Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Industrie-4.0-Informatik

Bachelorarbeit

von

Lukas Rupp

Konzeption und prototypische Umsetzung einer internen Webanwendung zur Auftragserstellung als Modernisierungsmaßnahme

Conception and Prototypical Implementation of an Internal Web Application for Order Creation as a Modernization Purpose

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Industrie-4.0-Informatik

Bachelorarbeit

von

Lukas Rupp

Konzeption und prototypische Umsetzung einer internen Webanwendung zur Auftragserstellung als Modernisierungsmaßnahme

Conception and prototypical Implementation of an internal Web Application for Order creation for Modernization purposes

Bearbeitungszeitraum: von 02. Oktober 2023
bis 09. Februar 2024

1.Prüfer: Prof. Dr.-Ing. Christoph P. Neumann
2.Prüfer: Prof. Dr.-Ing. Gerald Pirkl

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektro- und Informationstechnik

Eigenständigkeitserklärung gemäß § 27 (8) ASPO

Name und Vorname

des Studenten: **Rupp, Lukas**

Studiengang: **Industrie-4.0-Informatik**

Ich bestätige, dass ich die Bachelorarbeit mit dem Titel:

Konzeption und prototypische Umsetzung einer internen Webanwendung zur Auftragserstellung als Modernisierungsmaßnahme

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 09. Februar 2024

Unterschrift:

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektro- und Informationstechnik

Bachelorarbeit Zusammenfassung

Student (Name, Vorname):	**Rupp, Lukas**
Studiengang:	Industrie-4.0-Informatik
Aufgabensteller, Professor:	Prof. Dr.-Ing. Christoph P. Neumann
Durchgeführt in (Firma):	Siemens AG, Amberg
Betreuer in Firma:	Manfred Schmidt
Ausgabedatum:	02. Oktober 2023
Abgabedatum:	09. Februar 2024

Titel:

Konzeption und prototypische Umsetzung einer internen Webanwendung zur Auftragserstellung als Modernisierungsmaßnahme

---

Zusammenfassung:

Das Ziel dieser Arbeit ist die Konzeption und prototypische Implementierung einer internen Webanwendung zur Auftragserstellung als Modernisierungsmaßnahme. Die Vorgehensweise bei der Implementierung soll als Vorarbeit für die Weiterentwicklung des Systems dienen. Das erste Kapitel widmet sich der Zielsetzung für die Modernisierung und Konzeption der Webanwendung. Im zweiten Kapitel werden grundlegende Begriffe zum Verständnis der nachfolgenden Kapitel erläutert. Im darauffolgenden Kapitel wird die Implementierung der zu modernisierenden Webanwendung beschrieben und der IST-Zustand festgehalten. Im vierten Kapitel folgt die Anforderungsanalyse an das zu entwickelnde System, inklusive des Anwendungsszenarios. Anschließend werden die dafür zur Verfügung stehenden Möglichkeiten zur Systemumsetzung erklärt und eine Möglichkeit gewählt. In der darauffolgenden Beschreibung des Lösungskonzepts wird der verwendete Technologie-Stack vorgestellt, sowie die Designentwürfe für die Webanwendung. Darauf aufbauend folgt die Implementierung des Lösungskonzeptes, in welcher dabei verwendete Design-Muster und Architekturen beschrieben werden. Abschließend werden die Ergebnisse im Kapitel acht evaluiert, bevor in Kapitel neun eine Zusammenfassung und ein Ausblick auf die Zukunft der Webanwendung folgt.

Schlüsselwörter: Webanwendung, SAP, Responsive Webdesign, Mockup, Angular, .NET

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektro- und Informationstechnik

Abstract:

The aim of this thesis is the conception and prototypical implementation of an internal web application for order creation as a modernization measure. The implementation procedure is intended to serve as preparatory work for the further development of the system. The first chapter is dedicated to the objectives for the modernization and conception of the web application. The second chapter explains basic terms that help to understand the following chapters of the work. This is followed by a description of the implementation of the web application to be modernized. The fourth chapter analyses the requirements of the system to be developed, including the application scenario. The available options for system implementation are then explained and one option is selected. In the following description of the solution concept, the technology stack used and the design drafts for the web application are presented. This is followed by the implementation of the solution concept, in which the design patterns and architectures used are described. Finally, the results are evaluated in chapter eight, followed by a summary and an outlook on the future of the web application in chapter nine.

Keywords: web application, SAP, responsive web design, mockup, Angular, .NET

Inhaltsverzeichnis

1. Einleitung

Durch die Konstruktion des Zeigertelegrafen gelang es dem 30-jährigen Erfinder Werner von Siemens den Grundstein der heutigen Siemens AG zu legen. Er gründete am 12. Oktober 1847 zusammen mit dem Feinmechaniker Johann Georg Halske die „Telegraphen-Bauanstalt von Siemens & Halske“. In einer Zehn-Mann-Werkstatt eines Berliner Hinterhofes nahm er anschließend den Betrieb auf. Nur ein Jahr später (1848) erhielten sie bereits den Auftrag für den Bau der ersten Ferntelegrafenverbindung Europas zwischen Berlin und Frankfurt am Main. Die 670 Kilometer lange, weitgehend unterirdische, Strecke wurde schon im Februar 1849 fertiggestellt. In den folgenden Jahren entstanden durch eine erhöhte Auftragslage weitere Tochtergesellschaften in Russland und England [1].

Mittlerweile ist Siemens ein internationales Unternehmen mit Standorten in 193 Ländern und ca. 303.000 Mitarbeitern. Dabei ist Siemens in sechs verschiedenen Sektoren tätig. Diese sind Digital Industries, Smart Infrastructure, Mobility, Siemens Advanta, Portfolio Companies und Siemens Healthineers. Somit ist Siemens als Weltkonzern in fast allen Bereichen vertreten [2].

Der Standort Amberg besteht seit 1948 und begann mit dem Bau des Gerätewerks, welches mittlerweile Teile für die industrielle Schalttechnik produziert. Außerdem werden dort Leistungsschalter für die industrielle Anwendung entwickelt. Im später gebauten Elektronikwerk entsteht die Steuerungsfamilie der SIMATIC S7 und das Beobachtungssystem SIMATIC HMI. Der mehrfach preisgekrönte Standort Amberg gilt als Vorzeigewerk mit modernster IT und Fertigungstechnologie und ist somit ein Paradebeispiel für Industrie 4.0 [3]. In den zusammenhängenden Standorten Amberg/Cham werden rund 5000 Mitarbeiter zur Fertigung von Niederspannungsschalttechnikelementen sowie speicherprogrammierbaren Steuerungen beschäftigt. Im letzteren Bereich ist der Standort Amberg ebenfalls Weltmarktführer.

Auch mit dem Bau des Besucherzentrums zeigt sich der Standort Amberg erneut von seiner besten Seite und präsentiert neue Technologien rund um die Automatisierung und Digitalisierung bzw. der Industrie 4.0 für Gäste aus der ganzen Welt.

1.1 Abteilung

Die Werksinfrastruktur (WI) ist für das Planen, Entwerfen und Realisieren von Projekten in Bereichen der Büros, Labore und der Fertigung des Standortes zuständig. Dazu gehört die Büro- und Werksplanung allgemein, das Beschaffen von technischem Equipment und Medientechnik sowie die Betreuung der Versorgungstechnik im Werk. Des Weiteren gehört die Instandhaltung und Prüfung von Betriebsmitteln, wie zum Beispiel Leitern oder Gabelstaplern, zu den wichtigsten Aufgaben der WI. Die Funktionsfähigkeit und Sicherheit der Maschinen und Anlagen muss stets gewährleistet sein, um Ausfälle sowie Unfälle zu vermeiden.

Die Verwaltung der Instandhaltung erfolgt über SAP ERP. SAP bzw. „Systemanalyse Programmentwicklung“ ist ein Programm zur Steuerung von Geschäftsprozessen. Es bietet eine Lösung für zentrales Datenmanagement an, welche unternehmensweit genutzt werden kann [4]. In den SAP-Datenbanken werden zur Instandhaltung beispielsweise alle technischen Plätze oder Arbeitsplätze verwaltet. Diese sind in die Bereiche der Mechanik und Elektrik aufgeteilt. Die verschiedenen bereits kategorisierten Aufgaben werden an die dafür zuständigen technischen Plätze weitergeleitet und dort dann auf die verschiedenen Mitarbeiter der Werkstatt aufgeteilt. Darüber hinaus werden Informationen über alle Betriebsmittel gespeichert. Dazu gehören unter anderem deren Wartungspläne. Auch Personendaten oder wichtige Daten für die kaufmännische Abteilung werden hier verwaltet. Zusätzlich können die von den Mitarbeitern des Standortes angelegten „Stör- bzw. Kleinaufträge“ hier gesichtet und verwaltet werden. Der „Stör- bzw. Kleinauftrag“ dient der Beauftragung von Kleintransporten, Bestuhlungen, Reparaturen, Reinigungen oder weiteren Instandhaltungsthemen bis zu einem Betrag von 500€. Somit können administrative Aufgaben am Standort einfach und zentral verwaltet, an die Werksinfrastruktur weitergeleitet und dort anschließend bearbeitet werden. Ein Auftrag kann über eine siemensinterne Webanwendung von allen Standortmitarbeitern angelegt werden und soll im Rahmen dieser Bachelorarbeit modernisiert werden.

1.2 Motivation

Die fortschreitende Digitalisierung und der stetige Wandel der technologischen Landschaft haben einen maßgeblichen Einfluss auf die Entwicklung und den Betrieb von Webanwendungen. Um den wachsenden Anforderungen der Benutzer gerecht zu werden, stehen Unternehmen ständig vor der Herausforderung, ihre Online-Plattformen zu optimieren. Im Fokus dieser Bachelorarbeit steht die Modernisierung einer bestehenden Webanwendung in der Werksinfrastruktur der Siemens AG Amberg. Die aktuelle Implementierung der Webanwendung weist mehrere Einschränkungen auf, die nicht nur die Leistungsfähigkeit und Benutzerfreundlichkeit beeinträchtigen, sondern auch Auswirkungen auf die Effizienz der betrieblichen Abläufe haben.

Zu den Anforderungen an aktuelle Webanwendungen gehört unter anderem eine dauerhafte Verfügbarkeit. Ein modernes Webanwendungs-Design bietet dem Nutzer außerdem eine gute Benutzererfahrung und eine dementsprechende Benutzbarkeit, sodass diese häufiger oder vor allem ohne negative Gedanken verwendet wird. Zu weiteren Anforderungen an moderne Webanwendungen gehören außerdem der Besitz einer hohen Zuverlässigkeit sowie Effizienz und gutes Leistungsverhalten. Auch eine plattformübergreifende Nutzbarkeit der Webanwendung sollte möglich sein. Es ist wichtig, dass moderne Webseiten auf allen Endgeräten und vor allem auf jedem Bildschirm geöffnet und angezeigt werden können. Zur Erfüllung dieser Anforderung spielen Flexibilität und Skalierbarkeit eine große Rolle [6].

1.3 Zielsetzung

Das übergeordnete Ziel der Bachelorarbeit ist es ein Konzept für eine anwenderfreundliche und zeitgemäße Webanwendung zu entwickeln. Dabei sollen sowohl ein modernes Design als auch eine optimale Benutzererfahrung im Fokus stehen. Ein zentraler Aspekt ist hierbei die mobile Nutzbarkeit, wodurch den Nutzern ermöglicht wird, Aufträge von unterwegs zu erstellen. Des Weiteren sind die Integration der Anmeldung mittels firmeninternen Microsoft-Accounts sowie die Entwicklung einer bisher fehlenden Plattform zur Auswertung der Auftragsdaten mit Zugangsbeschränkung Hauptziele dieser Arbeit.

Die finale Produktentwicklung erfolgt in enger Zusammenarbeit mit dem Auftraggeber. Hierbei wird die bisherige Implementierung der Webanwendung analysiert, um spezifische Anforderungen und Verbesserungspotenziale zu identifizieren. Der darauffolgende Vergleich verschiedener Umsetzungsmöglichkeiten berücksichtigt vor allem den Aspekt der Umsetzung anhand der vorgegebenen Anforderungen. Abschließend erfolgt die prototypische Umsetzung, welche die Mindestanforderungen gemäß den ermittelten Anforderungen erfüllt und einen Ausblick auf zukünftige Entwicklungen ermöglicht.

2. Grundlagen

In diesem Kapitel werden grundlegende Begriffe und Konzepte erklärt, welche zum Verständnis der Bachelorarbeit notwendig sind.

2.1 Webanwendung

Eine Webanwendung ist ein Anwendungsprogramm, welches auf einem Webserver gespeichert ist und über einen Webbrowser aufgerufen wird. Zu den gängigen Webbrowsern gehören zum Beispiel Chrome, Firefox, Internet Explorer/Edge oder auch Safari. Da auf Webanwendungen auch über andere Möglichkeiten, wie Skripte oder die Kommandozeile, zugegriffen werden kann, wird im Allgemeinen vom Client gesprochen. Das HTTP (Hypertext Transfer Protocol) bzw. das darauf aufsetzende HTTPS (Hypertext Transfer Protocol Secure) werden für die Kommunikation zwischen Server und Client verwendet [8, S. 2]. Im Webbrowser bzw. Client wird die Adresse oder URL (Uniform Resource Locator) der Webanwendung eingegeben. Daraufhin sendet der Webbrowser eine Anfrage an den Webserver, auf welchem die Webanwendung gespeichert ist. Der Browser antwortet auf die Anfrage, indem er statische Daten an den Webbrowser zurücksendet (vgl. Abbildung 1).

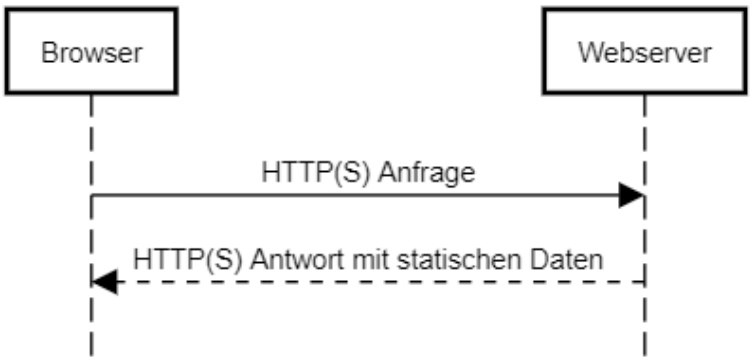


Abbildung 1: Kommunikationsdiagramm zwischen Webbrowser und Webserver

Die statischen Daten bestehen dabei aus HTML-, JavaScript- und CSS-Code, welcher durch den Browser interpretiert und dargestellt wird. HTML bezeichnet die Hypertext Markup Language und dient der Strukturierung von textbasierten Inhalten eines Webdokuments. Der JavaScript-Code ist für den Aufruf serverseitiger Webdienste verantwortlich und implementiert die Logik der Webanwendung. CSS sind die Cascading Style Sheets und dienen zur Anpassung des Designs einer Webanwendung über Style-Elemente [8, S. 2]. Bei der Übertragung der Webanwendungen wird zwischen statischen und dynamischen Seiten unterschieden. Eine statische Seite wird umgehend vom Webserver, ohne Änderungen, an den Webbrowser übertragen. Eine dynamische Seite wird stattdessen vor Übertragung an den Webbrowser an eine spezielle Software, den Anwendungsserver, zur Fertigstellung bzw. Änderung der Seite geleitet [9, S. 347]. Der Webserver kommuniziert anschließend mit Datenbanken, auf welchen alle Informationen, wie beispielsweise Nutzerdaten, Standortdaten oder Materialdaten, gespeichert werden.

Die Präsentationsebene bzw. das Frontend der meisten Webanwendungen wird in Programmiersprachen wie JavaScript, HTML5 und CSS geschrieben. Für das Backend, welches für die serverseitige Programmierung zuständig ist, werden Sprachen wie Python, C#, Java oder Ruby verwendet [9, S. 347]. Webanwendungen, die in HTML5 geschrieben sind, können Daten auch kurzfristig lokal zwischenspeichern. Dadurch können sie auch teilweise offline genutzt werden. Sobald die Verbindung zum Internet bzw. zum Webserver wieder steht, werden anschließend alle lokalen Daten synchronisiert. Der Vorteil von Webanwendungen liegt in der Abrufbarkeit auf allen Betriebssystemen und Geräten, welche über einen, von der Webanwendung unterstützten, Webbrowser verfügen. Des Weiteren kann der Nutzer, ohne einen Download von externer Software, stets auf die aktuelle Version bequem von überall zugreifen. Als ein Nachteil von Webanwendungen gilt der eingeschränkte Zugriff auf Systemressourcen, wodurch die Leistungsfähigkeit begrenzt werden kann. Zusätzlich können lokale Daten bei einem Absturz des Browsers nicht gespeichert werden.

2.2 SAP

Das im Jahr 1972, durch fünf IBM-Programmierer, gegründete Unternehmen SAP ist heute einer der weltweit führenden Anbieter von Software für die Steuerung von Geschäftsprozessen. SAP bzw. „Systemanalyse Programmentwicklung“ entwickelt Lösungen, welche effektive Datenverarbeitungen und den Informationsfluss in Unternehmen erleichtern sollen. Durch ein schnelles Wachstum konnte SAP zu einem multinationalen Konzern mit Sitz in Walldorf und weltweit mehr als 105.000 Mitarbeitern wachsen [4]. Die Grundidee des Unternehmens bestand darin, eine einzige betriebswirtschaftliche Standardsoftware zu entwickeln, welche alle Bereiche der Thematik abdeckt und dem Nutzer eine einheitliche Struktur und Nutzeroberfläche bietet. Bereits 1979 wurde das R/2-System für große Datenserver (Mainframes) eingeführt. Das „R“ steht dabei für „Realtime“ bzw. „Echtzeit“. Bis 1989 wurde es weltweit bereits bei über 1200 Kunden installiert. Fast 15 Jahre später, im Jahr 1992, wurde das R/3-System, welches auf der Server-Client-Technologie basiert, von SAP freigegeben. Zwei Jahre später wurde SAP mit beiden Systemen weltweiter Marktführer im Bereich betriebswirtschaftlicher Software, auch ERP-Software (Enterprise Resource Planning) genannt [13, S.1 ff.]. Dazu gehören alle Kerngeschäftsbereiche wie Beschaffung, Produktion, Materialwirtschaft, Marketing, Vertrieb, Finanzwesen und Personalwesen. Die nächste Generation der SAP-ERP-Software erfolgte durch SAP S/4 HANA. Durch die Realisierung in der Cloud können riesige Datenmengen in Echtzeit verarbeitet werden. Des Weiteren lassen sich neue Technologien, wie künstliche Intelligenzen (KI), einbinden. Bei der veralteten, dezentralen Datenhaltung wurden die jeweiligen operativen Daten der verschiedenen Geschäftsbereiche auf separaten Datenbanken gespeichert. Dadurch ist es Mitarbeitern nicht möglich abteilungsübergreifend auf Informationen zuzugreifen. Zusätzlich erhöhen doppelt angelegte/gespeicherte Daten die Speicherkosten und das Risiko von Datenfehlern. Durch die SAP-Software wird ein zentrales Datenmanagement geboten, welches unternehmensweit eine einheitliche Sicht auf alle Daten ermöglicht. Komplexe Geschäftsprozesse können somit effizienter verwaltet werden und es kann ein einfacher Echtzeit-Zugriff im gesamten System ermöglicht werden [4].

2.3 Business Server Page (BSP)

Business Server Pages sind ein von SAP entwickeltes Programmiermodell, das serverseitiges Scripting für die Entwicklung, Gestaltung und Implementierung von Webanwendungen ermöglicht. Durch das serverseitige Scripting wird ein direkter Zugriff auf sämtliche Elemente im Applikationsserver sowie auf von SAP bereitgestellte Funktionsbausteine, Datenbanktabellen oder andere Objekte ermöglicht. Ein zentrales Merkmal von Business Server Pages ist die Möglichkeit zur Trennung von Präsentations- und Business-Logik. Dies ermöglicht die Anwendung alternativer Frontend-Technologien [14].

Die Benutzeroberfläche setzt sich aus statischen und dynamisch generierten Webseiten zusammen, die mithilfe des serverseitigen Scripting erst zur Laufzeit der Webseite erzeugt werden. Dabei können die erstellten Seiten entweder eine gewisse Ablauflogik oder Views und Controller enthalten, wenn das MVC-Design-Pattern (Model-View-Controller) angewendet wird [16].

Wie in anderen SAP „Vorgängen“ auch, setzt sich die Business-Logik aus sogenannten BAPIs (Business Application Programming Interface) zusammen. Diese dienen als Programmierschnittstellen. Des Weiteren können auch SAP-Funktionsbausteine und -Klassenbibliotheken Teil einer BSP-Applikation sein.

Der grundsätzliche Aufbau von BSP-Applikationen besteht unter anderem aus den Business Server Pages (vgl. Abbildung 2), welche verschiedene Webseiten darstellen und bei der Verwendung der Applikation im Browser angezeigt werden. Diese können aus statischem HTML-Code, dynamischem Scripting-Code in ABAP (Advanced Business Application Programming) oder JavaScript bestehen. Der Scripting-Code wird dann auf dem Server ausgewertet und anschließend an den Browser weitergeleitet. ABAP ist die von SAP entwickelte Programmiersprache, welche für Anwendungen im SAP-Umfeld benutzt wird [36]. Falls die Applikation eine Ablauflogik besitzt, besteht diese aus einfachen Seiten mit Eventhandlern, aber ohne viel Anwendungslogik oder Visualisierungselementen.

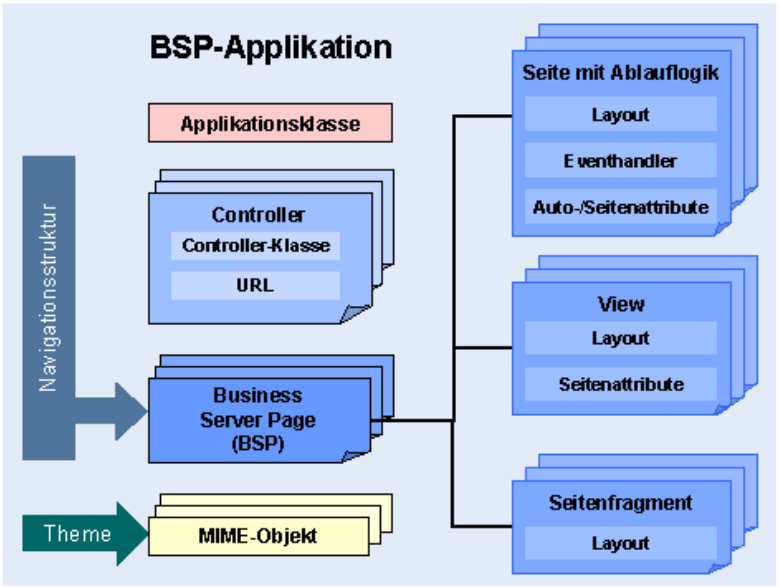


Abbildung 2: Aufbau einer BSP-Applikation [16]

Des Weiteren gibt es die Möglichkeit BSPs aus Views aufzubauen. Die Views dienen dann der Visualisierung der Daten und erfordern einen Aufbau nach dem Model-View-Controller (MVC)-Design (vgl. Abbildung 3). Eine weitere Möglichkeit ist die Visualisierung mit Seitenfragmenten. Diese unterscheiden sich von normalen BSPs durch eine spezielle Kennzeichnung, welche dafür sorgt, dass sie auch global von anderen BSPs verwendet werden können [14].

