



Praxisprojekt

Praxisprojekt

Bachelor of Science im Studiengang allgemeine Informatik
an der Fakultät für Informatik und Ingenieurwissenschaften
der Technischen Hochschule Köln

vorgelegt von: Marc Kevin Zenzen
Matrikel-Nr.: 111 317 24
Adresse: Richratherstraße 65
40764 Langenfeld
marckevinzenzen@gmx.de

eingereicht bei: Prof. Dr. Stefan Karsch

Langenfeld, 10.08.2021

Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und dabei keine anderen als die angegebenen Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum

Rechtsverbindliche Unterschrift

Kurzfassung/Abstract

Beim Eintritt neuer Mitarbeiter in ein Unternehmen, ist die Eingliederung in die Unternehmensstruktur erforderlich. Diese umfasst die Einführung in das Unternehmen, Vermittlung von Informationen, Wissen und Projekten. Im Vorfeld ist die Erteilung von Berechtigungen zu klären. Dieser Prozess wird Onboarding genannt. Das Ziel der vorliegenden Praxisarbeit ist, den Prozess des Onboardings bei der Firma Computacenter effizienter und kostengünstiger für alle Beteiligten zu gestalten. Damit dieser Zweck erfüllt werden kann, wird eine Internetseite entwickelt, die es den Mitarbeitern im Unternehmen erleichtern soll, neue Mitarbeiter zu onboarden. Durch das Onboarding soll sichergestellt werden, dass neue Mitarbeiter fachlich eingearbeitet und sie zügig in das Team und in die Unternehmensstruktur eingliedert werden. Es ermöglicht dem Personalverantwortlichen (PV) als auch dem neuen Mitarbeiter sicheres Agieren. Zudem trägt es dazu bei, dass sich der neue Mitarbeiter wertgeschätzt fühlt und die seine Motivation steigt.

Um den Prozess des Onboardings zu modifizieren, wird der herkömmliche Prozess von dem PV erläutert. Es zeigt sich, dass der PV viel Zeit benötigt, um neue Mitarbeiter zu onboarden. Aus diesem Grund wird eine Web Applikation für dieses Problem entwickelt, damit die neuen Mitarbeiter selbstständig das Onboarding durchlaufen und bei Rückfragen den PV fragen können. Dafür wird ihnen eine individuelle Checkliste auf der Internetseite bereitgestellt, die sie abarbeiten können.

Inhaltsverzeichnis

Erklärung	I
Kurzfassung/Abstract	II
Abbildungsverzeichnis	V
Einleitung	1
1 Prozess des Onboardings	3
1.1 Prozess des Onboardings allgemein	3
1.2 Prozess des Onboardings bei Computacenter	3
2 Anforderungen	6
2.1 Architektur	7
3 Durchführung	9
3.1 Agile Softwareentwicklung	9
3.1.1 Das agile Manifest	9
3.1.2 Agile Softwareentwicklung bei Computacenter	10
3.2 Iterationen der Entwicklung	11
4 Datenmodellierung	12
4.1 Microsoft SQL Server Management Studio	13
5 Backend	14
5.1 .Net	14
5.2 Entity Framework	16
5.3 REST API	16
5.4 Verbindung zur Datenbank herstellen	17
6 Microsoft Azure	18
6.1 Azure Tenant	18
6.2 Azure SQL Datenbank	18
6.3 Azure Web App	18
7 DevOps	20
7.1 Bedeutung DevOps	20
7.2 DevOps-Kultur	20
7.3 DevOps-Methoden	20
7.4 DevOps-Tools	21

8 Frontend	22
8.1 Vue.js und Vuetify	22
8.2 Umsetzung Manager Vue.js	22
8.3 Umsetzung Manager Vuetify	24
8.4 Umsetzung Checklist	24
9 Authentifizierung	26
9.1 Authentisierung, Authentifizierung und Autorisierung	26
9.2 OAuth2.0 Authorization Code Flow with PKCE	26
9.3 Azure	28
9.4 Generieren eines Token via Postman und Microsoft Graph	28
9.5 Backend	29
9.6 Frontend	30
10 Fazit und Ausblick	31
Anhang	34

Abbildungsverzeichnis

1	Architektur	4
2	Use Case	6
3	Architektur	7
4	Das agile Manifest	10
5	Datenbankmodell	12
6	Authorization Code Flow with PKCE	27

Abkürzungsverzeichnis

AD Active Directory

API Application Programming Interface

BPMN Business Process Model and Notation

CD Continuous Delivery

CI Continuous Integration

CLI Command Line Interface

CRUD Create, Read, Update und Delete

CSS Cascading Style Sheets

Dev Development

DI Dependency Injection

EF Entity Framework

ERM Entity-Relationship-Modell

ESS Employee Self Service

GUID Globally Unique Identifier

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ID Identifikator

MSAL Microsoft Authentication Library

MSS Manager Self-Service

NGSD NEXT GENERATION SERVICE DESK

OAuth Open-Authorization

OOP objektorientierte Programmierung

Ops Operation

ORM Objekt-Relationales Mapping

PKCE Proof Key for Code Exchange

PV Personalverantwortlicher

REST Representational State Transfer

SPS SharePoint Solutions

SQL Structured Query Language

UI User Interface

UUID Universally Unique Identifier

Einleitung

Die Firma Computacenter ist ein herstellerübergreifender IT-Dienstleister. Sie bietet einen Anwender-Support, die Lieferung qualitativ hochwertiger Hardware sowie die sichere Bereitstellung von Anwendungen und Daten. Optimal und individuell angepasste Technologien werden implementiert, deren Performance optimiert und/oder die IT-Infrastruktur gemanagt. Durch den Support der CIOs (Chief Investment Officer) und der IT-Abteilungen der Auftraggeber wird u.a. deren Produktivität erhöht (Computacenter, 2021a).

Bei der Firma Computacenter sind rund 16.000 Mitarbeiter beschäftigt, davon rund 6.800 in Deutschland.

Ebenso wie in den meisten anderen Unternehmen werden bei der Integration neuer Mitarbeiter Maßnahmen durch einen Personalverantwortlichen (nachfolgend PV oder Manager genannt) durchgeführt, die dem neuen Mitarbeiter ermöglichen sollen, sich in das Unternehmen zu integrieren. Dieser Prozess wird Onboarding (an Bord nehmen) genannt.

Für die Firma Computacenter hat sich der bisherige Prozess des Onboardings als sehr zeitintensiv und daher als kostspielig herausgestellt, da ein PV zur Verfügung gestellt werden muss, der in der Regel einen Tag mit der Einführung des neuen Mitarbeiters beschäftigt ist.

Aufgrund dieser Tatsache ist dieses Praxisprojekt entstanden. Demnach soll für die Firma Computacenter, durch die Entwicklung einer entsprechenden Internetseite, der Prozess des Onboardings optimiert und kostengünstiger gestaltet werden. Dafür werden eine Reihe von Technologien wie Microsoft .Net, Vue.JS, DevOps und Microsoft Azure verwendet.

Die Internetseite soll dem neuen Mitarbeiter ermöglichen, sich anhand einer Checkliste in das Unternehmen einzufinden, sodass die PVs weniger Zeit für das Onboarding investieren müssen. Dennoch stehen sie als Ansprechpartner bei Fragen für den neuen Mitarbeiter zur Verfügung. Anders als zuvor können sie dennoch ihren regulären Tagesablauf beibehalten und ihren täglichen Aufgaben nachkommen.

Im ersten Kapitel der Arbeit wird der Begriff des Onboardings im allgemeinen und speziell in Bezug auf das Unternehmen Computacenter beschrieben. Anschließend (2) folgt eine Darstellung der Anforderungen für die zu entwickelnde Internetseite, um den in Kapitel 1 beschriebenen Prozess des Onboardings zu optimieren. Zusätzlich wird in diesem Kapitel die verwendete Softwarearchitektur, angepasst auf die Anforderungen, beschrieben. Da bei Computacenter Softwareprojekte agil entwickelt werden, wird diese Art der Entwicklung in Kapitel 3 detailliert vorgestellt. Des Weiteren werden darauf aufbauend die Iterationen der Implementierung erläutert. Die Beschreibung des Aufbaus der Datenbank zum persistenten Speichern der

Daten folgt in Kapitel 4. Aufbauend auf die Datenmodellierung wird im nächsten Kapitel (5) die Entwicklung des Backends beschrieben. Im Anschluss (6) wird auf das Hosten des Backends in einem Cloud-Service (Azure) eingegangen. Die Beschreibung der Automatisierung des Prozesses des Hochladens in den Cloud-Service erfolgt in Kapitel 7. Danach (8) erfolgt die Darstellung der Umsetzung des User Interface (Frontend) basierend auf den Anforderungen und des Backends. Der Schutz der gesamten Anwendung erfolgt mittels einer Authentifizierung (9).

1 Prozess des Onboardings

1.1 Prozess des Onboardings allgemein

In diesem Abschnitt wird der Einarbeitungs- und Integrationsprozess eines neuen Mitarbeiters in ein Unternehmen nach Brenner, 2014 beschrieben. Dieser Prozess wird „Onboarding“ genannt. Es setzt sich aus drei Ebenen zusammen: Die fachliche, soziale und wertorientierte Integration.

Die fachliche Integration beschäftigt sich damit, dass der Mitarbeiter sich Kenntnisse über das Unternehmen und insbesondere über seine Arbeitsgebiete aneignet. „Der Schwerpunkt liegt dabei in der Einarbeitung in bestimmte Aufgabenstellungen, der Aneignung von Faktenwissen und der konkreten Umsetzung seiner Kenntnisse und Fähigkeiten im Sinne der Unternehmensziele.“ (Brenner, 2014, S. 7). Auch Kenntnisse über die Organisationsstruktur und die richtigen Ansprechpartner in den entsprechenden Fachbereichen werden vermittelt.

Bei der sozialen Integration muss der Mitarbeiter sich mit seinem neuen Arbeitsumfeld vertraut machen. Der Umgang zu Vorgesetzten, Kollegen, internen- und externen Kunden findet meistens über soziale Kontakte statt. Das Arbeiten in Teams oder Projektgruppen und das Absprechen mit Kollegen stellen dabei wichtige Elemente dar. „Erst wenn der Mitarbeiter als Teil der Gemeinschaft akzeptiert wird und sich ein „Wir-Gefühl“ entwickelt hat, kann von einer erfolgreichen sozialen Integration gesprochen werden.“ (Brenner, 2014, S. 8).

Bei der wertorientierten Integration muss sich der Mitarbeiter mit den Zielen, Werten und den Führungsgrundsätzen vertraut machen. Wertorientierte Integration ist ein langfristiger Prozess, der dem Mitarbeiter nicht nur über ein Leitbild sondern viel mehr über vorgelebte Werte vermittelt werden kann.

Ein erfolgreicher Onboardingprozess beinhaltet alle drei Elemente. Der Schwerpunkt, der bei der Einarbeitung und Integration gewählt werden sollte hängt sehr stark von den individuellen Voraussetzungen des Einzelnen ab. Es macht einen Unterschied, ob es sich um einen Hochschulabsolventen oder eine Führungskraft mit jahrelanger Erfahrung handelt, die eingestellt wird. Von zentraler Bedeutung ist auch das zukünftige Arbeitsgebiet des neuen Mitarbeiters. Ein Kaufmann muss z.B. anders auf seine Aufgaben vorbereitet werden als ein Entwickler.

Nachfolgend wird der Prozess der Onboardings bei der Firma Computacenter erläutert.

1.2 Prozess des Onboardings bei Computacenter

Zu dem jetzigen Zeitpunkt wird der Prozess des Onboardings in der Firma Computacenter eher manuell und daher nicht effizient abgearbeitet. Dies bedeutet, dass

der PV die neuen Mitarbeiter zunächst einen Tag lang intensiv begleitet und einarbeitet. An diesem Prozess sind hauptsächlich der neue Mitarbeiter und der PV beteiligt.

