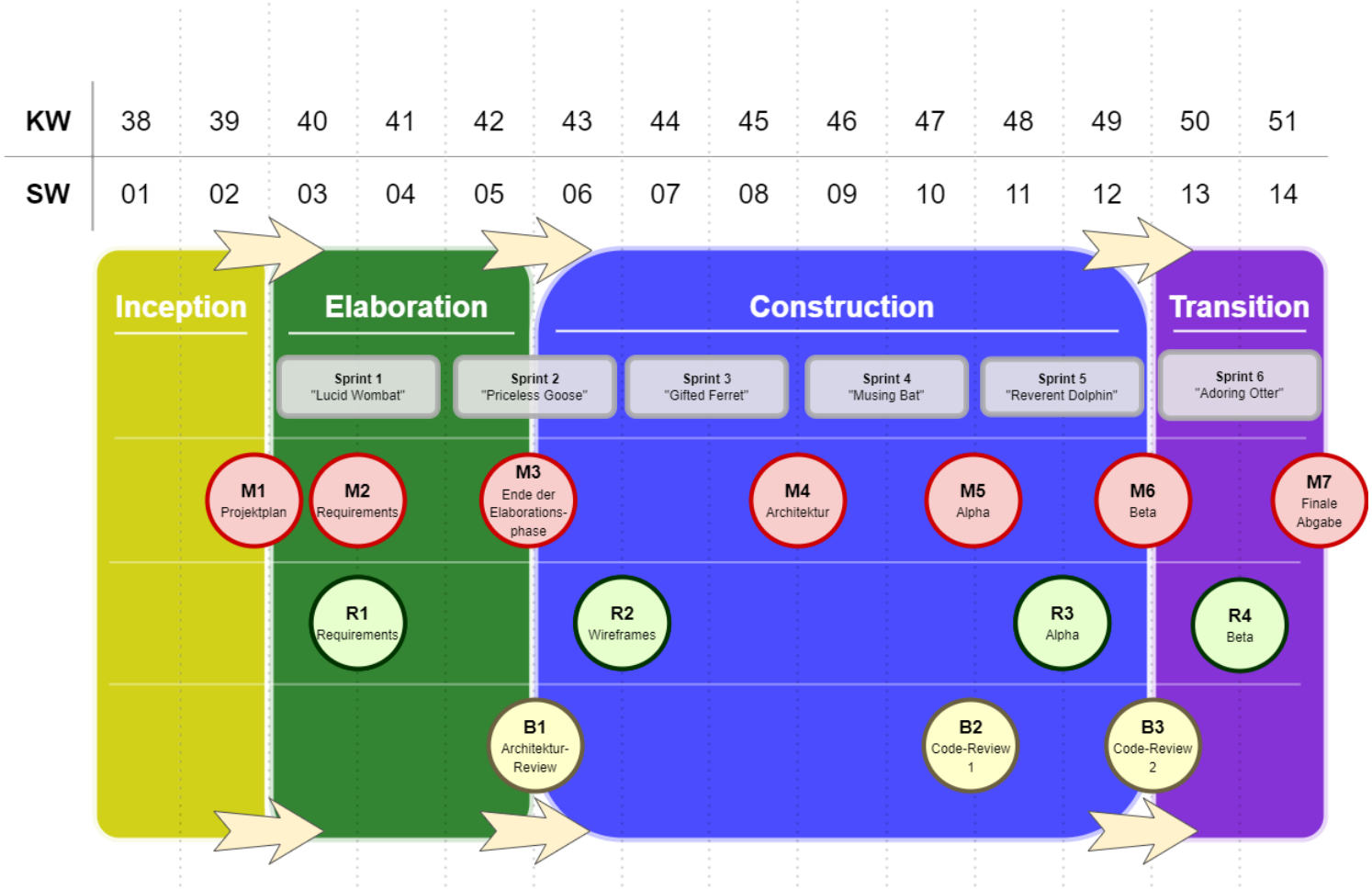


Abbildung 7.1: Projektplan

7.4 Meetings

Team-interne Meetings

Im Projekt-Team werden die folgenden Meetings durchgeführt:

Meeting	**Häufigkeit**	**Wann**	**Wo**
Sprint Planning	Alle zwei Wochen (vor jedem Sprint)	Dienstag, 17:00 - 18:00 (Reserve von 18:00 - 19:00)	MS Teams
Sprint Review	Alle zwei Wochen (nach jedem Sprint)	Dienstag, 17:00 - 18:00 (Reserve von 18:00 - 19:00)	MS Teams
Sprint Retroperspective	Alle zwei Wochen (nach jedem Sprint)	Dienstag, 17:00 - 18:00 (Reserve von 18:00 - 19:00)	MS Teams
Weekly Scrum	Jede Woche	Dienstag, 17:00 - 18:00 (Reserve von 18:00 - 19:00)	MS Teams
Backlog Refinement	Alle zwei Wochen (mit dem Weekly Scrum)	Dienstag, 17:00 - 18:00 (Reserve von 18:00 - 19:00)	MS Teams

Tabelle 7.2: Team-interne Meetings

Hinweis: Da die Projektmitglieder nur mit begrenztem Pensum (Yael Schärer 60%, Rolf Oberhänsli & Manu Weber 40%) am Projekt arbeiten, wird anstatt eines "Daily Scrumëin "Weekly Scrum"durchgeführt.

Reviews mit dem Betreuer

Wöchentlich wird ein Status-Meeting mit dem Betreuer Prof. Dr. M. Stolze durchgeführt. Dieses findet jeweils am Mittwoch von 16 - 17 Uhr über MS Teams statt.

Zudem finden die folgenden drei grösseren Reviews statt:

Review	**Datum**	**Teilnehmer**	**Wo**
B1: Architektur-Review	19.10.2022	M. Stolze, M. Gfeller, Projektteam	MS Teams
B2: Code-Review 1	23.11.2022	M. Gfeller, Projektteam	MS Teams
B3: Code-Review 2	07.12.2022	M. Gfeller, Projektteam	MS Teams

Tabelle 7.3: Grössere Reviews mit Betreuer oder Assistent

Des Weiteren wurden die folgenden Termine für die Zwischenreview und die Prüfung der SA/BA vereinbart:

Titel	**Datum**	**Teilnehmer**	**Wo**
Zwischenreview	16.11.2012	M. Stolze, Projektteam	MS Teams
Prüfung	09.02.2023	M. Stolze, F. Loch, Projektteam	Vor Ort

Tabelle 7.4: Termine SA/BA

Reviews mit 2BIT

Es werden vier Reviews mit 2BIT durchgeführt, um die Enduser (Support-Team von 2BIT) regelmässig über den Stand zu informieren und in die Priorisierung der Anforderungen zu involvieren. An diesen Reviews wird der aktuelle Stand präsentiert und Optimierungsvorschläge besprochen. Zudem werden während R2, R3 und R4 Usability Tests durchgeführt. Dies ist eine Massnahme, um das Risiko 7 'Anforderungen' (siehe Kapitel 7.8) möglichst früh zu adressieren und somit zu minimieren.

Review	**Datum**	**Teilnehmer**	**Wo**
R1: Anforderungen	07.10.2022	2BIT Product Owner und Lead Support, Projektteam	MS Teams
R2: Wireframes	28.10.2022	2BIT Product Owner und Lead Support, Projektteam	MS Teams
R3: Alpha	25.11.2022	2BIT Product Owner und Lead Support, Projektteam	MS Teams
R4: Beta	09.12.2022	Lead Support und Stellvertreter Product Owner, Projektteam	MS Teams

Tabelle 7.5: Reviews 2BIT

7.5 Rollen

Im Projekt-Team sind die folgenden Rollen verteilt:

Role	**Zugewiesene Person(en)**	**Stellvertreter**
Frontend	Manu Weber	Rolf Oberhänsli
API / Backend	Rolf Oberhänsli	Yael Schärer
Datenbank	Yael Schärer	Manu Weber
Meeting-Organisation	Yael Schärer	Rolf Oberhänsli
Protokollführer(in)	Rolf Oberhänsli	Manu Weber
Dokumentations-Master	Manu Weber	Yael Schärer
Scrum Master	Manu Weber	Rolf Oberhänsli
Product Owner	Yael Schärer	Rolf Oberhänsli
Entwickler(in)	Yael Schärer, Rolf Oberhänsli, Manu Weber	-
Softwarearchitektur	Yael Schärer	Rolf Oberhänsli
Layout-Master	Yael Schärer	Manu Weber

Tabelle 7.6: Rollenverteilung

Details zu den definierten Rollen

- Die Scrum-Rollen (Scrum Master, Product Owner, Entwickler) werden wie von Scrum [Scr] definiert eingesetzt.
- Die Rolleneinteilung (Frontend, API / Backend & Datenbank) ist nicht strikt. Alle Personen können in allen Bereichen mitarbeiten. Die Rolle definiert jedoch wer die Hauptverantwortung für einen Bereich hat.
- Meeting-Organisation: Organisiert Meetings und Reviews mit 2BIT und dem Betreuer.
- Protokollführer(in): Führt das Protokoll während Meetings und Reviews. Hat die Hauptverantwortung für die Protokolle.
- Dokumentations-Master: Hat die Hauptverantwortung dafür, dass die Dokumentation up-to-date gehalten wird. Die Dokumentation wird im Team geschrieben, diese Person sorgt lediglich dafür, dass die Dokumentation stetig aktuell gehalten wird.
- Softwarearchitektur: Hat die Hauptverantwortung für alle Architektur-Dokumente. Die Architektur wird vom Projekt-Team in Zusammenarbeit mit 2BIT definiert.
- Layout-Master: Ist verantwortlich für das Design und Layout der Dokumentation.

7.6 Ansprechpersonen bei 2BIT

Bei 2BIT stehen dem Projektteam primär die folgenden Ansprechpersonen zur Verfügung:

Name	**Rolle**
Product Owner	Product Owner der Applikation 2getHR und dem zugehörigen Admin-Portal
Lead Support	Supporter Lead der Applikation 2getHR und Enduserin vom Admin-Portal

Tabelle 7.7: Ansprechpersonen 2BIT

7.7 Arbeitspakete

Tabelle 7.8 verschafft einen groben Überblick über die Arbeitspakete der verschiedenen Phasen.

Name	**Phase**	**Meilenstein**	**Zeitschätzung**
• Initiale Planung • Projektplan • Risiko Management • Setup Dokumentation • Support Analyse • Issue Management aufsetzen • Zeiterfassung aufsetzen	Projektplan	M1	SW 1-2
• Analyse bestehendes System • User Stories • Nicht funktionale Anforderungen • Domain Analyse • Setup Entwicklungsumgebung • Domain Model	Anforderungen	M2	SW 2-3
• Architektur definieren (C4 Modell) • API-Definition • Wireframes • Papier Prototyp und Usability-Tests • Quality Management • Applikationsprototyp • End of Elaboration (EoE) Checkliste • Testkonzept	End of Elaboration	M3	SW 3-6

• Architektur vollständig umsetzen • Erste Features implementieren	Architektur	M4	SW 6-8
• MVP implementieren • Zweiter Usability-Test • Testing	Alpha	M5	SW 8-10
• Bugfixes • Usability-Test Ergebnisse umsetzen • Optionale Features Implementieren • Abstract	Beta	M6	SW 10-12
• Dokumentation • Thesis	Finale Abgabe	M7	SW 12-14

Tabelle 7.8: Arbeitspakete Übersicht

7.8 Risikomanagement

Risikomatrix

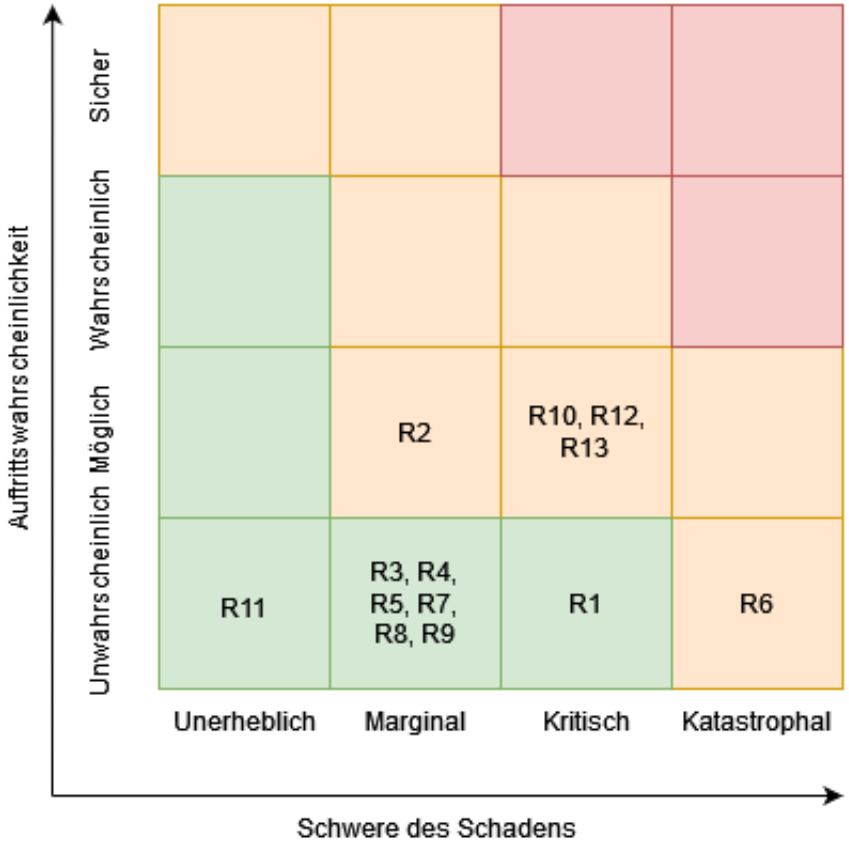


Abbildung 7.2: Risikomatrix

Risiken

Folglich werden alle erkannten Risiken im Projekt aufgelistet.