Zur Verarbeitung der Daten gibt es den Controller, welcher die Business-Logik und Anwendungsdaten beinhaltet. Der Controller ist die Instanz einer zentralen Controller-Klasse, welche von einer SAP-Basisklasse abgeleitet wird. Über eine URL kann der Controller von außen angesprochen werden. Im MVC-Design (vgl. Abbildung 3) ist er für die Bereitstellung von Daten, die Auswahl des richtigen Layouts und das Starten der Datenintialisierung zuständig [17]. Zusätzlich wertet er die Daten der eingehenden Anfrage auf Grundlage eines Modells aus und wählt entsprechend dieser Werte die passende View für die Antwort an den Benutzer [14]. Das Modell wird dabei für die Datenbeschaffung aller notwendigen Anwendungsdaten von der Datenbank benutzt. Es entspricht der verwendeten Applikationsklasse und repräsentiert die internen Datenstrukturen [17]. Applikationsklassen dienen der Kapselung von Business-Logik. Sie werden durch globale ABAP-Klassen realisiert und haben Zugriff auf die Business-Daten. Jede Seite einer BSP-Applikation kann dann auf Komponenten wie Attribute, Methoden etc. der Klassen zugreifen [18]. In einer Navigationsstruktur kann zusätzlich festgelegt werden, von welcher Seite und unter welcher Navigationsanfrage, zu welcher Folgeseite navigiert werden soll. Mit MIME-Objekten (Multipurpose Internet Mail Extensions) werden Grafiken, Style-Sheets, Audios und Videos verwaltet [14].

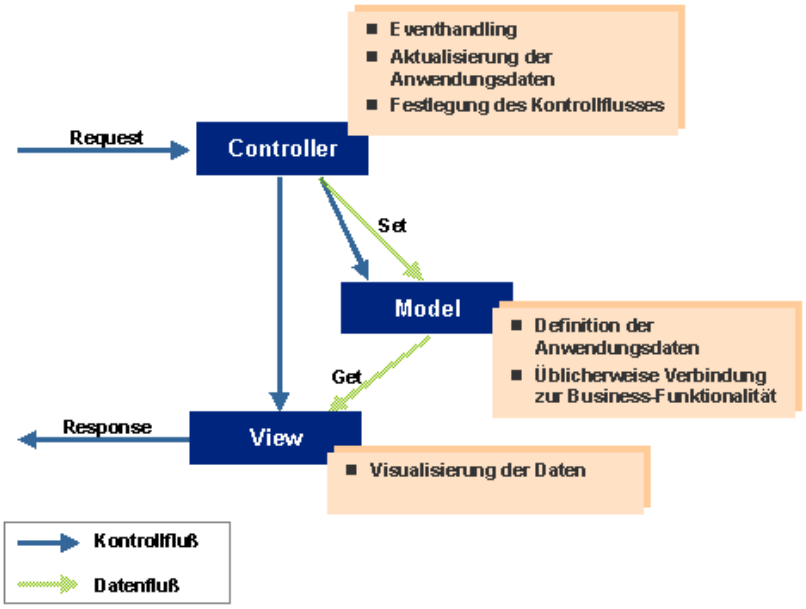


Abbildung 3: Abbildung der Model-View-Controller-Architektur bei BSP [17]

Business Server Pages sind im Wesentlichen geschlossene und funktionale Anwendungen, die vergleichbar mit herkömmlichen SAP-Transaktionen sind und es ermöglichen, Änderungen in der SAP-Datenbank vorzunehmen. Auf diese wird jedoch im Gegensatz zu den Transaktionen nicht über das SAPGUI bzw. die SAP-Desktopanwendung, sondern über den Webbrowser via URL/Adresse mittels HTTP bzw. HTTPS zugegriffen [15].

2.4 Single Page Application

Das Grundprinzip einer Single Page Application, kurz SPA, liegt darin, Informationen nur auf einer einzigen Webseite anzuzeigen. Demnach verbleibt man, selbst wenn sich durch eine Eingabe im Menü die Webseite verändert, immer noch auf derselben Seite, mit unveränderter URL [19, S. 311]. Die SPA-Webanwendungen laden dabei bereits alle Ressourcen bei der ersten Serveranfrage. Anschließend werden je nach Nutzerinteraktion verschiedene Komponenten unabhängig ausgetauscht bzw. angepasst, ohne die ganze Webanwendung neu laden zu müssen [20, S. 2876]. Der zu aktualisierende Inhalt wird über APIs modifiziert. Eine API, bzw. Application Programming Interface, stellt dabei eine Programmierschnittstelle dar [21, S. 141].

Bei der herkömmlichen Webanwendung sendet der Benutzer bzw. Webbrowser eine Anfrage an den Webserver. Daraufhin erhält er die Webseite als HTML-Dokument mit allen Bildern, CSS- oder Scripting-Code. Wenn der Benutzer nun mit dieser Seite agiert, z.B. einen Knopf anwählt, bekommt der Webserver eine erneute Anfrage und antwortet auf diese mit einem neuen HTML-Dokument. Der Webbrowser muss dieses HTML-Dokument nun erneut laden und interpretieren bzw. darstellen [20, S. 2877].

Bei Single Page Anwendungen werden dem Webbrowser mit der ersten Antwort auf seine Anfrage direkt alle Ressourcen übergeben. Dadurch können, wenn der Benutzer nun mit der Webanwendung interagiert, ohne die Seite neu zu laden, gewisse Änderungen direkt an der Webseite durch Scripts im Webbrowser dargestellt werden. Falls dem Webbrowser dennoch Ressourcen fehlen, fragt er diese beim Webserver an. Dieser antwortet nun jedoch nur mit einer JSON-Datei, welche spezifisch alle fehlenden Informationen beinhaltet (vgl. Abbildung 4). Demnach wird nur ein Teil der Seite verändert oder ausgetauscht [20, S. 2877].

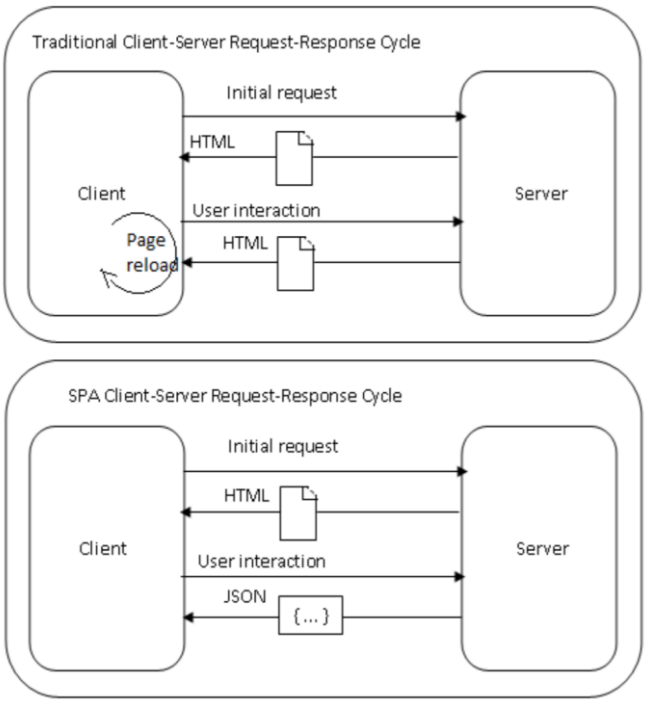


Abbildung 4: Kommunikationsablauf bei herkömmlichen Webseiten (oben) und bei Single Page Applications (unten) [20, S. 2877]

Die Technologie der Single Page Applications ermöglicht es, durch einen wesentlich geringeren Datenaustausch, die Bandbreite sowie die Serverbelastung zu verringern. Dadurch steigt wiederum die Geschwindigkeit. Des Weiteren kann durch SPAs die Benutzererfahrung erheblich gesteigert werden, da sich die Webanwendung, ohne neu zu laden, fast wie eine native Anwendung anfühlt. Die Seite wirkt dynamischer und es existieren keine Wartezeiten. Da der Webserver nur Daten liefert, welche anschließend im Browser verarbeitet werden, kann die Entwicklung des Frontend bzw. des Backend getrennt stattfinden, wodurch die generelle Entwicklung vereinfacht wird.

Single Page Applications weisen im Gegensatz zu anderen Webseiten eine schlechte Search Engine Optimization (SEO) auf. Da SPAs nur eine einzige URL besitzen, entstehen Einschränkungen bei der Suchmaschinenoptimierung. Sie besitzen eine schlechte Indexierung und es fehlen gute Analysen, Metadaten und eindeutige Links. Wenn auf einer Single Page Application der Zurück-Knopf benutzt wird, erfolgt beispielsweise die Weiterleitung auf die davor geladene Website.

2.5 AJAX

Der Begriff AJAX steht für „Asynchronous JavaScript and XML“ und bezeichnet einen Zusammenschluss mehrerer Technologien, welcher 2005 von Jesse J. Garret entwickelt wurde [33, S. 387]. Dieser führte zu einer erheblichen Verbesserung der damals aktuellen Client-Server-Kommunikation. Durch die Kombination von Techniken wie XHTML, CSS, DOM, XML, XMLHttpRequest und JavaScript konnte ein dynamischer Datenaustausch zwischen Webclient und Webserver, ohne lange Wartezeiten, ermöglicht werden [32, S. 25]. Eine AJAX-Anwendung kann ein desktoptypisches Verhalten, wie beispielsweise einen Doppelklick, aufweisen. Der größte Vorteil von AJAX ist, dass Webseiten nach Nutzereingabe nicht mehr vollständig neu laden müssen. Die Eingabe des Benutzers wird mit AJAX zum Server gesendet und kann dort weiterverarbeitet werden, ohne dass der Benutzer es merkt [32, S. 31]. Dadurch muss nicht die ganze Webseite, sondern nur der für die Benutzereingabe relevante Teil, neu geladen werden. Verantwortlich dafür ist die AJAX-Engine (vgl. Abbildung 5). Diese ist sowohl für die Darstellung der Benutzeroberfläche als auch für die Kommunikation mit dem Server zuständig [35].

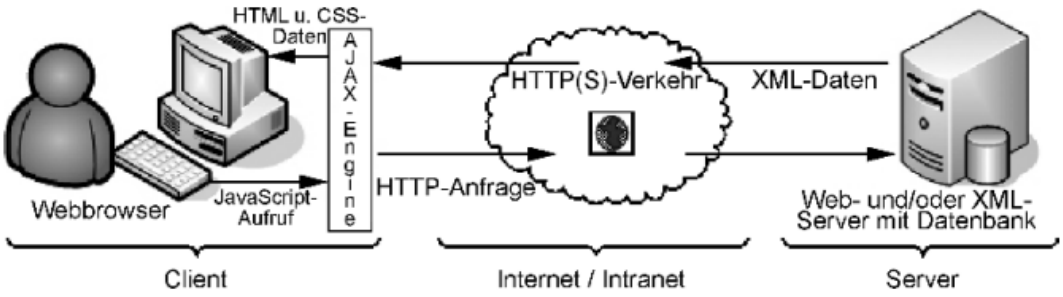


Abbildung 5: Kommunikationsablauf einer AJAX-Anwendung [32, S. 27]

Die Anfragen an den Server werden zunächst an die AJAX-Engine geleitet. Bei Anfragen, wie beispielsweise Gültigkeitsprüfungen, welche nicht zwingend an den Server geleitet werden müssen, erfolgt die Aufbereitung direkt in der AJAX-Engine bzw. im Webbrowser.

Falls die Anfrage doch relevant für den Webserver ist, findet die Kommunikation asynchron statt. Der Webserver sendet eine mit der AJAX-Engine kompatible Antwort, welche dort im Hintergrund verarbeitet wird. Anschließend kann diese auf der Benutzeroberfläche nachgeladen bzw. „geparsed“ werden [32, S. 33]. Weitere Vorteile, die durch AJAX entstehen, sind außerdem eine Verringerung der Netzwerkauslastung sowie eine Performancesteigerung, aufgrund des nicht benötigten Neuladens der ganzen Webseite [33, S. 400].

2.6 Rest

Rest bzw. „Representational State Transfer“ von Roy Fielding, ist eine Anwendungsprogrammierschnittstelle und entspricht den Beschränkungen des REST-Architekturstils. Eine API bzw. Schnittstelle wird als RESTful bezeichnet, wenn diese den Beschränkungen der REST-Architektur unterliegt [23, S. 893]. Wenn eine Client-Anfrage über eine RESTful-API gestellt wird, überträgt die RESTful-API die Ressourcen-Statusinformationen an den Benutzer bzw. Endpunkt. Diese Informationen können in Formaten wie einfachem Text, Python, JSON oder HTML über HTTP übermittelt werden.

Im Original werden dabei folgende Architekturen beschrieben [22, S. 78 ff.]:

- Client-Server-Architektur
    - Diese beschreibt die Aufspaltung der Applikation in einen Server-Teil, welcher die Logik beinhaltet und einen Client-Teil, in welchem die Darstellung implementiert wird.
- Zustandslosigkeit
    - Jede Anfrage vom Client an den Server muss alle erforderlichen Informationen enthalten, sodass keine auf dem Server gespeicherten Daten nötig sind. Dadurch wird die Zuverlässigkeit und Skalierbarkeit der Anwendung erhöht.
- Cache
    - Das Cachen bzw. Zwischenspeichern von Antworten vom Server muss möglich sein.
- Einheitliche Schnittstellen
    - Diese Eigenschaft gilt als Hauptmerkmal der REST-Architektur. Dadurch werden die einzelnen Implementierungen von den Dienstleistungen, welche sie zur Verfügung stellen, entkoppelt.
- Mehrschichtige Architektur
    - Das System kann mit zusätzlichen Schichten erweitert werden. Dabei sollten diese Schichten jedoch nur bis zur unmittelbar nächsten Schicht Einblick haben dürfen.
- Code auf Anfrage
    - Der Server darf Code an den Client schicken, welcher dort ausgeführt wird. Dazu zählen beispielsweise Applets oder Skripts.

2.7 SOAP

SOAP bzw. das Simple Object Access Protocol ist ein vom W3C standardisierter Nachrichtenrahmen [46, S. 710]. Dieser stellt ein einfaches Protokoll zum Datenaustausch in dezentralisierten und verteilten Umgebungen dar. Das Protokoll basiert auf XML und besteht aus drei Teilen (vgl. [48, 49]):

- Umschlag
  - Bezeichnet das Stammelement der SOAP-Nachricht und definiert einen Rahmen zur Beschreibung des Inhalts sowie der Verarbeitung einer Nachricht.
- Kopfzeile
  - Ist ein optionaler Bestandteil, welcher eine Reihe von Codierungsregeln zum Ausdrücken von Instanzen anwendungsdefinierter Datentypen definiert.
- Körper
  - Enthält die Rohdaten, welche zwischen dem Sender und Empfänger übertragen werden.

Bei SOAP wird anhand eines Service-Vertrags eine Menge an ausführbaren Methoden definiert bzw. reglementiert. Dieser Vertrag stellt den wesentlichen Unterschied zu einer REST-Schnittstelle dar, welche mit einem festen Satz an Methoden arbeitet. Des Weiteren enthält der SOAP-Header Metadaten, welche bei REST nicht vorhanden sind. SOAP ist robust und bietet eine serverseitige Zustandslosigkeit, wodurch eine bessere Skalierung erreicht werden kann und ist ein allgemein akzeptierter Standard. Die Schnittstelle ist plattform- bzw. programmiersprachenunabhängig und erlaubt es zusätzlich nicht-XML-entsprechende Informationen anzuhängen [46, S. 711].

2.8 Responsive Webdesign

Das Ziel vom Responsive Webdesign bzw. dem reaktionsfähigen Webdesign ist es Webanwendungen zu erstellen, welche auf allen Geräten, von mobilen Endgeräten bis zu Desktop Computern, darstellbar sind. Dabei geht es vor allem um die Optimierung der Darstellung für die unterschiedlichen Geräte. Um eine möglichst gute Benutzererfahrung zu erzielen, ist es wichtig, Navigationsmenüs oder einzelne Menüeingaben so anzupassen, dass diese stets einfach bedient werden können. Die Entwicklung des Responsive Webdesigns beginnt daher schon in der jeweiligen Eingabe-Komponente. Diese muss möglichst so programmiert werden, dass sie in sämtlichen Größenformaten einfach zu bedienen und darzustellen ist [26]. Eine Möglichkeit, diese Funktion zu realisieren, besteht darin, CSS-Medienabfragen zu nutzen. Diese Abfragen erlauben es, unterschiedliche Stilvorlagen je nach verwendetem Gerätetyp anzuwenden [27]. Abhängig von der ermittelten Bildschirmgröße verbraucht beispielsweise ein Eingabefeld auf einem Smartphone die ganze Zeile, während auf einem Desktop bequem drei Eingabefelder nebeneinander dargestellt werden können (vgl. Abbildung 6). Neben dieser Neuanordnung kann Responsive Webdesign beispielsweise auch über das Anzeigen bzw. Ausblenden von Interaktionselementen oder das Ersetzen dieser erreicht werden [47].

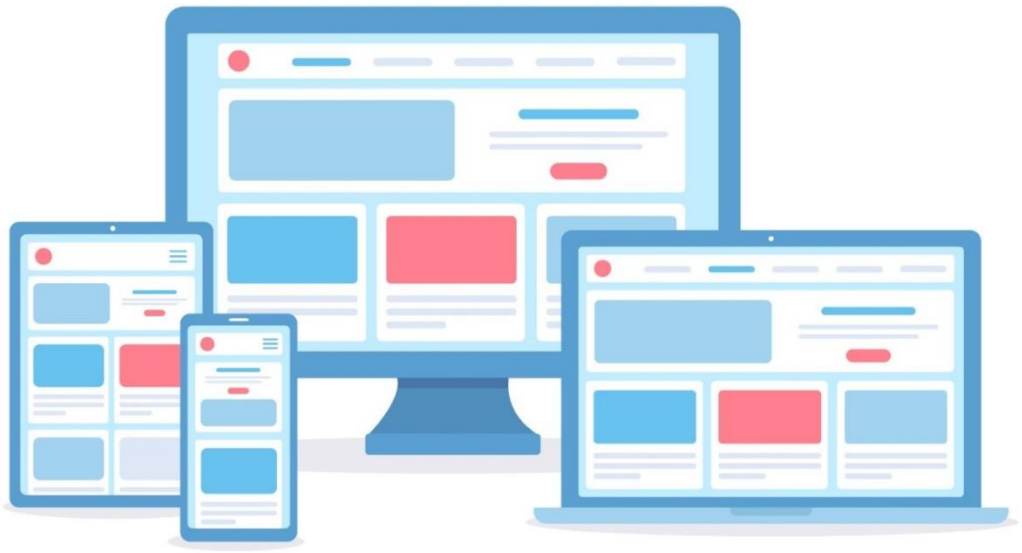


Abbildung 6: Beispiel für Responsive Webdesign [52]

Die Relevanz des Responsive Webdesigns nimmt mit der zunehmenden Verbreitung von internetfähigen Geräten stetig zu. Während im Jahr 2010 noch der Computer oder Laptop die präferierte Methode der Internetnutzung und der mobile Zugang zu vernachlässigen war [31], stieg der Anteil an mobilen Internetnutzern vom Jahr 2015 bis 2022 um ca. 30 Prozent [30]. Diese Verschiebungen spiegeln die sich wandelnden Präferenzen der Internetnutzer wider. 2020 nutzten die meisten Befragten das Internet vor allem über Smartphones (79%) bzw. über den Laptop (71%) und am wenigsten über den Desktop-PC (58%) [28]. Der Smartphone-Zugang erreichte im Jahr 2023 sogar 89,2 Prozent, während der Laptop-Zugang auf 66,6 Prozent und der Desktop-Zugang auf 43,9 Prozent gesunken ist [29]. Damit ist der mobile Internetzugriff die meist verbreitetste Methode.

3. Dokumentation des IST-Zustands

In diesem Kapitel erfolgt die detaillierte Dokumentation des IST-Zustands. Dabei wird sowohl auf die Benutzeroberfläche als auch auf den Quellcode eingegangen. Die Analyse der Bestandsanwendung dient als Grundlage für die Planung des weiteren Vorgehens, einschließlich der Überlegung, ob das bestehende System bzw. gewisse Teile dieses erweitert werden können oder eine vollständige Neuentwicklung erforderlich ist. Die aus dieser Analyse gezogenen Schlüsse sind für einen späteren Vergleich verschiedener Umsetzungsmöglichkeiten essenziell.

Die bisherige Webanwendung des Stör-/Kleinauftrags ist über eine Business-Server-Page-Applikation, kurz BSP, implementiert. Beim Aufruf der Seite wird zunächst der Auftraggeber abgefragt, da die vorherige Anforderung lautete, dass jeder Mitarbeiter am Standort die Möglichkeit haben muss, einen Auftrag zu erstellen.

Nach erfolgreicher Suche des Auftraggebers öffnet sich die Eingabemaske (vgl. Abbildung 7 und 8), in welcher über ein Dropdown-Feld (1) der Grund des Auftrages gewählt werden kann. Anschließend öffnet sich je nach Auswahl eine weitere Eingabemaske (2) mit den vom Grund abhängigen Eingabefeldern sowie einer entsprechenden Vorbelegung der Felder.