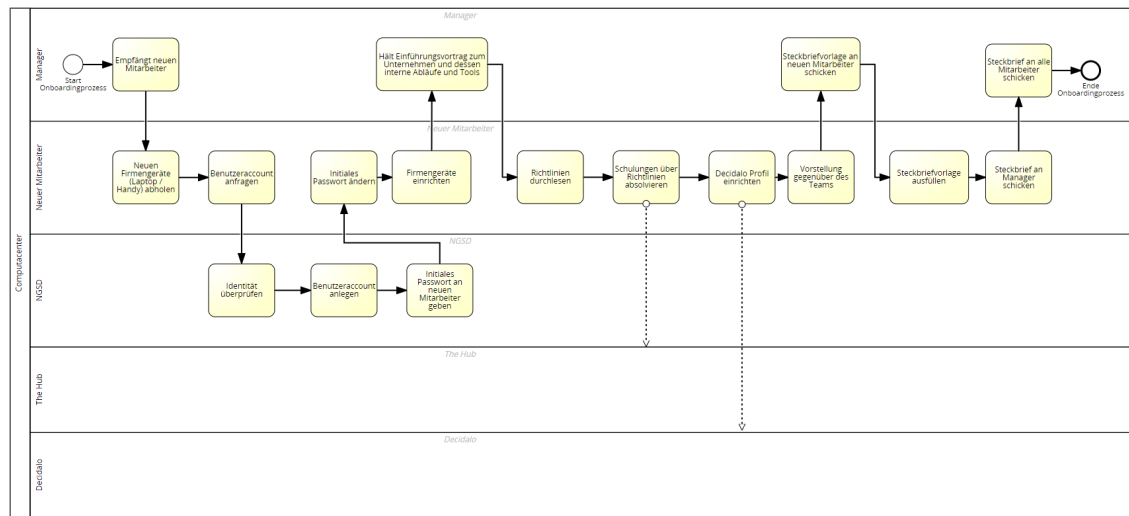


Abbildung 1 Prozess des Onboardings bei Computacenter

In Abbildung 1 ist dieser Prozess als Business Process Model and Notation (BPMN) dargestellt. „BPMN ist der führende Standard zur Erstellung von Geschäftsprozessmodellen.“ (Signavio, 2021). Nach Signavio, 2021 stellt ein ‚Pool‘ eine Einheit mit klar voneinander abgegrenzten organisatorischen Grenzen wie bspw. Organisationen oder Unternehmen dar. Eine Lane hingegen repräsentiert Abteilungen, Personen oder Rollen in einem Prozess und damit die Prozessteilnehmer. Aktivitäten sind BPMN-Elemente, die für bestimmte Handlungen und Aufgaben in einem Geschäftsprozess stehen. In der Grafik ist der ‚Pool‘ mit der Bezeichnung ‚Computacenter‘ zu sehen. Die dazugehörigen Lanes sind unterteilt in: Manager, neuer Mitarbeiter, NGSD, The Hub und Decidalo. Im nachfolgenden werden die Aktivitäten dieser Lanes erklärt.

Zu Beginn des Prozesses empfängt der Manager den neuen Mitarbeiter. Danach holt der Mitarbeiter die neuen Firmengeräte ab. Anschließend wird beim NEXT GENERATION SERVICE DESK (NGSD) das Benutzerkonto angefragt. Nach Computacenter, 2021b ist NGSD ein IT-Support-Service von Computacenter, welcher den Support für Mitarbeiter als auch für Kunden in Richtung Digitalisierung verbessern soll. Dafür überprüft es die Identität des neuen Mitarbeiters, legt daraufhin ein neues Benutzerkonto mit einem initialen Passwort an und gibt diese Daten an den Mitarbeiter zurück. Der Mitarbeiter ist nun aufgefordert das Passwort direkt zu ändern und kann danach mit der Einrichtung der Firmengeräte beginnen. Nach der Einrichtung der Geräte hält der Manager einen Einführungsvortrag über das Unternehmen und dessen interne Abläufe, Strukturen und Tools. Nach Beendigung des Vortrags muss der Mitarbeiter sich die Richtlinien des Unternehmens durchlesen. Zusätzlich werden Schulungen in The Hub (Computacenters internes Schulungssystem)

absolviert. Des Weiteren muss der neue Mitarbeiter sein Firmenprofil bei Decidalo einrichten. Decidalo, 2021 bietet die Möglichkeit, Kenntnisse und Erfahrungen der Mitarbeiter in Profilen abzubilden. In einem Teammeeting stellt sich der Mitarbeiter dem Team vor. Anschließend schickt der Manager dem Mitarbeiter eine Steckbriefvorlage, die der Mitarbeiter mit persönlichen Informationen (Berufserfahrungen und bspw. Hobbies) ausfüllt. Dieser Steckbrief wird an den Manager geschickt und von ihm an alle Mitarbeiter weitergeleitet.

Die o.g. abzuarbeitenden Aufgaben variieren je nach Position und Aufgabengebiet des neuen Mitarbeiters. Abhängig von dem Arbeitsbereich des neuen Mitarbeiters sollen vielfältige Informationen vermittelt werden. Sie können sich auf die allgemeine Firmenphilosophie, den Datenschutz, die Firmenstruktur, Arbeitsabläufe, aktuell die Hygieneregeln etc. beziehen.

2 Anforderungen

Da der aktuelle Prozess des oben beschriebenen Onboardings für den PV recht zeitintensiv und dadurch unwirtschaftlich ist, soll eine Software entwickelt werden, die dem neuen Mitarbeiter ermöglicht, es nahezu eigenständig zu durchlaufen. Der PV kann währenddessen einer anderen Aufgabe nachgehen, steht dem neuen Mitarbeiter selbstverständlich dennoch für Fragen zur Verfügung.

Die Anforderungen für die zu entwickelnde Software wurden von Computacenter definiert und beinhalten verschiedene Funktionalitäten für die Anwendung.

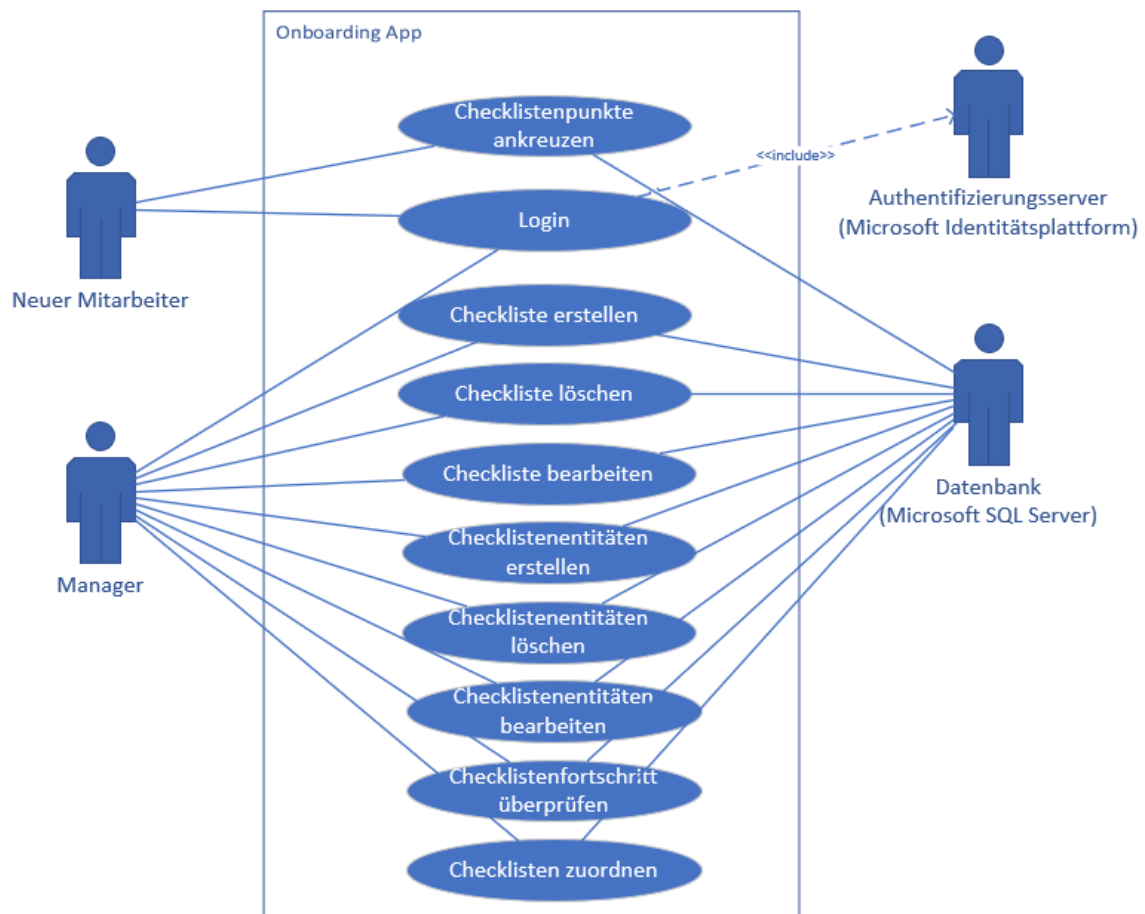


Abbildung 2 Use Case Diagramm zu den Funktionen für den Mitarbeiter und den Manager

In Abbildung 2 sind die geforderten Funktionen in Form eines Use Case Diagramms zu sehen. Für den neuen Mitarbeiter soll die Funktionalität entwickelt werden, dass er in einer individuell für ihn erstellten Checkliste diese Punkte ankreuzen kann. Der Manager soll die Möglichkeit haben, Checklisten zu erstellen, zu bearbeiten und zu löschen. Außerdem soll er diese Checklisten den Mitarbeitern zuweisen können, damit diese die Checklistenpunkte abarbeiten können. Zusätzlich sollen vom Manager die Checklistenentitäten (Checklistenpunkte) erstellt, bearbeitet und gelöscht werden können. Der Manager soll außerdem den Fortschritt der Checklisten der Mitarbeiter einsehen können.

Alle bisher aufgeführten Funktionalitäten erfordern das persistente Speichern der Änderungen oder Zuordnungen. Aus diesem Grund soll eine Datenbank erstellt werden. Die zu entwickelnde Anwendung soll außerdem von Fremdeinwirkungen geschützt werden, weshalb zusätzlich eine Authentifizierung implementiert werden soll. Die Umsetzung der zu entwickelnden Anwendung ist dem Unternehmen wichtig, da es durch die Anwendung sehr viel Geld und Aufwand sparen kann.