Hinweis: Die Schwere bezieht sich auf die Auswirkung im produktiven Betrieb und der Einfluss bezeichnet die Auswirkungen auf das Projekt.

Nr	Titel	Beschreibung	Wahrscheinlichkeit	Schwere	Einfluss
1	Technologiewandel	Im Projekt werden Technologien eingesetzt, die sich fortschreitend weiterentwickeln, was ein Mismatch in der Applikation verursacht. Die angestrebte Lösung muss daher ohne grossen Mehraufwand auf solche Änderungen reagieren können.	Unwahrscheinlich	Kritisch	Marginal

2	Gesundheit	Ein oder mehrere Teammitglieder fallen durch Krankheit oder Umfall für längere Zeit aus. Die Aufgaben dieses Teammitglieds werden nicht ausgeführt.	Möglich	Unerheblich	Gering
3	Kommunikation	Ein Teammitglied beteiligt sich nicht an Meetings oder diskutiert Unstimmigkeiten nicht offen. Dies kann zu weiteren Problemen führen, die negativen Einfluss auf das Projekt haben.	Unwahrscheinlich	Marginal	Medium
4	Projektumfang	Der Umfang des Projekts ist zu groß und die angestrebten Ziele sind nicht realistisch.	Unwahrscheinlich	Marginal	Gering
5	Wissen	Mangelndes technisches Wissen und Erfahrung von Teammitgliedern beeinflusst und verzögert die zeitliche Planung.	Unwahrscheinlich	Marginal	Medium
6	Externe Gefahren	Die Energiekriese, Unwetter oder andere nicht beeinflussbare Faktoren machen ein Weiterarbeiten am Projekt unmöglich.	Unwahrscheinlich	Marginal	Medium
7	Anforderungen	Falsche, unvollständige oder fehlende funktionale oder nicht funktionale Anforderungen führen zu unzureichender Funktionalität in der Anwendung.	Unwahrscheinlich	Marginal	Medium
8	Architektur	Die ausgearbeitete Architektur ist nicht realisierbar oder erfüllt funktionale oder nicht-funktionale Anforderungen nicht.	Unwahrscheinlich	Marginal	Medium
9	Zeitmanagement	Durch mangelhafte Zeitplanung kann die Applikation nicht im vorgegebenen Zeitrahmen umgesetzt werden.	Unwahrscheinlich	Marginal	Medium
10	Softwarequalität	Unzureichende Tests führen zu unentdeckten Fehlern und Bugs im Code.	Möglich	Marginal	Medium
11	Pandemie und Krieg	Einflüsse aufgrund des Krieges in der Ukraine oder der Pandemie bringen ausserordentliche Situationen hervor und schränken den Fortschritt des Projektes ein.	Unwahrscheinlich	Unerheblich	Gering

12	Usability	Die ausgearbeitete Lösung kann nicht von Personen ohne technischem Hintergrund bedient werden, da das Interface zu komplex und unübersichtlich strukturiert ist.	Möglich	Marginal	Gering
13	Security	Sicherheitsanforderungen wurden nicht erkannt und führen zu Sicherheitslücken in der Applikation	Möglich	Marginal	Gering

Tabelle 7.9: Risiken

Risikominimierung und -prävention

In der folgenden Tabelle werden alle Massnahmen zur Abschwächung von Risiken aufgelistet.

Betroffene Risiken	**Massnahme**
R1	Software wird modular und isoliert aufgebaut, so dass das Ersetzen von Technologien ohne grösseren Aufwand umgesetzt werden kann. Zusätzlich werden die Tools ausgiebig evaluiert und analysiert. Es werden nur Bibliotheken verwendet, die langlebig sind und aktiv gepflegt werden.
R2, R3, R4, R6, R11	Es wird Fokus auf Engagement innerhalb des Teams, direkte Kommunikation, wöchentliche Meetings und Statusaktualisierungen gelegt. Falls ein extremer Ausfall eintrifft wird eine Verzögerung des Abgabetermins beim Studiengang beantragt.
R4, R7, R12	Es werden intensive Reviews und Usability-Tests mit dem Berater und dem Industriepartner durchgeführt. Zudem wird das Projekt sorgfältig geplant. Dafür werden gängige Praktiken eingesetzt.
R5	Während der Inception-Phase arbeiten sich die Teammitglieder in unbekannte Technologien ein und es werden Pufferzeiten für die Entwicklung eingeplant.
R8, R12	Die Architektur wird im Team geplant und mit dem Berater besprochen. Für eine klare Kommunikation und Planung wird eine Domänenanalyse durchgeführt und Architekturdiagramme erstellt. Falls nötig werden zusätzliche Experten hinzugezogen um allfällige Probleme zu lösen.
R9	Die Planung wird sauber aufgegleist und Pufferzeiten werden miteinberechnet. Im Sprint Review werden geleistete sowie verbleibende Arbeitsstunden analysiert und mit den Arbeitspaketen verglichen. Wo nötig werden Anpassungen an den Arbeitspaketen vorgenommen.
R10, R13	Das Testen wird priorisiert und TDD wird als grundlegende Entwicklungspraktik eingesetzt. Erledigte Tasks müssen die Definition of Done (siehe Kapitel 6.3) erfüllen.

Tabelle 7.10: Massnahmen zur Risikominimierung und -prävention

Risikoentwicklung

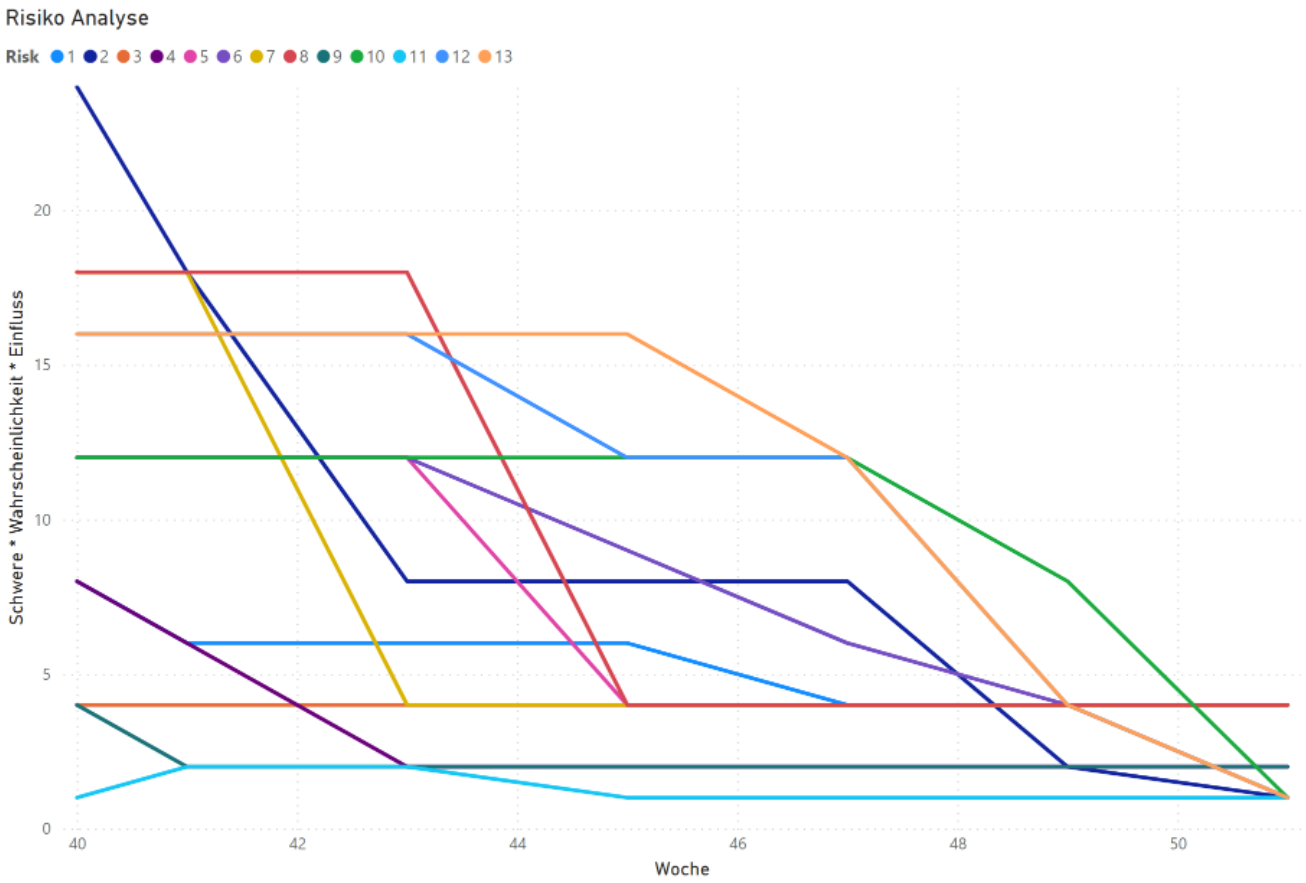


Abbildung 7.3: Risikoentwicklungsgraph

7.9 Issue-Management

Die Issues werden über Azure DevOps in der Infrastruktur von 2BIT verwaltet. Die Issues werden als sogenannte `Work Items` erfasst. Sie erhalten eine ID, auf welche sich in der Zeiterfassung (siehe Kapitel 7.11) bezogen wird.

Die Issues werden auf einem Board mit folgenden Stages organisiert:

1. Epics
   - Anforderungen werden als Epics in dieser Stage grobgranular ausgewiesen.
   - Der Aufwand von Epics wird aufgrund einer relativen Schätzungsskala (T-Shirt-Grössen) geschätzt.
   - Epics verbleiben als Übersicht in dieser Stage.
2. Product Backlog
   - Epics werden zu User Stories verfeinert.
   - Epics und User Stories werden miteinander verlinkt.
   - User Stories werden mit dem Präfix [US] im Product Backlog abgelegt.

- Der Aufwand von User Stories wird mit einer relativen Schätzungsskala (Story-Punkte) geschätzt.
- Aus den User Stories werden Tasks (Work Items) abgeleitet.
- Tasks werden mit den User Stories verlinkt und in einem ersten Schritt im Product Backlog abgelegt.
- Der Aufwand von Tasks wird absolut (in Stunden) geschätzt.

3. Sprint Backlog
   - Tasks, die im aktuellen Sprint bearbeitet werden sollen, werden während des Sprint Planning im Sprint Backlog abgelegt.
   - Wichtig: Nur Tasks, deren Aufwand bereits geschätzt wurde, können in einen Sprint eingeplant werden.
4. In Bearbeitung
   - Tasks, an denen die Teammitglieder aktiv arbeiten, befinden sich in dieser Stage.
5. In Review
   - Tasks, die bearbeitet und bereit für ein Review sind, befinden sich in dieser Stage.
6. Done
   - Tasks und User Stories, die gemäss Kapitel 6.3 abgeschlossen sind, werden in Done abgelegt.

7.10 Versionskontrolle

Dokumentation

Die Versionskontrolle der Dokumentation läuft über GitLab in der Infrastruktur der OST. Link auf das GitLab-Repository: https://gitlab.ost.ch/rolf.oberhaensli/sa-dokumentation.

Source-Code

Die Versionskontrolle des Source-Codes wird über Azure DevOps von 2BIT gehandhabt. Link auf das DevOps-Repository: https://2bit.visualstudio.com/_git/2payroll.

7.11 Zeiterfassung

Für die Zeiterfassung wird die Applikation 2getHR von 2BIT eingesetzt. Diese Applikation eignet sich optimal, da im Projekt das Admin-Portal von der 2getHR-Applikation ausgebaut wird. Somit sind die Mitglieder des Projektteams auch Enduser der Applikation 2getHR.

Für die Zeiterfassung wurden folgende Aktivitätsarten definiert:
- Dokumentation
- Code
- Management
- Research
- Team Meeting
- Betreuer Meeting

- Meeting mit 2BIT

Des Weiteren wird einem Time-Report der zugehörige Meilenstein (siehe Kapitel 7.2) zugewiesen. Für den Fall, dass an Issues gearbeitet wird, wird die ID des Issue / der Issues (siehe Kapitel 7.9) im Kommentarfeld des Time-Reports angegeben.

Kapitel 8

Übergabe Produkt an 2BIT

Das fertige Produkt konnte in der letzten Woche erfolgreich an 2BIT weiteregegeben werden. Das Meeting Protokoll kann im Anhang **??** eingesehen werden. Hierbei wurden folgende Ergebnisse abgegeben:

- Wiki Dokumentation des Projektes mit allen relevanten Informationen zur Weiterentwicklung. Hier wurden auch alle ausstehenden Arbeiten aufgeführt. Kann im Anhang als PDF-Auszug eingesehen werden.
- Codebasis als Merge Request in den 2getHR dev Branch.
- Laufende Demo Applikation auf 2BIT Azure Infrastruktur.
- Bestätigung der erfolgreich umgesetzten Anforderungen.
- Dokumentation Anforderungen für weitere Arbeit am Projekt.

Während des Meetings wurden folgende offene Punkte festgehalten, die noch zu überabreiten sind, um das Produkt erfolgreich in die produktive Umgebung zu mergen. Es wurde mit 2BIT bestimmt, dass diese erst nach Projektende von 2BIT ausgeführt werden, um den Merge in die Entwicklungsumgebung ohne zusätzliche Komplikationen durchführen zu können.

