



WHZ Westsächsische
Hochschule Zwickau
Hochschule für Mobilität



Bachelorarbeit

**Erstellung einer Java-Spring-Webanwendung zur
Verarbeitung von Projekttickets und Integration in eine
bestehende Anwendung.**

an der Fakultät Physikalische Technik/Informatik

Betreuerin an der Hochschule:	Prof. Dr. Tina Geweniger Westsächsische Hochschule Zwickau
Betreuer an der Hochschule:	Prof. Dr.-Ing. Rainer Wasinger Westsächsische Hochschule Zwickau
Betreuer beim Unternehmen:	Dipl.-Geoinf. Andreas Jubelt IAMT-Ingenieurgesellschaft für allgemeine Maschinentechnik mbH
vorgelegt von:	Beksultan Andashev Fachgruppe Informatik MatrikelNr. 47619
Abgabedatum:	4. Februar 2026

Inhaltsverzeichnis

Abkürzungsverzeichnis	iv
Abbildungsverzeichnis	v
Auflistungsverzeichnis	vi
Tabellenverzeichnis	vii
1 Einleitung	1
1.1 Unternehmensvorstellung	1
1.2 Problemstellung und Hintergrund	1
1.3 Zielsetzung der Arbeit	2
2 Theoretische Grundlagen	4
2.1 Ticketsysteme als Bestandteil betrieblicher Informationssysteme	4
2.2 Architekturen webbasierter Unternehmensanwendungen	5
2.3 Persistenz und relationale Datenhaltung	6
2.4 Webtechnologien zur Gestaltung der Präsentationsschicht	7
2.5 Benutzerfreundlichkeit, Human-Computer Interaction und UI-Patterns	8
2.5.1 Kriterien zur Bewertung von Interface-Patterns	8
2.5.2 One-Page Form	9
2.5.3 Wizard / Step-by-Step Form	11
2.5.4 Accordion Form	13
2.5.5 Card-Based Layout	14
3 Ist-Analyse der bestehenden Webanwendung	16
3.1 Gesamtüberblick über die bestehende Anwendung	16
3.2 Serverseitige Architektur und Anwendungslogik	16
3.3 Datenzugriff und Persistenz	16
3.4 Sicherheits- und Rollenmodell	17
3.5 Präsentationsschicht und Benutzeroberfläche	17
3.6 Organisatorischer Rahmen	17
4 Anforderungen an das Projektticket-Modul	19
4.1 Herkunft und Rahmenbedingungen der Anforderungen	19
4.2 Liste der funktionalen und nicht-funktionalen Anforderungen	20
4.3 Funktionale Anforderungen an Projekttickets	20
4.3.1 Grundlegende Ticketfunktionalitäten	20
4.3.2 Ticketinformationen und Datenfelder	21
4.3.3 Rollen und Berechtigungen	21
4.3.4 Dynamische und rollenabhängige Felder	22
4.4 Nicht-funktionale Anforderungen	22
4.4.1 Benutzerfreundlichkeit und mobile Nutzung	22
4.4.2 Benachrichtigungen	23

4.4.3	Technische und organisatorische Randbedingungen	23
4.5	Abgrenzung der Anforderungen	23
4.6	Zusammenfassung der Anforderungen	23
5	Konzeption des Projektticket-Moduls	24
5.1	Einordnung des Projektticket-Moduls in die bestehende Systemarchitektur	24
5.2	Konzeption zur Benutzerinteraktion und Formularstruktur	25
5.2.1	Ableitung konzeptioneller Prinzipien für die Benutzerinteraktion . .	25
5.2.2	Vorgehen zur Auswahl eines Interface-Patterns	26
5.2.3	Vergleich relevanter UI-Patterns	27
5.2.4	Begründung des ausgewählten Interface-Patterns	30
5.3	Konzeption der Rollen- und Berechtigungslogik	31
5.4	Konzeption des Ticket-Lebenszyklus und der Benachrichtigungen	33
5.4.1	Zustände eines Projekttickets	34
5.4.2	Konzeption der Benachrichtigungen	34
5.5	Konzeption der Datenstruktur	35
5.6	Zusammenfassung der Konzeption	36
6	Implementierung der Lösung	37
6.1	Technologische Grundlagen und Rahmenbedingungen	37
6.2	Entwurf der Datenstruktur des Projekttickets	38
6.3	Implementierung der Geschäftslogik	39
6.3.1	Ermittlung des Bearbeitungskontexts und der effektiven Benutzerrolle	40
6.3.2	Validierung von Bearbeitungsschritten und Zustandsübergängen . .	40
6.3.3	Verarbeitung fachlicher Aktionen im Ticketprozess	41
6.3.4	Abgrenzung zur Präsentations- und Persistenzschicht	42
6.4	Implementierung der Benutzerinteraktion	42
6.4.1	Umsetzung des One-Page-Formulars	42
6.4.2	Dynamische Formularbereiche (Meilensteine)	43
6.4.3	Rollen- und statusabhängige Feldsteuerung	45
6.4.4	Mobile Nutzbarkeit und responsives Verhalten	47
6.5	Implementierung des Ticket-Lebenszyklus und der Zustandslogik	48
6.5.1	Technische Abbildung der Zustände	48
6.5.2	Umsetzung der Zustandsübergänge	49
6.5.3	Verhinderung unzulässiger Bearbeitungsschritte	51
6.6	Implementierung der Benachrichtigungen	52
6.6.1	Entscheidung für eine datenbankseitige Umsetzung	52
6.6.2	Auslöser der Benachrichtigungen	52
6.6.3	Ziel der Benachrichtigungen	53
6.6.4	Vorteile der Umsetzung	54
6.7	Sicherheit und Zugriffskontrolle	54
6.7.1	Serverseitige Zugriffssicherung	55
6.7.2	Abgrenzung: Sicherheit vs. UI-Steuerung	55
6.8	Zusammenfassung der Implementierung	56
7	Evaluierung und Ergebnisse	57
7.1	Evaluierung	57