2.1 Architektur

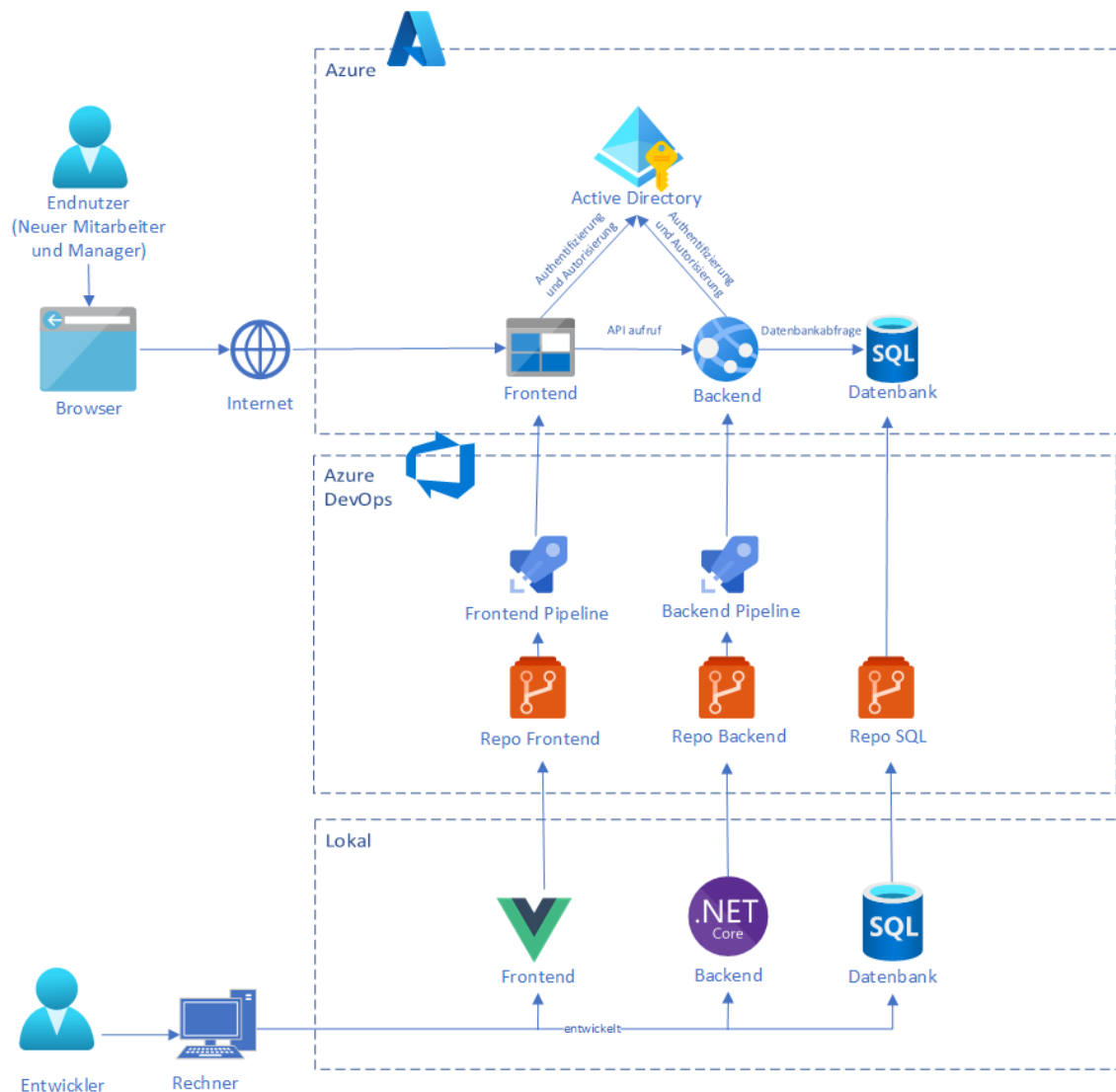


Abbildung 3 Architektur

Aufbauend auf den Anforderungen wird eine Architektur entwickelt, um die zu entwickelnde Anwendung umzusetzen. Für die Umsetzung werden hauptsächlich die Produkte von Microsoft verwendet, da diese von Computacenter vorgegeben sind. Ein großer Fokus in diesem Praxisprojekt liegt auf der in Abbildung 3 abgebildeten Architektur. Gerade in den ersten Iterationen wird kein großes Spektrum an Funktionalitäten umgesetzt, sondern eher das Gerüst, indem alle Plattformen wie Azure,

DevOps und die Lokale Entwicklungsumgebung miteinander verknüpft sind. Diese müssen auch innerhalb der Plattformen konfiguriert werden, indem z.B. das Frontend auf die Api (in Kapitel 5) des Backends zugreift oder dass das Backend mit der Datenbank kommuniziert etc.. Dazu wird im Backend das kostenlose Open-Source Webframework .Net Core von Microsoft verwendet. Für die Datenbank wird im lokalen Entwicklungsbereich der Microsoft SQL (Structured Query Language) Server verwendet, um die Daten persistent zu speichern. Im Frontend wird mit Vue.js gearbeitet, welches ein Javascript Webframework darstellt. Zusätzlich zu Vue.js wird Vuetify im Frontend für das styling verwendet. Das Back- und das Frontend werden entkoppelt voneinander entwickelt, so dass später ohne Probleme das Austauschen des Back- oder Frontends ermöglichen wird.

Das gesamte Projekt wird in einem DevOps Repository (Kapitel 7.3) gespeichert. Außerdem werden von DevOps zwei weitere Funktionalitäten verwendet: die Boards und die Pipelines (Kapitel 7.3).

Des Weiteren ist eine Authentifizierung per OAuth2.0 erforderlich. Hierfür werden die Benutzerdaten (Anmeldename und Passwort) nicht in der eigenen Datenbank gespeichert, sondern über die Microsoft Identity Plattform abgefragt. Dafür benötigen die User ausschließlich einen Microsoft Account um sich über diese in der Onboarding App anmelden zu können.

Das gesamte Projekt wird allerdings nicht nur lokal entwickelt, sondern wird anschließend bzw. auch während der Entwicklung in Azure gehostet. Zusätzlich wird die Anwendung in Azure automatisiert auf dem aktuellen Stand der Repositories vom Front- und Backend gehalten. Auf diese Themen wird in Kapitel 7 (DevOps) dieser Arbeit ausführlicher eingegangen.

3 Durchführung

Bevor auf den groben Ablauf (die Iterationen) der Implementierung eingegangen werden kann, muss vorher der Begriff der agilen Softwareentwicklung eingeführt werden, da die Softwarelösungen bei Computacenter mit Hilfe dieser entwickelt werden.

3.1 Agile Softwareentwicklung

Für dieses Projekt wird die Methode der agilen Softwareentwicklung gewählt. Agile Softwareentwicklung beinhaltet den Begriff agil. Agil bedeutet, dass jemand körperlich oder geistig flink oder wendig ist. Diese beiden Begriffe können ebenso gut auf die agile Softwareentwicklung angewendet werden. Es soll angepasst (wendig) vorgegangen werden und es sollen schnelle (flinke) vorzeigbare Ergebnisse erzielt werden (Wolf und Bleek, 2011).

Hinzu kommt die Komponente Kommunikation. In der agilen Softwareentwicklung spielt die Kommunikation zwischen allen Projektbeteiligten eine große Rolle. Der Kundenmanager soll mit den Anwendern über ihre genauen Ziele sprechen. Außerdem müssen die Anwender ihrem Kundenmanager und den Entwicklern ihre Bedürfnisse explizit mitteilen und sollten sich über diese am besten einig sein. Zusätzlich sollten die Entwickler untereinander auch mit allen anderen Projektbeteiligten kommunizieren, da diese im Späteren die Ziele der Anwender umsetzen müssen und sich über die auftretenden Problematiken, aber auch Lösungswege austauschen sollten. Die Kommunikation sollte direkt, offen und ehrlich sein. Direkt bezieht sich auf kurzfristige Kommunikation bei Bedarf. Offen und ehrlich ist die Kommunikation dann, wenn auch über Probleme die auftreten könnten, gesprochen wird.

Rückkopplung oder auch Feedback ist ein weiterer Aspekt der agilen Softwareentwicklung. Die Rückkopplung steht in einem sehr engen Zusammenhang mit der direkten Kommunikation. Möglichst frühe und häufige Auslieferungen von Softwareprodukten an den Kunden fördern die Rückkopplung. Bei Bedarf kann sie ergänzt oder korrigiert werden.

3.1.1 Das agile Manifest

Im Februar 2001 haben sich viele Vertreter der Agilen Methoden getroffen. Darunter vertreten waren Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland und Dave Thomas. Damit waren u.a. die Methoden eXtreme Programming, Scrum, Crystal und Feature Driven Development vertreten. Die Teil-

nehmer einigten sich als Grundlage aller agiler Methoden auf folgendes Manifest, das in Abbildung 4 zu sehen ist:



Abbildung 4 Das agile Manifest Beck et al., 2001

Darunter sind drei Prinzipien, welche die obigen präzise zusammenfassen und abrunden. Diese sind zum Ersten „Funktionierende Software mehr als umfassende Dokumentation“ d.h. der Fokus liegt erstmal auf einem laufenden Prototypen. Der Zweite ist „Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung“. Damit wird die Kommunikation priorisiert. Das dritte Prinzip „Reagieren auf Veränderung mehr als das Befolgen eines Plans“ rückt die Flexibilität in den Vordergrund.

3.1.2 Agile Softwareentwicklung bei Computacenter

Bei der Firma Computacenter wird in den meisten Softwareprojekten agil gearbeitet. Dies hat zur Folge, dass in kleinen Iterationen vorgegangen wird und nach jeder Iteration dem Kunden ein lauffähiger Prototyp vorgestellt werden kann. Der Kunde kann darauf hin bei Zufriedenheit die nächsten Funktionalitäten in Auftrag geben oder auf Basis des Prototypen mitteilen, was ihm nicht gefällt. Nach jeder oder auch während der Iterationen wird Rücksprache mit dem Kunden gehalten. Zusätzlich soll die agile Softwareentwicklung bezwecken, dass Risiken und Fehlentwicklungen minimiert werden, da ein ständiger Austausch mit dem Kunden stattfindet. Auf diese Weise wird vermieden, dass erst geplant und implementiert wird und der Kunde im Späteren feststellt, dass er sich das Ergebnis doch anders gewünscht hatte.

3.2 Iterationen der Entwicklung

Basierend auf den Anforderungen und der agilen Softwareentwicklung, wird die zu entwickelnde Internetseite in verschiedenen Iterationen entwickelt.

In der ersten Iteration soll ein lauffähiges Backend angelegt werden, auf welches per Postman zugegriffen werden kann, um die Daten aus dem Backend abzufragen. Postman bietet die Möglichkeit auf APIs zuzugreifen und dabei Funktionen zum Erstellen, Testen, Mocken und Debuggen von API-Anfragen zur Verfügung zu stellen. Gleichzeitig sollen die Daten persistent in einer Datenbank gespeichert werden, auf die das Backend später zugreifen kann. Entsprechend der Anforderungen an die Applikation werden Tabellen für die Datenbankmodellierung erstellt. Die Daten sollen in der ersten Iteration noch nicht visualisiert werden und nur über Postman abgefragt werden können. Zu diesen Daten zählen auf der einen Seite Daten über Personen, wie z.B. Manager oder Employees und auf der anderen Seite eine Liste mit den abzuarbeitenden Schritten. Diese Liste (auch Checkliste) stellt in der Anwendung die Kernanforderung da.

In der zweiten Iteration sollen sowohl das Backend als auch die Datenbank in der Cloud von Microsoft Azure gehostet werden. Außerdem wird dort auf die Möglichkeit zurück gegriffen, mit DevOps zu arbeiten, da dieses weitere Möglichkeiten bietet, um Prozesse zu automatisieren.

Die dritte Iteration beinhaltet die Bereitstellung des Frontendes mit Hilfe von Vuejs und Vuetify. Dort sollen die entwickelten Funktionalitäten aus dem Backend visualisiert werden, worauf in Kapitel 8 differenzierter eingegangen wird.

In der vierten Iteration soll das Frontend in der Cloud von Microsoft Azure bereitgestellt werden, damit auch dieses aus dem Internet erreichbar ist.

Für die folgenden Iterationen sind noch Funktionalitäten wie die Authentifizierung per OAuth2.0 (Open-Authorization) und die Chatmöglichkeit mit dem Personalverantwortlichen (Echtzeitkommunikation) geplant. Außerdem können Checklisten für neue Mitarbeiter oder Schnittstellen zu anderen Plattformen erstellt werden z.B. „The Hub“ (Computacenter internes Schulungsportal). Die Informationsaufbereitung direkt für den PV bietet den Vorteil, dass Informationen des Mitarbeiters von dem PV direkt aus dem Programm abgerufen werden können.

4 Datenmodellierung

Bevor mit dem Programmieren begonnen werden kann, muss erst überlegt werden, welche Daten überhaupt benötigt und somit bei der Datenmodellierung berücksichtigt werden müssen, damit alle Informationen entsprechend den Anforderungen persistent gespeichert werden. Für diesen Schritt wird ein Entity-Relationship-Modell (ERM) angelegt, welches in Abbildung 5 zu sehen ist.

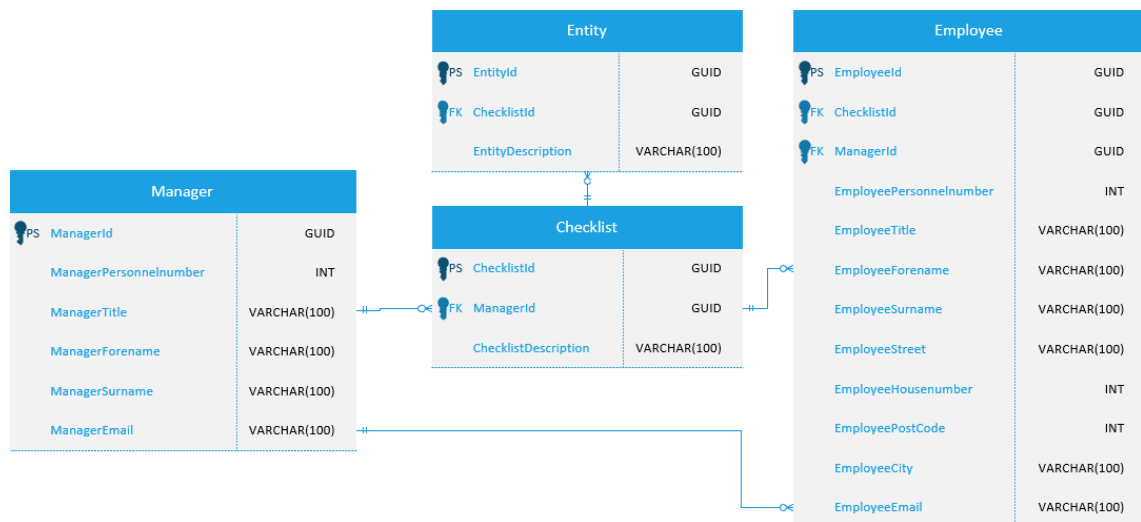


Abbildung 5 Datenbankmodell

In der Abbildung ist zu sehen, dass sich das Modell aus 4 Tabellen zusammen setzt: Manager, Employee, Checklist und Entity. Jede dieser Tabellen besitzt einen (eindeutigen) Primärschlüssel. Dieser setzt sich aus dem Namen der Tabelle und dem Kürzel ‚Id‘ zusammen. Der Datentyp des Primärschlüssels ist ein Globally Unique Identifier (GUID). Für diesen wurde zunächst ein Integer gewählt. Dies stellte sich jedoch bei der Entwicklung des Backends als nachteilig heraus, weil der Wertebereich im Gegensatz zur GUID kleiner ist. Die GUID wird gewählt, da diese laut (Microsoft, 2021c) weltweit eindeutig ist. Außerdem stellt sie einerseits eine Implementierung der UUID dar und andererseits wird der Begriff ‚GUID‘ meistens im Zusammenhang mit Microsoft verwendet. Zusätzlich haben der Manager und der Employee die Attribute Personalnummer, Titel, Vorname, Nachname und E-Mail Adresse. Der Employee hat zusätzlich seine Adresse hinterlegt.