- Es wurden Merge Konflikte in der bestehenden 2getHR Codebasis entdeckt. Diese werden von 2BIT gelöst. Die Codebasis des Admin-Portals weist keine Merge Konflikte auf.
- Die folgenden beiden Datenfelder in der Detailansicht des Mitarbeiters, 'Mitarbeiter freigegeben am' und 'Interviewer freigegeben am', sind fälschlicherweise mit 'unterzeichnet am' beschriftet.
- Sobald der Code in einer vollständig konfigurierten Umgebung läuft, können die beiden Funktionalitäten Tenant Löschen und Mitarbeiter E-Mail ändern vollständig durchgeführt werden. Entsprechend müssen zwei auskommentierte Codezeilen angepasst werden. Diese wurden mit einem TODO versehen und dokumentiert. Für genauere Informationen siehe 5.8.
- Der 404 Resource nicht gefunden Fehler (siehe 5.8) kann auf der produktiven Umgebung korrigiert werden. Eine Anleitung diesbezüglich findet sich unter folgendem Link https://antbutcher.medium.com/hosting-a-react-js-app-on-azure-blob-storage-azure-cdn-for-ssl-and-routing-8fdf4a48feeb.
- Da das Projekt auf einer Entwicklungsumgebung mit eingeschränkter Performance durchgeführt wurde, konnten die Lasttests nicht aussagekräftig getestet werden 3.45. Eine Empfehlung diesbezüglich wurde and 2BIT weitergereicht.

Ein Teammitglied wird bei 2BIT weiterarbeiten und steht für allfällige Fragen und Arbeiten zur Verfügung.

Teil V

Ergebnisse

Kapitel 9

Resultate

Im Anhang ist ein Video zu finden, das die Funktionalitäten der erarbeiteten Applikation demonstriert.

9.1 Funktionale Anforderungen

UC00 Login AAD

Die Loginfunktionaliät wurde von der bestehenden Lösung übernommen und optimiert. Das Login funktioniert auf Basis des Azure Active Directories von 2BIT. Nur Personen mit autorisierten Zugangsdaten können sich Zugang zum Admin-Portal verschaffen. Unautorisierte Personen haben somit keinen Zugang zum Portal. Navigiert ein nicht eingeloggter User auf das Admin-Portal wird dieser zur Login Seite von Microsoft umgeleitet.

UC01 (Optimierte) Listenansicht

Die Daten, die in den Listenansichten angezeigt werden, wurden zusammen mit 2BIT anhand einer Whitelist erarbeitet und entsprechend umgesetzt. Es wurden vier Listenansichten implementiert: die Tenantliste, die Treuhänderliste, die Mitarbeiterliste und die Mitarbeitergesprächeliste. Die Listenansichten sind paginiert und können nach jeder Spalte sortiert werden. Die Anordnung und Sichtbarkeit der Spalten der Listenansichten kann vom User selbst konfiguriert werden. Die Konfiguration wird im lokalen Speicher persistiert und bleibt somit Session-übergreifend erhalten. Für jede Listenansicht gibt es ein spezifisches Filterpanel, das ausgewählte Filter anbietet mit denen die Einträge der Listenansichten eingeschränkt werden können.

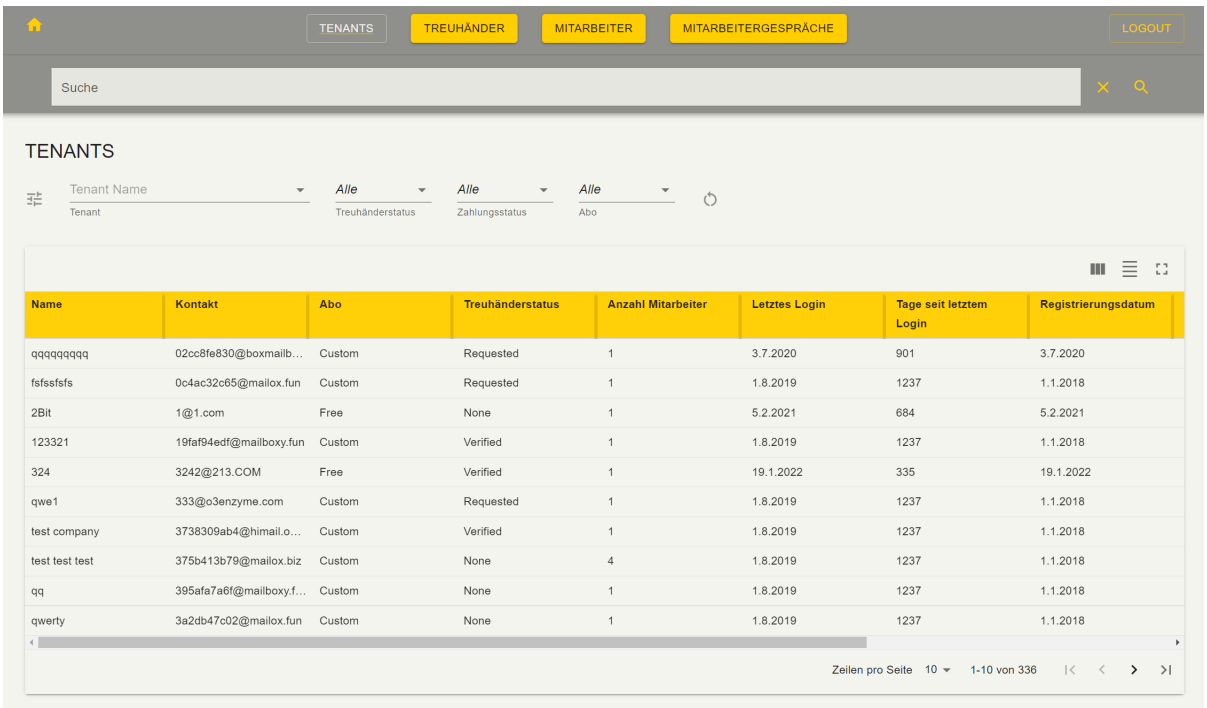


Abbildung 9.1: Exemplarische Listenansicht: Tenant

UC02 Crossnavigation Listenansicht

Für die Navigation zwischen den Listenansichten wurde ein Menu erstellt, das es dem Nutzer ermöglicht mit einem Klick zwischen den gewünschten Listenansichten zu navigieren. Die eingestellte Sortierung sowie

Filterung der Listenansichten wird in den lokalen Speicher geschrieben und ist somit Session-übergreifend verfügbar.

UC03 Aktionsjournal

Dieser Use Case wurde nur bedingt umgesetzt. Im Frontend werden alle Mutationsvorgänge geloggt. Jedoch war es nicht möglich diese Logs zu persistieren. Das Anbinden der Azure Storage Solution wurde für die Projektumgebung nicht konfiguriert. Momentan werden Logeinträge im Backend auf den Standard-Output-Stream geloggt (siehe auch Kapitel 5.8). Dass die vollständige Umsetzung dieser Anforderung noch aussteht, wurde im Rahmen der Übergabe an 2BIT ausgewiesen.

UC04 Anzeigegedächtnis

Die Navigation wurde mit Hilfe des React Routers umgesetzt. Die Anforderungen an den Back-Button im Browser konnte somit erfüllt werden. Der Benutzer hat die Möglichkeit über diesen auf die zuvor besuchten Ansichten zurück zu navigieren.

UC05 Tenant (Read & Update)

Es wurde eine Tenant-Detailansicht entwickelt. Diese ist über die Listenansicht sowie die Suche erreichbar und zeigt dem Nutzer alle benötigten Informationen zu einem Tenant. Zusätzlich wird die Möglichkeit angeboten den Treuhänderverifikationsstatus anzupassen sowie den Tenant zu löschen.

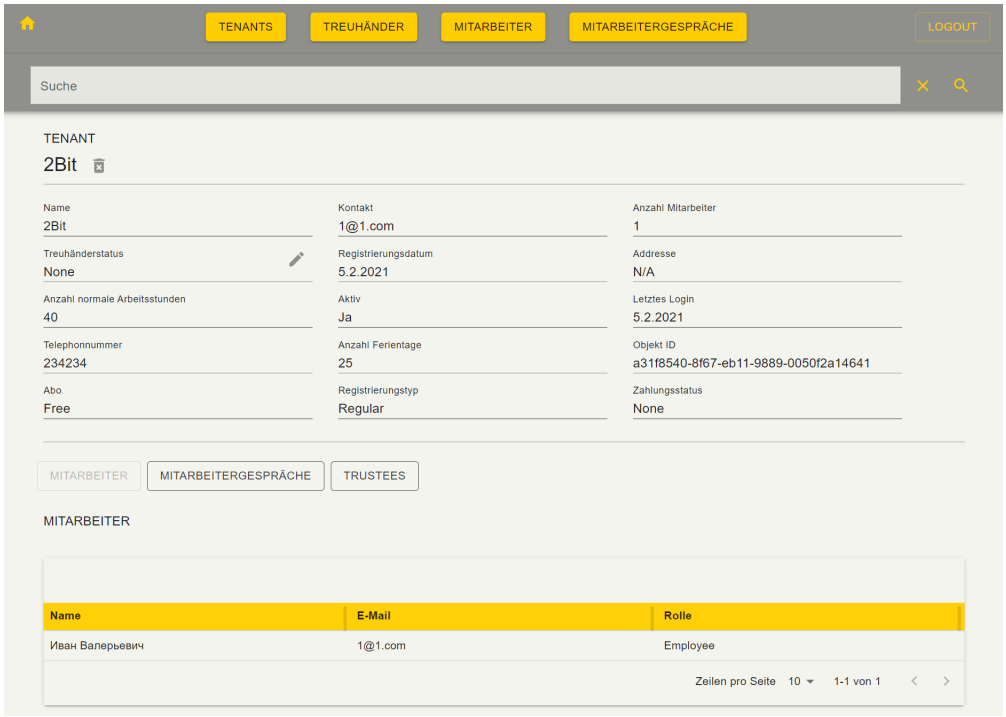


Abbildung 9.2: Detailansicht Tenant / Treuhänder

UC06 Mitarbeiter (Read & Update)

Es wurde eine Mitarbeiter-Detailansicht entwickelt. Diese ist über die Listenansicht sowie die Suche erreichbar und zeigt dem Nutzer alle benötigen Informationen zu einem Mitarbeiter. Zusätzlich wird die Möglichkeit angeboten die E-Mail-Adresse und die Rolle des Mitarbeitenden anzupassen. Die

Projektumgebung erlaubt den vollständigen E-Mail-Update-Vorgang nicht und wirft eine Fehlermeldung (siehe Kapitel 5.8 für Details). Die Korrektheit des Vorgangs konnte auf der Entwicklungsumgebung von 2BIT jedoch verifiziert werden.

UC07 Crossnavigation Objekte

In den Detailansichten der einzelnen Entitäten sind die zugehörigen Objekte verlinkt. Diese Links sind entweder direkt in den Detailinformationen eingebaut (zum Beispiel der Tenant) oder über eine Liste der zugehörigen Objekte (zum Beispiel die Mitarbeitergespräche). Über die Links kann direkt, mit einem Klick, zu der gewünschten Entität navigiert werden.

UC08 Mitarbeitergespräche (Read & Update)

Es wurde eine Mitarbeitergespräche-Detailansicht entwickelt. Diese ist über die Listenansicht sowie die Suche erreichbar und zeigt dem Nutzer alle benötigten Informationen zu einem Mitarbeitergespräch. Zusätzlich wird die Möglichkeit angeboten die Mitarbeitergespräche zu löschen.

UC10 Lösch-Funktionen Mitarbeiter

Die Löschfunktionalität für Mitarbeiter-Objekte wurde implementiert. Dabei werden beim Öffnen des Lösch-Dialogs alle Referenzen des Objektes angezeigt. Hat ein Mitarbeiter spezifische Referenzen, ist er zum Beispiel Abteilungsleiter, kann dieser nicht gelöscht werden. Ist dies der Fall, wird der Benutzer informiert und die Löschfunktionalität steht nicht zur Verfügung. Hat der Mitarbeiter keine Referenzen wird die Löschfunktionalität angeboten und eine Löschung kann durchgeführt werden.

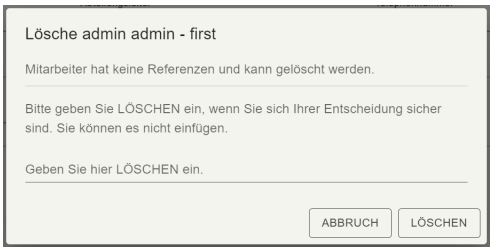


Abbildung 9.3: Löschen eines Mitarbeiters, Keine Referenzen vorhanden

Abbildung 9.4: Löschen eines Mitarbeiters, Referenzen vorhanden

UC11 Suche

Die Suchfunktion wurde umgesetzt und bietet die Möglichkeit Entitäten zu finden und direkt zu diesen zu navigieren. Die Suche gliedert Ergebnisse in drei Kategorien, Tenants, Mitarbeiter und Mitarbeitergespräche.

Abbildung 9.5: Suche mit Gliederung nach Kategorien

Ausstehende Use Cases

Folgende Use Cases wurden aus zeitlichen Gründen nicht in diesem Projekt umgesetzt. Diese Use Cases wurden an 2BIT übergeben.