7.1.1	Testaufbau und Vorgehensweise	57
7.1.2	Testergebnisse anhand der Anforderungen	58
7.1.3	Erweiterungsvorschläge und Auffälligkeiten aus der Testphase . . .	61
7.2	Fazit und Ausblick	62
Quellenverzeichnis		I

Abkürzungsverzeichnis

CRUD Create, Read, Update, Delete

CSS Cascading Style Sheets

DAO Data Access Object

ERP Enterprise Resource Planning

HCI Human–Computer Interaction

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JDBC Java Database Connectivity

JSP JavaServer Pages

MSSQL Microsoft SQL Server

MVC Model View Controller

SQL Structured Query Language

UI User Interface

VPN Virtual Private Network

Abbildungsverzeichnis

2.1	One-Page Form Darstellung (Quelle zur Abb. [Lea])	10
2.2	Wizard / Step-by-Step Form Darstellung	12
2.3	Accordion Form Darstellung	13
2.4	Accordion-basierter Checkout-Prozess	14
2.5	Card-based Layout Darstellung	15
6.1	Projektticket, Projektticket_Meilenstein	39
6.2	Add Ticket Formular (mobile Ansicht)	48
6.3	Ablehnungsprozess in der mobilen Ansicht	51
6.4	E-Mail-Benachrichtigungen im Ticketprozess (Neuanlage, Annahme, Statusänderung, Ablehnung)	54
7.1	Validierung der Meilenstein-Prozente (100%-Prüfung)	60
7.2	Darstellung einer Ticketablehnung (Status und Ablehnender sichtbar) . . .	60
7.3	Auffälligkeiten und Hinweise	61

Auflistungsverzeichnis

6.1	Controller – Initialisierung der Ticketerstellung (GET /addTicketPage) . .	43
6.2	Controller – Speichern eines Tickets inkl. Meilensteine (POST /addTicket- Page)	44
6.3	Controller – Laden eines Tickets und Rollenflags (GET /ticketInfo)	45
6.4	JSP/JavaScript – Kernlogik zur Feldsperrung und Projektleiter-Modus (ticke- tInfo.jsp)	46
6.5	Controller – Verarbeitung der Projektleiterentscheidung (Ablehnung) . . .	49
6.6	DAO – Ablehnung des Tickets (declineTicket())	50
6.7	ticketInfo.jsp – Clientseitige Sperrlogik bei abgelehnten Projekttickets . . .	51
6.8	Controller - Ermittlung des eingeloggten Benutzers über Spring Security .	55

Tabellenverzeichnis

5.1	Vergleich der UI-Patterns anhand definierter Bewertungskriterien	29
7.1	Übersicht der durchgeführten Tests und Ergebnisse	59

1 Einleitung

1.1 Unternehmensvorstellung

Die nachfolgenden Angaben zur IAMT-Gruppe basieren auf Informationen, die der offiziellen Website entnommen wurden¹.

Die IAMT-Gruppe ist ein international tätiger Anbieter von Ingenieurdienstleistungen mit Schwerpunkt auf der Entwicklung und Erprobung komplexer technischer Systeme. Das Leistungsspektrum des Unternehmens umfasst komplexe Systemlösungen in den Kompetenzbereichen MOVE (Produktkompetenz in den Bereichen Fahrwerke, Fördertechnik, Baumaschinen und Systementwicklung), SAFE (Produktsicherheit in den Bereichen Prüfung, Fahrversuche und Prüfstandbau), FLOW (Entwicklungs-, Test- und Implementierungskompetenz für technologische Prozesse) und NEXT (innovative Produkte im Bereich Simulation und Systemlösungen).