Außerdem sind in der Abbildung des ERM die Kardinalitäten der Tabellen zu sehen. Diese beschreiben die Beziehungen zwischen den Tabellen. Hierfür wird die Krähfußnotation aus der Vorlesung von Bertelsmeier, 2019, gewählt. Ein Manager kann beispielsweise eine oder mehrere Checklisten verwalten. Eine Checkliste kann immer nur von einem Manager verwaltet werden. Diese Checkliste kann dann einem oder mehreren Employees zugewiesen werden. D.h., Employees aus denselben Bereichen können die gleichen Checklisten erhalten, da sie meistens die selben Arbeitsschrit-

te abzuarbeiten haben. Zusätzlich besteht noch eine direkte Beziehung zwischen Manager und Employee, da Employees immer genau einen Manager als Vorgesetzten haben. Managern hingegen werden mehrere Employees zugeordnet. Die letzte Kardinalität beschreibt die Beziehung zwischen der Checkliste und der Entität (Entity). Eine Entität kann immer nur einmal in einer Checklist enthalten sein, da ein Manager bspw. die Bezeichnung dieser ändern könnte und diese Änderung in allen Checklisten angenommen werden würde. Für jeden anderen Manager würde diese Bezeichnung keinen Sinn ergeben, weshalb jede Entität nur in einer Checkliste besteht. Eine Checkliste kann sich jedoch aus mehreren Entitäten zusammen setzen.

Alle Tabellen, die auf ihrer Seite der Kardinalitäten die n-Beziehung haben, enthalten außerdem den Primärschlüssel der gegenüber liegenden Tabelle als Fremdschlüssel. Würde in diesem ERM auch eine n:m Beziehung existieren, dann müsste diese über eine Zwischentabelle aufgelöst werden.

4.1 Microsoft SQL Server Management Studio

Für die Umsetzung des Datenbankmodells in eine relationale Datenbank wird für die lokale Entwicklung das Microsoft SQL Server Management Studio gewählt, da es von Computacenter vorgegeben ist. Beim Anlegen der Tabellen tritt dann häufig das Problem auf, dass - sofern bei einer Tabelle Änderungen an dem Primärschlüssel vorgenommen werden müssen-, wie z.B. der Änderung des Datentyps von Integer auf GUID, dies nicht funktioniert und die ganze Tabelle gelöscht und wieder neu erstellt werden müsste. Außerdem wird die GUID im Microsoft SQL Management Studio nicht GUID betitelt, sondern ist dort unter der Bezeichnung ‚uniqueidentifier‘ als Datentyp zu finden.

5 Backend

5.1 .Net

Um die oben erstellte Datenbank zu verwenden, wird ein dazu angepasstes Backend entwickelt. Dieses verwendet das .NET Core Web Api Framework von Microsoft. Nach Augusten, 2020 wird .NET Core als Open-Source-Plattform bezeichnet und federführend von Microsoft entwickelt. Er postuliert weiter, dass .NET Core im Gegensatz zu seinem Vorgänger, dem .NET Framework zahlreiche Vorteile bietet. In .NET 5 werden die Framework- und Coreversion miteinander vereint. An .NET Core wird bereits seit 2014 gearbeitet, 2016 erschien es erstmals und wird seitdem regelmäßig aktualisiert. Die Open-Source-Plattform dient dazu, Apps zu entwickeln und diese ausführen zu können. Dabei ist .NET Core eine deutliche Modernisierung im Vergleich zum früheren .NET Framework. Dabei handelt es sich also nicht um eine komplette Neuentwicklung, sondern um eine Hybridlösung aus Redesign und Neuimplementierung.

Seit Ende 2020 werden diese jedoch nicht mehr nebeneinander weiter entwickelt, sondern sollen seitdem zusammenfließen. Das Ganze wird .NET 5 genannt. Für die folgende Entwicklungsziele soll eine entsprechende Basis geschaffen werden:

- Apps für Desktoprechner
- Anwendungen für Webserver
- Mobile Apps
- Spiele

In dem vorliegenden Projekt wird .NET 5 verwendet. Beim Erstellen des Projekts muss zunächst ausgewählt werden, welche Vorlage verwendet werden soll. Als Vorlage wird in diesem Fall die ASP .NET Core-Web-API und im nächsten Schritt die .NET 5 Version gewählt. Nachdem das Projekt erfolgreich erstellt wurde, erscheinen zwei Ordner mit den Namen „Properties“ und „Controllers“. Zusätzlich sind noch vier weitere Dateien zu sehen: die „appsettings.json“, „Programm.cs“, „Startup.cs“ und die „WeatherForecast.cs“. Die Dateien „WeatherForecast.cs“ und „WeatherForecastController.cs“ stellen ausschließlich ein kleines Demo Projekt dar. Sie werden daher nicht berücksichtigt und können direkt gelöscht werden.

In der „Programm.cs“ wird der Host für die App erstellt. Der Host ist ein Objekt, das alle Ressourcen der App kapselt, z.B. Abhängigkeitsinjektion, Protokollierung, Konfiguration und *IHostedService*-Implementierungen. Der Hauptgrund des Einschließens aller unabhängigen Ressourcen in einem Objekt ist die Verwaltung der Lebensdauer. Damit wird das Starten und ordnungsgemäße Herunterfahren der

App gesteuert. Der Host ruft außerdem die „Startup“ auf.

Die „Startup“-Klasse konfiguriert Dienste und die Anforderungspipelines der App. Sie enthält zwei Methoden, die *ConfigureServices* und die *Configure*. Die *ConfigureServices* wird verwendet, um die Dienste zu konfigurieren. Ein Dienst ist eine wiederverwendbare Komponente, welche App-Funktionalitäten bereitstellt. Dienste werden in der *ConfigureServices* registriert und in der App über *Dependency Injection (DI)* oder *ApplicationService* genutzt. Die *ConfigureService* wird dabei von dem Host vor der *Configure* aufgerufen, um die App-Dienste zu konfigurieren. In der *Configure*-Methode wird sodann festgelegt, wie die App auf HTTP-Anforderungen reagieren soll. Wenn die App gestartet wird, werden die *ConfigureService* und die *Configure* von der ASP .NET Core-Runtime aufgerufen.

In der „appsettings.json“ Konfigurationsdatei stehen wichtige Schlüssel-Wert-Paare, die von der Anwendung verwendet werden. Unter anderem sind dort die Konfigurationsinformationen zu Azure Active Directory (Azure AD) zu finden. Außerdem ist auch der *ConnectionString* in dieser Datei gespeichert, welcher die Verbindung zur Datenbank ermöglicht. Diese Datei kann in mehreren Varianten gespeichert werden: beispielsweise als „appsettings.Production_.json“ oder „appsettings.Development_.json“, um im späteren Verlauf verschiedene Konfigurationen für die Produktiv- oder Entwicklungsumgebung zu bilden.

Zunächst wird in dem Projekt, zusätzlich zu den anderen beiden Ordnern ein weiterer Ordner mit dem Namen „Models“ angelegt. In diesem Ordner werden alle zuvor erstellten Entitäten aus dem ERM als Klassen angelegt. Hierfür wird eine neue Klasse erstellt, die den selben Namen tragen sollte, wie die Entität. In der Datei wird für jedes Attribut der Klasse eine Variable deklariert, welche dem selben Datentyp zugrunde liegt, wie das Attribut in der Entität aus dem ERM (dazu zählen auch die Fremdschlüsselattribute). Hier stellt ein *String* einen *VarChar*, ein *int* einen *INT* und die *guid* die *GUID* dar. Jede Variable erhält zusätzlich in geschweiften Klammern einen *Getter* und einen *Setter*. Hier stellt sich die Frage, wie die Beziehung unterhalb der Klassen abgebildet werden kann. Dafür wird das Beispiel der 1:n Beziehung zwischen dem Manager und der Checkliste verwendet. Zur Erinnerung: ein Manager kann mehrere Checklisten erstellen, aber eine Checkliste kann nur von einem Manager erstellt werden. Um dies abzubilden, wird in der Manager-Klasse ein weiteres Attribut mit dem Namen *Checklists* hinzugefügt. Dieses Attribut ist eine *Navigationseigenschaft*. *Navigationseigenschaften* enthalten andere Entitäten, die dieser Entität (also dem Manager) zugehörig sind. In diesem Fall enthält die *Checklists*-*Navigationseigenschaft* alle Checklisten, die diesem Manager zugeordnet, bzw. von diesem Manager erstellt wurden. Da ein Manager mehrere Checklisten er-

stellen kann und ihm diese über die Navigationseigenschaften zugeordnet sind, sie auch hinzugefügt, gelöscht und aktualisiert werden müssen, bilden sie den Typ einer Liste, wie z.B. die `ICollection`. In der `Checklists`-Klasse muss für die 1:n Beziehung keine Ergänzung durchgeführt werden. Nachdem für alle Klassen die entsprechenden Navigationseigenschaften hinzugefügt wurden, sind alle Beziehungen im Backend abgebildet.

5.2 Entity Framework

In diesem Projekt wird als Objekt-Relationales Mapping (ORM) das Entity Framework (EF) nach Microsoft, 2021b verwendet. Beim ORM bildet man Konstrukte aus der objektorientierten Programmierung (OOP) auf die relationale Welt der Datenbanken ab.

Das EF ist ein Open-Source Framework für .NET Anwendungen, welches von Microsoft entwickelt wurde. Es ermöglicht den Entwicklern mit den Daten als Objekte zu arbeiten, ohne dabei auf die Datenbankebene herunter zu gehen, um dort mit Tabellen und Spalten einer Datenbank arbeiten zu müssen. Durch das EF kann auf einer höheren Abstraktionsebene gearbeitet werden. Außerdem können mit dem EF automatisch sogenannte „Migrationen“ erstellt werden. Migrationen bilden Datenbanken ab, deren Schema anhand der Models erstellt wird. Wird ein Model verändert, z.B. durch das Hinzufügen eines neuen Attributes, wird diese Änderung anschließend auch in der Migration vorgenommen. Hier wird nicht mit einer Migration gearbeitet, sondern mit dem Microsoft SQL Server, da im Endeffekt die Datenbank als Service in der Azure Cloud gehostet werden soll.

Das Arbeiten mit dem EF hat bereits bei der Erstellung der Models angefangen. Das EF erkennt anhand der vorhandenen Fremdschlüssel und der Navigationseigenschaften in einem „Model“ die Beziehung zu den anderen Models.

5.3 REST API

Schließlich können in dem Controllers Ordner die Controller für die zuvor erstellten Klassen generiert werden. Dabei wird für ein Model genau ein Controller erstellt. In jeder dieser Klassen werden Funktionen über REST API angeboten. Zu diesen Funktionen zählen die CRUD-Funktionen (CREATE, READ, UPDATE und DELETE). REST (REpresentational State Transfer) API (Application Programming Interface) stellt nach außen eine Schnittstelle bereit, worüber die Kommunikation zu dieser Anwendung ermöglicht wird. Nach dem Artikel von (Srocke und Karltetter, 2017) verwendet REST HTTP-Anfragen, um per GET, PUT, POST und DELETE auf Daten zuzugreifen. Zusätzlich ermöglicht dies, dass die Anwendung

Modular aufgebaut wird. Dies bedeutet, dass das Back- und Frontend, wie weiter oben bereits beschrieben, voneinander unabhängig sind und nur über die REST API Schnittstelle miteinander kommunizieren.