- UC09 CRUD von Custom-Subscription-Packages
- UC12 Support-Zugriff auf Kunden-Tenants
- UC13 SendGrid-Integration
- UC14 Zurzeit aktive 2getHR-User
- UC15 Flectra-Integration
- UC16 Reports & Statistiken zu 2getHR
- UC17 Mitarbeiter verifizieren
- UC18 Feiertage
- UC19 Breadcrumbs

9.2 Nicht-Funktionale Anforderungen

Die nicht funktionalen Anforderungen wurden je nach dem Alpha-, Beta- und Final-Release kontrolliert und entsprechende Massnahmen eingeleitet. Die genaue Analyse kann in Kapitel 3.6 eingesehen werden.

NFR01

Die neue Lösung soll mindestens gleich gut bedienbar sein wie die alte Lösung.

✔ Das NFR ist erfüllt.

Ein Direktvergleich der beiden Lösungen durch das Supportpersonal von 2BIT hat gezeigt, dass die Lösung fühlbar angenehmer zu bedienen ist als die bestehende Lösung.

NFR02

Das Admin-Portal läuft parallel zur 2getHR-Applikation ohne diese beträchtlich in der Performance einzuschränken.

✔ Das NFR ist bedingt erfüllt.

Es wurde ein Last- und Performance-Test auf der Entwicklungsumgebung durchgeführt (siehe Kapitel 3.45). Dieser ist inkonklusiv für die Performance-Messung, da in der Entwicklungsumgebung nur stark begrenzte Ressourcen zur Verfügung stehen. Jedoch ist die zusätzliche Last der Nutzer minimal im Vergleich zu den dokumentierten Benutzerzahlen von 2getHR selbst. Eine Empfehlung für einen Last- und

Performance-Test in der Produktivumgebung wurde an 2BIT weitergegeben, um dieses NFR vollständig zu verifizieren.

NFR03

Die Daten im Admin-Portal sind konsistent mit den Daten in 2getHR.

✔ Das NFR ist erfüllt.

Vordefinierte Aktionen auf dem Admin-Portal, wie das Wechseln der Ansicht und das Bearbeiten von Daten, provozieren einen Refresh der Daten. Somit werden immer aktuelle Daten angezeigt, solange der User aktiv auf der Applikation ist.

NFR04

Das Admin-Portal kann nur von Usern mit autorisiertem Azure Active Directory Zugriff benutzt werden.

✔ Das NFR ist erfüllt.

User können sich nur im Admin-Portal einloggen, wenn diese einen gültigen und registrierten Azure Active Directory Account mit Zugriff auf das Admin-Portal haben. Ohne diese Autorisierung ist es nicht möglich Funktionalitäten der Applikation zu benutzen oder Daten aus 2getHR einzusehen.

NFR05

Die Daten sind in Transit gesichert.

✔ Das NFR ist erfüllt.

Wird ein unsicheres Zertifikat in einer MitM-Attacke eingesetzt, wird eine Fehlermeldung angezeigt und keine Daten sind einsehbar.

NFR06

Es werden keine sensiblen Userdaten angezeigt.

✔ Das NFR ist erfüllt.

2BIT hat eine Whiteliste bereitgestellt von allen Daten, die vom Support gebraucht und somit im Admin-Portal angezeigt werden dürfen. Das Admin-Portal zeigt ausschliesslich diese Daten an. Desweiteren werden im Backend auch nur diese Daten bereitgestellt. Somit können sensible Userdaten auch nicht in Transit ausgelesen werden.

NFR07

Daten vom Backend werden vor der Anzeige sanitized.

✔ Das NFR ist erfüllt.

Der eingesetzte API-Client von Axios bietet eingebaute Sanitization-Funktionalitäten out-of-the-Box. Dies wurde per manuell eingefügtem gefährlichen Input verifiziert.

NFR08

Die Software ist modular aufgebaut und erlaubt somit ein einfaches Erweitern der Applikation.

✔ Das NFR ist erfüllt.

Die Software ist modular aufgebaut und kann ohne grössere Änderungen an bestehenden Funktionalitäten erweitert werden.

NFR09

Benutzereingaben werden validiert und sanitized.

✔ Das NFR ist erfüllt.

Im Admin-Portal sind nur zwei Usereingaben möglich: E-Mail-Adresse und Sucheingabe. Die E-Mail-Adresse wird auf formale Korrektheit validiert und kann nur gespeichert werden, wenn die E-Mail-Adresse ein gültiges Format hat. Es wurde im Projektteam beschlossen die Sucheingabe nicht zu validieren, um die Suchmöglichkeiten nicht einzuschränken. Für beide Inputs wird durch Axios automatisch ein Escaping der Strings durchgeführt. Auch wird durch die eingesetzte LINQ Bibliothek eine weitere Sanitization durchgeführt.

NFR010

Performance

✔ Das NFR ist erfüllt.

Funktionen, die unabhängig vom Backend sind, werden in der Entwicklungsumgebung in unter 1 Sekunde durchgeführt. Alle API-Aktionen werden in unter 2 Sekunden durchgeführt und es wird während der Ladezeit ein Loading-Indikator angezeigt.
Im produktiven Betrieb wird die Performance noch besser sein, da dort eine performantere Umgebung eingesetzt wird.

NFR011

Browser Kompatibilität

✔ Das NFR ist erfüllt.

Die Usability Tests wurden erfolgreich auf den beiden Browsern Chrome & MS Edge durchgeführt.

NFR012

Logging

✔ Das NFR ist bedingt erfüllt.

Im Frontend ist die Logging-Funktionalität vollständig umgesetzt. Bei jeder erfolgreichen Update- und Löschaktion wird ein entsprechender Logeintrag ans Backend geschickt. Die Logs werden jedoch momentan nicht persistiert. Für genauere Informationen siehe Kapitel 5.8.

NFR13

Test-Coverage

✔ Das NFR ist erfüllt.

Die Businesslogik des Admin-Portals wurde vollständig getestet. Die Code-Coverage berägt im Backend 96% und im Frontend rund 80%. Für genauere Informationen siehe Kapitel 6.4.

NFR14

Dem User werden relevante Fehlermeldungen angezeigt.

✔ Das NFR ist erfüllt.

Wenn ein Fehler auftritt wird dem User eine generische Fehlermeldung angezeigt und ein Seitenreload als Aktion angeboten. Benötigt der User spezifischere Informationen kann ein detaillierterer Fehlertext angezeigt werden, der Entwicklern bei der Behebung des Fehlers helfen kann. Aus Sicherheitsgründen wurde davon abgesehen noch genauere Informationen anzuzeigen.

NFR15

Angezeigte Texte werden in einem separaten File gehalten.

✔ Das NFR ist erfüllt.

Alle angezeigten Texte des Admin-Portals werden in einem separaten JSON Dokument gehalten und können so alle an einem Ort gewartet werden. Auf Basis dieser Architektur kann das Admin-Portal bei Bedarf ohne grossen Mehraufwand um eine Localization-Funktionalität erweitert werden.

NFR16

Eingesetzte Libraries sind auf dem neusten Stand.

✔ Das NFR ist erfüllt.

Für den Final-Release wurde am Tag des Code Freezes ein Update aller eingesetzten Libraries erfolgreich durchgeführt.

NFR17

Multi-User Funktionalität

✔ Das NFR ist erfüllt.

Um Sicherheit vor Multi-User Problemen wie Lost Updates zu bieten, wurde ein Snapshot-Test vor jeder Update Funktionalität umgesetzt. Wird ein Update ausgelöst wird das Objekt im Frontend zuerst mit dem aktuell in der Datenbank persistierten Objekt verglichen. Bestehen Unterschiede wird das Vorgang abgebrochen und der User entsprechend informiert.

9.3 Vergleich des alten und neuen Admin-Portals

Grundsätzlich zeigt sich, dass die neue Lösung einen grösseren Funktionsumfang besitzt, benutzerfreundlicher ist und ein schnelleres Lösen von Supportfällen zulässt. Dies wurde von der Benutzerbasis in den Usability Tests bestätigt.
Folgende Tabelle zeigt auf, welche funktionalen Anforderungen erfüllt wurden und wie die Klickdifferenz zur alten Lösung aussieht. Falls die Klickdifferenz leer ist (-), bedeutet dies, dass in der alten Lösung der betroffene Use Case nicht implementiert war.

Use Case	**Aktion**	**Klicks**	**Differenz**	**Kommentar**
UC00 - Login AAD	Einloggen	0	0	Wurde übernommen
UC01 - Listenansichten	Anzeigen	1	0	Tenants ist Homepage, Mitarbeiter und Mitarbeitergespräche können innerhalb eines Klicks erreicht werden.
UC01 - Listenansichten	Filtern	1	-4	Filter sind neu direkt oberhalb der Listenansichten platziert und können über einen Klick erreicht werden. Die Filterung wird neu im Backend durchgeführt.
UC01 - Listenansichten	Sortieren	1	0	Die Sortierfunktionalität ist nach wie vor direkt auf den Spalten der Tabelle angesiedelt. Die Sortierung wird neu im Backend durchgeführt.
UC01 - Listenansichten	Persistenz	-	-	Tabellenkonfigurationen werden im Local Storage gespeichert und sind somit über längere Zeit verfügbar.
UC01 - Listenansichten	Paginierung	1	0	Die Tabellen werden paginiert dargestellt und ist am unteren Tabellenrand einer Tabelle steuerbar. Die Paginierung wird neu im Backend durchgeführt.
UC02 - Crossnavigation	Ansicht Wechseln	1	-1	Es wurde eine Crossnavigation implementiert, die es erlaubt von jeder Entität auf alle relevanten verbundenen Objekte zuzugreifen. Dies erhöhte die Zugänglichkeit der Listen.
UC03 - Logging	Logs einsehen	-	-	Neu werden Aktionen des Benutzers geloggt.
UC04 - Anzeigegedächtnis	-	1	0	Das Anzeigegedächtnis funktioniert neu.
UC05 - Tenant	Read	1	1	Eine Tenant-Detailansicht wurde erstellt. Darauf sind alle benötigten Informationen enthalten.
UC05 - Tenant	Update	3	-	Die für den Tenant relevanten Datenfelder können bearbeitet werden.

UC06 - Mitarbeiter	Read	1	0	Mitarbeiter können in einer separaten Listenansicht eingesehen werden. Zudem wurde eine Detailseite erstellt worauf alle zu einem Mitarbeiter relevanten Daten sowie seine verbundenen Objekte eingesehen werden können.
UC06 - Mitarbeiter	Update	3	-	Auf der Detailansicht der Mitarbeiter können die relevanten Datenfelder editiert werden.
UC07 - Crossnavigation Objekte	-	2	-	Innerhalb jeder Detailansicht wurden die für eine Entität relevanten verbunden Objekte verlinkt.
UC08 - Mitarbeitergespräche	Read	1	0	Mitarbeitergespräche können in einer separaten Listenansicht eingesehen werden. Zudem wurde eine Detailseite erstellt, worauf alle zu einem Mitarbeitergespräch relevanten Daten eingesehen werden können.
UC09 - CRUD Custom Package	Read	-	-	Nicht implementiert. Nur die Information, ob ein Custom Package verwendet wird, ist vorhanden.
UC09 - CRUD Custom Package	Create, Update, Delete	-	-	Nicht implementiert.
UC10 - Mitarbeiter Löschen	-	3	-	Ein Mitarbeiter kann über die Detailansicht gelöscht werden. Zudem kann eingesehen werden, welche Referenzen der Mitarbeiter besitzt.
UC11 - Suche	-	-	-	Über das Suchfeld können spezifische Objekte gesucht werden. Die Resultate werden in drei Kategorien unterteilt, Tenant, Mitarbeiter, und Mitarbeitergespräche.
UC12 - Support-Zugriff	-	-	-	Nicht implementiert.
UC13 - SendGrid	-	-	-	Nicht implementiert.
UC14 - Aktive USer	-	-	-	Nicht implementiert.
UC15 - Felctra	-	-	-	Nicht implementiert.
UC16 - Reports	-	-	-	Nicht implementiert.
UC17 - Mitarbeiter verifizieren	-	-	-	Nicht implementiert.
UC18 - Feiertage	-	-	-	Nicht implementiert.
UC19 - Breadcrumbs	-	-	-	Nicht implementiert.

Tabelle 9.1: Use Case Vergleich zu bestehendem System

9.4 Zeiterfassungsreport

Die Zeiterfassung erfolgte gemäss Definition von Kapitel Zeiterfassung im Projektplan (Kapitel 7.11). Der Zeitplan konnte erfolgreich eingehalten werden, inklusive allen Meetings mit 2BIT, Feature- und Code-Freeze und Meilensteinen. Das Team konnte ein mehrheitlich stetiges Arbeitstempo von Anfang an beibehalten und kam nie in Verzögerung.