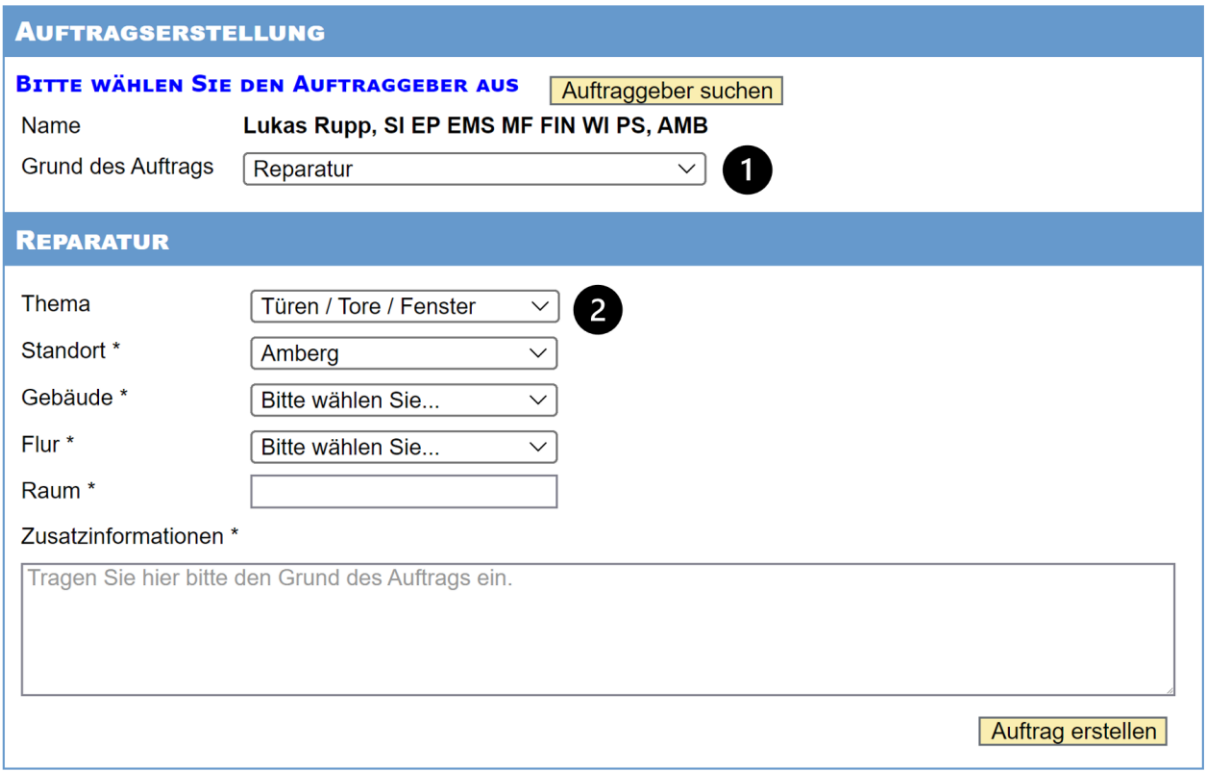


Abbildung 7: Ausschnitt aus der „Reparatur“ Webmaske der WI-Stör-/Kleinauftragserstellung

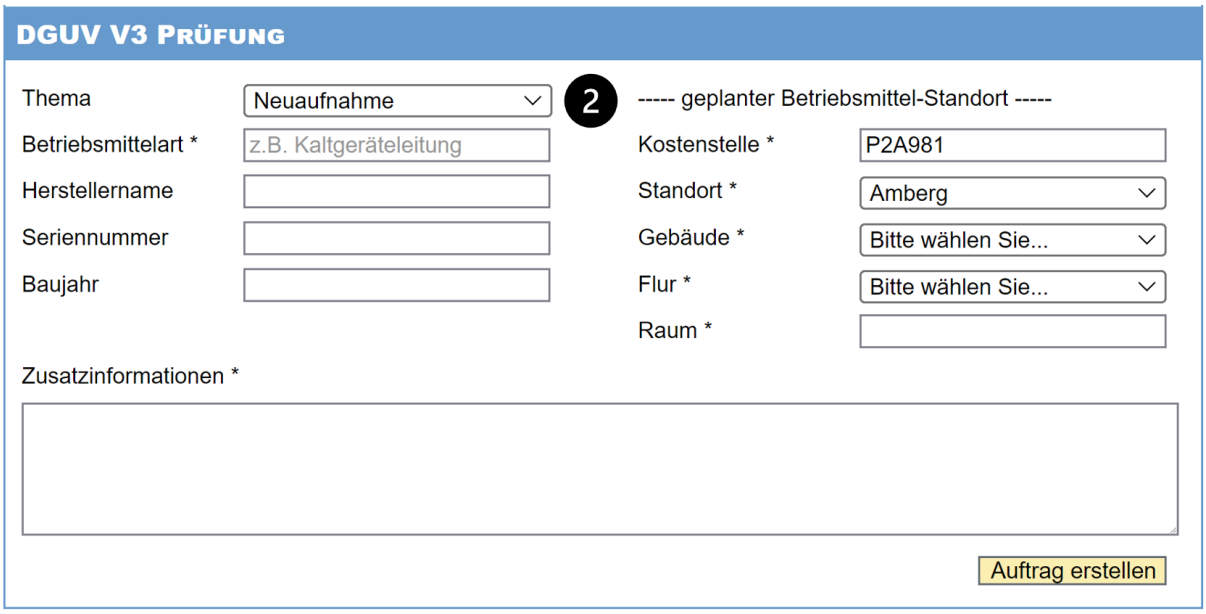


Abbildung 8: Ausschnitt aus der "DGUV V3 Prüfung" Webmaske der WI-Stör-/Kleinauftragserstellung

Damit auch Mitarbeiter ohne SAP-Account einen Stör-/Kleinauftrag anlegen können, erfolgt die Vorgangsabwicklung über einen sogenannten Default-Benutzer. Dieser hat alle nötigen Rechte, um die benötigten Aktionen, zum Beispiel die Erstellung des SAP PM Auftrages, auszuführen. Am Anfang jeder Seitenerzeugung wird einmalig die do_init-Methode des Controllers durchlaufen (vgl. Quellcode 1). Diese wird von der Basisklasse vererbt und verhält sich wie ein Konstruktor [37]. In der BSP-Applikation des Stör-/Kleinauftrags wird zuerst das Modell durch einen Aufruf erstellt, falls dieses noch nicht existiert. Danach werden die Optionen für die Webmasken befüllt (vgl. Quellcode 2). Zuletzt erfolgt das Befüllen der Hilfstabellen über die im Modell definierten Methoden (vgl. Quellcode 3). Die nähere Beschreibung des Modells erfolgt später. Zuerst soll der grobe Ablauf der BSP-Applikation beschrieben werden.

```
METHOD do_init.

  IF model IS  INITIAL.
    model ?= create_model( class_name = '/SIE/AD_ZC_CL_M_IHMELDUNG'
                           model_id   = 'model' ).
  ENDIF.
  set_model( model_id = 'model'
             model_instance = model ).

* Optionen für die Masken füllen
  me->initialize_mask_options( ).

* Hilfstabellen füllen (Wertehilfen)
  CALL METHOD model->fill_wh_grund.
  CALL METHOD model->fill_wh_standort.
  CALL METHOD model->fill_wh_art.
  CALL METHOD model->fill_wh_tims.
  CALL METHOD model->fill_wh_storung.

ENDMETHOD.
```

Quellcode 1: do_init-Methode

```
* Maskenkonfiguration aufbauen
*  <fs_dialog_maske>-grund         =  Grund -> muss gefüllt sein
*  <fs_dialog_maske>-objekt        =  Thema -> kann gefüllt sein
*  <fs_dialog_maske>-maske         =  CREATE_BEE_MASKE_+ -> Methode im Controller
*  <fs_dialog_maske>-vorbelegung   =  PREALLOCATE_MASKE_1 -> Methode im Model
  field-symbols: <fs_dialog_maske> type typ_dialog_mask.
  append initial line to t_dialog_masken assigning <fs_dialog_maske>.
  <fs_dialog_maske>-grund         = '1'.
  <fs_dialog_maske>-objekt        = ''.
  <fs_dialog_maske>-maske         = 'CREATE_BEE_MASKE_1'.
  <fs_dialog_maske>-vorbelegung   = 'PREALLOCATE_MASKE_1'.
```

Quellcode 2: Ausschnitt aus der initialize_mask_options-Methode

```
  FIELD-SYMBOLS <fs_line> TYPE shsvalstr.
  REFRESH t_wh_grund.
  APPEND INITIAL LINE TO t_wh_grund ASSIGNING <fs_line>.
  APPEND INITIAL LINE TO t_wh_grund  ASSIGNING <fs_line>.
  <fs_line>-key   = '1'.
  <fs_line>-value = cl_bsp_runtime=>get_otr_text( alias = '/SIE/AD_ZC_IH
   MELDUNG/GR_KLEINTRANSPORT_MIT_BESTUHL').
```

Quellcode 3: Ausschnitt aus der fill_wh_grund-Methode

Im Anschluss wird die zentrale Methode do_request aufgerufen. Diese ist zur Verarbeitung der Requests bzw. Anfragen zuständig. In der Implementierung der Kleinauftragsseite wird diese Methode im obersten Controller der Komponente genutzt und behandelt dadurch sowohl die Eingabe- als auch die Ausgabeverarbeitung. Falls die do_request-Methode nicht im obersten Controller genutzt wird, behandelt diese nur die Ausgabeverarbeitung [37].

In der do_request-Methode wird zuerst die initiale View definiert und daraufhin die Instanz dieser durch die Methode „create_view“ erstellt. Parallel dazu erfolgt der Aufbau der einzelnen Masken. Nachdem die erste Maske und somit der erste Teil der Webanwendung erstellt wurde, erfolgt der Aufruf zur Erstellung der nächsten Maske. Dieser Prozess wird wiederholt, bis alle erforderlichen Masken erstellt wurden.

Zusätzlich werden zwei JavaScript-Funktionen generiert. Die erste Funktion füllt nach dem vollständigen Laden der Seite bestimmte Felder mit vordefinierten Standardwerten und hängt zudem Ereignisse an die Felder, um das Ein- und Ausblenden der Werte zu steuern. Die zweite JavaScript-Funktion ist für die Überprüfung der Pflichtfelder zuständig. Wenn Pflichtfelder nicht ausgefüllt wurden, markiert die Funktion die Namen der Eingabefelder in Rot. Abschließend werden die Seitenattribute der Ansicht, wie die Referenz der Model-Klasse, gesetzt [38]. Danach wird die Ansicht aufgerufen [14].

In der do_handle_event-Methode findet darauffolgend die eigentliche Verarbeitung der Benutzereingaben statt [37]. Im Falle der Stör-/Kleinauftrags-Webanwendung werden zuerst der Auftragstext, die Benutzerdaten sowie die Auftragsdaten im Modell durch dafür vorgesehene Methoden gesetzt bzw. ausgelesen. In Abhängigkeit von der Auswahl in den Dropdown-Feldern werden Hilfstabellen im Modell befüllt oder Masken gelöscht. Nach Eingabe aller erforderlichen Daten wird der Auftrag im Modell angelegt. Wenn der Auftrag abgeschickt wird, ist die Methode ebenfalls dafür zuständig, eine Eingangsmail mit den Auftragsdaten an den Auftraggeber zu versenden.

Die Erklärung und Funktionsweise aller Methoden zur Maskenerstellung erfolgt exemplarisch am Beispiel der Methode „create_bee_maske_1“ (vgl. Quellcode 4). Am Anfang dieser Methoden erfolgt die Deklaration der einzelnen Variablen. Dies betrifft zum Beispiel die Eingabeelemente wie Buttons, Eingabefelder, Dropdown-Feld und Check-Boxen. Weiterhin werden Variablen für die Breite, Höhe und den Text deklariert. Anschließend erfolgt die Implementierung der Zeilen. Die erste Zeile ist das Eingabefeld für die Kostenstelle. Am Anfang jeder Zeilenimplementierung wird eine Methode zum Hinzufügen eines Labels bzw. Namensschildes aufgerufen. In dieser werden zunächst die benötigten Variablen deklariert. Anschließend erfolgt die Überprüfung der Eingabewerte. Das Label wird über weitere Variablen, welche die HTML-Elemente oder Bezeichnungen beinhalten, zusammengebaut. Im nächsten Schritt wird das Eingabefeld über eine ABAP-Standardmethode erstellt. Über eine ID kann später auf das Feld zugegriffen werden. Zusätzlich erfolgt die Zuweisung einer Breite sowie eines Anzeigewertes aus dem Modell.

```
html_add_label(   if_otr_alias = '/SIE/AD_ZC_IHMELDUNG/KOSTENSTELLE'
                  if_new_table = 'X'
                  if_required = 'X'
                  if_id = 'if_kostenstelle' ).

  CALL METHOD cl_htmlb_inputfield=>factory
    EXPORTING
      id      = 'if_kostenstelle'
      width   = lf_width
      _value  = '//MODEL/S_AUFTRAG.KOSTENSTELLE'
    RECEIVING
      element = lref_ifield.


  ref_bee_table->add( element = lref_ifield level = f_level ).

  html_add_close_row( ).
  add_used_elements( if_id          = 'if_kostenstelle'
                     if_type        = 'I'
                     if_required    = 'X'
                     if_label       = '/SIE/AD_ZC_IHMELDUNG/KOSTENSTELLE'
                     if_fieldname   = 'KOSTENSTELLE'
                     if_group       = '0020'
                     if_sort_number = '0010'
                     if_print_crlf  = 'X'
```

Quellcode 4: Ausschnitt aus der create_bee_maske_1-Methode

Das Modell beinhaltet alle wichtigen Werte, welche für die Auftragserstellung benötigt werden und ist dementsprechend auch für die letztendliche Erstellung des Auftrages selbst zuständig. Nach Eingabe des Global Identifiers, kurz GID, welche eine eindeutige Nummer zur Identifizierung der Siemens Mitarbeiter darstellt, ruft das Modell über eine eigene Methode alle wichtigen Nutzerdaten aus der Datenbank ab. Somit können Eingabefelder für den Standort und die Kostenstelle bereits ohne weitere Benutzereingaben vorbelegt werden. Wie zuvor beschrieben, beinhaltet das Modell auch diverse Methoden zum Befüllen der Hilfstabellen, dessen Inhalte später in den Dropdown-Feldern der Masken angezeigt werden. Weiterhin implementiert das Modell alle Methoden, welche die Eingaben aus den Eingabefeldern auslesen und in dafür vorgesehene interne/lokale Tabellen speichern. Diese Methoden sind essenziell, um den Auftrag und das Dokument zu erstellen.

Die View wird in HTML geschrieben. Über <htmlb:page title=“”> wird hier zuerst die Überschrift festgelegt. Anschließend erfolgt der Aufruf von jQuery-Skripten und anderen CSS-Dateien, welche für eine bessere Darstellung bzw. Gestaltung der Benutzeroberfläche sorgen. Daraufhin werden die Masken über die „create_bee“ Methoden erstellt.

Im nächsten Schritt erfolgt die Definition verschiedener Funktionen, die je nach Benutzeraktion getriggert werden. Ein Beispiel hierfür ist die Personensuche im Siemens Corporate Directory (SCD), einer internen Kontaktsuche, nach Eingabe des Nachnamens. Die Funktion für die Kontaktsuche läuft über AJAX-Aufrufe. Außerdem werden für bestimmte Eingabefelder Formate festgelegt oder nach Klicken eines Buttons bestimmte Felder automatisch angewählt. Ein weiterer Teil der View ist die Implementierung des Pop-Ups für die SCD-Suche sowie das Kontrollieren der Pflichtfelder auf Vollständigkeit. Zuletzt erfolgt die Definition der verschiedenen Styles.

Ein weiterer Bestandteil der BSP-Applikation sind die Dictionary-Objekte, welche alle benötigten Datenbanktabellen beinhalten. Dazu gehört zum Beispiel eine Tabelle mit allen Gebäuden des Standortes und weitere Tabellen, welche bestimmte Themen der Kleinaufträge mit den dazugehörigen Sachbearbeitern bzw. Arbeitsplätzen verknüpfen. In Tabellentypen werden Vorlagen für wichtige oder mehrfach genutzte Tabellen definiert.

Zu den Dictionary-Objekten gehören zudem Strukturen, die eine Zeile einer Tabelle definieren und sozusagen nur die Kategorien/Spaltenbezeichner beinhalten. Während der Programmausführung können diese Strukturen auch mit dem Inhalt einer Zeile einer Tabelle befüllt werden. Dieses Verfahren wird angewendet, wenn Tabellen in Schleifen durchgearbeitet werden sollen, um zum Beispiel Daten zu modifizieren. In der bestehenden BSP-Applikation gibt es Strukturen, welche alle für den Kleinauftrag benötigten Felder, inklusive Datentypen, Längen und Kurzbeschreibungen, aber auch Wertehilfsstrukturen für die Datierung enthalten.

In Datenelementen werden einzelne Spaltenbezeichner näher beschrieben. Neben dem eigentlichen Namen dieser Spalten können dort noch andere Namen hinzugefügt werden, falls die Zeile zum Beispiel eine andere Breite hat und somit eine Abkürzung erforderlich ist. Durch Definition einer Domäne kann zusätzlich konfiguriert werden, ob die Werte in der Spalte auch Kleinbuchstaben enthalten dürfen, da die Standardkonfiguration nur Großbuchstaben erlaubt. Eine Beschränkung auf gewisse Werte oder Intervalle ist in diesem Zusammenhang ebenfalls möglich.

4. Anforderungsanalyse

In diesem Kapitel erfolgt die Anforderungsanalyse der protypischen Webanwendung, welche sich an den grundlegenden Systemzielen und den daraus abgeleiteten Anwendungsszenarien orientiert.

4.1 Grundlegendes Ziel des Systems

Die wesentliche Funktion der protypischen Webanwendung besteht in der Erstellung von WI-Stör-/Kleinaufträgen, wobei die Modernisierung nicht nur eine Verbesserung der bestehenden Funktionalitäten, sondern auch die Implementierung neuer Features beinhaltet. Ein zentraler Aspekt ist dabei eine zukunftsorientierte Arbeitsweise, um flexibel weitere Anpassungen vornehmen zu können. Dazu müssen mögliche Anwendungsszenarien beschrieben werden.

4.2 Ablauf eines WI-Stör-/Kleinauftrags

Der allgemeine Ablauf umfasst die Erstellung eines Stör- bzw. Kleinauftrages beim Auftreten von Defekten oder die Beauftragung einer durch die Werksinfrastruktur bereitgestellten Dienstleistung sowie die anschließende Bearbeitung durch einen WI-Sachbearbeiter.

Der Prozess beginnt mit der Auftragseröffnung durch den Auftraggeber über die Webanwendung (vgl. Abbildung 9). Als Kunde gelten alle Mitarbeiter der Standorte Amberg und Cham, welche einen eigenen firmeninternen Microsoft-Account besitzen. Dieser Schritt wird im SAP-System durch den Status „eröffnet“ gekennzeichnet. Gleichzeitig wird eine Eingangsmail an den Auftraggeber versandt, um den erfolgreichen Eingang des Auftrages zu bestätigen. Nach der Auftragseröffnung erfolgt die Zuweisung, Bearbeitung und Kontierung durch den WI-Sachbearbeiter. Während des Prozesses ändert sich der Auftragsstatus in SAP auf „An Bearbeiter übergeben“. Intern wird der Auftrag an einen WI-Werker übergeben, während bei externer Bearbeitung die Zuständigkeit an die Partnerfirma übertragen wird. Der SAP-Status wechselt dadurch auf „In Bearbeitung“. In der nächsten Phase erfolgt die Klärung und Auftragsbearbeitung vor Ort, wobei bei interner Bearbeitung die angefallenen Arbeitsstunden und mögliche Materialberechnungen hinzugefügt werden. Im Falle der externen Bearbeitung werden durch Dritte Rechnungen erstellt und eingereicht. Nach Abschluss der Bearbeitung erfolgt die Kostenabrechnung des Kleinauftrages. Im abschließenden Schritt wird diese der entsprechenden Kostenstelle oder Werknummer zugeordnet. Der Status des Auftrages wird als „abgeschlossen“ vermerkt und automatisch eine Abschlussmail an den Auftraggeber versandt, um über die erfolgreiche Durchführung des Auftrages zu informieren.

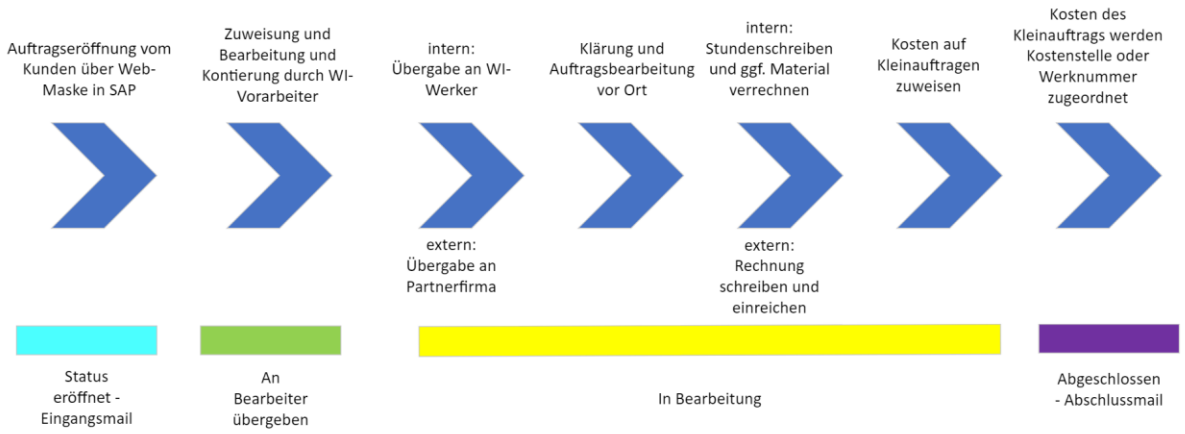


Abbildung 9: Ablauf des WI-Stör-/Kleinauftrages

Während dieser Schritte sind die Auftragsbearbeiter in Kontakt mit dem Auftraggeber, um ggf. weitere Fragen zur Bearbeitung zu klären. In den Phasen der Bearbeitung kommt es jedoch auch vor, dass die Kommunikation bereits stattgefunden hat, sich die Bearbeitung selbst jedoch durch unvorhersehbare Gründe verzögert. Zu diesem Zeitpunkt benötigt der Auftraggeber Informationen über Terminänderungen bzw. eine generelle Rückmeldung über den aktuellen Auftragsstatus. Dieser kann bisher jedoch nur recht umständlich über die Eingabe der Auftragsnummer auf einer weiteren Webseite oder über einen Link in der Auftragsbestätigung eingesehen werden. Um den Rückmeldeprozess künftig zu erleichtern und dem Auftraggeber mehr Transparenz und Information zu bieten, sollen im neuen System Funktionalitäten zur Abdeckung dieser Aufgaben implementiert werden.

Aus dem gegebenen Ablauf des Auftrages sowie der neuen Funktionalität der Rückmeldung ergeben sich folgende Anwendungsbeispiele:

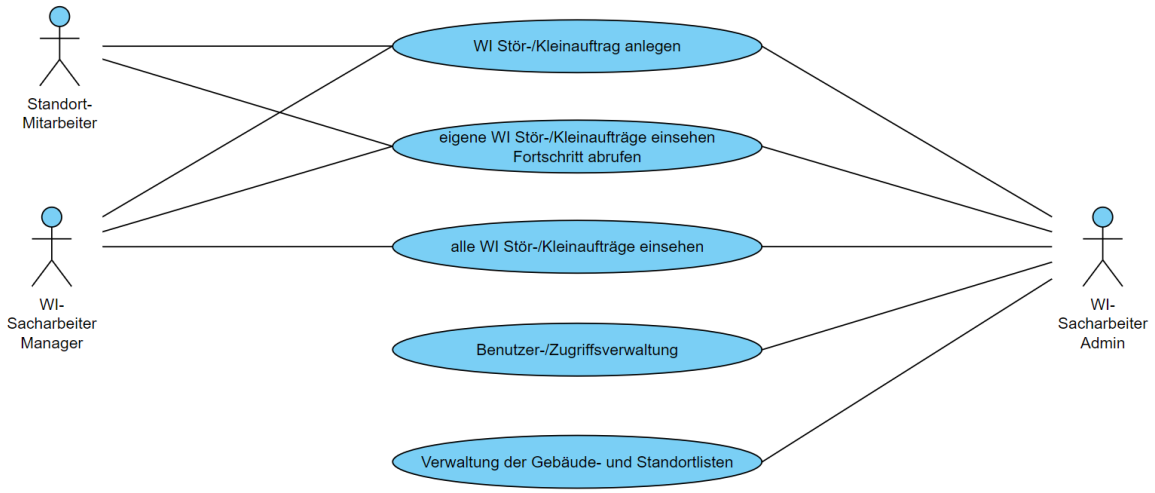


Abbildung 10: UML Use Case Diagramm der WI-Stör-/Kleinauftrags-Webanwendung

4.3 Funktionale Anforderungen

Um die bestehende Funktionalität der Webanwendung zu bewahren, müssen verschiedene Aspekte berücksichtigt werden. Für diesen Prozess wurde die Oberfläche der Webseite ausgiebig dokumentiert. Die Dokumentation und ein veraltetes Lastenheft tragen daher zur Erfassung der gesamten Anforderungen bei.

Kleinauftragserstellung und -verfolgung

Der Nutzer muss die Möglichkeit haben, Kleinaufträge über die Webanwendung zu erstellen. Des Weiteren soll er die Möglichkeit besitzen seine Aufträge, inklusive Status, in einer Übersicht darstellen zu lassen. Zur Datenerhebung soll die Übersicht Möglichkeiten zur Filterung und Suche von Einträgen besitzen. Für die Verfolgung seines Auftrages soll der Benutzer die Möglichkeit haben E-Mails mit dem aktuellen Auftragsstatus zu erhalten, sobald dieser sich ändert. Sachbearbeiter der Werksinfrastruktur sollen dabei zusätzliche Rechte zum Einsehen aller Aufträge besitzen.

Der Abruf der Aufträge über die Webanwendung ist hierbei ein Schlüsselmerkmal der Modernisierung. Die SAP-Oberfläche bietet keine benutzerfreundliche Möglichkeit zur Visualisierung der einzelnen Aufträge, stattdessen müssen diese jeweils einzeln über die entsprechende Auftragsnummer gesucht werden und können anschließend einzeln ausgelesen werden. Durch die Filterung der Aufträge nach bestimmten Kriterien wird des Weiteren die Möglichkeit geboten nach auffälligen Mustern zu suchen. Die Speicherung der Daten in einer eigenen Datenbank ist die Grundvoraussetzung für die Entwicklung weiterer Datenanalysen, um wiederkehrende Aufträge ausfindig zu machen, Maßnahmen zu ergreifen und beispielsweise häufig defekte Objekte auszutauschen.

Benutzeranmeldung und -rechte

Die Benutzeranmeldung soll über den firmeninternen Microsoft-Account erfolgen. Diesen besitzen alle für die Auftragserstellung relevanten Mitarbeiter der Standorte. Zudem ist es notwendig, bestimmten Nutzern erweiterte Rechte zuzuweisen. Dabei muss die Anwendung sicherstellen, dass nur autorisierte Benutzer auf bestimmte Funktionalitäten der Webanwendung Zugriff haben.