In den Controllern ist es wichtig, dass in den *GetAll()*-Methoden mit Includes gearbeitet wird. Das bedeutet, dass z.B. in der Manager Klasse über die Include auch die Employee und die Checkliste eingebunden werden, damit diese später auch im JSON mit zurückgegeben werden können. Im Frontend kann anschließend genau entschieden werden, welche Daten angezeigt werden und welche nicht.

5.4 Verbindung zur Datenbank herstellen

Nachdem im Backend nun alle Klassen und Controller mit den nötigen CRUD Funktionen implementiert sind, kann dieses mit der Datenbank verbunden werden. Dazu muss seitens des Backends der Connection String (siehe Anhang Abbildung ??), welcher von der Datenbank zur Verfügung gestellt wird, verwendet werden. Er wird im Backend in der „appsettings.json“-Datei ergänzt. Ebenso muss ein neuer Service in der *ConfigureService()* in der „Startup.cs“ angelegt werden, welcher einen DbContext hinzufügt und auf den Connection String in der „appsettings.json“ verweist. Nun öffnet sich beim Starten der Anwendung automatisch ein Browserfenster, welches die Daten aus der Datenbank im JSON-Format über die API abrufen und anzeigt.

6 Microsoft Azure

Damit die zuvor erstellte Datenbank und Backend nicht nur lokal sondern auch online zur Verfügung stehen, werden diese zusätzlich in einer Cloud gehostet. Dazu wird Microsoft Azure verwendet.

Das Azure Portal bietet eine große Reihe an Möglichkeiten. Es bietet mehr als 200 Produkte an, wodurch diese verwaltet, erstellt und überwacht werden können. In diesem Projekt wird sich auf das Azure Active Directory, das Erstellen von Azure Web Apps und das Erstellen von Azure SQL Servern, konzentriert.

6.1 Azure Tenant

Zuerst muss ein neuer Tenant (Verzeichnis) erstellt werden, da bis jetzt noch der Tenant von Cumputacenter verwendet wird. Anschließend wird die Visual Studio Professional Subscription von dem Coputacenter Tenant auf den neuen Tenant umgezogen.

6.2 Azure SQL Datenbank

Eine Azure SQL Datenbank wird angelegt. Dabei ist wichtig, dass diese vor der Web App (Backend) angelegt wird, da diese auf die Azure SQL Datenbank zugreift. Hierfür muss im Azure Portal unter „create new resource“ die Azure Datenbank angelegt werden. Dazu muss das darauf folgende Formular ausgefüllt werden. In diesem müssen unter anderem Informationen wie Datenbankname, unter welcher Subscription die Datenbank laufen soll, Wahl des Servers (Region) und die Art der Konfiguration angegeben werden. Von der Art der Konfiguration hängen die entstehenden Kosten pro Monat ab. Anschließend müssen die SQL Scripte zum Erstellen der lokalen Datenbank exportiert und danach verwendet werden, um dieselbe Datenbank auch in der Azure SQL Datenbank anzulegen.

6.3 Azure Web App

In diesem Schritt wird eine Azure Web App erstellt, in der das Backend laufen soll. Dafür wird zunächst im Azure Portal unter „create new resource“ eine neue Azure Web App erstellt. In dem darauf folgenden Formular müssen wieder Informationen wie Subscription, Web App Name, welches Framework (in dem Fall .NET), erneut die Region und auch wieder eine Konfiguration angegeben werden. Im Anschluss kann die Web App erstellt und der Connection String von der Azure SQL Datenbank angegeben werden. Danach wird in Visual Studio auf das Projekt im *Projektmappen-Explorer* mit der rechten Maustaste geklickt, um das Projekt zu veröffentlichen. Es muss zusätzlich angegeben werden, dass das Projekt in Azure veröffentlicht werden

soll. Abschließend erfolgt die Anmeldung mit dem zuvor erstellten Tenant über die Auswahl der zuvor erstellte Web App. Durch das Klicken auf *Fertig stellen* wird das Projekt in der Azure Web App veröffentlicht.

Nun kann mit der URL, welche in der Übersicht der Web App im Azure Portal angegeben ist, auf das Backend zugegriffen werden. Dort werden die Daten aus der Azure SQL Datenbank im JSON Format angezeigt, d.h., dass das Backend und die Datenbank in Azure erfolgreich gehostet werden (Anhang Abbildung 8).

7 DevOps

In diesem Kapitel wird beschrieben wo der Source Code abgelegt ist und wie das automatische Bereitstellen der zuvor erstellten Dienste in Azure funktioniert.

7.1 Bedeutung DevOps

Im dem folgenden Abschnitt wird sich auf die Dokumentation von DevOps von (Microsoft, 2021a) bezogen. DevOps setzt sich aus dem Begriff Dev (Development, Entwicklung) und Ops (Operation, Vorgänge) zusammen. Es soll die Menschen, die Prozesse und die Technologien vereinen, damit für die Kunden eine optimal angepasste Lösung entwickelt werden kann. Die Bereiche, die unabhängig voneinander agieren, wie z.B. Entwicklung, IT-Betrieb, Qualitätstechnik und Sicherheit, bekommen durch DevOps die Möglichkeit, sich besser zu koordinieren und zusammenzuarbeiten, um bessere und zuverlässigere Produkte abzuliefern. „Durch die Einführung der DevOps-Kultur mit DevOps-Methoden und -Tools können Teams besser auf die Anforderungen ihrer Kunden reagieren, das Vertrauen in ihre eigenen Anwendungen steigern und Geschäftsziele schneller erreichen“ (Microsoft, 2021a).

7.2 DevOps-Kultur

Alles beginnt mit der Kultur in dem Unternehmen und den Menschen, die an den Prozessen mitwirken. Um die DevOps-Kultur zu etablieren sind Änderungen der Arbeitsweise und Zusammenarbeit der Mitarbeiter nötig. Wenn diese Voraussetzungen geschaffen werden und die DevOps-Kultur umgesetzt wird, sind die Voraussetzungen für enorm leistungsfähige Entwicklungsteams geschaffen.

7.3 DevOps-Methoden

DevOps wird nicht nur durch die DevOps-Kultur in einem Unternehmen umgesetzt, sondern auch durch die Verwendung der DevOps-Methoden. Darunter fallen z.B. das Beschleunigen, Automatisieren oder Optimieren von Prozessen. In diesem Projekt werden vor allem folgende DevOps-Methoden verwendet: Agile Softwareentwicklung, Versionskontrolle und Continuous Integration und Continuous Delivery (CI/CD).

Die agile Softwareentwicklung wurde bereits in Kapitel 3.1 erläutert.

Die Versionskontrolle ermöglicht, den Programmcode online zu speichern, mit mehreren Entwicklern daran zu arbeiten und von diesem verschiedene Versionen zu verwalten. Dabei wird der Änderungsverlauf festgehalten, damit vorherige Versionen wieder hergestellt werden können. Außerdem ermöglicht der Einsatz eines Versionskontrollsystems ein geradlinigen Prozess für das Mergen (verschmelzen) von Codeänderungen in einer Datei und für die Konfliktbehandlung von Änderungen in

früheren Versionen. Diese Methode wird meistens mit einem Versionskontrollsystem wie Git implementiert. Die Entwickler werden bei der Zusammenarbeit unterstützt. Versionskontrollsysteme stellen sodann eine grundlegende DevOps-Methode dar.

Bei der Continuous Integration handelt es sich um eine Softwareentwicklungsmethode, in der Entwickler regelmäßig die Codeänderungen in dem Hauptbranch mergen. Automatisierte Tests werden immer bei einem commit ausgeführt. Dadurch wird im Hauptbranch immer ein stabiler Code sichergestellt.

Unter Continuous Delivery wird das regelmäßige und automatisierte Bereitstellen des aktuellen Codes in der Produktivumgebung verstanden. Durch das automatisierte Bereitstellen können Bereitstellungsfehler verhindert und häufiger neue Versionen veröffentlicht werden.

Beide Methoden zusammen stellen den CI/CD- Prozess dar, welcher dafür sorgt, dass alles zwischen dem Comitten des Codes und der Bereitstellung in der Produktivumgebung vollständig automatisiert ist.

7.4 DevOps-Tools

Zu Beginn wurde das Azure DevOps Board verwendet, um die Anforderungen genau zu definieren. Dieses bietet die Möglichkeit, agil zu arbeiten und Arbeitselemente anzulegen, sich ein Board mit allen Arbeitselementen anzeigen zu lassen oder diese beispielsweise in drei Zustände einzusortieren und Sprints anzulegen.

Als Versionskontrollsystem werden die Azure DevOps Git-Repos gewählt. Diese bieten nun alle Möglichkeiten, welche auch bei Git möglich wären. Darunter zählen die Aspekte, die bereits im Abschnitt *DevOps-Methoden* erläutert wurden. Hierbei werden das Front- und Backend in unterschiedlichen Repositories verwaltet und gespeichert.

Die Azure DevOps Pipelines werden verwendet, um den CI/CD Prozess in diesem Projekt zu implementieren. Beim Erstellen der Pipelines ist das Problem aufgetreten, dass einige Funktionalitäten von den Pipelines nicht umgesetzt werden können, da die Azure DevOps Versionen, die auf den firmeninternen Servern laufen, veraltet und nicht mit .NET kompatibel sind. Aus diesem Grund wird ein eigener Azure DevOps Account angelegt und alles (Boards, Repos und Pipelines) auf den neuen Account übertragen. Zusätzlich wird der Support von Azure DevOps Pipelines angeschrieben, da die Pipelines nicht direkt für den freien Gebrauch zur Verfügung stehen, da sie zum Minen von Kryptowährung verwendet wurden. Anschließend werden die Pipelines freigeschaltet.

8 Frontend

Damit sowohl der Manager als auch der neue Mitarbeiter die Anwendung benutzen können, wird aufbauend auf dem Backend ein Frontend entwickelt. Dieses wird in Form einer Internetseite bereitgestellt.

8.1 Vue.js und Vuetify

Im Frontend wird Vue.js verwendet. Vue.js ist ein clientseitiges JavaScript Framework zum Entwickeln von Single-Page-Webanwendungen. Zusätzlich zu Vue.js wird für das Gestalten der Anwendung Vuetify verwendet. Dabei handelt es sich um eine Vue UI Bibliothek (User Interface) mit vorgefertigten Komponenten. In diesem Projekt wird sich hauptsächlich an die offiziellen Dokumentationen der beiden Produkte gehalten.

8.2 Umsetzung Manager Vue.js

Das Projekt wird mittels der Vue CLI (command-line interface) angelegt, dabei können am Anfang einige Pakete mit angegeben werden, welche im Folgenden mit installiert werden sollen. Hierbei wird zusätzlich das *Router*-Paket ausgewählt. Über die Vue CLI können Pakete auch nachträglich noch auf einfache Weise installiert werden. Nach dem Erstellen des Projektes in der CLI befinden sich in diesem Ordner und vorkonfigurierte Dateien. Dabei muss zunächst nur der „src“-Ordner dieses Projektes berücksichtigt werden. In ihm befindet sich der Ordner „assets“, in welchem im späteren Verlauf z.B. Bilder abgelegt werden können. In dem Ordner „components“ werden die Komponenten für das Projekt abgelegt. Komponenten sind Codeteile, die wiederholt verwendet werden. In dem Ordner „router“ befindet sich eine Datei mit dem Namen „index.js“ in welcher die Logik für das Routing steckt. In dieser wird festgehalten, welche Komponente oder View (im Nachfolgenden erklärt) bei einem bestimmten path in der URL aufgerufen werden soll. Der letzte Ordner ist der „views“-Ordner. In diesem sind die verschiedenen Views enthalten, die die Anwendungen zur Verfügung hat. Die „App.vue“ ist eine der beiden Dateien im „src“-Ordner. Sie ruft die Komponenten auf. Die nächste Datei ist die „main.js“, welche beispielsweise den Router so einbindet, dass dieser überall innerhalb der Vue Instanz funktioniert. Hier können im weiteren Verlauf auch noch andere Pakete eingebunden werden.