Die gesamten Arbeitszeiten des Teams sind folgende:

- Manu Weber: 240 h 44 min
- Rolf Oberhänsli: 240 h 20 min
- Yael Schärer: 360 h 5 min
- Total: 841 h 20 min

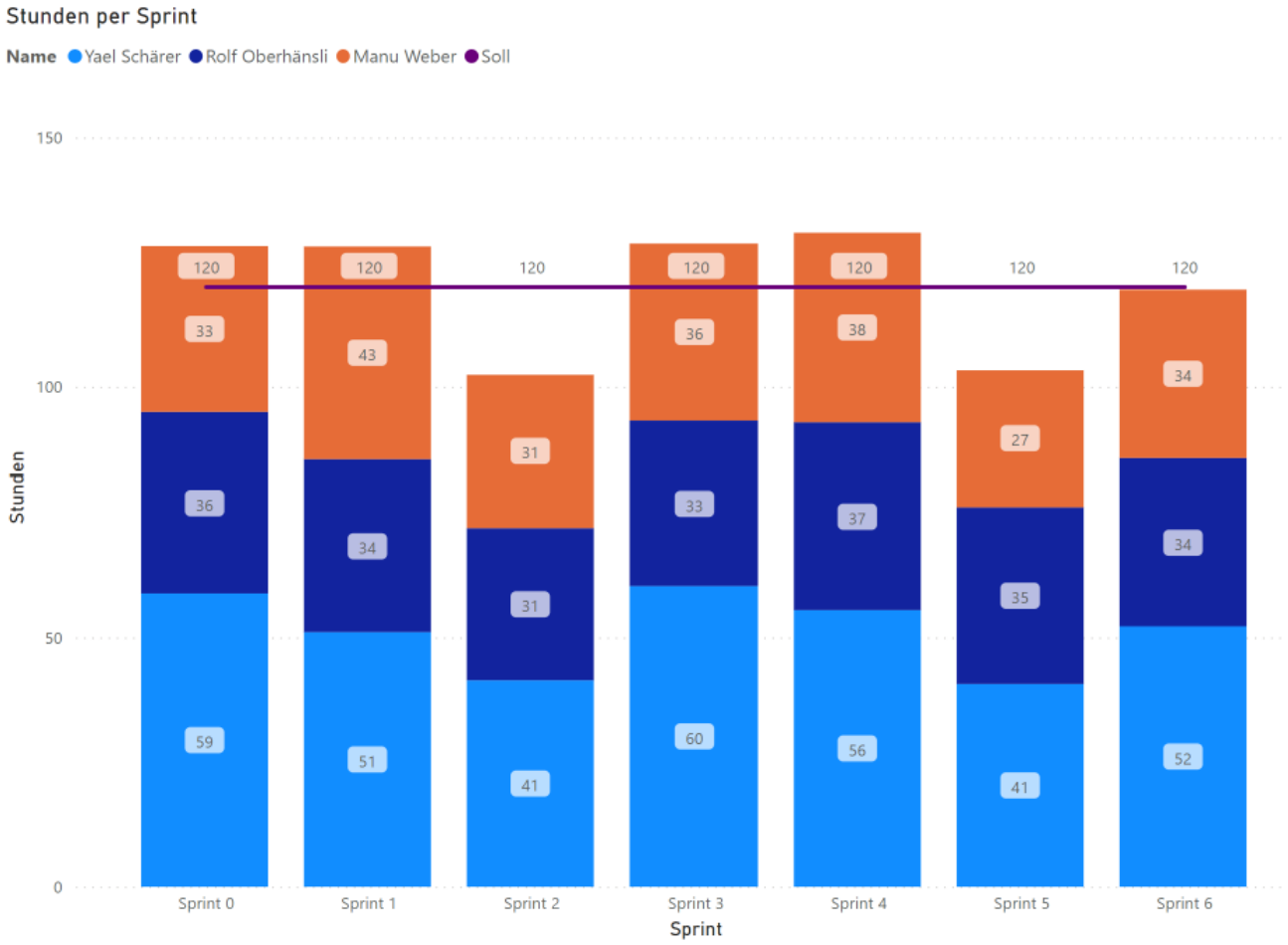


Abbildung 9.6: Zeiterfassung im Gesamten

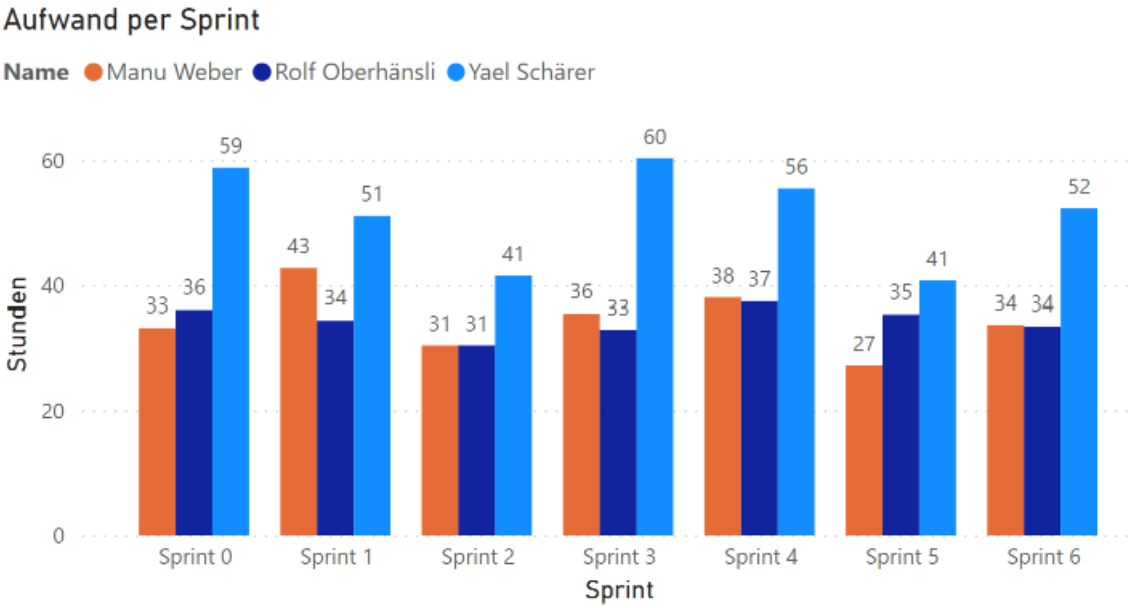


Abbildung 9.7: Zeiterfassung per Sprint

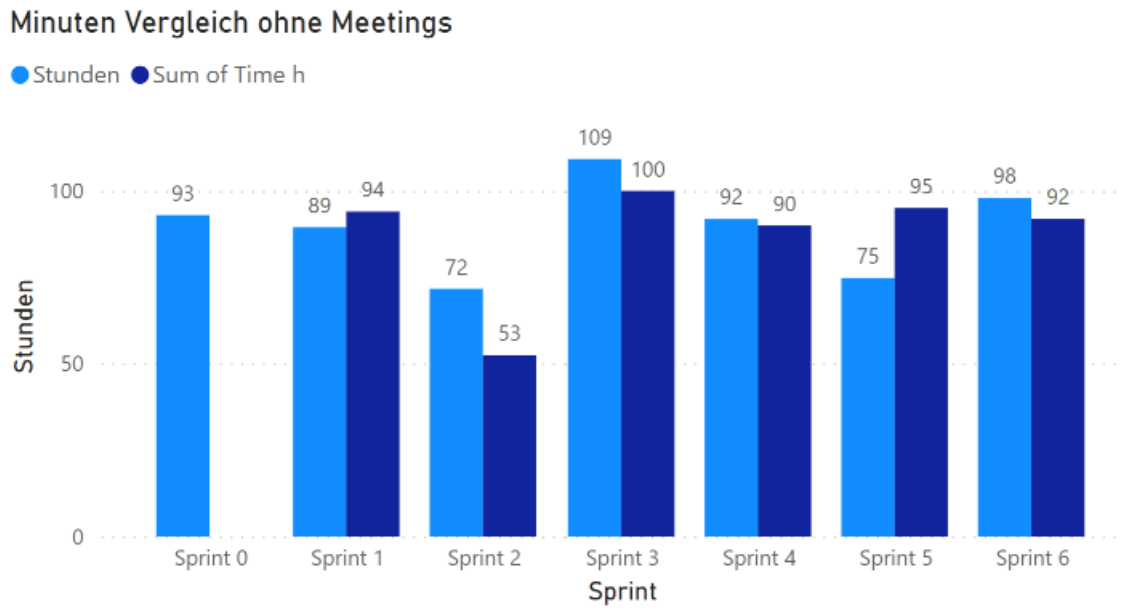


Abbildung 9.8: Zeiterfassung Differenz zu Schätzung

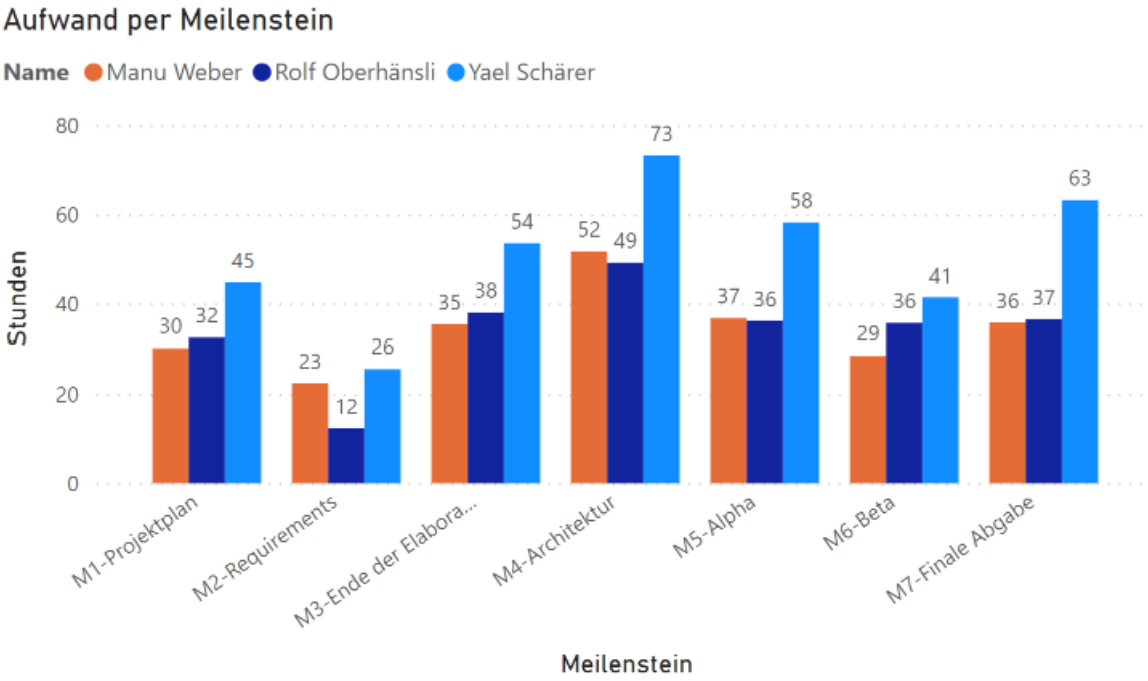


Abbildung 9.9: Zeiterfassung per Meilenstein

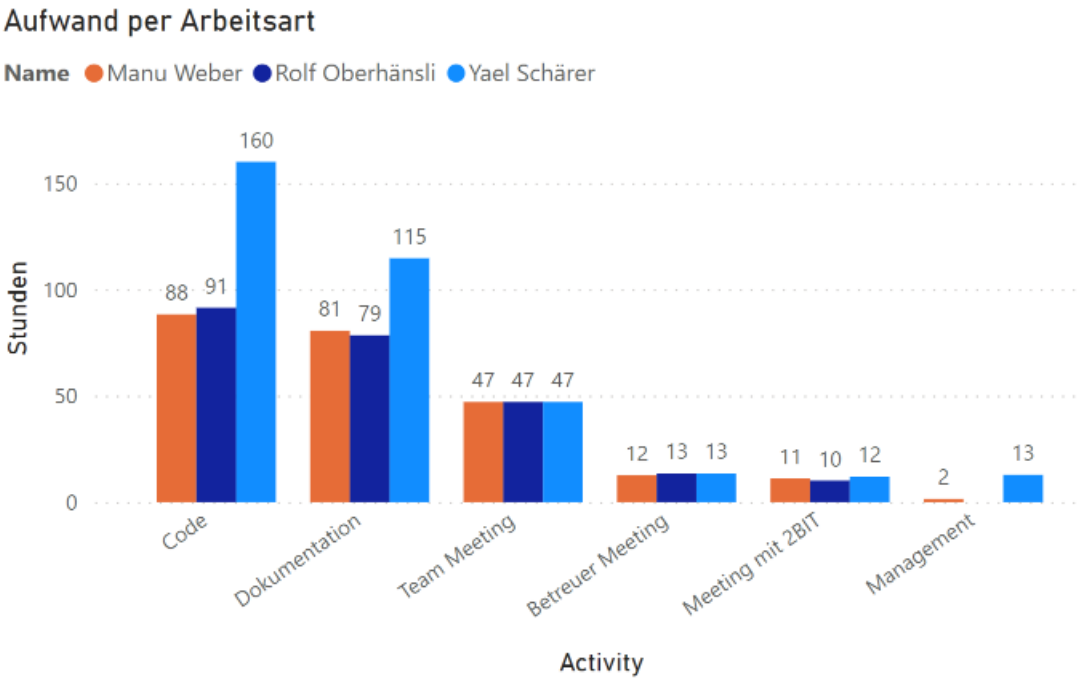


Abbildung 9.10: Zeiterfassung per Aktivitätsart

Kapitel 10

Fazit

10.1 Retrospektive

Produkt Als Minimum Viable Product (MVP) wurde das Umsetzen der Funktionalität des alten Admin-Portals mit stark erhöhter Bedienbarkeit definiert. Das Admin-Portal sollte keine Funktionen durch dieses Projekt verlieren, jedoch für den User einfacher und intuitiver zu bedienen sein. Durch die Usability Tests mit den Endnutzern konnte dies erfolgreich verifiziert werden. Zusätzliche Features konnten erfolgreich eingebaut werden und diese unterstützen den Supportprozess zusätzlich. Die zusätzlichen Features wurden zusammen mit 2BIT priorisiert, um die dringend benötigten Funktionalitäten für den Support umsetzen zu können. Der letzte Usability Test hat gezeigt, dass die Benutzerbasis über die zusätzlichen Möglichkeiten erfreut ist und das Portal als stark verbessert einstuft. Die nicht umgesetzten Projekt- und Produktanforderungen wurden für den Supportprozess als nicht prioritär eingestuft und können zu einem späteren Zeitpunkt von 2BIT umgesetzt werden. Da ein Mitglied des Projektteams weiterhin bei 2BIT arbeiten wird, kann der Ausbau des Admin-Portals ohne Informationsverlust fortgeführt werden.
Einzig der Use Case 'CRUD Custom Package' ist eine Funktionalität, die mit erhöhter Priorität im Support benötigt wird, jedoch im Projekt aufgrund Zeitmangels nicht umgesetzt werden konnte.