Seit ihrer Gründung im Jahr 1992 verfolgt sie das Ziel, eine zuverlässige und systemübergreifende Produktentwicklung für die Automobil- und Nutzfahrzeugindustrie sowie deren Zulieferer zu ermöglichen.

Die Unternehmenskultur zeichnet sich durch Merkmale wie flache Hierarchien, kurze Entscheidungswege und flexible Strukturen aus, die eine Vielzahl von Kompetenzen umfassen. Diese organisatorischen Rahmenbedingungen unterstützen eine effiziente Zusammenarbeit und ermöglichen schnelle Reaktionszeiten innerhalb projektbezogener Prozesse.

1.2 Problemstellung und Hintergrund

Die IAMT-Gruppe unterhält Niederlassungen und beschäftigt Mitarbeiter, die an externen Standorten tätig sind. Die dort eingesetzten Arbeitsplatzrechner befinden sich in der IT-Infrastruktur des jeweiligen Partnerunternehmens und ermöglichen keinen direkten Zugriff auf das interne Unternehmensnetzwerk der IAMT-Gruppe. Dadurch ist es Mitarbeitenden, die sich außerhalb der eigenen Infrastruktur befinden, nicht möglich, interne Ressourcen uneingeschränkt zu nutzen.

Diese Rahmenbedingungen schränken die Gestaltung und Umsetzung interner Workflows erheblich ein. Insbesondere fehlt die Möglichkeit einer flexiblen und ortsunabhängigen Bearbeitung von Anfragen und Projekttickets. Damit entsteht ein deutlicher Gegensatz zwischen den Anforderungen an eine moderne, standortunabhängige Arbeitsweise und den technischen Gegebenheiten der eingesetzten Infrastruktur bei externen Partnern.

¹ <https://www.iamt-gruppe.de>

Um diese Einschränkung zu umgehen, wurde der Beschluss gefasst, mobile Geräte zu verwenden. Diese können im Gegensatz zu stationären Computern in externen Büros über einen VPN-Tunnel mit dem Unternehmensnetzwerk der IAMT-Gruppe verbunden werden. Es wird den Mitarbeitern ein sicherer und direkter Zugriff auf das interne Netzwerk ermöglicht, selbst wenn sie sich physisch am Standort eines externen Partners befinden.

Aus diesem Grund ist die Entwicklung oder Erweiterung einer Weblösung erforderlich, die eine effiziente und sichere Bearbeitung interner Prozesse und Projektanträge über mobile Geräte ermöglicht. Die Integration dieser Lösung in die bestehende Systemumgebung sowie die Erfüllung der organisatorischen und technischen Anforderungen der IAMT-Gruppe sind dabei von entscheidender Bedeutung.

1.3 Zielsetzung der Arbeit

Für verschiedene interne ERP-Prozesse nutzt die IAMT-Gruppe bereits eine Webanwendung, die auf Java Spring, HTML, JSP, JavaScript und MSSQL basiert. Die vorhandene Lösung deckt grundlegende CRUD-Funktionalitäten ab und wird im Unternehmen seit 2 Jahren eingesetzt. Da die technische Basis bereits vorhanden ist und zuverlässig funktioniert, wurde bewusst darauf verzichtet, eine komplett neue Anwendung zu entwickeln. Stattdessen soll das bestehende System um Funktionen zur Bearbeitung von Projekttickets erweitert werden.

Im Mittelpunkt dieser Arbeit steht die Entwicklung und Integration eines Moduls, das die Projektticketverarbeitung innerhalb der bestehenden Webanwendung ermöglicht. Das Ziel besteht darin, eine Lösung zu konzipieren, die sich nahtlos in die vorhandene Systemarchitektur einfügt und den organisatorischen sowie technischen Rahmenbedingungen des Unternehmens Rechnung trägt.

Aus dieser Zielsetzung ergibt sich folgende praxisorientierte Forschungsfrage:

- Wie kann ein Projektticket-Modul unter Berücksichtigung bestehender Systemarchitektur, rollenbasierter Zugriffskonzepte und mobiler Nutzung sinnvoll umgesetzt und in eine bestehende Java-Spring-Webanwendung integriert werden?

Zur Beantwortung der Forschungsfrage werden zunächst der Ist-Zustand der bestehenden Anwendung sowie die relevanten technischen und organisatorischen Rahmenbedingungen analysiert. Auf dieser Grundlage wird ein konzeptioneller Lösungsansatz erarbeitet, der die Grundlage für die spätere Implementierung bildet.