Benutzbarkeit

Die Interaktion des Nutzers mit der Webanwendung sollte intuitiv und verständlich sein. Eine effiziente Benutzerführung, klare Navigationen und gut platzierte Interaktionselemente müssen dazu beitragen, den Nutzer schnell und problemlos durch die Webmasken zu leiten, sodass eine einfache Bedienbarkeit ermöglicht wird. Hierbei sollten auch Aspekte wie die Anordnung der Eingabeelemente sowie die Minimierung von Eingabeaufforderungen berücksichtigt werden. Des Weiteren muss die Webanwendung Aspekte der Zugänglichkeit aufweisen, dazu gehören der Zugriff auf Deutsch und auf Englisch.

Responsive Webdesign

Für eine ansprechende Darstellung auf allen Geräten muss die Webanwendung über ein Responsive Webdesign verfügen. Der Benutzer soll die Möglichkeit haben, die Kleinauftragserstellung von überall aus zu erreichen. Dies gewährleistet die Flexibilität, auch von unterwegs oder direkt vom Ort des Defekts einen Störauftrag anlegen zu können.

Schrittweise Auftragserfassung

Die Webmasken zur Auftragserfassung sollen schrittweise gefüllt werden. Der Benutzer wählt zunächst eine passende Kategorie für seinen Auftrag aus einer Liste aus. Anschließend werden die erforderlichen Eingabefelder eingeblendet, wobei die Pflichtfelder deutlich markiert sind. Insbesondere bei Feldern, die personenbezogene oder Standortdaten enthalten, sollen sinnvolle Vorbelegungen vorhanden sein. Während der Auftragserstellung hat der Nutzer die Möglichkeit, Dateien anzuhängen, um zusätzliche Informationen wie Bilder, Tabellen oder andere Dokumente zu übermitteln.

4.4 Nicht-Funktionale Anforderungen

Um den Anforderungen einer modernen Webanwendung zu entsprechen, muss die prototypische Umsetzung einige nicht-funktionale Anforderungen erfüllen. Diese werden im Folgenden kurz beschrieben.

Wartbarkeit

Das System sollte nicht nur Möglichkeiten zur einfachen Modifizierung bieten, sondern auch darauf ausgelegt sein, eine nachhaltige Weiterentwicklung der Anwendung zu ermöglichen. Eine entscheidende Voraussetzung dafür ist, dass der Code leicht verständlich ist. Darüber hinaus ist es essenziell, dass der Code in übersichtlichen Modulen implementiert wird, um einzelne Implementierungen austauschen bzw. ersetzen zu können.

Leistung und Effizienz

Das System sollte innerhalb einer angemessenen Antwort- und Verarbeitungszeit reagieren und die zur Verfügung stehenden Ressourcen effizient nutzen. Dazu gehört die Fähigkeit, die anfallende Arbeitslast zu bewältigen und stets schnelle Antwortzeiten zu gewährleisten.

5. Implementierungsmöglichkeiten

Zur Umsetzung der genannten Anforderungen stehen grundsätzlich mehrere Möglichkeiten zur Verfügung. Im Folgenden werden diese erläutert sowie Vor- und Nachteile beschrieben.

5.1 BSP-Applikation

Eine Möglichkeit zur Umsetzung der Modernisierung ist die Anpassung der bisherigen Implementierung an die neuen Anforderungen. Die grundsätzliche Infrastruktur ist bereits implementiert und Datenbanken sowie Modelle sind erstellt. Die größten Anpassungen würden das Frontend betreffen. Nachteilig an dieser Umsetzung ist jedoch die benötigte intensive Einarbeitung in die Business-Page-Application sowie in die Implementierungssprache ABAP. Teile des Programms sind bereits über 10 Jahre alt und benötigen eine Modernisierung. SAP selbst hat mittlerweile neue Möglichkeiten für die Implementierung von Webanwendungen entwickelt und stellt mit SAPUI5 und SAP Fiori modernere Optionen zur Verfügung.

Vor allem im Bereich des Frontend sind in der bisherigen Implementierung viele Methoden enthalten, welche nur zur dynamischen Erstellung der einzelnen Masken zuständig sind. Dieser Prozess lässt sich durch alternative Ansätze, wie der Ausführung von Skripten im Webbrowser des Nutzers, wesentlich einfacher implementieren. Im Gegensatz zum Bestandsprogramm, bei dem jede Maske komplett neu erzeugt werden muss, ermöglichen es moderne Methoden, einzelne Elemente der Seite je nach Bedarf ein- und auszublenden. Dies führt zu einer Steigerung der Effizienz der Anwendung. Eine Integration eines Frameworks, beispielweise Angular, ist zwar durchaus möglich, jedoch muss hier der Aufwand und die Langfristigkeit betrachtet werden.

Momentan kann jeder Benutzer, für jede Person am Standort, einen Auftrag erstellen. Durch die neue Anforderung einer Übersichtsliste der aktuellen Aufträge müsste eine neue Methode der Authentifizierung implementiert werden. Andernfalls könnte jeder Benutzer die erstellten Aufträge jeder anderen Person am Standort einsehen, was zu einer Verletzung der Datenschutzrichtlinien führen würde. Gegenwärtig ist es nur schwer möglich, den in den Anforderungen beschriebenen Login über firmeninterne Microsoft-Accounts mit dem SAP-System zu verknüpfen. Auf Grund des Vorhandenseins verschiedener Authentifizierungsmechanismen, welche jeweilige Benutzerprofile benötigen, würde diese Umsetzungen einen enormen Implementierungsaufwand erfordern. Darüber hinaus würde dies bedeuten, dass nur Mitarbeiter mit einem SAP-Account die Befugnis haben, Aufträge zu erstellen.

5.2 SAP Fiori

Eine alternative Umsetzungsmöglichkeit besteht in der Nutzung von SAP Fiori als Grundlage der Webanwendung. SAP Fiori ist eine einfache, rollenbasierte Benutzeroberfläche, welche sich durch eine optimale Darstellung, unabhängig von der Bildschirmgröße, auszeichnet [39, S. 26]. Dadurch kann es auf Desktop-PCs, aber auch auf Tablets und Smartphones angewendet werden [40, S. 13]. Die Benutzeroberfläche besteht größtenteils aus Kacheln (vgl. Abbildung 11), welche verschiedene Funktionen, wie Schaltflächen und Anzeigeelemente, repräsentieren.

Ein bedeutender Vorteil von Fiori liegt in der Verwendung wiederverwendbarer Designkomponenten, welche zu erheblichen Kostensenkungen bei der Frontend-Entwicklung von bis zu 80% führen können [41]. Dies ermöglicht eine unkomplizierte Gestaltung der Benutzeroberfläche.

Trotz der genannten Vorteile weist die Implementierung von SAP Fiori auch erhebliche Nachteile auf. Insbesondere beschränkt die Nutzung von SAP Fiori den Zugriff auf die Webanwendung auf Mitarbeiter mit einem SAP-Zugang. Dies steht im Widerspruch zu der Anforderung, dass alle Standortmitarbeiter mit einem firmeninternen Microsoft-Account die Möglichkeit haben sollten Störaufträge anzulegen. Bestimmte Gruppen von Mitarbeitern der Produktion besitzen keinen SAP-Zugang, da sie diesen für ihre Tätigkeiten nicht benötigen. Des Weiteren soll die Anmeldung mittels des Microsoft-Accounts erfolgen. Entgegen dieser Anforderung verwendet SAP jedoch eigene Anmeldeverfahren. Zudem schränkt SAP Fiori die Oberfläche durch vorgegebene Designbausteine ein, wodurch potenziell die Flexibilität der Anwendung beeinträchtigt werden kann.

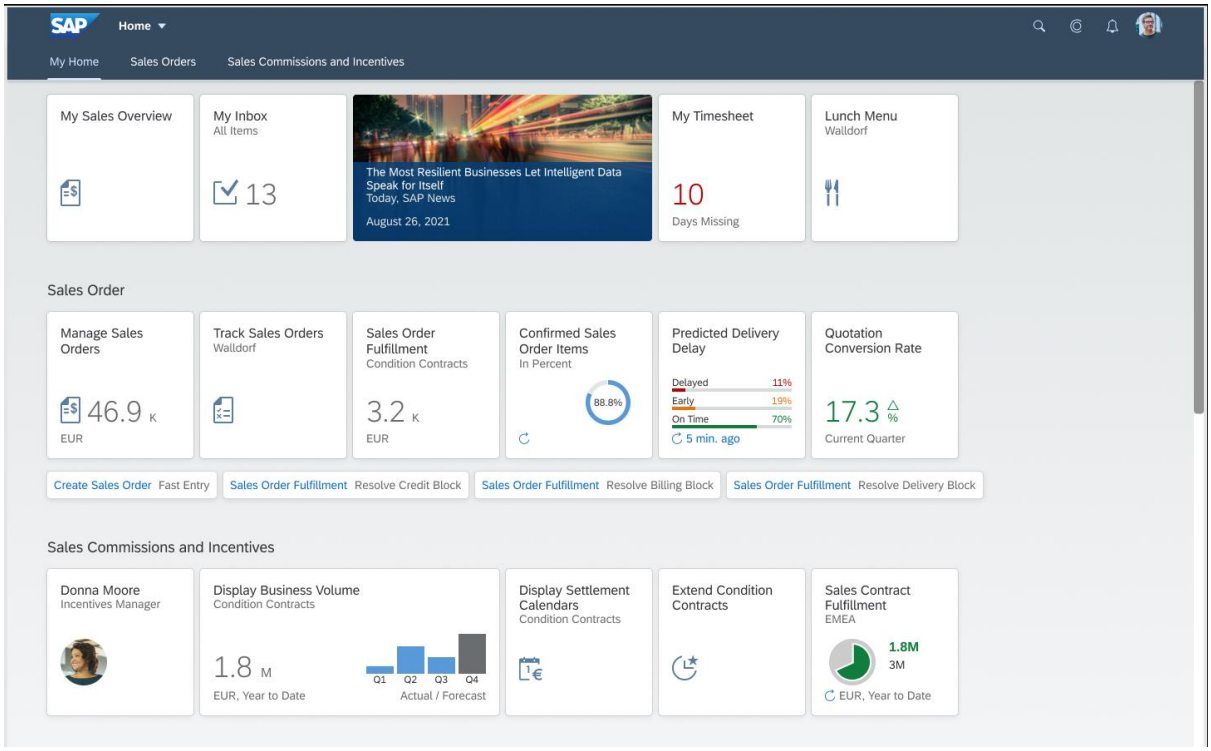


Abbildung 11: Beispielbild einer SAP Fiori Webanwendung [51]

5.3 Eigene Implementierung

Eine weitere Variante ist die Entwicklung einer Webanwendung, frei von bisherigen Systemen, über einen unabhängigen Technologie-Stack. Das Loslösen von anderen vorgefertigten Werkzeugen ermöglicht die Implementierung der Webanwendung nach beliebigen Vorstellungen. In den bisher genannten Möglichkeiten wird die Benutzeroberfläche durch vorgegebene Elemente eingeschränkt. Für die unabhängige Implementierung der Webanwendung würden lediglich die Unternehmensrichtlinien oder Designvorgaben der IT-Abteilung Grenzen setzen. Hier muss vor allem auf IT-Security-Richtlinien und generelle Programmierrichtlinien geachtet werden.

Das Frontend ließe sich beispielsweise über das Angular-Framework implementieren. Zur Datenspeicherung könnte die SAP-Datenbank wie bisher auch fungieren, jedoch ist zur Vereinfachung eine eigene Datenbank vorgeschaltet.

Die Kommunikation mit dem SAP-System müsste über das SOAP ablaufen, da es hier bereits Implementierungen gibt, welche in abgeänderter Form verwendet werden können (vgl. Abbildung 12).

Gemäß den Anforderungen ist die selbst entwickelte Umsetzung die optimale Option unter den hier aufgeführten Möglichkeiten. Das System kann dadurch modern und nach aktuellen Anforderungen an IT-Systeme individuell entwickelt werden. Diese Möglichkeit bietet zusätzlich die größtmögliche Flexibilität, da sie nicht an bestimmte Baukastenmodelle oder andere Vorgaben gekoppelt ist. Ein Nachteil ist jedoch die Systemwartung, welche die Werksinfrastruktur langfristig nicht übernehmen kann. Daher ist eine enge Zusammenarbeit mit der IT-Abteilung des Standortes und eine Abgabe der Betreuung an diese notwendig.

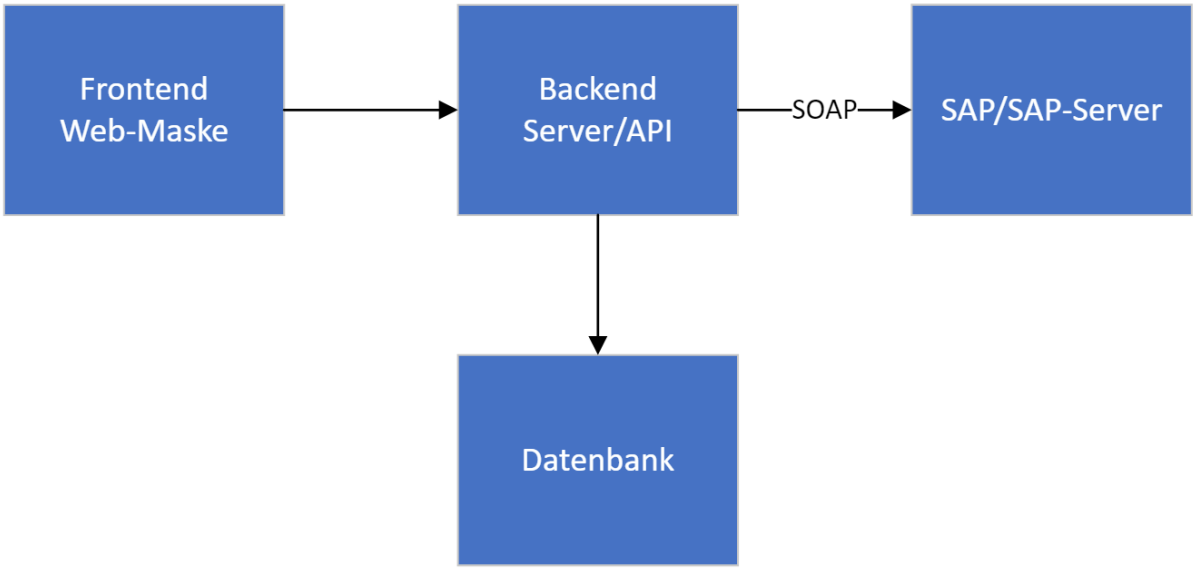


Abbildung 12: Entwurf einer möglichen eigenen Implementierung

6. Lösungskonzept des WI-Stör-/Kleinauftrags

Auf der Grundlage der Systemanforderungen wird ein Lösungskonzept für das Projekt erarbeitet. Zuerst wird die benötigte Infrastruktur beschrieben. Anschließend erfolgt der Entwurf der Oberfläche sowie die Modellierung der Datenbank.

6.1 Technologie-Stack des WI-Stör-/Kleinauftrags

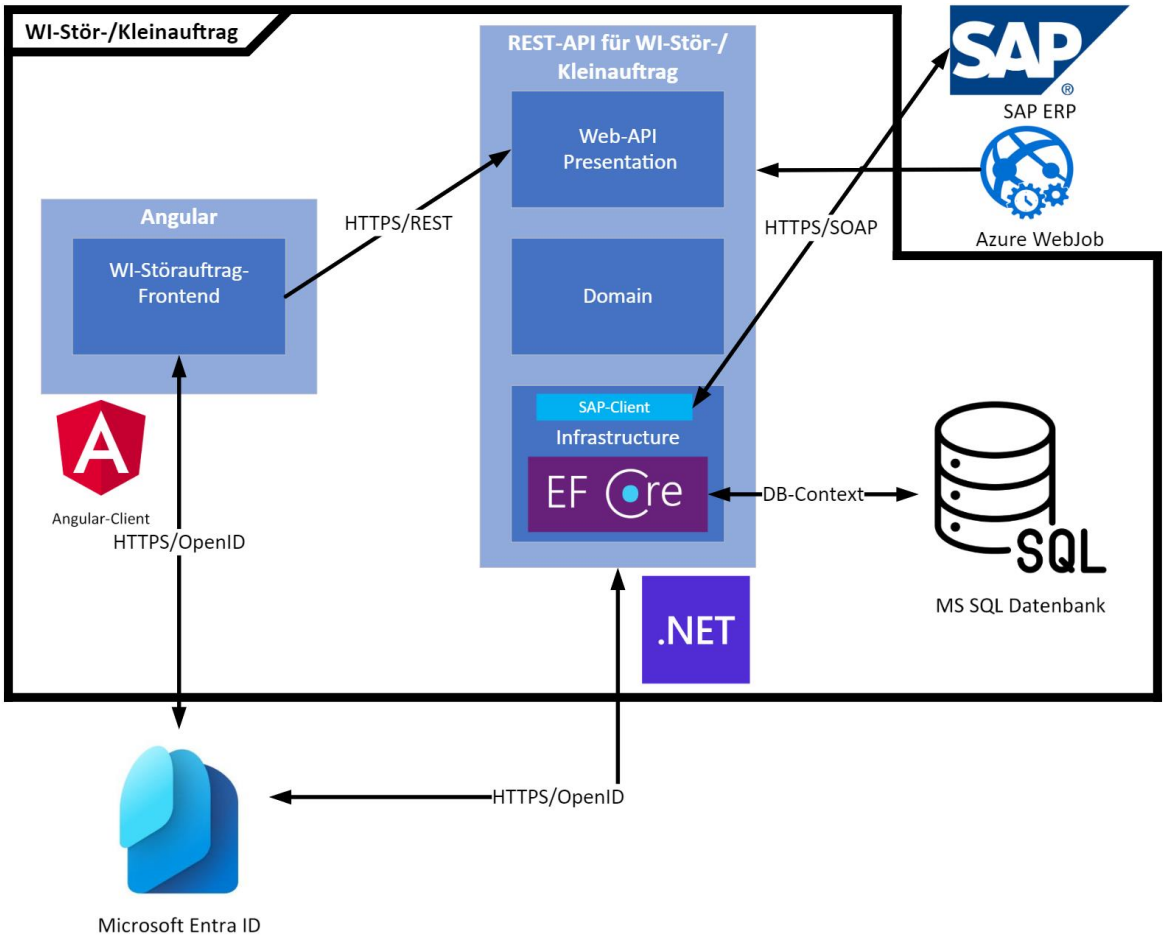


Abbildung 13: Architekturdiagramm des WI-Stör-/Kleinauftrags

Frontend – Angular

Die Entwicklung der Benutzeroberfläche erfolgt mithilfe des Angular Frameworks, welches eines der weltweit bekanntesten Frameworks für die Erstellung von Web-, Mobil- und Desktopanwendungen ist und von Google entwickelt wird [42, S. 209]. Angular zeichnet sich durch eine komponentenbasierte Struktur aus und bietet die Möglichkeit, skalierbare Webanwendungen zu entwickeln. Die Benutzeroberfläche selbst wird unter Verwendung der Auszeichnungssprache HTML5, Cascading Style Sheets und der Skriptsprache TypeScript implementiert. HTML5 strukturiert die Webinhalte, während CSS verwendet werden, um diese Inhalte nach beliebiger Vorgabe zu formatieren. TypeScript ermöglicht anschließend die Interaktion des Anwenders mit der Webanwendung.

Backend – .NET

.NET ist ein open-source, plattformübergreifendes Framework für die Erstellung moderner Apps und Cloud-Services. Es bietet die Verwendung verschiedener Sprachen, wie C#, F# oder Visual Basic, sowie die Verwendung vieler Bibliotheken zur Erweiterung an [44]. Die Grundlage aller .NET-Anwendungen stellt die Common Language Runtime (CLR) dar. Sie ist die Laufzeitumgebung, in welcher die .NET-Programme auf dem Zielrechner ausgeführt werden [23, S. 9].

Kommunikation und Mapping – Entity Framework Core

Das Entity Framework Core ist eine Sammlung von Technologien für die Entwicklung datenorientierter Softwareanwendungen. Es dient als Objektbeziehungszuordnung zur Erstellung von Datenzugriffsebenen für eine Vielzahl von Datenbanken. Durch das Entity Framework können Daten in Form von domänenspezifischen Objekten und Eigenschaften verwendet werden. Dies ermöglicht die Behandlung der Daten auf einer höheren Abstraktionsebene und die Entwicklung datenorientierter Anwendungen mit weniger Code [45].

Datenbank – SQL-Server 2022

Der Microsoft SQL Server 2022 ist ein relationales Datenbankmanagementsystem. Es stellt eine Hybridplattform dar, welche auf Innovation in Hinsicht auf Sicherheit, Leistung, Verfügbarkeit und Datenvirtualisierung setzt [50].

Versionsverwaltung und Deploying – Azure DevOps

Azure DevOps bietet die umfassende Verwaltung des Lebenszyklus eines Softwareprojekts. Im Rahmen der DevOps-Kultur soll eine enge Zusammenarbeit zwischen Entwicklern und Projektmanagern gefördert werden. DevOps bietet die Integration von Git-Repositories zur Verwaltung und Zusammenführung von Quellcode. Darüber hinaus kann eine Suite von Werkzeugen die agile Entwicklung unterstützen. Außerdem werden Pipelines für Build- und Releasedienste sowie Testpläne angeboten, um die Testphasen von Anwendungen zu erleichtern und zu optimieren [43].

6.2 Designentwürfe und Mockups

Die Entwicklung einer modernen Webanwendung erfordert eine sorgfältige Planung und Gestaltung der Benutzeroberfläche, um sicherzustellen, dass die Anforderungen des Auftraggebers sowie des Endnutzers erfüllt werden. Frühzeitige Mockups dienen als visuelle Vorlagen und Entwürfe, welche das Design und die Interaktionselemente der Anwendung veranschaulichen.

Die Webanwendung wird von zwei verschiedenen Anwendergruppen genutzt, den im Werk tätigen Mitarbeitern und den Sachbearbeitern der Werksinfrastruktur. Um den vielfältigen Anforderungen gerecht zu werden, wurden sorgfältige Designentscheidungen getroffen, die im nachfolgenden Abschnitt beschrieben werden.

Das Ziel besteht darin, bereits bei der ersten Anwendung eine klare Übersichtlichkeit zu bieten und eine sofortige Nutzbarkeit zu ermöglichen. Im Weiteren werden die initialen Mockups aus Figma, einer Software zum Erstellen von Oberflächen-Prototypen, präsentiert.

Der erste Entwurf dient als Grundlage um die inhaltliche Struktur und den Aufbau der Webanwendung zu beschreiben. Dieser konzentriert sich vor allem auf die Neuanordnung der vorhandenen Eingabefelder (vgl. Abbildung 14). Der Fokus liegt auf einem zentral platzierten Eingabebereich mit kontrastreichen, gut sichtbaren Feldern. Die Dropdown-Felder, welche später für die dynamische Anpassung der Webmasken verantwortlich sind, befinden sich im oberen Bereich des Bildschirms. Je nach Auswahl im Dropdown-Feld werden die benötigten Eingabefelder darunter eingeblendet.

Als potenziellen Hintergrund bietet sich ein Siemens-typischer Farbverlauf an, welcher bereits von anderen internen Webanwendungen, wie der Siemens-Hello-Startseite, genutzt wird. Durch die eher dunkel gehaltenen Farben soll ein angenehmes Benutzererlebnis erzielt werden.