Design Die Vorgaben von 2BIT gemäss dem DEsign der neuen Applikation waren minimal gehalten. Das Projektteam hat sich dazu entschlossen, das Design der Applikation stark am bestehenden Admin-Portal zu orientieren. Dies um einen komfortablen Übergang vom alten Portal zum neuen zu ermöglichen. Hierzu wurde ein high-fidelity Wireframe ausgearbeitet und daruaf Usabilitytests mit 2BIT durchgeführt. Dieser Ansatz wurde gewählt um einen professionellen Eindruck zu geben, der dem bestehenden Admin-Portal mindestens ebenbürdig ist.
In Review des Vorgehens, auch mit Input von Experten, kam die Frage auf, ob andere Layoutvarianten getestet wurden. Eine andere Grundsatzlösung, zum Beispiel ganz ohne Listenansichten, wäre eine andere mögliche Lösung für die Bedürfnisse des Supportpersonals gewesen. Obwohl das Endpordukt mit grossem Enthusiasmus von 2BIT aufgenommen wurde, ist es sicherlich von Vorteil in einem zukünftigen Projekt ein offenerer Ansatz in Betracht zu ziehen. Zwei Usabilitytests auf Papier durchzuführen, einer mit einem sehr reduzierten Design, der sich auf die Funktionalitäten und Arbeitsweisen konzentriert und erst ein späterer mit ausgearbeitetem Design könnte ein Vorteil für die Qualität der entstehenden Applikation sein.

Anforderungen Es konnte das Minimal Viable Product und ein Grossteil der zusätzlichen funktionalen Anforderungen erfolgreich umgesetzt werden. Die nicht funktionalen Anforderungen konnten vollständig eingehalten werden. Das Projekt ist auf einem ausgezeichneten Stand für die Übergabe an 2BIT und wurde als äusserst positiv von der Benutzerbasis eingestuft. Die für den Support wichtigen Funktionalitäten können nun über das Admin-Portal eingesehen werden und sind intuitiv zu finden.

Verifikation Nebst dem Austausch mit 2BIT wurden die Lösungsansätze mit verschiedenen Tests verifiziert. Es wurde mit automatisierten Tests gearbeitet, um die funktionale Korrektheit zu garantieren. Zudem haben manuelle System- und Usability Tests den Umgang und die Intuitivität der Software bestätigt.

Usability Tests Die Aufgaben des ersten Usability Tests wurden nicht auf Basis von realen Supporttickets ausgearbeitet, sondern aus den Erkenntnissen der Anforderungsanalyse. Dies beeinflusste die Aufgabenstellungen negativ, da zu technische und genaue Fragen gestellt wurden. Zudem beinhaltete der Usability Test ziemlich viele Aufgaben, was dazu führte, dass die letzten Aufgaben in einer gewissen Hektik durchgeführt wurden. Auch wurde im ersten Usability Test kein Direktvergleich zwischen der neuen und alten Lösung durch die User gemacht.
Dies wurde in zweitem Usability-Test erfolgreich nachgeholt. Die Aufgaben wurden möglichst stark an reale Support-Tickets angelehnt. Dies um die User zu zwingen die Lösung so nahe am normalen Supportprozess wie möglich zu benutzen. Auch wurde der Umfang des Tests reduziert, was merklich angenehmer für die Test-Teilnehmer war und mehr Zeit für Reflexion und Diskussion übrig lies.
Der letzte Usability Test wurde auf Rat vom Betreuer zusätzlich mit einer im Projekt bisher nicht involvierten Person durchgeführt. Dies führte zu neuen Inputs, welche genutzt werden konnten, um die Softwarequalität zu verbessern.

Zusammenarbeit mit 2BIT Es wurde auf einen engen und offenen Austausch mit 2BIT gesetzt, damit die Lösung so gut wie möglich auf die Bedürfnisse der Benutzerbasis angepasst ist und den Ansprüchen von 2BIT bezüglich produktiver Software genügt. Durch die hilfreiche Teilnahme von 2BIT konnte dies erfolgreich umgesetzt werden. Missverständnisse konnten vermieden oder zeitnah aufgeklärt werden. Entscheidungen und Optimierungen konnten umgehend erkannt und gelöst werden. Dies hat das Risiko einer konzeptionell falschen Lösung minimiert und die Qualität der Software erhöht.

Planung Im Verlauf des Projektes wurde viel Zeit in das Projektmanagement und die Planung der Software investiert. Diese Investition hat sich ausbezahlt und die Qualität des Produktes sowie die Zusammenarbeit des Teams konnte kontinuierlich verbessert werden. Als Resultat konnten alle Meilensteine erfolgreich eingehalten werden, es kam zu keinen Verzögerungen und alle geplanten Termine wurden eingehalten. Der Entscheid viel Zeit in das Projektmanagement und die Planung zu investieren basierte darauf, dass das im Projekt entwickelte Produkt schlussendlich im produktiven Betrieb zum Einsatz kommt. Das neue Admin-Portal hat die Ansprüche an Qualität mehr als erfüllt und kann somit ohne Bedenken in den produktiven Betrieb übergeben werden.

Zeitschätzung Das Projektteam konnte die Dokumentationsaufgaben zeitlich akkurat einschätzen. Jedoch gab es zu Beginn starke Abweichungen zwischen den geschätzten und tatsächlichen Werten der Entwicklungsaufgaben. Vor allem zu Beginn der Konstruktionsphase wurde der Aufwand oft stark unterschätzt und nicht alle Aufgaben konnten in einem Sprint umgesetzt werden. In den letzten Sprints der Konstruktionsphase wurden die Schätzungen immer besser und die Sprints konnten präziser geplant werden.

Team Das Team hat sehr gut zusammengearbeitet und konnte das Projekt ohne Probleme abschliessen. Es wurde viel Wert auf die offene Kommunikation und harmonischen Umgang gelegt. Die Stärken der Teammitglieder wurden gezielt eingesetzt und konnten durch intensive Reviews miteinander geteilt werden. Jedes einzelne Teammitglied hatte während der Projektdauer immer 100% gegeben. Die gute Zusammenarbeit und hohe Motivation haben zu einem gelungenen Projekt geführt.

10.2 Ausblick

Obwohl viele Anforderungen umgesetzt werden konnten, blieben einige übrig, welche die Unterstützung des Supportprozesses durch das Admin-Portal noch weiter verbessern können. Diese Anforderungen wurden im Items Board des Admin-Portal-Projektes auf Azure DevOps gepflegt, geschätzt und an 2BIT übergeben. Die folgende Liste beinhaltet die Use Cases, die während des Projektverlaufs nicht umgesetzt wurden.

- Erstellen und Bearbeiten von Custom-Subscription-Packages
- Support-Zugriff auf Kunden-Tenants
- SendGrid-Integration
- Zurzeit aktive 2getHR-User
- Felctra-Integration
- Reports und Statistiken zu 2getHR
- Mitarbeiter verifizieren
- Feiertage
- Breadcrumbs

Das Projekt hat den grossen Vorteil, dass ein Teammitglied weiterhin bei 2BIT arbeiten wird. So geht kein Wissen verloren und die weiteren Anforderungen können direkt von Entwicklern des Produktes implementiert werden.

Literaturverzeichnis

[12f] The twelve-factor app.

[Cyp] Cypress.

[iso] Iso - iso/iec 25010:2011 - systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models.

[loa] Loadninja.

[Mar09] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson, 2009.

[mit] mitmproxy.

[OWA] Owasp top ten.

[Rea] Vue.js vs react.

[RUP] Rational unified process.

[Scr] Scrum.

[uc] Craig larman. applying evolutionary use cases.

Glossar

.NET C# Framework. 15, 20, 89, 90

API Application Programming Interface. Äussere Schnittstelle einer Software, womit andere Programme interagieren können.. 92

Axios JavaScipt Library für API Funktionalitäten.. 152

BA Bachelorarbeit. 133

Bexio Bexio ist eine Schweizer Business Software, die mit dem Modul Bexio in 2getHR eingebunden werden kann. Die Verbindung erlaubt ein teilen und synchronisieren von Userdaten.. 25, 73, 115

CD Continuous Deployment. 62, 101, 104

CI Continuous Integration. 62, 101, 104

coverlet .NET Code Coverage Command Line Tool. 105

CRUD Create, Read, Update und Delete. 42, 51, 151, 156, 161

Design Contraint Eine Grenze der Entscheidungsfreiheit. Beispiel wäre Entwickler Wissen. Entwickler müssen sich in Unbekannte Technologien einarbeiten, was Zeit kostet und in fehlerhaften Resultaten enden kann.. 83

DoD Definition of Done. 102

DTO Data Transfer Object. 21, 63

Figma Kollaborative Software zum Erstellen von Prototypen im Bereich des UX bzw. UI Design. Siehe https://www.figma.com/de/. 87

Flectra Ein Geschäftsinformationssystem. 2BIT verwaltet mit diesem Tool Supporttickets.. 24

JSON Java Script Object Notation.. 69, 154

LINQ .NET Library für OR-Mapping.. 65, 69, 153

MitM Man in the Middle. 55, 152

MVP Minimal Viable Product. 30, 92, 94, 96, 129, 136, 161

NFR Nicht-funktionale Anforderungen. 52, 58, 61, 63–65, 67, 68, 108, 151–154, 168

nuget Package Manager von .NET.. 89

Objekt Oberbegriff von Tenants (inkl. Treuhänder), Mitarbeiter und Mitarbeitergespräch. 33, 35, 40, 44, 167

OST OST - Ostschweizer Fachhochschule. 142

PostGres Opensource Datenbank.. 24

Progressive Web Application Eine Webapplikation die sich wie eine native Applikation verhält. Sie basieren wie normale Webseiten auf HTML, CSS und JavaScript, bieten aber Fähigkeiten wie Offline Modus, Single Page Application und weiteres wie native Aps.. 15

pycobertura Command Line Tool für xml Formatierung. 105

SA Studienarbeit. 133

SPA Single Page Applikation. 16, 36, 79, 90

SW Semesterwoche. 135, 136

TDD Test-Driven Developement. 140

UI User Interface. 122, 124

Yarn React Dependency Management Tool.. 17, 89

Abbildungsverzeichnis

Tabellenverzeichnis

Teil VI

Anhang

Kapitel 11

Anhang

11.1 Historie der Systemtests

Wann	Wer	Ref.	Effektives Resultat	Massnahmen
30.11.	Yael Schärer	1	1. Login erfolgreich und wurde auf Tenantliste weitergeleitet. Logout erfolgreich und wurde auf Microsoft Login Seite weitergeleitet. 2. Kann nicht einloggen, Konto wurde nicht gefunden. 3. Wird die Seite nicht per 'https://a-2gethr-admin.azurewebsites.net/' aufgerufen wird ein HTTP 404 Error angezeigt.	HTTP 404 Error muss korrigiert werden.
30.11.	Yael Schärer	2	1. Nach erfolgtem Login wird Tenantliste angezeigt. 2. Alle Listen werden korrekt angezeigt. 3. Paginierung funktioniert wie gewünscht. Einziges Problem ist, dass Filter setzen den Seitenindex nicht zurücksetzt. 4. Filter Setzen löst korrekter API-Call aus und liefert gefilterte Listenelemente. Filter werden erfolgreich in den Redux Store gespeichert. 5. Navigation zwischen Listen ist erfolgreich und gesetzte Filter und Paginierung bleibt erhalten. 6. Listenkonfiguration wird erfolgreich eingesetzt und in Redux Store gespeichert. Über Session hinweg wird die Konfiguration nicht gespeichert. 7. Listenkonfiguration wird nicht wieder hergestellt. 8. Listenkonfiguration wird nicht wieder hergestellt.	Das Setzen von Filtern muss Page Index zurücksetzen. Funktioniert in der Logik aber im UI wird der Wert nicht auf 1 zurückgesetzt. Konfiguration der Tabelle muss in Lokalem Speicher abgelegt werden.
30.11.	Yael Schärer	3	Momentan sind keine zu Loggenden Useraktionen möglich.	Aktionen müssen protokolliert werden, um diesen UC zu testen.

30.11.	Yael Schärer	4	1. Tenantliste wird dargestellt. 2. Mitarbeiterliste wird dargestellt. 3. Treuhänderliste wird dargestellt. 4. Mitarbeiterliste wird dargestellt. 5. Tenantliste wird dargestellt.	UC funktioniert einwandfrei.

Tabelle 11.1: Systemtest 30.11.

Wann	**Wer**	**Ref.**	**Effektives Resultat**	**Massnahmen**
14.12.	Yael Schärer	1	1. Login erfolgreich und wurde auf Tenantliste weitergeleitet. Logout erfolgreich und wurde auf Microsoft Login Seite weitergeleitet. 2. Kann nicht einloggen, Konto wurde nicht gefunden. 3. Wird die Seite nicht per 'https://a-2gethr-admin.azurewebsites.net/' aufgerufen wird ein HTTP 404 Error angezeigt.	UC funktioniert einwandfrei. HTTP 404 Error wird in development Umgebung nicht korrigiert, erst in 2getHR Umgebung.