Zunächst muss nun eine neue View „Managers.vue“ angelegt werden. In der View sollen alle Daten über die API des Backendes per HTTP GET Request angefragt und angezeigt werden. Jeder .vue Datei beinhaltet dabei drei Abschnitte: ‚template‘, ‚script‘ und ‚style‘. Der ‚template‘-Abschnitt beinhaltet ein HTML-Fragment. Der Abschnitt ‚script‘ befasst sich mit JavaScript und im ‚style‘-Abschnitt befindet sich der Cascading Style Sheets (CSS)-Code für die aktuelle Komponente. Das bedeutet,

dass jede Komponente seinen eigenen Style und die eigene Logik enthält. In dem Abschnitt ‚template‘ wird ein *div*-Tag verwendet, welcher als Parameter eine *v-for*-Direktive enthält, die von Vue vordefiniert ist und analog einer For-Schleife funktioniert. Diese For-Schleife soll alle Manager ausgeben. Zusätzlich muss in dem *div* eine *key*-Direktive von Vue für die *v-for* angegeben werden, die eindeutig sein muss. Dafür wird die *managerId* verwendet. Im ‚script‘-Teil wird in der *data*-Funktion ein leeres *managers* Array angelegt. Dieses wird anschließend in einem Axios HTTP GET request mit dem Ergebnis dieser Anfrage gefüllt und anschließend in dem ‚template‘-Abschnitt in der *v-for*-Direktive entsprechend ausgegeben. Axios verbindet das Back- und Frontend miteinander, indem es per HTTP mit der API des Backends kommuniziert. Siehe Anhang Abbildung 9

Nun soll von der *Manager.vue* aus auch ein neuer Manager angelegt werden können. Dafür wird in der „*Managers.vue*“ oberhalb von der Ausgabe der Manager ein Button hinzugefügt, der den User auf die Seite weiterleiten soll, auf welcher der neue Manager angelegt werden kann. Dafür muss der Router entsprechend angepasst werden, damit das Weiterleiten des Buttons funktioniert. Anschließend wird in der neuen View, der „*AddManager.vue*“, ein Formular in dem Abschnitt ‚template‘ angelegt, in welchem die Informationen für den neuen Manager eingegeben werden können. In dem ‚script‘-Teil wird in der *data* Methode ein Array angelegt. Anders als zuvor enthält es die Attribute vom Manager, welche als leere Strings definiert werden. Diese Attribute werden per *v-model*-Direktive an die *v-text-fields*-Direktive im Formular gebunden. Wird im Formular eins dieser Felder ausgefüllt, ist dieser Wert auch in dem Array gespeichert, da die *v-model*-Direktive den Wert aus dem Array an das Input Feld der *v-text-fields*-Direktive bindet. Dieses Array wird anschließend beim Klicken auf *save* per Axios HTTP POST request ans Backend geschickt und als neuer Manager in der Datenbank gespeichert. Außerdem wird der User beim Klicken auf *save* automatisch wieder zurück auf die „*Managers.vue*“ geleitet. Siehe Anhang Abbildung 10

Als nächstes wird die Funktionalität implementiert, um die angezeigten Manager in der „*Managers.vue*“ auch bearbeiten zu können. Dafür muss in der „*Managers.vue*“ in der *v-for*-Direktive ein neuer Button zum Bearbeiten des Managers hinzugefügt werden, der den User auf eine neue View leiten wird. Dies muss im Router angepasst werden, damit das Routing auf die neue Komponente funktioniert. In der neuen View „*EditManager.Vue*“ wird, wie in der „*AddManager.vue*“, ein Formular erstellt, in dem nun die Informationen des ausgewählten Managers enthalten sind. Beim Klicken auf den Bearbeitungs-Button wird die ID des ausgewählten Managers mitgegeben. Sie wird benötigt, um die Informationen von diesem per Axios HTTP

GET zu erhalten. Die geladenen Daten von dem Manager werden dem User direkt in dem Formular angezeigt, da die Daten über die *v-model*-Direktive verbunden sind. Sie können jetzt beliebig bearbeitet werden. Klickt der User auf den *Save*-Button, werden alle Daten per Axios PUT ans Backend geschickt und in der Datenbank gespeichert. Siehe Anhang Abbildung 11

Um ein Manager aus der Datenbank zu löschen wird in der „Managers.vue“ in der *v-for*-Direktive ein weiterer Button benötigt. Beim Klicken auf das entsprechende Papierkorbicon wird eine Methode aufgerufen, die zusätzlich die Id des zu löschenden Managers als Parameter erhält. In dieser Methode wird also erneut ein Axios Request an das Backend gesendet, um den Manager aus der Datenbank zu löschen. Hier wird der HTTP DELETE Request verwendet.

Nun sind auch alle CRUD Funktionen für den Manager im Frontend implementiert, sodass die Kommunikation über die REST-API des Backends funktioniert. Siehe Anhang Abbildung 12.

8.3 Umsetzung Manager Vuetify

Für das Stylen des Frontends wird Vuetify benutzt. Hierfür können von der Herstellerseite verschiedene Komponenten verwendet werden. In der „App.vue“ wird ein Wireframe implementiert. Zusätzlich wird eine vorgefertigte *App bar* als Navigationsleiste verwendet. In der „Managers.vue“ werden Listen und Buttons von Vuetify genutzt. Bei der „AddManager.vue“ und „EditManager.vue“ werden Formularvorlagen verwendet. Diese Komponenten werden entsprechend angepasst, damit sie in die Onboarding App implementiert werden können.

8.4 Umsetzung Checklist

Im nächsten Schritt folgt die Umsetzung der View für die Checkliste. Dafür wird eine neue .vue Datei angelegt, in dem die schon vorhandene „Home.vue“ in „Checklist.vue“ umbenannt und verwendet wird. Anschließend wird auf der Vuetify Seite nach einer passenden Checklist-Komponente gesucht. Diese wird in den Abschnitt ‚template‘ der Checklist.vue kopiert. Im ‚script‘-Abschnitt muss die *data* Funktion ein leeres *entity*-Array zurückgeben. In einem GET-Request, der wieder über Axios getätigt wird, werden die Checklisteinträge von einer Checkliste angefragt und in das zuvor erstellte *entity*-Array eingetragen. Um einen GET-Request auf genau einer Checkliste tätigen zu können, muss die *getChecklist(id)*-Methode im Backend in der *ChecklistControllers*-Datei entsprechend angepasst werden. Im ‚template‘-Abschnitt kann wieder eine *v-for*-Direktive von Vue verwendet werden, um alle Entitäten zu durchlaufen und in der zuvor ausgewählten Vuetify-Checklisten-Komponente anzei-

gen zu lassen. Auf diese Weise ist im Frontend auch das Ausgeben einer Checkliste für einen Mitarbeiter über die Datenbank bzw. das Backend möglich. Siehe Anhang Abbildung 13

9 Authentifizierung

Das Thema Authentifizierung betrifft in diesem Projekt mehrere Anwendungsbereiche, da die Login Informationen von den Usern nicht in der Internen Datenbank gespeichert werden sollen, sondern die Authentifizierung über die Microsoft Identity Platform, Azure AD, stattfinden soll. Hierzu wird der Authorization Code Flow with PKCE (Proof Key for Code Exchange) verwendet.

9.1 Authentisierung, Authentifizierung und Autorisierung

Bei dem gesamten Prozess, von der Anmeldung bis zur Nutzung der Daten sind die drei Faktoren Authentisierung, Authentifizierung und Autorisierung, welche nach Perseus, 2021 beschrieben sind, von Bedeutung.

Die Authentisierung stellt den Nachweis einer Person dar. Ist die Person wirklich diejenige, die sie vorgibt zu sein? Als Nachweis kann eine geheime Information wie z.B. ein Passwort verwendet werden.

Die Authentifizierung ist die Überprüfung der behaupteten Identität. Dabei werden die Angaben auf ihre Echtheit überprüft. Die Authentifizierung findet zeitlich nach der Authentisierung statt.

Die Autorisierung ist die Einräumung von speziellen Rechten, z.B. wird unterschieden, ob ein User die Rechte erhält, ausschließlich auf Daten der Mitarbeiter zuzugreifen oder ob er darüber hinaus, je nach Position, Zugriff auf die Managersicht erhält.

9.2 OAuth2.0 Authorization Code Flow with PKCE

Bei OAuth2.0 (Open-Authorization) wird auf die Dokumentation von Microsoft, 2021d verwiesen. OAuth2.0 ist ein Protokoll und eine Anmeldemethode, die Identitäten auf Grundlage von bestehenden, verifizierten Authentifizierungen bestätigt. Das Protokoll verwendet eine tokenbasierte Autorisierung und Authentifizierung. Der Microsoft Identity Platform-Endpunkt implementiert Authentifizierung und Autorisierung unter anderem mit diesem Protokoll. In den meisten OAuth2.0-Flows sind vier Parteien an dem Austausch beteiligt. Das vorliegende Projekt befasst sich mit dem Authorization Code Flow with PKCE. Der Authentifizierungsserver (Authorization Server), der Ressourcenbesitzer (Resource Owner), der OAuth-Client (OAuth client) und der Ressourcenserver (Resource Server) bilden diese vier Parteien ab.

Der Authentifizierungsserver stellt die Microsoft Identitätsplattform zur Verfügung. Sie ist für die Sicherstellung der Identität des Benutzers verantwortlich sowie für die Gewährung oder das Vermeiden des Zugriffs auf die Ressourcen. Außerdem stellt er die Tokens aus. Er verarbeitet Informationen des Benutzers, seinen Zugriff und die Vertrauensbeziehungen zwischen den Parteien.

Ressourcenbesitzer ist in diesem Fall der User. Er besitzt die Daten und kann Clients den Zugriff darauf gewähren.

Der OAuth-Client stellt das Frontend dar. Der Benutzer interagiert mit dem Frontend, das Tokens von dem Authentifizierungsserver anfordert. Der Benutzer muss dem Client die Berechtigung zum Zugriff auf die Ressourcen erteilen.

Der Ressourcenserver beschreibt in diesem Projekt das Backend, in dem sich die Ressourcen und Daten befinden. Das Backend vertraut dem Authentifizierungsserver, um das Frontend sicher zu authentifizieren und zu autorisieren. Es werden Bearer-Zugriffstoken verwendet, um sicherzustellen, dass das Frontend berechtigt ist, auf die Ressourcen zuzugreifen.

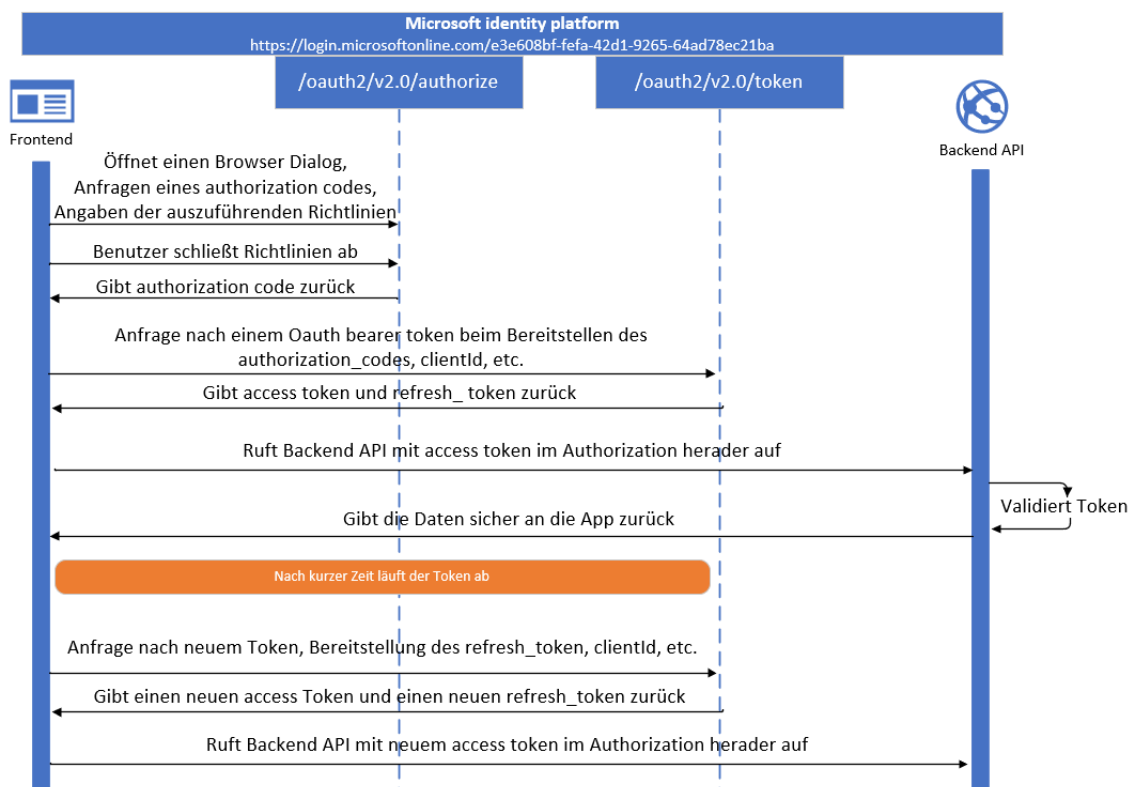


Abbildung 6 Authorization Code Flow with PKCE (Mit geringfügigen Veränderungen von Microsoft, 2021d)

In Abbildung 4 ist ein Sequenzdiagramm zu sehen, welches den Prozess des Authorization Code Flow with PKCE in der zeitlichen Reihenfolge darstellt. Auf der linken Spalte des Diagramms ist eine Zeitlinie zu erkennen, die das Frontend darstellt. Auf der gegenüberliegenden Seite befindet sich die Backend API. Zwischen diesen beiden sind die Endpunkte dargestellt, die von Azure AD bereitgestellt werden.