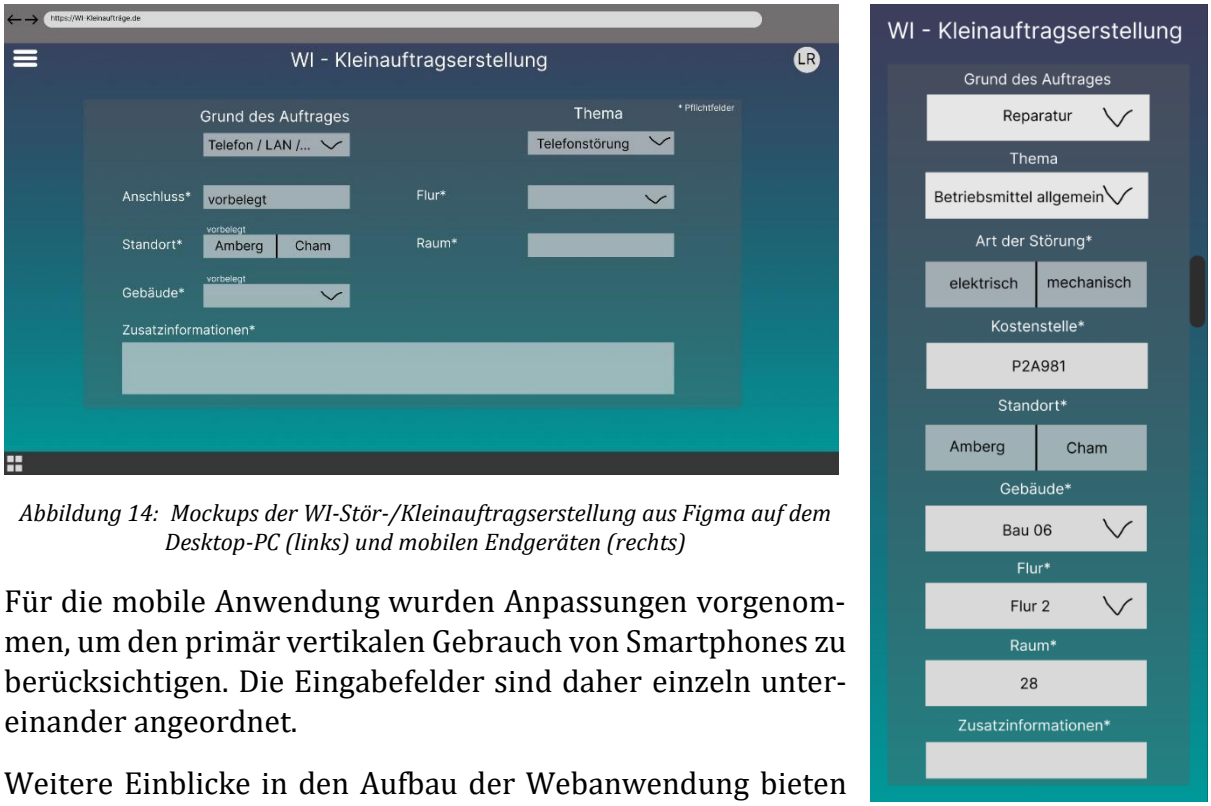


Abbildung 14: Mockups der WI-Stör-/Kleinauftragserstellung aus Figma auf dem Desktop-PC (links) und mobilen Endgeräten (rechts)

Für die mobile Anwendung wurden Anpassungen vorgenommen, um den primär vertikalen Gebrauch von Smartphones zu berücksichtigen. Die Eingabefelder sind daher einzeln untereinander angeordnet.

Weitere Einblicke in den Aufbau der Webanwendung bieten die Mockups, wie beispielsweise die Startseite. Diese stellt klare Entscheidungsmöglichkeiten zwischen dem Anlegen eines neuen Auftrages und der Übersicht eigener Aufträge, über große Auswahlflächen mit eindeutigen Icons bzw. Bildern, dar.

Für die Integration der Anmeldung mit dem internen Microsoft-Account wurde das gewohnte Design für die Verwaltung in der oberen rechten Ecke übernommen, während der Menü-Button sich in der oberen linken Ecke befindet. Für Sachbearbeiter der Werksinfrastruktur besteht zusätzlich die Möglichkeit über den Startbildschirm oder das Menü auf eine Übersicht aller Aufträge zuzugreifen (vgl. Abbildung 15). Dadurch entfällt die aufwändige Auftragssuche im SAP-System.

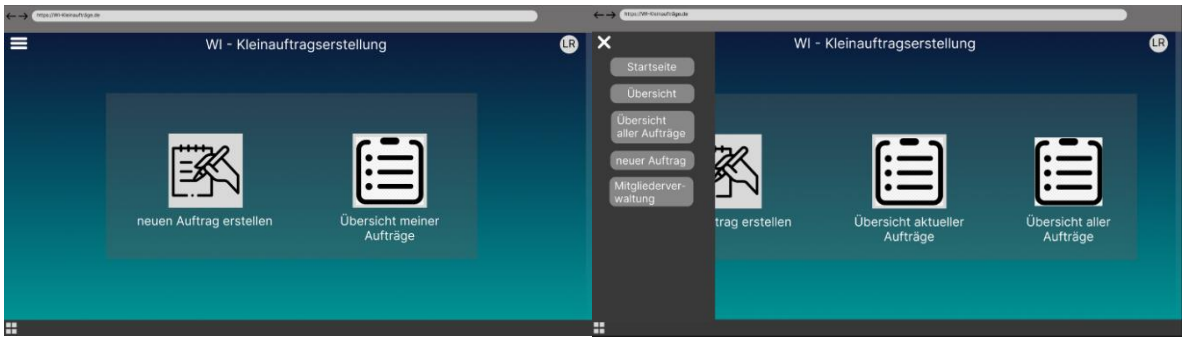


Abbildung 15: Mockups der Startseite des WI-Stör-/Kleinauftrags

Wenn sich der Nutzer für die Auftragsübersicht entscheidet, wird er entsprechend weitergeleitet (vgl. Abbildung 16). In dieser Übersicht werden alle benutzereigenen Aufträge mit den wichtigsten Informationen angezeigt. Zur Verbesserung der Benutzerfreundlichkeit wurden verschiedene Funktionen integriert:

- **Sortierfunktionen und Suchleiste**
  - Die Benutzer können die Reihenfolge der angezeigten Aufträge durch Auswahl des dafür vorgesehenen Buttons anpassen. Eine Suchleiste im oberen rechten Bereich ermöglicht es den Benutzern, gezielt nach Listeneinträgen zu suchen.
- **Scrollbar für den Listenbereich**
  - Um einen reibungslosen Überblick zu gewährleisten, ist im rechten Bereich der Liste eine Scrollbar angebracht. Diese ermöglicht es den Benutzern, den angezeigten Listenbereich nach Bedarf zu verschieben.

Abbildung 16: Mockup der Liste der erstellten Kleinaufträge

Weiterhin können Mitarbeiter mit entsprechender Autorisierung für die Datenbank- und Rollenpflege auf eine Standort-, Gebäude- und Mitgliederverwaltung zugreifen. Für die Listenverwaltung bietet sich die Bearbeitung über ein Dialogfenster an, welches über einen Button geöffnet werden kann (vgl. Abbildung 17). Über einen entsprechenden, mit einem Mülleimer-Icon versehenen Button können die Listeneinträge gelöscht werden. Zum Bearbeiten der Listeneinträge reicht das Anklicken des jeweiligen Eintrages, wodurch das Dialogfenster geöffnet wird.

Abbildung 17: Mockup des Dialogfensters der Verwaltungs- bzw. Pflegelisten

6.3 Datenbankmodellierung

Die Datenbank bildet das Herzstück des Systems und dient als Speicherort für sämtliche Aufträge sowie die Einträge einzelner Dropdown-Felder. Um alle relevanten Felder zu bestimmen, muss die Webanwendung zunächst detailliert betrachtet werden.

Die Webanwendung offenbart zunächst das Dropdown-Feld für den Grund des Kleinauftrages (vgl. Abbildung 18). Obwohl die Einträge dieser Liste variabel sind, ist jeder Eintrag jeweils mit einer Webmaske verknüpft. Die Liste wird demnach nur geändert, wenn auch die Webmasken angepasst werden. Dadurch ist es nicht notwendig, diese Liste in der Datenbank zu speichern. Ähnlich verhält sich dies mit dem Dropdown-Feld für die Auswahl des Themas.

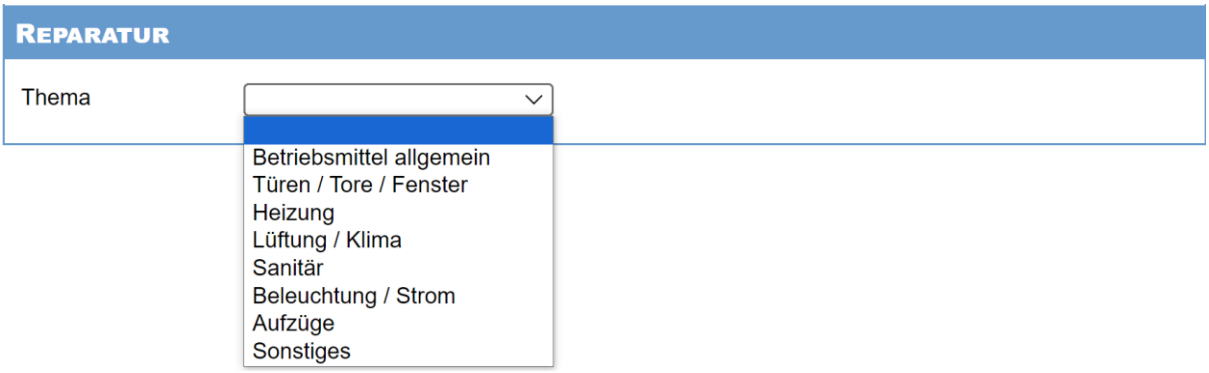


Abbildung 18: Ausschnitt aus der Webmaske der BSP-Applikation mit Fokus auf das Dropdown-Feld des Themas der Reparatur

Ein weiteres essenzielles Eingabefeld in den Masken ist die Auswahl des Gebäudes (vgl. Abbildung 19). Am Standort Amberg existieren etwa 50 Gebäude und für den Standort Cham müssen weitere 15 berücksichtigt werden. Eine Implementierung dieser ca. 65 Gebäude im Quellcode ist nicht praktikabel. Die Gebäude benötigen demnach eine eigene Datenbanktabelle. Im Datenbankmodell wird diese als „buildings“ bezeichnet. Die Tabelle enthält eine ID im Datenformat Integer als Primärschlüssel. Um Redundanzen zu vermeiden, wird eine Location-ID als Fremdschlüssel benötigt. Darüber hinaus muss die Tabelle den ausgeschriebenen sowie den abgekürzten Gebäudenamen jeweils als String enthalten. In einer zweiten Datenbanktabelle, namens „locations“, werden die zugehörigen Standorte der Gebäude gespeichert. Die „locations“-Tabelle besitzt eine ID als Primärschlüssel, welcher als Abkürzung des Standortes gilt, sowie einen ausgeschriebenen Namen, jeweils im Datenformat String. Beide Tabellen sind über den Fremdschlüssel „location_id“ und den Primärschlüssel „ID“ der „locations“-Tabelle mit einer n:1-Verbindung verknüpft.

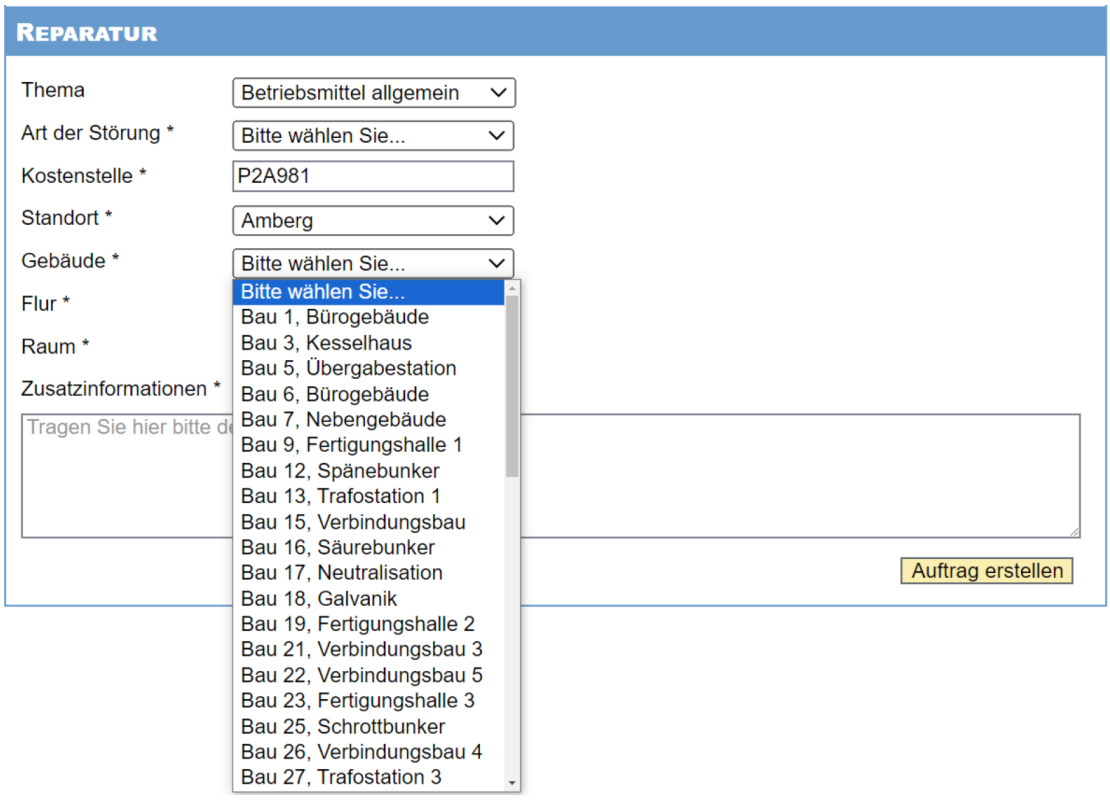


Abbildung 19: Ausschnitt aus der Webmaske der BSP-Applikation mit Fokus auf das Dropdown-Feld der Gebäude

Für die Mitglieder- bzw. Rollenverwaltung wird eine zusätzliche Tabelle mit einer ID als Integer sowie einer E-Mail und der Rolle benötigt. Die Rolle wird dabei als Integer gespeichert und entspricht in den späteren Implementierungen der Logik einer Enum.

Zur Darstellung der aktuellen und abgeschlossenen Kleinaufträge in der Webanwendung wird zusätzlich die „order“-Tabelle angelegt. Diese enthält alle relevanten Informationen, die von den Sachbearbeitern gewünscht sind, sowie zusätzliche Daten, welche für die Datenerhebung bzw. -analyse nötig sind. Zum Zweck der Datenanalyse werden zunächst alle Eingabefeldwerte der Webmaske gespeichert. Dadurch wird eine ausgiebige Analyse nach verschiedenen Kriterien gewährleistet.

Aus den ermittelten Anforderungen ergibt sich dabei folgendes Datenbankmodell:

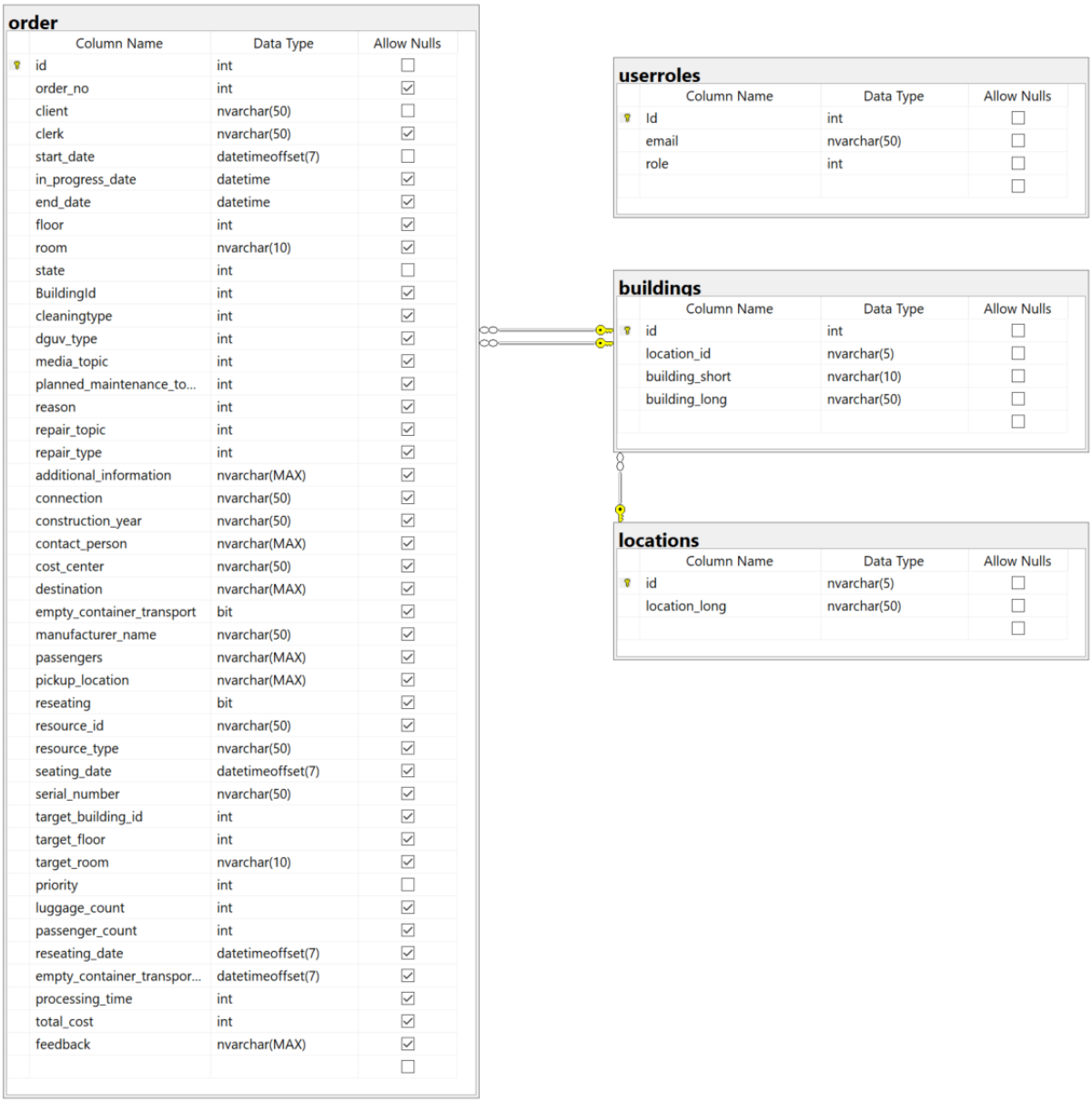


Abbildung 20: Datenbankmodell des WI-Stör-Kleinauftrages

7. Umsetzung des Lösungskonzepts

Im folgenden Kapitel wird die prototypische Umsetzung anhand des Frontends sowie des Backends erklärt. Das Basisprojekt wurde dabei aus einer Vorlage der IT-Abteilung entnommen und entsprechend den Anforderungen angepasst. Im Folgenden werden die einzelnen Implementierungen detailliert beschrieben und erklärt.

7.1 Frontend-Entwicklung

Das Frontend basiert auf dem Angular Framework. Des Weiteren werden durch die Nutzung der Siit-Bibliothek der Siemens AG bereits eigene HTML-Elemente bereitgestellt und für die Oberfläche genutzt. Die Siit-Bibliothek ist eine interne Sammlung an Standardkomponenten, welche für eine einheitliche Benutzeroberfläche aller Bereichsanwendungen sorgt. Um die Anforderung der Zugänglichkeit an die Webanwendung zu garantieren, ist diese auf Deutsch und Englisch verfügbar. Den Abruf der eingestellten Sprache sowie die Änderung des Texts übernimmt dabei ein durch die IT-Abteilung bereitgestelltes Skript.

Beim Aufruf der Webanwendung erfolgt zunächst der Login, mittels des firmeninternen-Siemens-Accounts, welcher bereits in der Vorlage des Angular Projektes vorhanden war. Der erfolgreiche Login wird anschließend durch eine grüne Bildschirmbenachrichtigung bestätigt. Nach Ablauf eines gewissen Zeitintervalls wird durch eine weitere Bildschirmbenachrichtigung vor dem Ablauf der Session gewarnt und eine Erneuerung der Session gefordert.

7.1.1 Startseite

Um die im Mockup vorgestellten Auswahlflächen zu implementieren, wird auf der Startseite das Card-Element der Bibliothek genutzt (1) (vgl. Abbildung 21). Dieses Element bietet diverse Konfigurationsoptionen, darunter die Einstellungen für ein Bild, einen Titel, eine Beschreibung sowie einen Button. In Abhängigkeit von der Rolle des Benutzers werden auf der Startseite zwei bzw. drei Karten angezeigt, welche für die Weiterleitung auf die entsprechenden Seiten zuständig sind.

Auf der linken Seite befindet sich das Navigationsmenü (2), welches ein- bzw. ausgeblendet werden kann. Über die verschiedenen Einträge kann der Benutzer zu den entsprechenden Seiten weitergeleitet werden. In Abhängigkeit der Benutzerrolle beinhaltet das Navigationsmenü verschiedene Einträge. Für den normalen Standortmitarbeiter ohne Rolle ist nur die „Startseite", die „Stör-/Kleinauftragserstellung" sowie die „Auftragsübersicht" einsehbar. Benutzer mit der Rolle Manager haben zusätzlich die Möglichkeit die „Auftragsübersicht - Alle" anzuwählen. Bei WI-Sachbearbeitern mit der Admin-Rolle wird zusätzlich der Reiter „Admin-Menü", inklusive der Unterpunkte, angezeigt.

Über ein Dropdown-Feld (3) in der oberen rechten Ecke der Webanwendung kann die Sprache eingestellt werden. Des Weiteren befindet sich dort ein Button (4), um sich aus der Webanwendung auszuloggen.

Das Navigationsmenü sowie die Kopfleiste sind ebenfalls Teil der Anwendungsvorlage, die Konfiguration dieser ist im Zuge der Modernisierung der Webanwendung erfolgt.