14.12.	Yael Schärer	2	1. Nach erfolgtem Login wird Tenantliste angezeigt. 2. Alle Listen werden korrekt angezeigt. 3. Pagination funktioniert wie gewünscht. 4. Filter Setzen löst korrekter API-Call aus und liefert gefilterte Listenelemente. Filter werden erfolgreich in den Redux Store gespeichert. 5. Navigation zwischen Listen ist erfolgreich und gesetzte Filter und Pagination bleibt erhalten. 6. Listenkonfiguration wird erfolgreich eingesetzt und in Redux Store gespeichert. Konfiguration wird zusätzlich in Lokalem Speicher persistiert um eine Session übergreiffende Einstellung zu ermöglichen. 7. Listenkonfigruation wird korrekt hergestellt. 8. Listenkonfigruation wird korrekt hergestellt.	UC funktioniert einwandfrei.
14.12.	Yael Schärer	3	1. Nach erfolgtem Login wird Tenantliste angezeigt. 2. Tenantliste wird angezeigt. 3. Update E-Mail und Rolle von Mitarbeiter sowie Treuhänderstatus von Tenant funktioniert einwandfrei. Änderungen sind in Datenbank persisitert und werden auch bei Reload der Seite korrekt angezeigt. 4. Momentan keine Logfiles vorhanden. Backend erhält korrekte Logeinträge von Frontend und schreibt diese auf den Standard Out Stream.	Logs werden auf Standard Out Stream im Backend geschrieben. Müssen auf Azure Storage persistiert werden.

14.12.	Yael Schärer	4	1. Tenantliste wird dargestellt. 2. Mitarbeiterliste wird dargestellt. 3. Treuhänderliste wird dargestellt. 4. Mitarbeiterliste wird dargestellt. 5. Tenantliste wird dargestellt.	UC funktioniert einwandfrei.
14.12.	Yael Schärer	5	1. Tenantliste wird dargestellt. 2. Tenant-Detailansicht wird korrekt geladen und zeigt korrekte Daten des Tenants an gemäss UC. 3. Treuhänderstatus wird gemäss Userinput (Select Liste) angepasst. Loadingindicator wird auf dem Feld angezeigt wärend Updatevorgang läuft. Tenant wird neu gesetzt wenn Updatevorgang erfolgreich durchgeführt wurde. Fehlermeldung wird angezeigt wenn Updatevorgang fehlgeschlagen ist. 4. Wird auf Abfalleimer Icon gedrückt, wird ein Dialog geöffnet mit Warnmeldung. Wird nicht 'LÖSCHEN' eingegeben und auf LÖSCHEN Button gedrückt wird eine Warnmeldung angezegit. Wird korrekter Input eingegeben wird lösch Call ans Backend geschickt. Tenant löschen ist momentan nicht möglich und muss auskommentiert werden. Wird normal durchgeführt sobald Code auf 2getHR Umgebung migriert wird. Wärend Löschvorgang wird ein Loading Indikator angezeigt. War Vorgang erfolgreich wird eine Erfolgsmeldung angezeigt und der User nach 2 Sekunden auf vorherige Ansicht zurückgeleitet. Ist Löschvorgang misslungen wird eine entsprechende Fehlermeldung angezeigt.	Löschvorgang ist momentan auskommentiert im Backend. Wird korrigiert sobald Code auf 2getHR Umgebung migriert wird.

14.12.	Yael Schärer	6	1. Mitarbeiterliste wird dargestellt. 2. Mitarbeiter-Detailansicht wird korrekt geladen und zeigt korrekte Daten des Mitarbeiters an gemäss UC. 3. Rolle wird gemäss Userinput (Select Liste) angepasst. Loadingindicator wird auf dem Feld angezeigt während Updatevorgang läuft. Mitarbeiter wird neu gesetzt, wenn Updatevorgang erfolgreich durchgeführt wurde. Fehlermeldung wird angezeigt, wenn Updatevorgang fehlgeschlagen ist. 4. E-Mail wird gemäss Userinput (Textfeld) angepasst. Loadingindikator wird angezeigt. Mitarbeiter wird neu gesetzt wenn Updatevorgang erfolgreich war. Ist E-Mail Adresse im System bereits vorhanden wird eine Warnmeldung mit entsprechendem Dialog angezeigt. Backend Funktionalität wirft momentan Fehlermeldung welche im Frontend ignoriert wird. Wird korrigiert sobald Code in 2getHR Umgebung migriert wird.	E-Mail Update Funktionalität wirft momentan Fehlermeldung, die ignoriert wird. Wird korrigiert sobald Code in 2getHR Umgebung migriert wird.

14.12.	Yael Schärer	7	1. Tenant-Listenansicht wird korrekt angezeigt. 2. Tenant-Detail wird korrekt angezeigt. 3. Mitarbeiter-Liste in Tenant-Detail wird korrekt angezeigt. 4. Mitarbeiter-Detail wird korrekt angezeigt. 5. Mitarbeitergespräch-Liste in Mitarbeiter-Detail wird korrekt angezeigt. 6. Mitarbeitergespräch-Detail wird korrekt angezeigt. 7. Tenant-Detail wird korrekt angezeigt. 8. Mitarbeitergespräch-Liste in Tenant-Detail wird korrekt angezeigt. 9. Mitarbeitergespräch-Detail wird korrekt angezeigt. 10. Mitarbeiter-Detail wird korrekt angezeigt. 11. Tenant-Detail wird korrekt angezeigt. 12. Treuhänder-Liste in Tenant-Detail wird korrekt angezeigt. 13. Tenant-Detail wird korrekt angezeigt.	UC funktioniert einwandfrei.

14.12.	Yael Schärer	8	1. Mitarbeitergespräch-Listenansicht wird dargestellt. 2. Mitarbeitergespräch-Detailansicht wird korrekt geladen und zeigt korrekte Daten des Mitarbeitergespräch an gemäss UC. 3. Wird auf Abfalleimer Icon gedrückt, wird ein Dialog geöffnet mit Warnmeldung. Wird nicht 'LÖSCHEN' eingegeben und auf LÖSCHEN Button gedrückt wird eine Warnmeldung angezeigt. Wird korrekter Input eingegeben wird der Mitarbeiter gelöscht. Während des Löschvorganges wird ein Loading Indikator angezeigt. War Vorgang erfolgreich wird eine Erfolgsmeldung angezeigt und der User nach 2 Sekunden auf vorherige Ansicht zurückgeleitet. Ist Löschvorgang misslungen wird eine entsprechende Fehlermeldung angezeigt.	UC funktioniert einwandfrei.

14.12.	Yael Schärer	10	1. Mitarbeiter-Detailansicht wird korrekt geladen. 2. Liste der gesetzten Referenzen wird angezeigt. Löschfunktionalität wird nicht eingeblendet. 3. Der normale Löschdialog wird angezeigt mit einer zusätzlichen Information, dass Mitarbeiter keine Referenzen hat. 4. Mitarbeiter kann nicht gelöscht werden und das Textfeld zeigt eine Fehlermeldung. 5. Wird korrekter Input eingegeben wird der Mitarbeiter gelöscht. Während Löschvorgang wird ein Loading Indikator angezeigt. War Vorgang erfolgreich wird eine Erfolgsmeldung angezeigt und der User nach 2 Sekunden auf vorherige Ansicht zurückgeleitet. Ist Löschvorgang misslungen wird eine entsprechende Fehlermeldung angezeigt.	UC funktioniert einwandfrei.

4.12.	Yael Schärer	10	1. Tenant-Listenansicht wird angezeigt. 2. Suche wurde nach 500ms ausgelöst und Resultat entsprechend angezeigt gemäss UC Spezifikation angezeigt. 3. Mitarbeiter-Detail wird korrekt und mit richtigem Mitarbeiter angezeigt. 4. Mitarbeiter-Detail wird korrekt und mit richtigem Mitarbeiter angezeigt. Mitarbeiter wurde erfolgreich ausgetauscht. 5. Tenant-Detail wird korrekt und mit richtigem Tenant angezeigt. 6. Mitarbeitergespräch-Detail wird korrekt und mit richtigem Mitarbeitergespräch angezeigt. 7. Input wird gelöscht und Suchresultat wird nicht mehr angezeigt. 8. Suche wurde vor 500ms Timeout ausgelöst und Resultat korrekt angezeigt. 9. Suchresultat wird nicht mehr angezeigt. Input bleibt erhalten.	UC funktioniert einwandfrei.

Tabelle 11.2: Systemtest 14.12.2022

11.2 Historie der Usability-Tests

Durchführung vom Freitag, 28.10.2022 Die Usability Tests wurden je mit einem Developer und einem Support durchgeführt (siehe Kapitel 2.5).

Wer	**Ref.**	**Effektives Resultat**	**Geplante Massnahmen**
Developer	1	Developer würde über die Suche gehen, hat es aber auch in der Liste gefunden. Feld Einladungsstatus wurde korrekt gefunden. Developer fand es einfach und ist sich sicher das richtige gefunden zu haben.	Priorisierung der Suche nach dem Alpha-Release neu evaluieren. In der Listenansicht Mitarbeiter modellieren, (nicht Personen). Tenant direkt in der Person (neu Mitarbeiter) als Feld anzeigen um Personen als Mitarbeiter Tenants zuweisen zu können.
Developer	2	Developer fand das korrekte Feld. Developer ist sich sicher, das richtige Feld gefunden zu haben.	-

Developer	3	Developer war sich nicht sicher, was er genau machen sollte. Developer meint, er würde bei Bexio Problemen direkt auf die Datenbank gehen.	-
Developer	4	Developer konnte den Supportfall lösen, hat jedoch nicht auf Anhieb bemerkt was genau von ihm verlangt wird. Developer ist verwirrt, weil eigentlich alle Informationen im Test bereits enthalten sind. Developer wurde darauf hingewiesen, dass er den Tenant, für welcher er den Zugriff benötigt, finden muss. Developer fand die verlinkten Objekte und konnte die Informationen einsehen.	Usability-Test umschreiben / klarer formulieren
Developer	5	Developer löste den Test wie erwartet und fühlt sich sicher. Developer hat nachgefragt, wie es mit gleichzeitigem Editiermodus bei mehreren Felder auf der gleichen Ansicht aussieht (Multiuserfähigkeit).	Bevor geänderte Formulardaten übermittelt werden, wird geprüft ob die ursprünglichen Daten des Formulars vom Zustand auf der Datenbank abweichen. Ist dies der Fall, darf die Änderung nicht übermittelt werden. Der User wird entsprechend informiert und zur Aktualisierung der Daten gezwungen.
Developer	6	Developer fand das Verifizierungsfeld wie erwartet. Developer hat nachgefragt, ob ein Treuhänder ünverifiziert"werden kann. → Ja, wird so implementiert. Developer ist damit einverstanden.	-
Developer	7	Developer löste den Test wie erwartet und fühlt sich sicher.	-
Developer	8	Developer löste den Test wie erwartet und fühlt sich sicher.	-
Developer	9	Developer löste den Test wie erwartet und fühlt sich sicher.	-
Support	1	Support hat auf den ersten Anlauf etwas Mühe das richtige Feld zu finden auf der Detailansicht. Support findet das richtige Feld anschliessend und ist sich sicher den korrekten Wert gefunden zu haben. Der Wunsch von Developer Mitarbeiter anstelle von Personen zu modellieren, wurde mit Support besprochen. Support stimmt Developer zu.	Anpassungen nicht nötig. Erste Verwirrung kam von Funktionalität von vorherigem Portal. In den folgenden Tests hatte Support weit weniger Mühe, Felder zu identifizieren.

Support	2	Support würde die Suche einsetzen und nicht die Listenansicht verwenden. Support löst die Aufgabe korrekt und fühlt sich sicher.	Priorisierung der Suche nach dem Alpha-Release neu evaluieren.
Support	3	Support findet das von uns angedachte Feld, war jedoch etwas verwirrt, da sie nicht wusste was sie mit dieser Information anfangen soll.	Wird bei der nächsten Durchführung verifiziert.
Support	4	Support löste den Test problemlos und fühlt sich sicher.	-
Support	5	Support löste den Test problemlos und fühlt sich sicher.	-
Support	6	Support löste den Test problemlos und fühlt sich sicher.	-
Support	7	Support löste den Test problemlos. Support ging zuerst über die Tenants Liste, hat sich aber dann umentschieden und ging über die Suche Support hat sich gefragt, was passiert, wenn sie über Tenant navigiert? Support wurde über die verschiedenen Wege aufgeklärt. Support hat bestimmt, dass sie grundsätzlich über Tenant oder Mitarbeiter geht und die Mitarbeitergesprächsliste nicht nötig sei. Sie meinte, so fühlt sie sich sicherer, das richtige Gespräch gefunden zu haben.	Keine geplante Massnahme, da das Projektteam zum jetztigen Zeitpunkt davon ausgeht, dass die Listenansicht der Mitarbeitergespräche einen Mehrwert bietet - die User sich aber zuerst an die zusätzliche Möglichkeit gewöhnen müssen.
Support	8	Support löste den Test problemlos und fühlt sich sicher. Support wünscht sich, Kunden würden den Tenant selbst löschen. Oft seien Tenant-Namen in solchen Fällen nicht eindeutig. Support hat erwähnt, dass hier die E-Mail-Adresse und der Kontakt von Tenants äusserst wichtig sind.	Anpassungen siehe Design Entscheide - Darstellung Datenfelder 5.4
Support	9	Support hat Crossnavigation sehr schnell gefunden und gebraucht. Support ist sich nicht sicher und versteht den Lösch-Mechanismus nicht sofort. Es wurde erläutert wie der Mechanismus mit dem Löschen des Users funktioniert. Anschliessend konnte Support die Aufgabe lösen bzw. nachvollziehen. Alternative Buttons und Funktionalitäten wurden mit Support besprochen.	Da die Funktionalität, die für den Test benötigt wird, nicht vollständig in den Wireframes abgebildet ist, ist dies ein nicht eindeutiges Ergebnis und es werden keine Massnahmen geplant. Sobald die Funktionalität umgesetzt ist, wird verifiziert, dass das Problem behoben wurde.