Zunächst wird der Benutzer von dem Client (Frontend) auf den `/authorize`-Endpunkt von Azure AD weiterleitet, um die Berechtigungen vom Benutzer zu erhalten. Dieser wird dazu aufgefordert, seine Anmeldeinformationen einzugeben, um sich anzumel-

den und die Authentifizierung abzuschließen. Wenn dieser Nutzer die erforderlichen Berechtigungen erfüllt, wird vom *authorize*-Endpunkt ein authorization code (eine Art Passwort) zurück an den Client gesendet. Anschließend erfolgt eine Anfrage des Clients nach einem OAuth bearer Token, mit den nötigen Informationen, wie authorization code, clientId etc. an den */token*-Endpunkt. Von dort wird ein access_token und ein refresh_token zurück an den Client gesendet. Nun kann der Client die API des Backends mit dem access_token im Authorization header aufrufen. Das Backend validiert das Token und schickt bei erfolgreichem Validieren die Daten ans Frontend. Nach ca. 60 Minuten ist ein access_token abgelaufen. Der refresh_token dient aus Sicht des Clients dazu, beim token-Endpunkt ein neues access_token anfragen zu können.

9.3 Azure

Die vorhandenen App registration stellt Authentifizierungs- und Autorisierungsdienste für die Anwendung und die Benutzer über die Microsoft Identity Plattform bereit. Im Azure Portal muss dort ein neues Secret und ein neuer Scope angelegt werden. Der Scope stellt die Berechtigungen für den zu authentifizierenden Benutzer dar.

9.4 Generieren eines Token via Postman und Microsoft Graph

Im nächsten Schritt wird die Authentifizierung in das Backend implementiert. Zunächst wird mittels Postman und Microsoft Graph (RESTful-Web-API) versucht, das Verfahren ohne Backend durchzuspielen. Dabei agiert Microsoft Graph zu Testzwecken als API und wird im Späteren auf die Onboarding API umgestellt.

Um ein Token zu erstellen, werden zunächst Daten, wie zum Beispiel die ‚clientId‘ und die ‚tenantId‘, von der oben genannten App registration benötigt. Danach wird ein neuer GET Request in Postman angelegt. Bei diesem muss im Reiter Authorization der Type OAuth2.0 ausgewählt werden. Nun werden die nachfolgenden Parameter angegeben:

Für den zu erstellenden Token kann ein Name vergeben werden. Außerdem muss ein ‚grant Type‘ (Gewährungsablauf) ausgewählt werden. In dem vorliegenden Projekt ist der ‚grant Type‘ der Authorization Code Flow with PKCE. Um den User nach der Anmeldung wieder zurück auf die angegebene URL zu leiten, wird die ‚redirectUrl‘ angegeben. Diese wird auch in der App registration hinterlegt und muss mit der dort vorhandenen ‚redirectUrl‘ übereinstimmen. Als nächstes müssen zwei Endpunkte für den Authentifizierungsserver und für den Autorisierungsserver angegeben werden. Diese beiden sind in der Übersicht unter Endpoints in der App registration zu finden. Anschließend erfolgt die Übertragung der clientId und des Client Secrets

aus der App registration. Das voreingestellte SHA-256 wird bei der Code Challenge Method nicht verändert. Der Code Verifier bleibt leer, da dieser dann automatisch generiert wird. Schließlich muss der Scope benannt werden, den der User zugewiesen bekommen soll. Dieser zeigt die Berechtigungen auf. Dafür wird in das Feld „https://graph.microsoft.com/User.read“ eingetragen. Dieser Scope ist standardmäßig in der App registration vorhanden und kann somit direkt verwendet werden.

Durch das Klicken auf „Get New Access Token“ öffnet sich eine neue Seite, auf der sich mit einem Microsoft Account angemeldet werden kann. Nach der Anmeldung erhält man den Access Token, welcher nun für den Get Request verwendet werden kann. Bei der Request Url wird die Url von Microsoft Graph angegeben: „https://graph.microsoft.com/v1.0/me/“. Wenn der Access Token gültig ist, werden die Daten des Users im JSON-Format angezeigt. Sollte der Token abgelaufen oder invalide sein oder der User über keine entsprechende Berechtigung verfügen, wird die Fehlermeldung „401 Unauthorized“ ausgegeben. Eine Screenshot ist im Anhang Abbildung 14 zu finden.

9.5 Backend

In dem Backend müssen in zwei Dateien Änderungen vorgenommen werden. Die Erste in der „appsettings.json“ und die Zweite in der „Startup.cs“. Zusätzlich muss das „Microsoft.Identity.Web“ Pakage installiert werden. Es stellt sich zu diesem Zeitpunkt die Frage, was genau von den Parametern in den Dateien angegeben werden muss, da bei den verschiedenen Blogs und Internetseiten verschiedene, aber ähnliche Abschnitte angegeben sind.

Hierzu wird wieder auf die Dokumentation von Microsoft Azure zurückgegriffen. In der „appsettings.json“ müssen drei Variablen aus der App registration angegeben werden: Instance, ClientId und der Tanent. In der „Startup.cs“ muss ein Service zur Authentication mit Hilfe der Daten aus der „appsettings.json“ angelegt werden. In der *Configure*-Funktion wird der Service aufgerufen. Nun kann in einem beliebigen Controller (entweder für einen gesamten Controller oder nur für einzelne Funktionen) das „[Authorize]“-Tag aufgerufen werden. Dadurch muss sich der User automatisch authentifizieren. Anschließend wird in Postman das Erstellen des Tokens so angepasst, dass die Daten vom Backend und nicht vom Microsoft Graph abgerufen werden. Danach kann ein Access Token erstellt und ein GET Request ans Backend geschickt werden. Nun werden die Daten angezeigt. Ohne oder mit einem invaliden Access Token, bekommt Postman die Meldung: *401 Unauthorized* vom Backend zurück.

9.6 Frontend

In diesem Schritt entsteht die Implementierung der Authentifizierung für das Frontend. Für die Authentifizierung im Frontend ist es sinnvoll, die MSAL (Microsoft Authentication Library) Bibliothek von Microsoft zu nutzen. Diese stellt viele wichtige Funktionalitäten zur Verfügung.

Dafür wird ein eigenes „myAuthService.js“ Modul gebaut, das die komplette Login Logik enthalten soll. Nach dem Erstellen es Moduls, wird dieses in das Vue Projekt eingebunden und entsprechend verwendet. Dafür muss in der „App.vue“ in der Navigationsleiste der Web Applikation ein Login und ein Logout Button angelegt werden. Beim Klicken einer dieser Buttons wird entsprechend die *SignIn*-Methode oder die *SignOut*-Methode aus der „myAuthService.js“ aufgerufen. Damit sind die Funktionalitäten des Ein- und Ausloggens implementiert und der Benutzer muss für diesen Prozess seine Anmeldeinformationen seines Microsoft Accounts eingeben. Um diesen Access Token verwenden zu können, wird an dieser Stelle eine weitere Funktionalität benötigt, die den aktuellen Access Token hervorruft oder die einen neuen Access Token anfragt, falls keiner existiert. Hierfür wird aus der „myAuthService.js“ die Funktion *getTokenPopup* aufgerufen. Damit dieser nach einem refresh der Seite nicht verloren geht, muss der Access Token zusätzlich gespeichert werden. Für diesen Zweck wird ein zusätzliches Paket ‚Vuex‘ installiert, das einen Store zur Verfügung stellt, in dem der Access Token abgelegt und bei Bedarf angefragt werden kann. Anschließend muss in allen Axios Anfragen an das Backend der Access Token im Header mitgegeben werden, da das Frontend andernfalls keinen Zugriff auf die Daten bzw. Ressourcen erhält.

Schließlich ist die vollständige Authentifizierung für das Back- und Frontend abgeschlossen. Nur, wenn ein User eingeloggt ist, werden im Frontend Daten angezeigt.

10 Fazit und Ausblick

Im Rahmen der vorliegenden Praxisarbeit konnte der Onboarding Prozess erfolgreich digitalisiert werden. Durch die Entwicklung einer neuen Software ergab sich eine Optimierung des Prozesses. Zusätzlich könnte durch die Einführung der neuen Onboarding-Struktur die Kosten für das Unternehmen reduziert werden.

Für die Entwicklung der Software wurden die vorgestellten Technologien verwendet, die das Onboarding für die neuen Mitarbeiter hinsichtlich Einarbeitung und Eingliederung in die Unternehmensstruktur verbessern. Dies wurde mit Hilfe eines neu entwickelten Back-, Frontends und einer Datenbank realisiert. Mit der Anwendung ergibt sich für den User eine gute Funktionalität und Selbstorganisation. Zu berücksichtigen ist jedoch, dass ein guter Kompromiss zwischen diesen Gesichtspunkten und der persönlichen Betreuung gefunden wird.

Neben dem wirtschaftlichen Aspekt für den Betrieb sollte die Onboarding App auch für die Mitarbeiter einen Vorteil bieten. Ein Vorteil könnte die Entwicklung von Funktionen zur Kommunikationsverbesserungen darstellen. Hierzu zählen Maßnahmen von aktiven, lebendigen Chats wie beispielsweise eine Face-Video Option, die darüber hinaus auch wieder einen Vorteil für die persönliche Betreuung darstellt.

Zusätzlich zu diesen Funktionalitäten kann die Anwendung so erweitert werden, dass sie nicht nur für das Onboarding verwendet wird sondern auch für die Einarbeitung des neuen Mitarbeiters begleitend im Fachbereich. Die Applikation könnte eine Verbindung zu arbeitsrelevanten Repositories und Dokumenten bereitstellen.

Literatur

- Augsten, S. (2020). *Was ist .NET Core?* <https://www.dev-insider.de/was-ist-net-core-a-914978/>
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., James, Grenning, Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). *Das agile Manifest*. <https://agilemanifesto.org/iso/de/manifesto.html>
- Bertelsmeier, B. (2019). *Datenbanken und Informationssysteme, Phasenmodell der DB-Entwicklung + Entity-Relationship-Modellierung*. Technische Hochschule Köln.
- Brenner, D. (2014). *Onboarding: Als Führungskraft neue Mitarbeiter erfolgreich einarbeiten und integrieren* [HBZ: TT050431179; Verfasserangabe: von Doris Brenner; Umfang: IX, 33 S. 4 Abb; Quelldatenbank: DE-605; Besitznachweise: DE-832: <http://lobid.org/organisations/DE-832>; Online-Ressource [Kann nicht per Fernleihe bestellt werden!]; Erscheint auch als: Druck-Ausgabe, 9783658065270]. Springer Fachmedien Wiesbaden. <https://doi.org/10.1007/978-3-658-06528-7>
- Computacenter. (2021a). *Das Unternehmensprofil von Computacenter*. <https://www.computacenter.com/de/%C3%BCber-uns/unternehmensprofil>
- Computacenter. (2021b). *NEXT GENERATION SERVICE DESK*. <https://www.computacenter.com/de/referenzen/computacenter-ngsd>
- Decidalo. (2021). *decidalo – die Branchenlösung für IT-Services und Consulting*. <https://www.data-assessment.com/de/decidalo/>
- Microsoft. (2021a). *Definition von DevOps*. <https://azure.microsoft.com/de-de/overview/what-is-devops>
- Microsoft. (2021b). *Entity Framework Core*. <https://docs.microsoft.com/de-de/ef/core/>
- Microsoft. (2021c). *GUID*. <https://docs.microsoft.com/de-de/dotnet/api/system.guid?view=net-5.0>
- Microsoft. (2021d). *Microsoft Identity Platform und der OAuth 2.0-Autorisierungscodeflow*. <https://docs.microsoft.com/de-de/azure/active-directory/develop/v2-oauth2-auth-code-flow>
- Perseus. (2021). *Authentisierung, Authentifizierung und Autorisierung*. <https://www.perseus.de/wissen/glossar/glossarbegriff/authentisierung-authentifizierung-und-autorisierung/>
- Signavio. (2021). *Business Process Model and Notation (BPMN) — Einführung*. <https://www.signavio.com/de/bpmn-einfuehrung/>

- Srocke, M. D. & Karlstetter, F. (2017). *Was ist eine REST API?* <https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/>
- Wolf, H. & Bleek, W.-G. (2011). *Agile Softwareentwicklung: Werte, Konzepte und Methoden*. dpunkt.verlag GmbH.