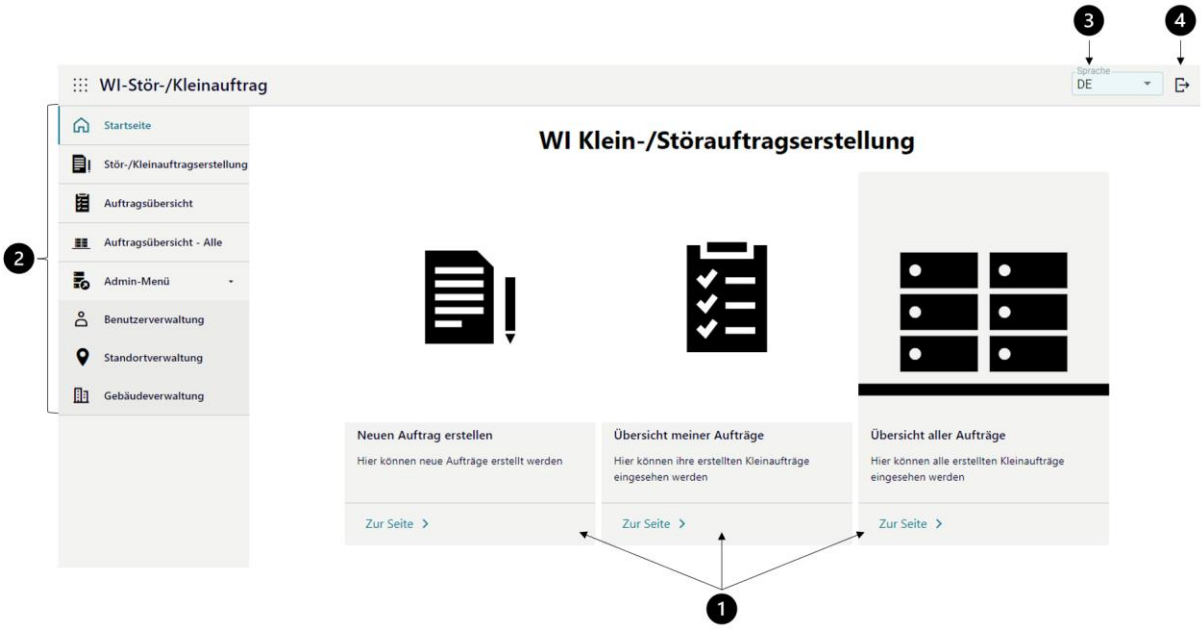


Abbildung 21: Startseite der neuen Webanwendung des WI-Klein-/Störauftrags

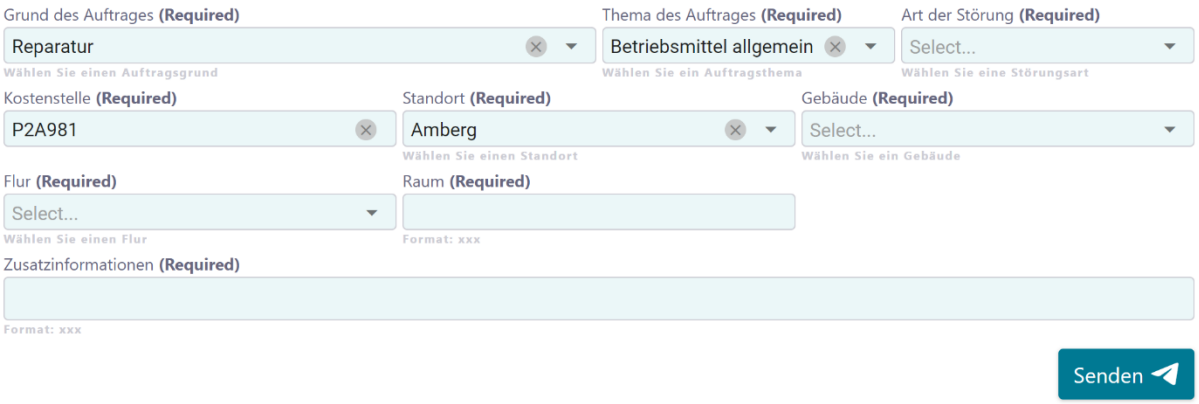
7.1.2 Formular zur Auftragserstellung

Das Eingabeformular wird über ein Forms-Element umgesetzt, welches umfassende Funktionalitäten für die Erstellung von Eingabemasken bietet. Dazu gehören Check-Boxen, Datumseingaben, Dropdown-Felder, Texteingabefelder oder Radio-Buttons. Zusätzlich bietet das Element eine Funktion zur automatischen Textvervollständigung sowie die Möglichkeit, Tabellen aus verschiedenen Eingabefeldern zu erstellen.

Die Konfiguration des Forms-Elements erfolgt durch Arrays in der TypeScript-Datei der Angular-Komponente, welche anschließend als Eingabeparameter im HTML-Element eingelesen werden. Die Konfiguration der Eigenschaften wie Typ, Pflichtfeld, Breite, Titel und Eingabevalidierung sowie die Reihenfolge auf der Webmaske erfolgt dabei direkt im Array (vgl. Quellcode 5). Bei Eingabemasken, welche nur ein Dropdown-Feld für den dynamischen Aufbau der Webmaske benötigen, ist hier darauf zu achten, dass Platzhalter eingefügt werden, um eine stets gleiche Sortierung der Eingabefelder zu gewährleisten. Des Weiteren bieten die Eingabefelder die Möglichkeit verschiedene Breiten, abhängig von der Bildschirmgröße, festzulegen. In der mobilen Ansicht entspricht ein Eingabefeld ca. der Breite des Bildschirmes, während auf größeren Geräten zwei bis drei Kurztexteingabefelder angezeigt werden (vgl. Abbildung 22).

```
{
        dataField: 'cost_center',
        editorType: SiitFormEditorType.Input,
        label: this._translate('cost_center.title'),
        cols: { xs: 12, md: 4 },
        required: true,
}
```

Quellcode 5: Definition eines Eingabefelds der Forms-Komponente



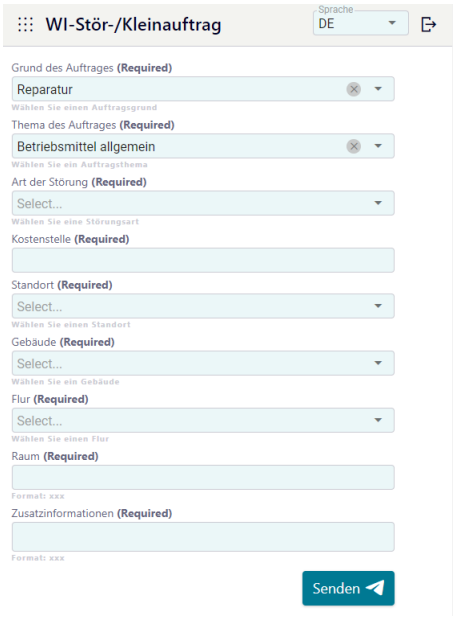


Abbildung 22: WI-Stör-/Kleinauftragserstellung der neuen Webanwendung auf dem Desktop-PC (oben) und mobilen Endgeräten (rechts)

Um eine bessere Benutzererfahrung zu erzielen, werden die Benutzereingaben der Eingabefelder nach dem Absenden oder der Änderung von Dropdown-Feldern, welche für die dynamische Anpassung verantwortlich sind, zurückgesetzt. Dabei werden die Felder jeweils mit entsprechenden Vorbelegungen aus den Account-Daten gefüllt, die mithilfe des Authentication-Services aus der Vorlage bereitgestellt werden. Die Inhalte der Dropdown-Felder für die Standort- und Gebäudewahl werden direkt aus der Datenbank abgerufen und eingefügt. Das Dropdown-Feld der Gebäude wird dabei abhängig vom gewählten Standort automatisch mit den zugehörigen Gebäuden gefüllt.

Durch den Senden-Button wird zusätzlich die Funktionalität der Eingabevalidierung implementiert. Vor Absenden des Auftrages erfolgt eine Überprüfung aller Benutzereingaben auf Gültigkeit. Eine Bildschirmbenachrichtigung informiert den Nutzer bei fehlerhafter Eingabe, während gleichzeitig die betroffenen Felder rot markiert werden. Der Hilfstext weist zusätzlich auf das geforderte Datenformat hin (vgl. Abbildung 23). Erst nach Korrektur eines der markierten Felder kann der Benutzer erneut versuchen, den Auftrag abzusenden. Im Falle einer erfolgreichen Übermittlung wird der Anwender ebenfalls über eine Bildschirmbenachrichtigung informiert. Darüber hinaus erfolgt automatisch der Versand einer Auftragseingangsemail, inklusive der Auftragsdaten und einem Link zur Auftragsübersicht. Der dafür benötigte E-Mail-Service konnte aus der Vorlage übernommen werden.

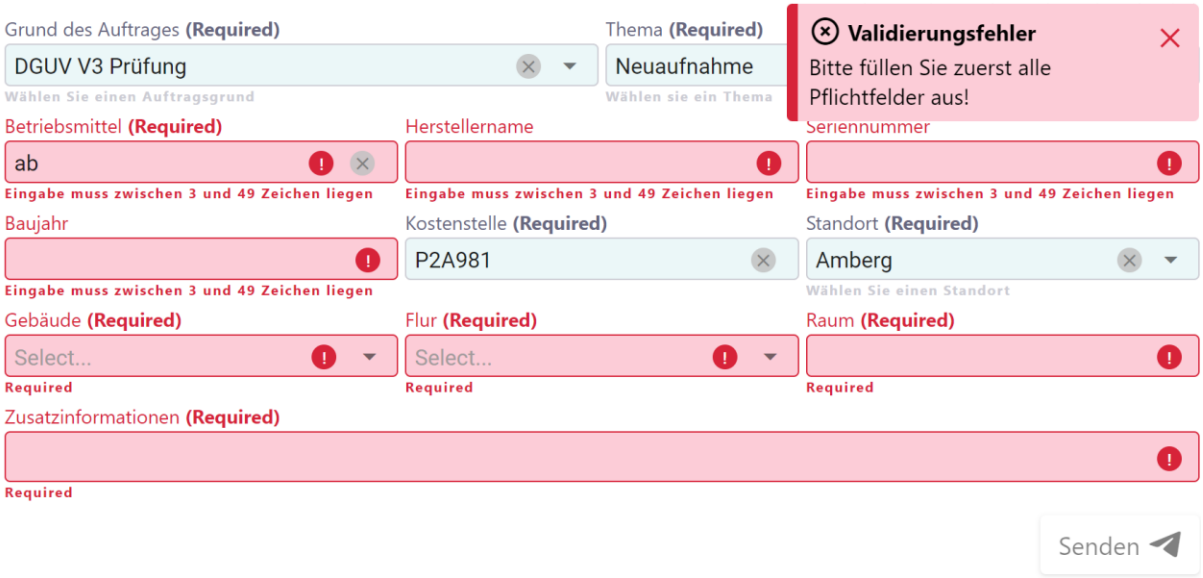


Abbildung 23: Eingabevalidierung der Stör-/Kleinauftragserstellung

7.1.3 Auftragslisten und Pflegedialoge

Die Implementierung der verschiedenen Listen findet mittels des Data-Table-Elementes statt. Das HTML-Listenelement bietet allgemein die Möglichkeit der Sortierung nach Spalten sowie eine Möglichkeit zum Filtern. Des Weiteren kann die Anzahl der Listenreihen angepasst werden. Ähnlich zur Konfiguration der Eingabemaske werden die Spalten auch hier in einem Array konfiguriert (vgl. Quellcode 6). Die Spalten beinhalten dabei die Eigenschaften des Daten- und Darstellungstyps sowie des Spaltennamens (vgl. Abbildung 24). Zusätzlich besteht die Möglichkeit, die Spalten zu sortieren (1) und filtern (2), ebenso wie die Anzahl der präsentierten Listeneinträge zu verwalten (3). In der unteren rechten Ecke der Liste (4) kann zudem die Listenseite gewechselt werden. Durch Anklicken eines Listenelements öffnet sich ein Dialogfenster, welches detaillierte Auftragsinformationen präsentiert. Aufgrund der Abhängigkeit der Spaltenbreite vom Bildschirm können die Texte teilweise abgeschnitten werden. Das Dialogfenster trägt dazu bei dieses Problem zu beheben.

```
{
        "display": this._translate('room.title'),
        "dataField": "room",
        "cellType": "text",
        "dataType": "text",
        "allowSort": true,
        "allowFilter": true
}
```

Quellcode 6: Definition einer Spalte des Listen-Elements

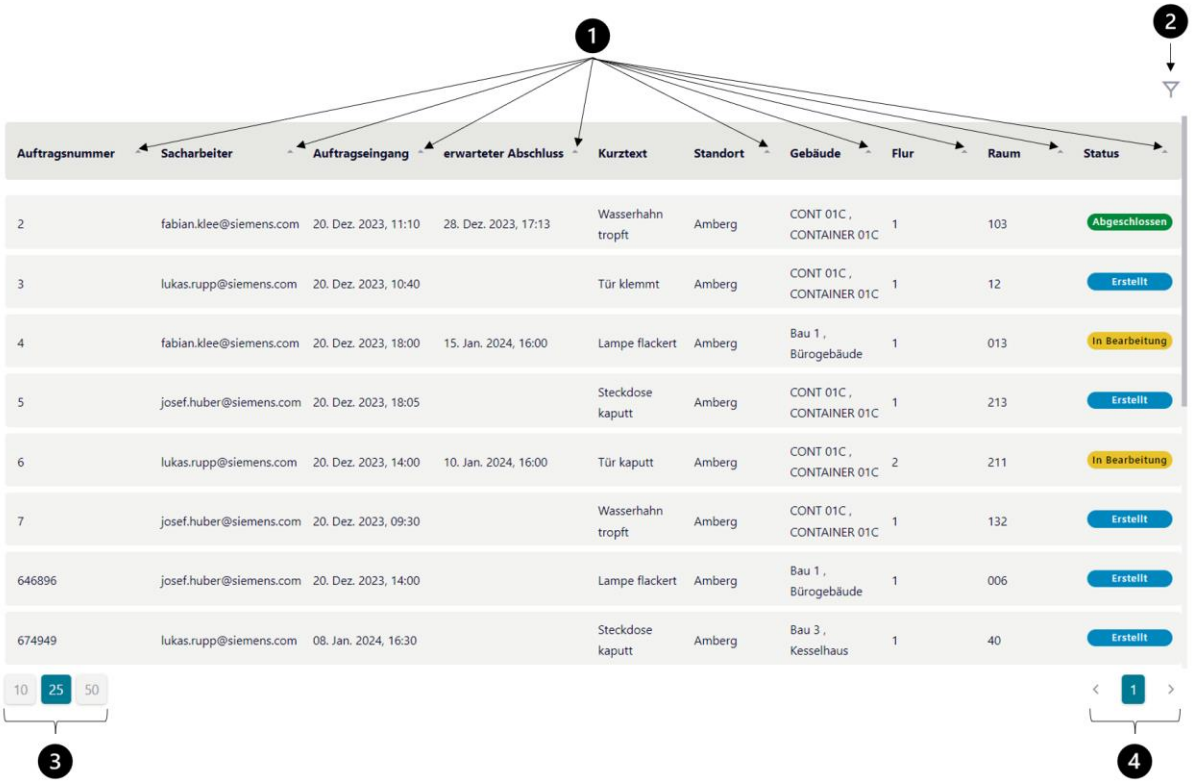


Abbildung 24: List der eigenen erstellten Stör-/Kleinaufträge

Die Auftragslisten sind über eine Angular-Komponente implementiert. In Abhängigkeit des gewählten Pfades wird entweder die Liste der eigenen Aufträge oder die Liste aller Aufträge angezeigt. Der Zugriff auf die Liste aller Aufträge ist jedoch nur für Nutzer mit entsprechender Rolle möglich, welche zusätzlich beim Seitenaufruf vom Server angefragt wird. Die Liste aller Aufträge beinhalten neben den allgemein relevanten Spalten, wie der Auftragsbeschreibung oder dem Übermittlungs- sowie Fertigstellungsdatum, den Auftraggeber. Abhängig von der Benutzerrolle erfolgen dementsprechend verschiedene Datenanfragen an verschiedene Endpunkte. Da im Auftrag nur die Gebäude-ID gespeichert wird, ist es zusätzlich notwendig, die benötigten Gebäudeinformationen vom entsprechenden Endpunkt abzurufen. Dazu wird ein Mapping-Profil erstellt, welches anschließend die Gebäude-ID mit dem jeweiligen Namen in den Tabellendaten verknüpft. Gleiches erfolgt für die Standorte, dessen IDs sich in den Gebäudeinformationen befinden.

Die Pflege- bzw. Verwaltungslisten bieten die zusätzlichen Möglichkeiten der Listenbearbeitung (vgl. Abbildung 25). Über die Eigenschaften „enableAdding“ bzw. „enableRemoving“ sowie den entsprechenden Events, beispielsweise „onRowClicked“, können die jeweiligen Aktionen konfiguriert werden. Für das Erstellen (1) bzw. Löschen (2) von Listeneinträgen wird ein dafür zuständiger Button eingeblendet. Die Bearbeitung findet durch das Anklicken des jeweiligen Eintrages statt. Je nach Benutzeraktion öffnet sich anschließend ein Dialog-Fenster (vgl. Abbildung 26), welches ebenfalls durch das Siit-Package implementiert wird. Das Dialog-Fenster beinhaltet ein Forms-Element mit den benötigten Eingabefeldern bzw. entsprechender Vorbelegung. Zur Vorbeugung des versehentlichen Entfernens von Listeneinträgen wird im Dialogfenster für das Entfernen von Listeneinträgen eine Bestätigung der Aktion gefordert.

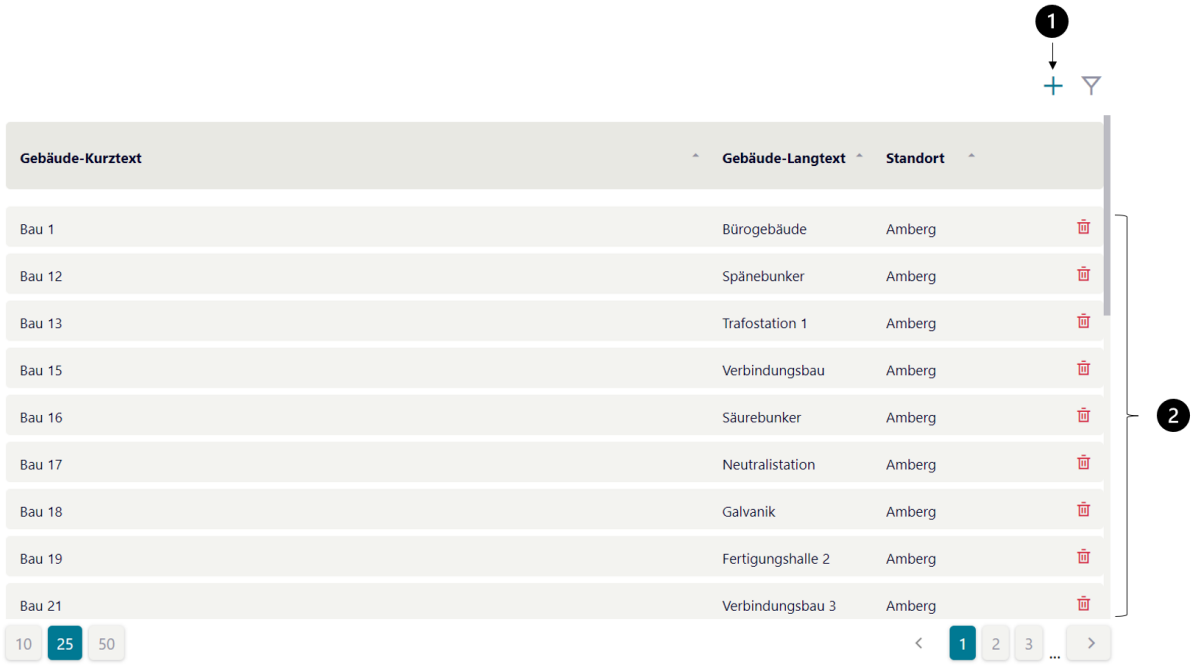


Abbildung 25: Verwaltungsliste der Gebäude

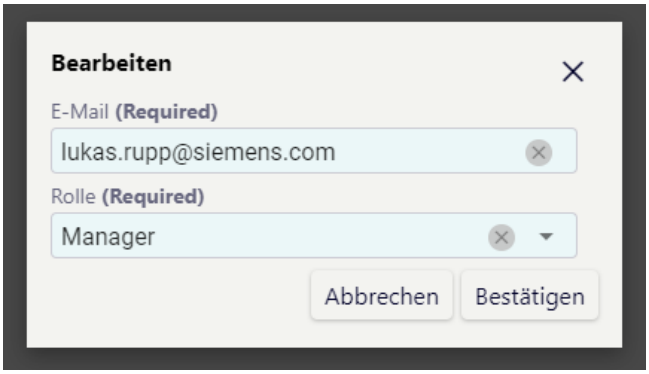


Abbildung 26: Dialogfenster zur Listenbearbeitung

Zur Reduzierung von Datenverkehr wird die Liste nur nach der Erstellung eines neuen Listeneintrages erneut vom Backend abgefragt, da hier eine neue ID erstellt wird. Bei der Bearbeitung bzw. dem Löschen eines Listenelementes werden die Eingabedaten an den Endpunkt versendet. Die Anpassung der Liste erfolgt jedoch lokal, ohne erneute Daten-Anfrage. Des Weiteren wird beim Abruf der Gebäude- und Standortinformationen vom Server ein Mapping-Profil erstellt. Dieses referenziert die jeweiligen Gebäude- und Standort-IDs mit den dazugehörigen Namen. Durch die Erstellung des Mapping-Profils müssen diese Endpunkte ebenfalls nur einmalig beim Seitenaufruf abgefragt werden.

7.2 Backend

Das Backend basiert auf einem .NET-Server unter Verwendung des Entity Framework Core, welches die Kommunikation mit der Datenbank abwickelt. Für die Struktur bzw. den Aufbau des Servers wurden unterschiedliche Patterns/Muster/Vorlagen genutzt, welche im Folgenden erklärt werden.

7.2.1 Verwendete Architekturen und Designmuster

Clean Code

Im Folgenden werden die wichtigsten Aspekte des Clean Codes anhand von Quelle 10 beschrieben:

Der Clean Code ist eine von Robert C. Martin erfundene Software-Architektur, welche sich an verschiedenen anderen Architekturen orientiert und die Unabhängigkeit von Frameworks, Benutzeroberflächen, Datenbanken und anderen externen Diensten erhöht. Diese Unabhängigkeit entspricht einer Modularisierung des Projektes, da alle externen Dienste einfach ausgetauscht werden können, ohne dass der Kern des Programms seine Funktionalität verliert.

Das Diagramm (vgl. Abbildung 27) beschreibt die verschiedenen Ebenen der Software. Je weiter innen sich der Kreis befindet, desto höher ist das Niveau der Software. Während die äußeren Kreise nur Mechanismen darstellen, sind die inneren Kreise Richtlinien.

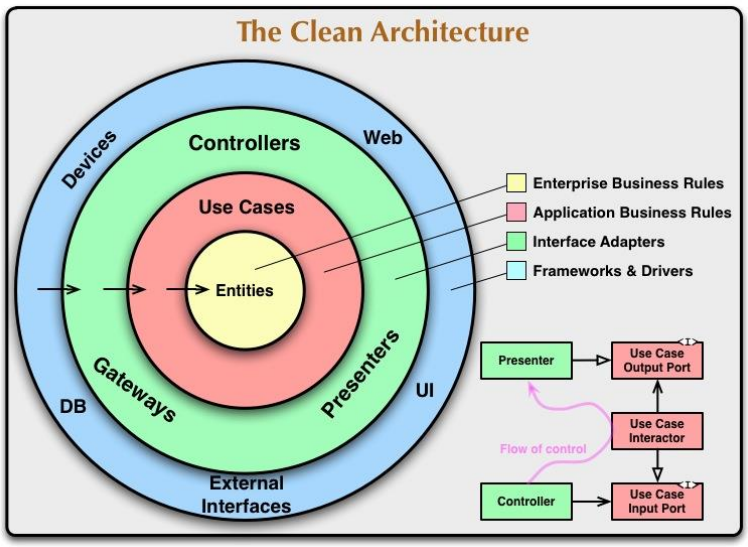


Abbildung 27: Architekturdiagramm des Clean Codes inklusive Beispiel zur Grenzüberschreitung [10]

Im Mittelpunkt des Clean Codes steht die Abhängigkeitsregel. Diese besagt, dass Abhängigkeiten im Quellcode nur nach innen gerichtet sein dürfen. Demzufolge haben die inneren Kreise keine Informationen über die äußeren Kreise. Keine Klassen, Funktionen oder Variablen der äußeren Kreise dürfen in einem inneren Kreis verwendet werden. Die äußeren Kreise dürfen somit grundsätzlich keine Auswirkungen auf die inneren Kreise haben.

Der innerste Kreis enthält Entitäten, welche die allgemeinsten Geschäftsregeln systemweit kapseln. Die Entität selbst kann ein Objekt mit Methoden oder ein Satz von Datenstrukturen und Funktionen sein. Die Form der Entität ist unwichtig, solange sie von verschiedenen Anwendungen im System verwendet werden können. In einer einzelnen Anwendung repräsentieren die Entitäten das Geschäftsobjekt.

Im nächst-äußeren Kreis befinden sich die anwendungsfallspezifischen Geschäftsregeln. Auf dieser Ebene werden alle Anwendungsfälle des Systems gekapselt und implementiert. Um die Ziele des Anwendungsfalls zu erreichen, steuert dieser Softwarebereich den aus- und eingehenden Datenfluss der Entitäten und weist diese darauf hin, ihre systemweiten Geschäftsregeln zu verwenden. Die Anwendungsfallebene ist ebenfalls isoliert von externen Diensten und kann auch nicht von diesen beeinflusst werden. Nur wenn sich Anwendungsfälle ändern, muss auch der Code dieser Schicht geändert werden.