Support	10	Ist sich nicht ganz sicher, findet es nicht auf Anhieb.	Da die Funktionalität, die für den Test benötigt wird, nicht vollständig in den Wireframes abgebildet ist, ist dies ein nicht eindeutiges Ergebnis und es werden keine Massnahmen geplant. Sobald die Funktionalität umgesetzt ist, wird verifiziert, dass das Problem behoben wurde.
Support	11	Ist sich zuerst nicht sicher, da Sie den Button "Listenkonfiguration"nicht sofort fand. Support konnte den Task anschliessend lösen.	Anpassungen siehe Design Entscheide - Zugang Listenkonfiguration 5.4

Tabelle 11.3: Historie Usability Test

Allgemeine Kommentare zur Durchführung vom Freitag, 28.10.2022 mit Developer

1. Overlay: Wenn man ausserhalb des Overlays drückt, soll man automatisch auf die zuvor geöffnete Ansicht kommen.
2. Developer hat das Suchfeld sehr oft benutzt. Deshalb sollte Suchfeld soll deshalb immer sichtbar sein. Zudem muss die Priorisierung des Suchfeldes durch 2BIT in Frage gestellt / neu evaluiert werden.
3. Farben: Weiss auf Gelb ist nicht sehr geeinget. → Farbgebung der Buttons überabrieten. 5.4
4. Developer meint im Allgemein sieht das Admin-Portal cool aus.
5. Eine Suche innerhalb der Listenansicht wäre eventuell auch angenehm, je nachdem wie die Suche selbst implementiert ist. Wenn diese aber alle Objekte durchsucht ist dies nicht notwendig.
6. Developer ist sich noch unsicher bezüglich der Navigation mit mehreren Overlay-Fenstern übereinander. → Muss im nächsten Usability Tests nochmals genauer betrachtet werden.
7. Developer fühlt sich grundsätzlich wohl mit der Navigation.

Allgemeine Kommentare zur Durchführung vom Freitag, 28.10.2022 mit Support

1. Support fühlt sich grundsätzlich wohl mit der vorgestellten Lösung.
2. Support findet die Listenansichten sehr angenehm und hat sich schnell an das Suchfeld gewöhnt und dieses oft benutzt.
3. Es soll in der Listen- sowie der Detailansicht direkt ersichtlich sein, ob ein Tenant ein Treuhänder ist oder nicht. 5.4
4. Support wird für uns in den kommenden Wochen Supportfälle dokumentieren, damit wir für die nächsten Usability Test konkrete Beispiele haben und näher an der Realität testen können.

Durchführung vom Freitag, 25.11.2022 Die Usability Tests wurden je mit einem Developer und einem Support durchgeführt (siehe Kapitel 2.5).

Wer	Ref.	Effektives Resultat	Geplante Massnahmen
Developer	1	Mitarbeiter Liste: Tenantfilter wurde als Mitarbeiterfilter interpretiert. Einladungsstatus wurde gefunden (Invited). Filterfeld in Mitarbeiter wird intuitiv als Name des MA und nicht als Name des Tenant verstanden.	Anpassungen siehe Design Entscheide - Mitarbeiter-Listenansicht 5.4.
Developer	2	Überlegt sich, ob die Lösung im Portal überhaupt gefunden werden kann. Verstand nicht sofort, dass er sich Ïst Einheit Vorsteheränschauen muss, fand es aber anschliessend. Developer viel auf, dass ïsActiveïn der Listenansicht der Mitarbeiter noch nicht abgefüllt ist (war bekannt, Erklärung wurde gegeben).	Feld wird umbenannt in Abteilungsleiter und isActive wird korrigert.
Developer	3	Startschwierigkeiten: Verstand nicht wie er den Fall lösen kann. Hilfestellung wurde gegeben. Ging über Tenantansicht und suchte nach dem Treuhänder Status.	-
Developer	4	Test konnte ohne Probleme gelöst werden, fonjallaz97 ist Admin.	-
Developer	5	Test konnte ohne Probleme gelöst werden. Detail im UX: Es ist nicht direkt ersichtlich, zwischen welchen Spalten eine Spalte nach dem Verschieben liegt.	Out-of-the-box Funktionalität der Tabelle kann leider nicht angepasst werden. Entsprechende Erklärung wurde gegeben.
Developer	6	Test konnte ohne Probleme gelöst werden, Daten von Home wurden gefunden.	-

Support	1	Startschwierigkeiten aufgrund anderem Aufbau als bestehende Lösung, Hilfestellung wurde gegeben. Anschliessend filterte der Support zuerst die Tenants, navigierte anschliessend auf Mitarbeiter. Fand den Mitarbeiter 'ext1 employee' und den Einladungsstatus. Detailview, resp. Link auf Liste aller Mitarbeteiter des Tenant fehlt (→ wird mit Detailview kommen). Fragte sich auf der Listenansicht, ob die Filter getrennt sind oder nicht (Erklärung wurde gegeben).	-
Support	2	Test wurde richtig verstanden. Folgende Anmerkungen wurden gemacht: • Warum sucht man auf der Mitarbeiter Seite mit dem Tenant? • Abteilungsleiter nicht ïst Organisation Vorsteher". • Mit der Erfahrung des Supports war aufgrund der Problemstellung, von Anfang an klar was die Ursache sein muss. • Filter nach Tenantname auf der Liste der Mitarbeiter ist nicht intuitiv. • 'Ist Einheit Vorsteher' ist nicht wirklich eine gute Übersetzung? → Ist Abteilungsleiter? • Horizontaler Scrollbar ist standardmässig nicht ersichtlich → somit sieht man nicht alle benötigen Spalten auf Anhieb	Anpassungen am Design siehe Design Entscheide werden entsprechend gemacht, sowie Tabellen Implementation 5.4 und Mitarbeiter Listenansicht 5.4 angepasst.
Support	3	Tenantfilter auf Mitarbeiter macht wenig Sinn, Valentina erwartet immer Mitarbeiterfilter. Treuhänder Status wurde gefunden. Für diesen Support-Fall wird eine Cross-Navigation gewünscht (Detail-View)	Test, mit einem Tenant welcher den Status "Requestet"hat, anpassen. Anpassungen am Design siehe Design Entscheide - Spaltenbezeichnungen 5.4

Support	4	billing bing bing wurde über Mitarbeiter gesucht. Test konnte ohne Probleme gelöst werden.	-
Support	5	Visuell wäre es schöner, wenn die Spalte zwischen bestehende Spalten eingefügt werden kann. Beim verschieben der Spalten wird erwartet, dass dies über den gesamten Eintrag möglich ist (dies sei aber Gewöhnungssache). Zudem ist die Lösung im 2getHR analog.	Spaltenkonfiguration kann nicht angepasst werden. Erklärung wurde gegeben.
Support	6	Tenant Home wurde ohne Probleme gefunden.	-

Tabelle 11.4: Historie Usability Test vom Freitag, 25.11.2022

Kommentare bezüglich dem Vergleich der alten zur neuen Lösung vom 25.11.2022 mit Developer und Support

1. Managerfeld ist auf altem System nicht vorhanden.
2. Übersicht ist in neuer Lösung grundsätzlich besser.
3. Die Navigation erleichtert das Arbeiten.
4. Grundsätzlich ist die neue Lösung viel schneller, dies dank besserer Zugänglichkeit.
5. Allgemeines Feeling ist gut, der grösste Vorteil sind die Ansichten der verschiedenen Entitäten.

Allgemeine Kommentare zur Durchführung vom Freitag, 25.11.2022 mit Developer

1. Durchgeführt auf Chrome.
2. Login konnte ohne Probleme durchgeführt werden.
3. Kompakte Liste findet er sehr schön.
4. Mitarbeitergespräche: eventuell E-Mail von Mitarbeiter auch anzeigen.
5. Neue Anforderung "Feiertage anzeigen und eintragen".
6. Suchfeld wird vermisst.

Allgemeine Kommentare zur Durchführung vom Freitag, 25.11.2022 mit Support

1. Durchgeführt auf Edge.
2. Login hat funktioniert.
3. Filter ist besser verfügbar.
4. Wünschenswerte Filter auf Tenant: Name und Kontakt.
5. Wünschenswerte Filter auf Mitarbeiter: Name und E-Mail-Adresse.

6. Scrollbare View innerhalb einer scrollbaren View ist suboptimal. Horizontales Scrollen ist unschön - Anpassungen siehe Design Entscheide 5.4.

7. Aufruf der Filter ist mühsam, Suche wird stark gewünscht.

Durchführung vom Freitag, 09.12.2022 Die Usabilitytests wurden je mit einem Developer und einem Supporter durchgeführt (siehe Kapitel 2.5). Der Developer hat bevor keine Einsicht in das Projekt gehabt, sprich für diese Person ist es ein Pilottest.

Wer	**Ref.**	**Effektives Resultat**	**Geplante Massnahmen**
Support	1	Konnte den Fall Problemlos lösen. Ging über die Suche auf Treuhänder und fand dort über die Detailansicht den Mitarbeiter. Fühlte sich gut an.	-
Support	2	Hat das Problem extrem schnell gelöst. Ging über Tenantfilter in die Tenantdetail-View und hat dort den Mitarbeiter in den Gelinkten Objekten gefunden. Über Suche funktioniert es auch gut, jedoch geht sie lieber über den Tenant zum Mitarbeiter, damit sie sich ganz sicher sein kann, den richtigen erwischt zu haben.	-
Support	3	Konnte Problem gut lösen. War positiv überrascht das die Info vorhanden ist, bis anhin wurde keine Unterscheidung zwischen None, Requested und Verified angeboten.	-
Support	4	Konnte ohne Probleme gelöst werden.	-
Support	5	Wurde übersprungen, da dies schon bekannt war.	-
Support	6	Konnte ohne Probleme gelöscht werden. Bei den Mitarbeitergesprächen fehlt der Typ (ID braucht es nicht).	Das Datenfeld wird entsprechend mit der Bezeichnung des Typs und nicht der ID angegeben.
Aussen-stehender	1	Kann sich intuitiv auf dem Tool bewegen und findet den Mitarbeiter schnell. Crossnavigation wurde intuitiv erkannt. Navigiert über die Suche auf den Tenant und von da auf den Mitarbeiter (Detailliste Mitarbeiter).	-
Aussen-stehender	2	Konnte den Fall ohne Probleme lösen. Inkonsistente Testdaten führen zu Diskussionen, da es eigentlich ohne Abteilung nicht möglich wäre Abteilungsleiter zu sein.	-

Aussen-stehender	3	Symbol auf Updateablefield sollte eine Disk sein und kein Hacken und Kreuz. Diese Symbole wurden als Parameter interpretiert (werden als Status Treuhänder - nicht Treuhänder angesehen).	Anpassungen siehe Design Entscheide - Darstellung Datenfelder 5.4.
Aussen-stehender	4	Konnte ohne Probleme gelöst werden.	-
Aussen-stehender	5	Konnte ohne Probleme gemacht werden.	-
Aussen-stehender	6	Konnte ohne Probleme gelöst werden.	-

Tabelle 11.5: Historie Usability Test vom Freitag, 09.12.2022

Allgemeine Kommentare zur Durchführung vom Freitag, 09.12.2022 mit Aussenstehender

1. Coole Sachen
2. Neues Feature erkannt: Path anzeigen und rückwärts Navigation explizit anbieten
3. Vergleich alte und neue Lösung: neue Lösung ist viel komfortabler und intuitiver, alte Lösung ist umständlich und bietet das meiste gar nicht an.

Allgemeine Kommentare zur Durchführung vom Freitag, 09.12.2022 mit Support

1. Hat Suche nicht direkt gefunden
2. Hat Gefragt ob man auch innerhalb der Detailansicht in den Listenansichten suchen kann.
3. Standardmässig ein Gelinktes Objekt anzeigen (Tenantdetail Ansicht soll Standard Mitarbeiter sein)
4. Warum sind Eingabefelder Grau?
5. Tabellenkonfiguration wurde weggelassen
6. Mitarbeiter Abteilungsleiter Filter noch in English
7. Mitarbeiter Rolle ist Leer wenn Administrator
8. ToS in Deutsch angeben (AGB)
9. ToS braucht es nicht
10. Mitarbeitertyp braucht es nicht
11. Pass Identität überprüft oder nicht, neuer UseCase erkannt
12. Gedankengang kann auf die Navigation abgebildet werden

13. Vergleich alte und neue Lösung: nicht mehr so grosse Überraschung seit letzten Mal. Viel Fokus auf neue Funktionalitäten gelegt, Detailview ist neu und Support muss sich an Differenz in Layout gewöhnen. Vorschlag für Reihenfolge der Felder wurde gegeben. Viel schöneres und ansprechenderes Design als alte Lösung, besser benutzbar. Hat das Gefühl wird schlussendlich weniger verwirrend sein als alte Lösung, wenn sie sich umgewöhnt hat. Freut sich stark auf die neue Lösung.

Erkannte Fehler

1. Key Sensitivität bei Löschen ist nicht vorhanden
2. Admin wird in Rolle nicht angezeigt
3. Warum Buttons und nicht Tabs für Listenansichten?
4. Aus Detailansicht kann man nicht von Suche aus navigieren
5. Listenansicht Tenants: Spalte "Letztes Login"kann nicht umsortiert werden