Anhang

Der Code für das Backend ist unter:

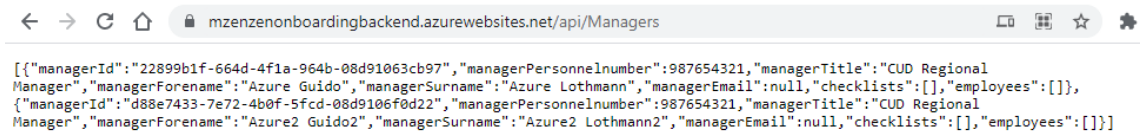
https://dev.azure.com/Marc-KevinZenzen/CC%20Onboarding%20App/_git/Backend
zu finden

Der Code für das Frontend ist unter:

https://dev.azure.com/Marc-KevinZenzen/CC%20Onboarding%20App/_git/Frontend
zu finden.

```
"ConnectionStrings": {  
  //Lokale Datenbank  
  "Database": "Data Source=DE5CD037272B\\ONBOARDING;Initial Catalog=Onboarding;Integrated Security=True"
```

Abbildung 7 Der Connection String, um eine Verbindung vom Backend mit der Datenbank aufbauen zu können.



The screenshot shows a web browser window with the address bar displaying `mzenzenonboardingbackend.azurewebsites.net/api/Managers`. The main content area shows a JSON array of two manager objects. The first object has a manager ID of `22899b1f-664d-4f1a-964b-08d91063cb97`, a personnel number of `987654321`, a title of `CUD Regional Manager`, a forename of `Azure Guido`, a surname of `Azure Lothmann`, and a null email. The second object has a manager ID of `d88e7433-7e72-4b0f-5fcd-08d9106f0d22`, the same personnel number and title, and a forename of `Azure2 Guido2`. Both objects have empty `checklists` and `employees` arrays.

```
[{"managerId":"22899b1f-664d-4f1a-964b-08d91063cb97","managerPersonnelNumber":987654321,"managerTitle":"CUD Regional Manager","managerForename":"Azure Guido","managerSurname":"Azure Lothmann","managerEmail":null,"checklists":[],"employees":[]}, {"managerId":"d88e7433-7e72-4b0f-5fcd-08d9106f0d22","managerPersonnelNumber":987654321,"managerTitle":"CUD Regional Manager","managerForename":"Azure2 Guido2","managerSurname":"Azure2 Lothmann2","managerEmail":null,"checklists":[],"employees":[]}]
```

Abbildung 8 Daten im JSON Format, welche über die REST API vom Backend in der Azure Cloud aufgerufen werden.

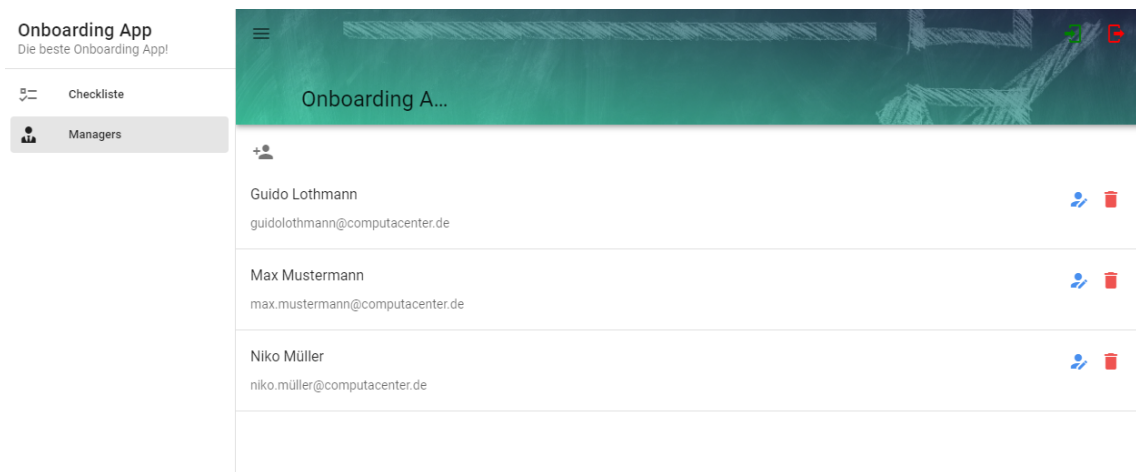


Abbildung 9 Übersicht über aller vorhandenen Manager.

The screenshot displays the 'Onboarding App' interface. On the left, a sidebar contains the app title 'Onboarding App' with the subtitle 'Die beste Onboarding App!', a 'Checkliste' option with a checklist icon, and a 'Managers' option with a group of people icon. The main content area has a green header with a hamburger menu icon, the title 'Onboarding A...', and a close icon. Below the header, there is a form with the following fields: 'Personalnummer' and 'Titel' in the first row; 'Vorname' (containing 'Peter') and 'Nachname' in the second row; and 'E-Mail' in the third row. At the bottom of the form are two buttons: 'CANCEL' and 'SAVE'.

Onboarding App
Die beste Onboarding App!

Checkliste

Managers

Onboarding A...

Personalnummer

Titel

Vorname
Peter

Nachname

E-Mail

CANCEL SAVE

Abbildung 10 Einen neuen Manager erstellen.

The screenshot displays the 'Onboarding App' interface. On the left, a sidebar contains the app title 'Onboarding App' with the subtitle 'Die beste Onboarding Appl', and two menu items: 'Checkliste' (with a checklist icon) and 'Managers' (with a person icon). The main content area has a green header with a hamburger menu icon, a title 'Onboarding A...', and a green plus icon and a red minus icon. Below the header, the form contains the following fields:

Personalnummer 12345	Titel qwe
Vorname Max	Nachname Mustermann
E-Mail max.mustermann@computacenter.de	

At the bottom of the form are two buttons: a dark grey 'CANCEL' button and a green 'SAVE' button.

Abbildung 11 Einen vorhandenen Manager bearbeiten.

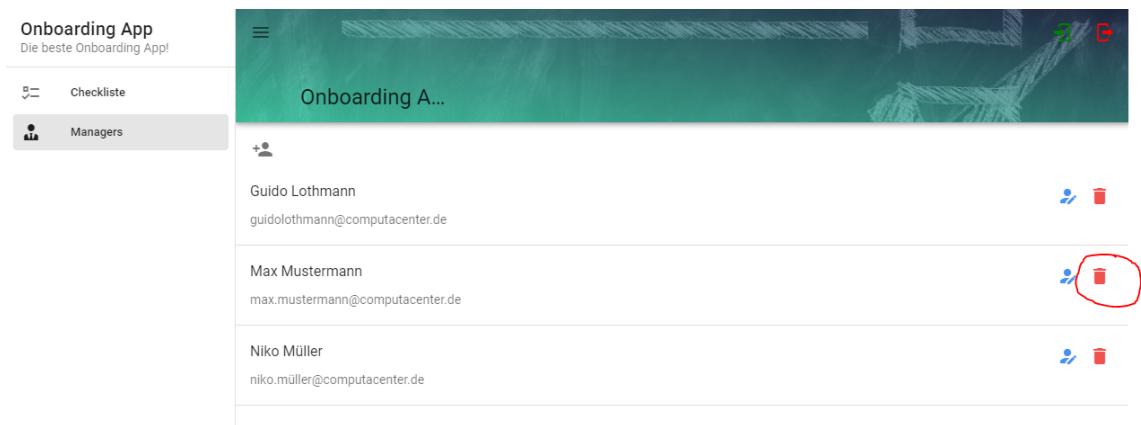


Abbildung 12 Einen Manager löschen.

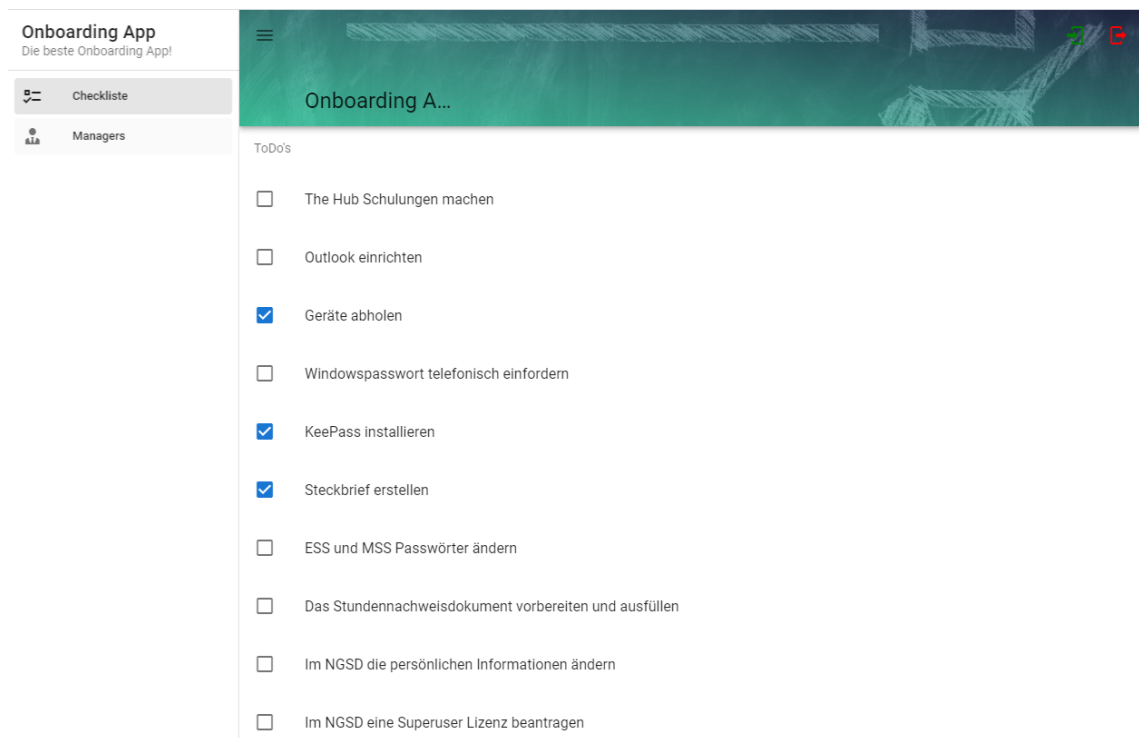


Abbildung 13 Die abzuarbeitende Checkliste für die neuen Mitarbeiter.

Configuration Options ●

Advanced Options

Token Name

2

Grant Type

Authorization Code (With PKCE) ▼

Callback URL ⓘ

http://localhost:8080/

☐ Authorize using browser

Auth URL ⓘ

https://login.microsoftonline.com/e3e608b1...

Access Token URL ⓘ

https://login.microsoftonline.com/e3e608b1...

Client ID ⓘ

e77281f7-d35a-46ee-acfa-b0a4104193 ... ⚠

Client Secret ⓘ

Q0-.S7NFLZSFN_Uy9G34_1As0xV2q.52 ... ⚠

Code Challenge Method ⓘ

SHA-256 ▼

Code Verifier ⓘ

Automatically generated if left blank

Scope ⓘ


https://graph.microsoft.com/User.read

State ⓘ

State

Client Authentication

Send as Basic Auth header ▼

 Clear cookies ⓘ

Get New Access Token

Abbildung 14 Manuell einen Access Token für die Authentifizierung via Backend erstellen.