In der dritten Schicht befinden sich Schnittstellenadapter. Die Software aus dieser Ebene hat die Aufgabe, die Daten von einem für die Anwendungsfall- und Entitätsebene geeigneten Datenformat in ein für die externen Dienste geeignetes Format zu konvertieren. Entsprechend der Abhängigkeitsregel dürfen in dieser Ebene keine Informationen über die externen Dienste enthalten sein. Sie dient lediglich zur Konvertierung von Daten.

Allgemein besteht der äußerste Kreis aus Frameworks, Datenbanken oder anderen Diensten, welche für das System benötigt werden. Der Quellcode dieser Schicht umfasst den Bindungscode und sorgt für die Kompatibilität mit den externen Diensten.

In der unteren rechten Ecke von Abbildung 27 wird zudem der Vorgang des Überschreitens der Grenzen dargestellt. Im Beispiel kommunizieren der Controller sowie der Presenter mit der Anwendungsfallebene. Die Quellcodeabhängigkeiten zeigen dabei in Richtung der Anwendungsfälle, da diese auf der nächstinneren Ebene implementiert sind. Die Ebenen kommunizieren dabei nicht direkt miteinander, sondern ausschließlich über die Schnittstellenadapter, um die Abhängigkeitsregel nicht zu verletzen. Diese Schnittstellenadapter müssen daher auch in der äußeren Ebene implementiert werden, um angenommen bzw. richtig interpretiert werden zu können.

Die dabei übergebenen Daten bestehen größtenteils aus einfachen Datenstrukturen bzw. sogenannten Datentransferobjekten. Diese Transferobjekte kapseln eine Reihe von Werten und erlauben es dem Aufrufer, einen gesamten Datensatz in einem einzigen Aufruf anzufordern bzw. zu empfangen [11].

Schichtenarchitektur

Bei der Schichtenarchitektur kann die Anwendung in beliebig viele Schichten aufgeteilt werden. Eine für einen Webserver sinnvolle Methode ist hierbei die Aufteilung in die Präsentationsschicht, die Domänenlogikschicht und die Datenzugriffsschicht. Einige Vorteile dieser Unterteilung liegen hierbei in der Modularität und somit der Ersetzbarkeit der verschiedenen Module, aber auch in der Einbindung mehrerer Module in eine Schicht [12].

Repositorymuster

Das Repositorymuster ist ein Domain-Driven Design-Muster, welches genutzt wird, um die Persistenzprobleme bzw. die Datenspeicherung außerhalb der Domänenschicht zu halten. Die Domänenschicht enthält Schnittstellen mit Implementierungen aus persistenzspezifischen Adaptern. Diese Adapter sind wiederum an anderer Stelle in der Anwendung definiert.

Die Repositoryimplementierungen stellen Klassen dar, welche die für den Zugriff auf Datenquellen erforderliche Logik kapseln. Diese Klassen zentralisieren allgemeine Funktionen für Datenzugriffe [24]. Dadurch kann die Logik für den Abruf und die Zuordnung der Daten zum Entitätsmodell von der Geschäftslogik getrennt werden. Das Repository selbst vermittelt anschließend zwischen der Datenquellenebene und der Geschäftsebene der Anwendung [25].

Unit-of-Work-Muster

Eine Unit-of-Work bezieht sich auf einzelne Transaktionen, welche mehrere Einfüge-, Aktualisierungs- oder Löschvorgänge umfasst. Bei einer Geschäftstransaktion fallen mehrere solcher Vorgänge gleichzeitig an und würden im Einzelnen jeweils eine eigene Transaktion ausführen. Die Unit-of-Work bündelt diese einzelnen Vorgänge zu einer einzigen Transaktion und führt diese anschließend einmalig aus. Durch die Anwendung des Unit-of-Work-Musters lässt sich in zahlreichen Situationen eine Verbesserung der Anwendungsleistung im Speicher erzielen. Des Weiteren wird das Blockieren von Transaktionen in Datenbanktabellen verringert, da alle beabsichtigten Operationen bereits in einer Transaktion ausgeführt werden [5].

Abhängigkeitsinjektion– Dependency Injection

Die Abhängigkeitsinjektion ist ein Entwurfsmuster, welches verwendet wird, um die Abhängigkeiten zwischen verschiedenen Komponenten eines Systems zu verwalten. Das Hauptziel ist hierbei die Entkopplung der konkreten Typen vom Code, welcher von diesen Typen abhängt. Dadurch wird unter anderem die Modularität, aber auch die Wartbarkeit bzw. Erweiterbarkeit verbessert.

Bei der Dependency Injection liegt die Steuerung für die Erstellung und Verwaltung von Abhängigkeiten nicht direkt in der Klasse bzw. der Komponente selbst, sondern erfolgt durch Container. Diese Container verwalten eine Liste von Registrierungen und Zuordnungen zwischen Schnittstellen und abstrakten Typen sowie den konkreten Implementierungen dieser Typen. Der Container übernimmt damit die Verantwortung für die Bereitstellung der benötigten Abhängigkeiten.

Durch die Verwendung dieser Container entfällt die Notwendigkeit der Klasse, ihre Abhängigkeiten zu finden und die Lebensdauer dieser zu verwalten. Der Container ermöglicht zusätzlich die Zuordnung implementierter Abhängigkeiten ohne Auswirkungen auf die Klasse selbst. Des Weiteren wird durch die Benutzung der Container die Testbarkeit erleichtert und die Wartungsfreundlichkeit erhöht [7].

7.2.2 Implementierung

Der .NET-Server ist nach der Schichtenarchitektur in drei Ebenen aufgeteilt. Den Kern des Systems stellt die Domänenebene dar, welche die Geschäftslogik enthält. Die zweite Schicht ist die Datenebene, über welche sämtliche Kommunikation mit Drittanwendungen abläuft. Für die Präsentationslogik ist die äußerste Schicht zuständig. Durch die Präsentationsschicht wird die Kommunikation, Rechteverwaltung und der Zugriff der Daten für das Frontend gesteuert.

Im folgenden Abschnitt wird von verschiedenen Entitäten gesprochen. Für eine einheitliche Zuordnung der jeweiligen Entitäten im System bzw. um eine Unterscheidung zu ermöglichen, sollen diese nun unterschiedlich betitelt werden. Die Geschäftsobjekte bzw. Entitäten der Domänenebene werden weiterhin als Entitäten bezeichnet. Der Begriff Modell wird verwendet, wenn die Entitäten der Datenebene gemeint sind, welche die persistenten Daten repräsentieren. Die API-Modelle sind die Entitäten der Präsentationsschicht bzw. die Datentransferobjekte.

Domänenebene

Die Domänenebene des Systems stellt gemäß der Clean-Architecture den innersten Kreis dar. Um die Abhängigkeitsregel zu beachten, ist die Domänenebene von keiner anderen Ebene abhängig. Die Implementierung der Schicht basiert allgemein auf den Strukturen des Repository- und Unit-of-Work-Musters. Zusätzlich wird Abhängigkeitsinjektion verwendet. Im Folgenden werden die wichtigsten Klassen und Schnittstellen dieser Ebene erläutert.

Entitätsklassen

Im Kern der Ebene stehen die Entitätsklassen (z.B. „Buildings“), welche die Geschäftsobjekte repräsentieren. Diese Klassen dienen als Datenmodelle und bilden die Grundlage für die Persistenzebene. Sie definieren die Eigenschaften der Entitäten und ermöglichen die Abbildung von Entitätsinformationen in der Datenbank. Durch die klare Definition der Entitätsklassen wird eine zentrale Struktur für die Interaktion zwischen der Geschäftslogik und der Datenzugriffschicht geboten.

Service-Klassen

Die Service-Klassen (z.B. „BuildingsService“) fungieren als zentrale Schnittstelle zwischen der Präsentationsschicht und der Datenzugriffsschicht. Sie sind die Implementierungen der dazugehörigen Schnittstellen (z.B. „IBuildingsService“) und steuern die Operationen auf die Entitäten (vgl. Quellcode 7). Neben dem Aufruf der grundlegenden CRUD-Opera-tionen (CRUD, Abk. Create, Read, Update, Delete) sind sie für die Aufrufe weiterer Methoden, unter anderem zur Abfrage der Gesamtanzahl von Entitäten, unter Berücksichtigung von optionalen Filterkriterien, zuständig. Im Konstruktor der Klasse wird eine Instanz der Unit-of-Work-Schnittstelle über eine Abhängigkeitsinjektion bereitgestellt.

```
public class BuildingsService : IBuildingsService
{
     private readonly IUnitOfWork _unitOfWork;
     public BuildingsService(IUnitOfWork unitOfWork)
     {
         _unitOfWork = unitOfWork;
     }
     public Task<int> CountAsync(Filter? filter = null)
     {
         return _unitOfWork.Buildings.CountAsync(filter);
     }
     public Task<List<Building>> GetListAsync()
     {
         return _unitOfWork.Buildings.GetListAsync();
     }
     […]
}
```

Quellcode 7: Ausschnitt aus dem Quellcode der BuildingsService-Klasse

Unit-of-Work-Schnittstelle

Durch die Unit-of-Work-Schnittstelle wird die Zusammenstellung von Repositories, welche für den Zugriff auf die Tabellen der Datenbank verantwortlich sind, definiert. Die in der Schnittstelle enthaltenen Eigenschaften repräsentieren die einzelnen Repository-Schnittstellen (z.B. „IBuildingsRepository“) der Modelle. Diese Repository-Schnittstellen basieren dabei auf einem, durch die Vorlage bereitgestellten, generischen „IGenericRepository“.

SAP Client Schnittstelle

Die SAP Client Schnittstelle definiert die Kommunikation mit dem SAP-System. Dazu gehört bisher eine Methode zum Senden von Aufträgen an das SAP-System.

Datenebene

Die Datenebene des .NET-Servers bildet die Brücke zwischen der Domänenebene und externen Diensten wie der Datenbank oder dem SAP-System. Sie ist von der Domänenebene abhängig und implementiert unter anderem die erforderlichen Repositories inklusive des Unit-of-Work-Musters, um den Zugriff auf die Datenbank zu gewährleisten. Die Datenzugriffsebene ist in zwei verschiedene Hauptkomponenten unterteilt, welche für die Datenpersistenz verantwortlich sind, die „Infrastructure.SAP“ und die „Infrastructure.SqlData“. Im Folgenden werden die wichtigsten Bestandteile dieser Ebene erläutert.

SAP-Datenklasse

Diese Klasse ist darauf ausgerichtet, die Interaktion mit der SAP-Datenbank zu erleichtern. Da die Implementierung einer funktionalen Schnittstelle mit dem SAP-System im Anschluss an die Bachelorarbeit erfolgt und somit nicht zum Umfang der Bachelorarbeit gehört, dient diese Klasse zurzeit nur als Platzhalter. Die „SAPClient“-Klasse ist die Implementierung der „ISAPClient“-Schnittstelle und implementiert die Methoden „SendOrder“ und „GenerateRandomOrderNumber“. Aktuell gibt diese Klasse eine positive Übermittlungsmeldung sowie eine zufällige Auftragsnummer zurück.

SQL-Datenklasse

DBContext - Datenbankkontext

Der Datenbankkontext ist die zentrale Klasse, welche die Entity Framework Core Funktionalitäten bereitstellt. In dieser Klasse erfolgt die Definition und Konfiguration der Modelle, bzw. der DBSets der verschiedenen Modelle, die die Datenbanktabellen repräsentieren.

Mapper und Profil

Der Mapper und das Mapping-Profil (z.B. „BuildingProfile“) ermöglichen die Umwandlung von Entitätsobjekten in Modelle und umgekehrt. Für diese Funktionalität wird die AutoMapper-Bibliothek verwendet, um die Abbildungen zwischen den Schichten zu erleichtern. Das Mapping-Profil definiert, wie die Eigenschaften zwischen den Entitäten und Modellen abgeglichen werden.

Repositories

Die Repositories dieser Ebene (z.B. „BuildingsRepository“) sind für den Zugriff auf die Datenbanktabellen sowie die Ausführung der Datenbankoperationen verantwortlich. Die jeweiligen Repository-Klassen sind die Implementierungen der Repository-Schnittstellen ihrer Modelle. Dabei erben alle diese Klassen das Verhalten von der generischen Klasse „GenericRepository“, welche wiederum von der „IGenericRepository“-Schnittstelle, der Domänenebene definiert wird (vgl. Quellcode 8). Diese Klasse stellt sicher, dass alle grundlegenden CRUD-Operationen sowie komplexe Abfragen durch die Repositories unterstützt werden. Das „GenericRepository“ sowie das „IGenericRepository“ wurden durch die Vorlage der IT-Abteilung bereitgestellt.

```
public interface IGenericRepository<TEntity>
{
        public Task<int> CountAsync(Filter? filter = null);
        [...]
        public void Update(TEntity entity);
        public void Delete(int id);
        public void Delete(string id);
        public void Delete(TEntity entity);
}
```

Quellcode 8: Ausschnitt aus dem Quellcode des IGenericRepositorys

Unit-of-Work

Die „UnitOfWork“-Klasse ermöglicht die Verwendung mehrerer Repository-Operationen innerhalb einer Transaktion. Sie ist die Implementierung der „IUnitOfWork“-Schnittstelle der Domänenebene, welche alle Repositories der Modelle bereitstellt. Der Aufruf der Repositories besitzt ein Lazy Loading Verhalten, wodurch diese nur im ersten Aufruf instanziiert werden. Dadurch erfolgt die Initialisierung der Repository-Instanzen nur bei Bedarf. Anschließend können diese dann wiederverwendet werden.

Präsentationsebene

Die Präsentationsebene der Web-API-Anwendung definiert die API-Modelle, Authorisierungseigenschaften sowie Controller. Für den Zugriff auf die zugrunde liegende Domänenlogik und die Daten ist die Präsentationslogik von der „Domain.Core“, der „Infrastructure.SAP“ sowie der „Infrastructure.SqlData“ abhängig. Im Folgenden werden die wichtigsten Bestandteile dieser Ebene erläutert.

API-Modelle

In den API-Modellen werden die für die Antwort (z.B. „BuildingApiResponse“) und Anfrage benötigten Modelle (z.B. „BuildingApiRequest“) definiert. Sie enthalten die notwendigen Eigenschaften für den Informationsaustausch zum Frontend und stellen die eigentlichen Datentransferobjekte dar.

Autorisierung und Authentifizierung

Für die Autorisierung und Authentifizierung sorgen verschiedene Klassen:

Die „PermissionAttribute“-Klasse ist ein benutzerdefiniertes auf Rollen basierendes Autorisierungsattribut und wird für die Definition von Berechtigungsrichtlinien verwendet. Sie besteht aus einer Konstante als Präfix sowie der dazugehörigen Rolle.

Der „PermissionHandler“ ist ein Autorisierungshandler, welcher überprüft, ob der Benutzer die erforderliche Rolle besitzt (vgl. Quellcode 9).
Dieser geht dabei wie folgt vor:

- Überprüfung der Benutzerauthentifizierung
    - Abrufen der Benutzer-E-Mail aus den Claims, welche sich im Autorisierungstoken befinden (in Microsoft-Profil enthalten)
    - Überprüfung der Benutzerrolle über E-Mail aus Datenbank
    - Vergleich der Nutzerrolle mit erforderlicher Rolle
- pro Aktion werden LogEvents ausgerufen

Claims oder Ansprüche sind identitätsabhängige Informationen, z.B. E-Mail, Name und Abteilungsbezeichnung eines Benutzers und werden vor allem für die Autorisierung verwendet [34].

Für die dynamische Erstellung von Autorisierungsrichtlinien, basierend auf den Anforderungen des „PermissionAttributes“, ist der „PermissionPolicyProvider“ zuständig. Dazu extrahiert dieser die Rolle aus dem Policy-Namen und fügt diesen zur Berechtigungspolitik hinzu. Der „PermissionPolicyProvider“ ist lediglich die Middleware, welche die Anfrage auffängt, bevor diese an den Controller gesendet wird.

Die „PermissionRequirement“-Klasse stellt die Anforderung an die Autorisierung dar und wird vom „PermissionHandler“ verwendet.

```
protected override async Task HandleRequirementAsync(AuthorizationHandlerContext
context, PermissionRequirement requirement)
{
    if (context.User?.Identity?.IsAuthenticated == true)
    {
        string? userEmail = context.User.FindFirst("mail")?.Value;
        if (userEmail != null)
        {
            _logger.LogTrace(AppLogEvents.Debug, $"Request from user: {user
            Email}");
            using IServiceScope scope = _serviceScopeFactory.CreateScope();
            var userroleService = scope.ServiceProvider.GetRequiredService<IUser
            rolesService>();
            var userrole = await userroleService.GetByEmailAsync(userEmail);
             if (userrole.Role >= requirement.Role)
            {
                context.Succeed(requirement);
            }
        } else {
            _logger.LogError(AppLogEvents.MissingData, message: "Missing user
            Email");
            context.Fail(new AuthorizationFailureReason(this, "Missing user
            Email"));
        }
    } else {
        _logger.LogError(AppLogEvents.Unauthorized, message: "Unauthenticated u
        ser");
        context.Fail(new AuthorizationFailureReason(this, "Unauthenticated u
        ser"));
    }
}
```

Quellcode 9: Ausschnitt aus dem Quellcode der PermissionHandler-Klasse

Controller

Die API-Controller (z.B. „BuildingsController“) stellen die verschiedenen Endpunkte bereit (vgl. Quellcode 10). Sie werden dabei durch einen von der Vorlage bereitgestellten „BaseDataController“ erweitert, welcher für den Abruf der Mapper und Logger zuständig ist. Für sicherheitsrelevante Aktionen können Autorisierungsattribute bzw. eine vom Benutzer benötigte Rolle konfiguriert werden.
Die Controller der Gebäude enthalten folgende Endpunkte:

- Count: gibt die Anzahl der Gebäude zurück
- Page: gibt eine paginierte Liste von Gebäuden zurück
- FullList: gibt eine vollständige Liste von Gebäuden zurück
- Details: gibt Details zu einem bestimmten Gebäude anhand der ID zurück.
- Create, Edit, Delete: Aktionen für das Erstellen, Bearbeiten und Löschen von Gebäuden, erfordern Administratorrechte

Eine Adresse eines Endpunktes könnte zum Beispiel “https://webservices.siemens.com/faultsignal/api/Buildings/Details/{id}” lauten.

Diese setzt sich wie folgt zusammen: Route aus „environment.ts“ / Applikationsname (aus Azure Gateway) / api / Controllername / Endpunkt / ggf. Parameter.

```
[HttpGet("Details/{id}")]
public async Task<ActionResult<BuildingApiRequest?>> Details([FromRoute] int id)
{
    var data = await _buildingsService.GetByIdAsync(id);
    if (data != null)
    {
        return Ok(_mapper.Map<BuildingApiResponse?>(data));
    }
    else
    {
        return NotFound();
    }
}
```

Quellcode 10: Ausschnitt aus dem BuildingsController

AutoMapper und Mapping-Profile

Die AutoMapper-Profile (z.B. „BuildingProfile“) sind für die Abbildung von Entitäten auf die API-Modelle und umgekehrt zuständig.

7.2.3 Datenfluss

Im Folgenden wird der Abruf von Daten aus dem Backend am Beispiel von allen Auftragsdaten mit Adminrechten beschrieben.

Wenn der User die Auftragsübersicht aller Aufträge aufruft, stellt das Frontend eine Datenanfrage an den „OrdersController“ bzw. den Endpunkt „PageAll“ sowie „CountAll“ über die URL des Endpunktes (vgl. Abbildung 28). Am Beispiel des „OrdersControllers“ sind diese „/api/Orders/PageAll“ (1) bzw. „/api /Orders/CountAll“. Dabei übergibt das Frontend in der Anfrage die Parameter, wie die Anzahl der Listenelemente pro Seite, welche standardmäßig auf 25 eingestellt ist sowie weiteren Filterkriterien.

Die Endpunkte „PageAll“ und „CountAll“ sind nur mit mindestens der Manager-Rolle erreichbar (vgl. Quellcode 11). Dadurch wird im Backend zuerst durch den „PermissionPolicyProvider“ eine Autorisierungsrichtlinie bzw. ein „Permission-Requirement“ mit der entsprechend, vom Endpunkt benötigten, Rolle erstellt. Der „PermissionHandler“ (2) beginnt anschließend mit der Überprüfung des Nutzers auf Autorisierung. Zudem wird die E-Mail-Adresse des Aufrufers aus den Claims sowie die dazugehörige Rolle aus der Datenbank abgerufen und mit der benötigten Rolle verglichen.

Als nächstes erfolgt der Datenabruf über die „GetPageAsync“-Methode (3) aus dem „IOrdersService“ in der Domänenebene (vgl. Quellcode 11). Die „IOrdersService“-Schnittstelle ist eine Referenz für die Implementierung des „OrdersService“ und wird über Abhängigkeitsinjektion bereitgestellt. Im „OrdersService“ erfolgt der Aufruf der „GetPageAsync“-Methode aus dem „OrdersRepository“ (4), welches durch die Unit-of-Work bereitgestellt wird. Die Unit-of-Work wird dabei ebenfalls über Abhängigkeitsinjektion zur Verfügung gestellt. Das „OrdersRepository“ erbt das Verhalten des „GenericRepository“, über welches anschließend der eigentliche Zugriff auf die Datenbank erfolgt. Im Endpunkt des Controllers werden die Daten zuletzt über den Mapper zugeordnet und über die „OrderApiResponse“ (5) zurück an das Frontend übergeben.

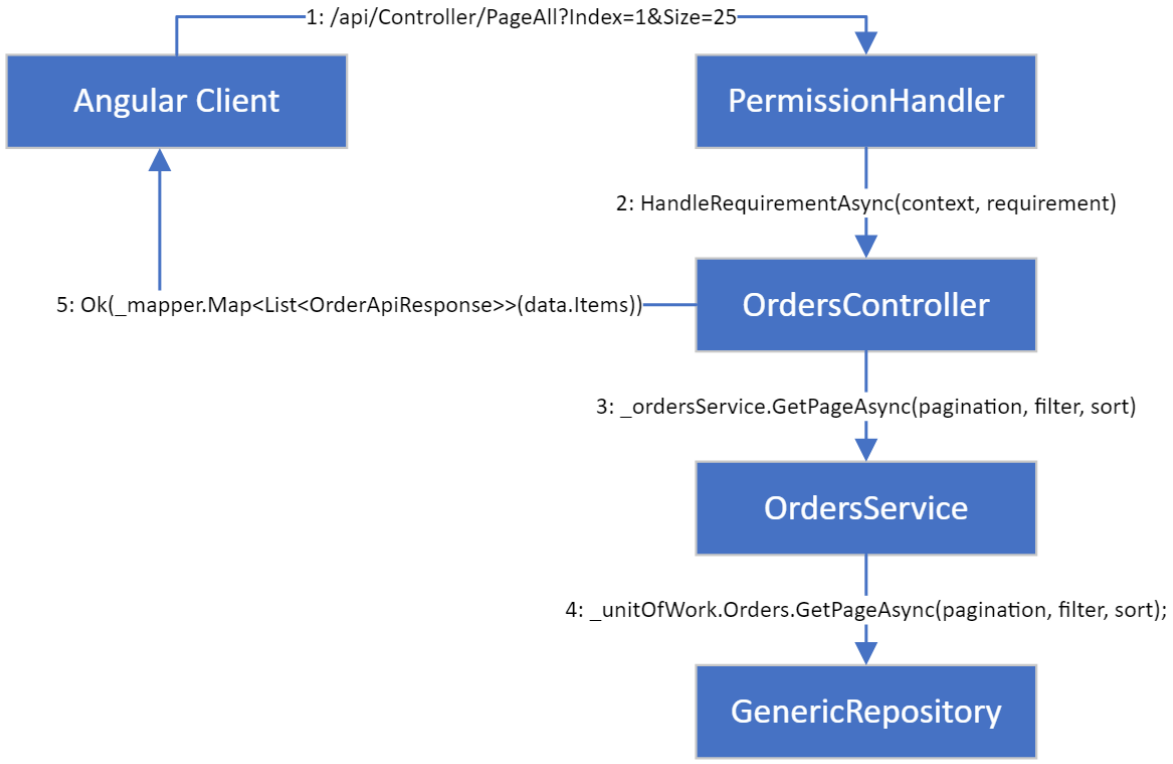


Abbildung 28: UML-Kommunikationsdiagramm des Aufrufs von PageAll aus dem OrdersController

```
[HttpGet("PageAll")]
[Permission(Role.MANAGER)]
public async Task<ActionResult<List<OrderApiRequest>>> Page([FromQuery] Pagination
pagination, [FromQuery] Filter? filter, [FromQuery] Sort? sort)
{
    //Datenabruf
    var data = await _ordersService.GetPageAsync(pagination, filter, sort);
    if (data != null)
    {
        return Ok(_mapper.Map<List<OrderApiResponse>>(data.Items));
    }
    else
    {
        return NotFound();
    }
}
```

Quellcode 11: Ausschnitt aus dem OrdersController

8. Evaluation

In diesem Kapitel wird die prototypische Implementierung anhand der aufgestellten Anforderungen evaluiert.

Im Rahmen dieser Arbeit wurde ein Konzept sowie eine prototypische Implementierung einer Webanwendung zur Auftragserstellung entwickelt. Eine detaillierte Einarbeitung in das vorhandene System erfolgte insbesondere im Rahmen der Anforderungsanalyse. Die Umsetzungsmöglichkeiten wurden in einer Zusammenstellung präsentiert. Auf Basis des entwickelten Lösungskonzeptes konnte die prototypische Webanwendung realisiert werden.

Diese nutzt den Anforderungen entsprechend den firmeninternen Microsoft Account zur Anmeldung. Auf der Startseite wird eine klare Strukturierung geboten, mit welcher die Webanwendung nativ bedient werden kann. Die Erstellung von Stör-/Kleinaufträgen ist über eine dynamische Eingabe-Webmaske möglich, durch die der Benutzer anhand der Eingaben geführt wird. Der Anwender wird mittels Hilfstexten an den Eingabefeldern beim Anlegen des Auftrages unterstützt. Zusätzlich sind die Eingabefelder sinnvoll vorbelegt, sofern die benötigten Daten aus dem Benutzerprofil entnommen werden können. Bei fehlerhafter Eingabe wird unmissverständlich auf das Problem hingewiesen und entsprechende Felder markiert.

Eine Möglichkeit zur Datenerhebung wird durch Listen auf der Webanwendung realisiert. Dabei wird zwischen einer Liste zur Übersicht eigener Aufträge, welche für alle Standortmitarbeiter zugänglich ist, sowie einer Liste aller Aufträge unterschieden. Der Zugriff auf die Liste aller Aufträge ist auf Benutzer mit entsprechender Rolle beschränkt. Das System unterscheidet die Benutzer anhand der fehlenden Rolle, einer Manager-Rolle und einer Admin-Rolle. Die Manager-Rolle befugt zum Einsehen der Liste aller Aufträge im System, während die Admin-Rolle zusätzlich zur Benutzung der Zugriffsverwaltung sowie der Pflege der Standort- und Gebäudeliste befugt. Die Abfrage der Rolle ist im Backend über den Vergleich der Benutzerrolle mit der benötigten Rolle implementiert. Die Benutzerrolle kann dabei mittels Suche der E-Mail aus Benutzerinformationen, welche bei der Anfrage an den Server mitversandt werden, in der internen Datenbank bestimmt werden. Die sicherheitsrelevanten Endpunkte sind durch die benötigte Rolle geschützt und ein unautorisierter Zugriff dadurch ausgeschlossen.

Die Aufträge in den Listen enthalten einen Status, über welchen der Benutzer Rückmeldung erhalten kann. Für weitere Fragen ist der jeweilige Sachbearbeiter als Ansprechpartner durch die E-Mail hinterlegt. Zusätzlich bekommt der Auftraggeber eine E-Mail über den Eingang des Auftrages.

Das Responsive Webdesign verbessert die Zugänglichkeit der Webanwendung und sorgt für die Anpassung an verschiedene Bildschirmgrößen. Das allgemeine Oberflächendesign orientiert sich dabei an den erstellten Mockups. Durch die Nutzung der durch Siemens bereitgestellten Bibliothek stellt die Webanwendung vorerst nur ein helles Design zur Verfügung. Die Bibliothek wird zeitnah auf Grundlage der Mockups um ein dunkles Design erweitert.

Das Anhängen von Bildern oder anderen Dateien an den Auftrag konnte aufgrund der zeitlichen Beschränkung dieser Arbeit nicht realisiert werden. Vor Integration dieser Funktionalität muss zuerst durch die dafür zuständige Institution geprüft werden, ob die SAP-Datenbank dieser Anforderung entsprechen kann. Des Weiteren ist es im bisherigen Systemstand nicht möglich, den Benutzer via E-Mail über eine Auftragsstatusänderung zu informieren. Da die Statusänderung im SAP-System stattfindet, sollte die Benachrichtigung durch dieses versandt werden. Alternativ dazu wäre es ebenfalls möglich, das SAP-System den E-Mail-Endpunkt des WI-Stör-/Kleinauftrag-Servers aufrufen zu lassen.

Zusammenfassend lässt sich feststellen, dass die prototypische Implementierung der Webanwendung die gestellten Anforderungen in weiten Teilen erfüllt. Die Weiterentwicklung des Systems stellt eine erhebliche Verbesserung im Vergleich zur vorherigen Version dar. Das neue Design und die optimierte Benutzerfläche bieten sowohl den Auftraggebern als auch den WI-Sachbearbeitern eine verbesserte Möglichkeit zur Übersicht und effizienten Nutzung der Webanwendung. Die Integration einer eigenen Datenbank eröffnet die Chance für eine unkomplizierte Datenanalyse, wodurch interne Prozesse in der Werksinfrastruktur fundiert untersucht und dargestellt werden können. Hieraus lassen sich wertvolle Erkenntnisse gewinnen, die als Grundlage für zukünftige Verbesserungen dienen können. Die prototypische Umsetzung präsentiert eine moderne Architektur, die mit einer verbesserten Systemleistung und Effizienz einhergeht. Damit schafft die Webanwendung eine zeitgemäße Plattform, die den Anforderungen an eine moderne Arbeitsumgebung gerecht wird.

Darüber hinaus wurde das System umfassend getestet, wobei insbesondere Komponenten-Tests des .NET-Servers mit einer Code-Abdeckung von etwa 66% durchgeführt wurden. Das Frontend wurde aufgrund Vorgaben der IT-Abteilung ausschließlich manuellen Tests unterzogen. Diese Tests gewährleisten eine hohe Qualität und Stabilität der implementierten Funktionen.

9. Zusammenfassung und Ausblick

Im folgenden Kapitel erfolgt die Zusammenfassung der Arbeit sowie ein Ausblick in die Zukunft des Projektes.

9.1 Zusammenfassung

Das Ziel dieser Arbeit war die Konzeption und prototypische Implementierung einer internen Webanwendung zur Auftragserstellung als Modernisierungsmaßnahme. Die Vorgehensweise bei der Implementierung soll als Vorarbeit für die Weiterentwicklung des Systems dienen. Die Anforderungsanalyse beschreibt das Anwendungsszenario der Webanwendung und die dafür benötigten Funktionen, welche das System bereitstellen muss. Neben der bestehenden Funktionalität der alten Webanwendung, die das Erstellen eines Auftrages ermöglicht, sollten neue Features implementiert werden, um die Nutzererfahrung zu erweitern. Dazu gehörte insbesondere die Visualisierung der Aufträge, die einen umfassenden Überblick über den Auftrag inklusive einer Rückmeldung in Form eines Status ermöglicht. Des Weiteren sollte der Login über den internen Firmenaccount erfolgen und die Verfügbarkeit der Webanwendung soll auf allen Endgeräten gewährleistet sein. In einem Vergleich der dafür zur Verfügung stehenden Umsetzungsmöglichkeiten wurden diese beschrieben und die optimale ausgewählt.

Für die genutzte Umsetzungsmöglichkeit wurde anschließend ein ausgiebiges Lösungskonzept erstellt, welches unter anderem die Systemarchitektur beinhaltet. Für das Frontend soll das Angular Framework verwendet werden. Das Backend basiert auf einem .NET-Server, welcher mittels des Entity Framework Core auf die SQL-Server 2022 Datenbank zugreifen kann. Des Weiteren wurden Designentwürfe entwickelt, welche als Vorlage für die Webanwendung gelten. Zur Vorbereitung der Datenbank wurde die alte Website analysiert und auf Grundlage dieser ein Datenbankkonzept entwickelt. Im Anschluss erfolgt die Implementierung der Webanwendung auf Grundlage der Anforderungen und der aufgestellten Mockups sowie dem Datenbankkonzept. Zuletzt wurden die Arbeitsergebnisse evaluiert und das Programm verschiedenen Komponenten-Tests unterzogen, welche die Funktionalität der Webanwendung bestätigten.

Die Webanwendung erleichtert die Auftragserstellung und -verwaltung für alle Benutzer des Systems erheblich und bietet Möglichkeiten für die Erweiterung der bisherigen Funktionalitäten.

9.2 Ausblick

Die Implementierung der Webanwendung bietet eine gute Grundlage zur Weiterentwicklung. Nachdem die grundsätzlichen Funktionen implementiert und die Mindestanforderungen erfüllt sind, können nun weitere Funktionalitäten hinzugefügt werden.

Ein wichtiger Schritt ist dabei die Anbindung der Webanwendung an die SAP-Datenbank und das Deployment des anschließenden Release-Standes auf das Produktivsystem. Dazu gehört die Übertragung der auf der Datenbank erstellten Stör-/Kleinaufträge, als auch die Aktualisierung der SQL-Datenbank mit der SAP-Datenbank. Mit der darauffolgenden standortweiten Nutzung können Rückmeldungen der Nutzer eingeholt werden, um stetige Verbesserungen vorzunehmen und anfallende Fehler zu beseitigen.

Weiterhin können auf Grundlage der dabei entstehenden Daten Algorithmen zur Datenanalyse und -darstellung entwickelt werden, um zukunftsorientierte Maßnahmen zur Verbesserung des Standortes zu planen. Hierzu gehören beispielsweise Prozesse für die vorausschauende Wartung (predictive Maintenance) oder der gezielte Austausch häufig defekter Objekte. Darüber hinaus können die internen Abläufe der Werksinfrastruktur mithilfe der neu gewonnenen Analysedaten von leicht zugänglichen Auswertungen profitieren. Ein Beispiel dafür wäre die verbesserte Einsicht in die Bearbeitungszeiträume. Durch die detaillierte Analyse dieser Zeitpunkte können effizientere Prozesse identifiziert und geplant werden, um die Gesamtleistung zu optimieren. Dies ermöglicht eine präzisere Steuerung von Arbeitsabläufen und trägt dazu bei, Engpässe zu minimieren sowie den gesamten Auftragsabwicklungsprozess zu optimieren.

Die Systemdatenbank bietet außerdem die Möglichkeit zur Implementierung einer Zwischenspeicherung der Aufträge, falls die SAP-Datenbank nicht erreichbar sein sollte. Durch einen zusätzlichen Button in der Übersichtsliste der erstellten Kleinaufträge könnte der Auftrag dann erneut abgesendet werden, sobald die SAP-Datenbank wieder verfügbar ist, ohne den Auftrag komplett neu anlegen zu müssen. Alternativ dazu könnte das System eigenständig den Status der SAP-Datenbank abfragen und die Aufträge eigenständig abschicken, sobald das SAP-System wieder verfügbar ist.

Um die Informationssicherheit des Systems zu verbessern, könnten künftig durch den Einsatz des Sicherheitskonzepts „Content Security Policy“ Maßnahmen gegen Cross-Site-Scripting-Angriffe ergriffen werden. Darüber hinaus wäre es diesbezüglich möglich, Maßnahmen gegen Flooding Angriffe zu implementieren, um einer Systemüberlastung vorzubeugen und die kontinuierliche Verfügbarkeit des Systems zu gewährleisten. Für die Sicherstellung der Leistung und Effizienz des Systems wäre es zusätzlich möglich, weitere Tests anzulegen.

Zur Erfüllung der noch nicht implementierten Anforderungen sowie der zuvor genannten Erweiterungen bietet die gewählte modulare Architektur eine optimale Grundlage. Diese Architektur ermöglicht eine effiziente Integration neuer Funktionalitäten und bietet somit eine solide Basis für die Weiterentwicklung des Systems.

Abbildungsverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

WI - Werksinfrastruktur

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

URL – Uniform Resource Locator

CSS – Cascading Style Sheets

ERP – Enterprise Resource Planning

KI – künstliche Intelligenz

BSP – Business Server Pages

ABAP – Advanced Business Application Programming

BAPI – Business Application Programming Interface

MIME – Multipurpose Internet Mail Extensions

SPA – Single Page Application

API – Application Programming Interface

JSON – JavaScript Object Notation

XSS – Cross-Site-Scripting

SEO – Search Engine Optimization

AJAX – Asynchronous JavaScript and XML

Literaturverzeichnis

[1] „Siemens Konzerngeschichte 1847-1865“, *Siemens Deutschland*. [Online]. Available: https://new.siemens.com/de/de/unternehmen/konzern/geschichte/unternehmen/1847-1865.html (zugegriffen 6. September 2023).

[2] „Unser Angebot“, Siemens Deutschland. [Online]. Available: https://new.siemens.com/de/de/unternehmen/konzern/unternehmensstruktur.html (zugegriffen 6. September 2023).

[3] „Siemens | Fertigungs- und Entwicklungsstandort Amberg“, Siemens Deutschland. [Online]. Available: https://new.siemens.com/de/de/unternehmen/standorte/fertigungs-und-entwicklungsstandort-amberg.html (zugegriffen 6. September 2023).

[4] „Was ist SAP? | Definition & Bedeutung | SAP Abkürzung“, SAP. [Online]. Available: https://www.sap.com/germany/about/what-is-sap.html (zugegriffen 6. September 2023).

[5] James Montemagno, et al. „Designing the infrastructure persistence layer - .NET,” Microsoft Learn, Feb. 21, 2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design (zugegriffen 29. Dezember 2023).

[6] Steve Smith, et al. „Merkmale moderner Webanwendungen - .NET“, Microsoft Learn, 8. Juni 2023. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics (zugegriffen 6. September 2023).

[7] Michael Stonis, et al. „Dependency injection,” Microsoft Learn, Nov. 28, 2022. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/architecture/maui/dependency-injection (zugegriffen 30. Dezember 2023).

[8] M. Rohr, Sicherheit von Webanwendungen in der Praxis: Wie sich Unternehmen schützen können – Hintergründe, Maßnahmen, Prüfverfahren und Prozesse. Springer-Verlag, 2018.

[9] M. Kappes, Netzwerk- und Datensicherheit: Eine praktische Einführung. Springer Vieweg, 2019.

[10] „Clean Coder Blog,” Aug. 13, 2012. [Online]. Available: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (zugegriffen 29. Dezember 2023).

[11] A. Pantaleev and A. Rountev, „Identifying Data Transfer Objects in EJB Applications,” Fifth International Workshop on Dynamic Analysis (WODA’07), May 2007, DOI: 10.1109/woda.2007.6.

[12] M. Fowler, „bliki: PresentationDomainDataLayering,” martinfowler.com. [Online]. Available: https://martinfowler.com/bliki/PresentationDomainDataLayering.html (zugegriffen 29. Dezember 2023).

[13] P. Wenzel, Betriebswirtschaftliche Anwendungen mit SAP R/3®: Eine Einführung inklusive Customizing, ABAP/4, Accelerated SAP (ASAP), Projektsystem (PS). Springer-Verlag, 2013.

[14] „Business Server pages (SAP-Bibliothek - Business server pages)“. [Online]. Available: https://help.sap.com/saphelp_snc700_ehp01/helpdata/de/e9/bb153aab4a0c0ee10000000a114084/content.htm (zugegriffen 1. September 2023).
[15] „Was ist eine BSP-Applikation? (SAP-Bibliothek - Business Server Pages)“. [Online]. Available: https://help.sap.com/saphelp_snc700_ehp01/helpdata/de/5a/f8b53a364e0e5fe10000000a11405a/content.htm (zugegriffen 1. September 2023).
[16] „Aufbau einer BSP-Applikation (SAP-Bibliothek - Business Server Pages)“. [Online]. Available: https://help.sap.com/saphelp_snc700_ehp01/helpdata/de/16/ac1e3a0088e042e10000000a11402f/content.htm (zugegriffen 1. September 2023).
[17] „Model-View-Controller (MVC) (SAP-Bibliothek - Business Server pages).” [Online]. Available: https://help.sap.com/saphelp_snc700_ehp01/helpdata/de/0f/ab3a3c9ca75402e10000000a114084/content.htm (zugegriffen 1. September 2023).
[18] „Applikationsklasse einer BSP-Applikation (SAP-Bibliothek - Business Server Pages).” [Online]. Available: https://help.sap.com/saphelp_snc700_ehp01/helpdata/de/21/8cec3ada87e076e10000000a11405a/content.htm (zugegriffen 1. September 2023).
[19] M. Amin, A. Sutrisman and Y. Dwitayanti, „Single page application for business intelligence dashboard“, Atlantis highlights in engineering, Jan. 2022, DOI: 10.2991/ahe.k.220205.055.
[20] Madhuri A. Jadhav, Balkrishna R. Sawant and Anushree Deshmukh, „Single Page Application using AngularJS”, International Journal of Computer Science and Information Technologies, Vol. 6 (3), 2015, 2876-2879
[21] Y. Sun, Practical Application Development with AppRun: Building Reliable, High-Performance Web Apps Using Elm-Inspired Architecture, Event Pub-Sub, and Components. Apress, 2019.
[22] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.
[23] J. Kotz and C. Wenz, C# und .NET 6 - Grundlagen, Profiwissen und Rezepte. Carl Hanser Verlag GmbH Co KG, 2022.
[24] James Montemagno, et al. „Entwerfen der Persistenzebene der Infrastruktur - .NET,” Microsoft Learn, Mar. 29, 2023. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design (zugegriffen 29. Dezember 2023).
[25] „The repository pattern,” Microsoft Learn, Apr. 27, 2010. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649690(v=pandp.10) (zugegriffen 29. Dezember 2023).
[26] J.-P. Voutilainen, J. Salonen, and T. Mikkonen, „On the Design of a Responsive User Interface for a Multi-device Web Service,” 2nd ACM International Conference on Mobile Software Engineering and Systems, May 2015, DOI: 10.1109/mobilesoft.2015.16.

[27] „Using media queries - CSS: Cascading Style Sheets | MDN," MDN Web Docs, Dec. 12, 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries (zugegriffen 02. Januar 2024).
[28] „Welches der folgenden Endgeräte benutzen Sie zum Surfen im Internet?" [Graph], Postbank, 7. Mai, 2020. [Online]. Available: https://de.statista.com/statistik/daten/studie/520893/umfrage/endgeraete-zur-internet-nutzung-in-deutschland/ (zugegriffen 02. Januar 2024).
[29] „Internetnutzer in Deutschland nach Endgeräten der Internetnutzung im Jahr 2023" [Graph], IfD Allensbach, 19. Juni, 2023. [Online]. Available: https://de.statista.com/statistik/daten/studie/940649/umfrage/umfrage-unter-internetnutzern-zu-endgeraeten-der-internetnutzung/ (zugegriffen 02. Januar 2024).
[30] „Anteil der mobilen Internetnutzer in Deutschland in den Jahren 2015 bis 2022" [Graph], Initiative D21, 17. Februar, 2023. [Online]. Available: https://de.statista.com/statistik/daten/studie/633698/umfrage/anteil-der-mobilen-internetnutzer-in-deutschland/ (zugegriffen 02. Januar 2024).
[31] „Anteil der Befragten in ausgewählten Ländern, die mit folgenden Gerätetypen das Internet nutzen" [Graph], Horizont, 26. Mai, 2011. [Online]. Available: https://de.statista.com/statistik/daten/studie/187955/umfrage/internetnutzung-nach-geraetetyp-weltweit/ (zugegriffen 02. Januar 2024).
[32] H. E. Moussaoui and K. Zeppenfeld, AJAX: Geschichte, Technologie, Zukunft. Springer-Verlag, 2007.
[33] T. Walter, Kompendium der Web-Programmierung: Dynamische Web-Sites. Springer, 2007.
[34] Bill Mathers, et al. „Rolle von Ansprüchen," Microsoft Learn, Aug. 28, 2023. [Online]. Available: https://learn.microsoft.com/de-de/windows-server/identity/ad-fs/technical-reference/the-role-of-claims#what-are-claims (zugegriffen 02. Januar 2024).
[35] J. James Garrett, Ajax: A New Approach to Web Applications. adaptive path, 2005.
[36] „ABAP-Programmiersprache - Übersicht - ABAP-Schlüsselwortdokumentation". [Online]. Available: https://help.sap.com/doc/abapdocu_751_index_htm/7.51/de-DE/abenabap_overview.htm (zugegriffen 1. September 2023).
[37] „Klasse CL_BSP_CONTROLLER2 (SAP-Bibliothek - Business Server Pages)". [Online]. Available: https://help.sap.com/saphelp_snc700_ehp01/helpdata/de/95/1e97386a89484f8eeaab29da1639e6/content.htm (zugegriffen 1. September 2023).
[38] P. Weber, „BSP applikation MVC Design Pattern". [Online]. Available: https://weberpatrick.de/bsp-applikation-mvc-design-pattern/ (zugegriffen 1. September 2023).
[39] F. Keijzer, SAP S/4HANA Embedded Analytics: Experiences in the Field. Apress, 2021.
[40] S. Guerrero, Custom Fiori Applications in SAP HANA: Design, Develop, and Deploy Fiori Applications for the Enterprise. Apress, 2020.
[41] „Benutzererfahrung und Apps | User Experience | SAP Fiori". [Online]. Available: https://www.sap.com/germany/products/technology-platform/fiori.html (zugegriffen 14. September 2023).

[42] S. B. Uzayr, N. Cloud and T. Ambler, JavaScript frameworks for modern web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now. Apress, 2019.
[43] Johanan Liebermann, et al. „Was ist Azure DevOps? - Azure DevOps“, Microsoft Learn, 20. Juli 2023. [Online]. Available: https://learn.microsoft.com/de-de/azure/devops/user-guide/what-is-azure-devops?view=azure-devops (zugegriffen 9. Oktober 2023).
[44] „What is .NET? an open-source developer platform.“, Microsoft. [Online]. Available: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet (zugegriffen 10. Oktober 2023).
[45] Mark LeBlanc, et al. „Übersicht über Entity Framework - ADO.NET“, Microsoft Learn, 8. Juni 2023. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/framework/data/adonet/ef/overview (zugegriffen 10. Oktober 2023).
[46] E. Tiemeyer, Handbuch IT-Projektmanagement: Vorgehensmodelle, Managementinstrumente, Good Practices. Carl Hanser Verlag GmbH Co KG, 2018.
[47] Quinn Radich und Jim Walker, „Reaktionsfähige Designtechniken - Windows apps,” Microsoft Learn, Apr. 06, 2023. [Online]. Available: https://learn.microsoft.com/de-de/windows/apps/design/layout/responsive-design (zugegriffen 02. Januar 2024).
[48] „IBM documentation.” [Online]. Available: https://www.ibm.com/docs/de/sc-and-ds/8.5.0?topic=stack-simple-object-access-protocol (zugegriffen 04. Januar 2024).
[49] Don Box, et al., „Simple Object Access Protocol (SOAP) 1.1.” [Online]. Available: https://www.w3.org/TR/soap11/ (zugegriffen 04. Januar 2024).
[50] Bob Ward, „Einführung - Training,” Microsoft Learn. [Online]. Available: https://learn.microsoft.com/de-de/training/modules/introduction-to-sql-server-2022/1-introduction (zugegriffen 05. Januar 2024).
[51] Mindsquare, „SAP fiori“, 31. Januar 2022. [Online]. Available: https://mindsquare.de/knowhow/sap-fiori/
[52] Webvisio, “Responsive Webdesign“, Feb. 03, 2023. [Online]. Available: https://webvisio.de/webdesign/responsive-webdesign/