Daraus lassen sich mehrere Teilziele ableiten, die sich unmittelbar aus den Anforderungen ergeben:

- Integration eines Ticketverarbeitungsmoduls in das vorhandene System,

- Implementierung der notwendigen CRUD-Funktionalitäten (Erstellen, Lesen, Bearbeiten, Löschen) für Projekttickets,
- Ausarbeitung einer rollenbasierten Berechtigungslogik, die die unterschiedlichen Zuständigkeiten und Rechte von Ersteller, Projektleiter und Teamleiter abbildet,
- Entwicklung einer adaptiven und benutzerfreundlichen Oberfläche, die für die Arbeit auf mobilen Geräten geeignet ist,
- Die Möglichkeit für die künftige Erweiterung der Funktionalität.

Das Ergebnis dieser Arbeit ist ein erweitertes Websystem, das es IAMT-Mitarbeitern ermöglicht, Projekttickets effizient und ortsunabhängig über mobile Geräte zu bearbeiten und damit die Einschränkungen externer Arbeitsbereiche zu entschärfen.

2 Theoretische Grundlagen

In diesem Abschnitt werden die theoretischen Grundlagen sowie allgemein verbreitete technologische Konzepte vorgestellt, die für das Verständnis der in dieser Arbeit behandelten Themen von Relevanz sind. Sie bilden die theoretische Grundlage dieser Bachelorarbeit.

2.1 Ticketsysteme als Bestandteil betrieblicher Informationssysteme

Ticket-Systeme werden als eine Klasse betrieblicher Informationssysteme bezeichnet, die zur strukturierten Erfassung, Bearbeitung und Verfolgung von Prozessen eingesetzt werden. Ein Ticket repräsentiert eine formalisierte Darstellung einer Anfrage, einer Aufgabe oder eines Prozesses und dient als zentraler Informationsbaustein innerhalb eines spezifischen Arbeitsprozesses. Der Einsatz von Ticketsystemen findet insbesondere in Organisationen statt, in denen Aufgaben auf der Grundlage der Arbeitsteilung ausgeführt werden und eine nachvollziehbare Dokumentation erforderlich ist [Axe19, S. 28–34].

Ein wesentliches Ziel von Ticketsystemen besteht darin, Prozesse transparent und kontrollierbar zu gestalten. Die systematische Erfassung von Vorgängen stellt sicher, dass die entsprechenden Informationen vollständig dokumentiert und jederzeit verfügbar sind. Gleichzeitig unterstützen Ticketsysteme die strukturierte Bearbeitung von Aufgaben, indem sie den Bearbeitungsstatus, die Zuständigkeiten und die zeitlichen Abläufe klar abbilden. Auf diese Weise tragen sie zur Standardisierung und Stabilisierung von Betriebsabläufen bei [Axe19, S. 30–34].

Die Anwendung von Ticketsystemen findet in verschiedenen betrieblichen Kontexten statt. Häufig werden sie im IT-Service-Management zur Bearbeitung von Störungen, Serviceanfragen oder Änderungsanfragen verwendet. Darüber hinaus werden Ticketsysteme auch im Projektmanagement sowie in administrativen Bereichen eingesetzt, etwa zur Abbildung von Genehmigungs-, Freigabe- oder Unterstützungsprozessen. In den beschriebenen Szenarien dienen Tickets dazu, Aufgaben über mehrere Bearbeitungsschritte hinweg konsistent zu verwalten und zu dokumentieren [LL20, S. 73–75].

Unabhängig vom spezifischen Anwendungsbereich weisen Ticketsysteme gemeinsame strukturelle Merkmale auf. Hierzu zählen insbesondere Statusmodelle, die den Bearbeitungszustand eines Tickets beschreiben. Ein Ticket durchläuft dabei typischerweise mehrere Zustände, beispielsweise von der Erfassung über die Bearbeitung bis hin zum Abschluss. Ergänzend dazu verfügen Ticketsysteme über rollenbasierte Zugriffsmöglichkeiten, die

festlegen, welche Benutzer bestimmte Aktionen ausführen dürfen oder welche Informationen für sie sichtbar sind [Axe19, S. 28–29].

Ein weiteres charakteristisches Merkmal von Ticketsystemen ist die Nachverfolgbarkeit von Prozessschritten. Änderungen am Inhalt eines Tickets, Statusänderungen oder Bearbeitungsschritte werden häufig protokolliert und mit Zeitangaben sowie Benutzerkennungen versehen. Diese Form der Protokollierung ermöglicht eine transparente Darstellung des Prozessverlaufs und erleichtert die spätere Analyse oder Überprüfung von Abläufen, wie sie in vielen betrieblichen Kontexten erforderlich ist [Wes19, S. 11–17].

In betrieblichen Informationsumgebungen werden Ticketsysteme in der Regel in bestehende Anwendungssysteme eingebettet. Typische Verbindungen bestehen zu Benutzer- und Rollenverwaltungssystemen, Projekt- oder Kundendatenbanken sowie anderen operativen Informationssystemen. Durch diese Integration können konsistente Datenbestände sichergestellt und redundante Erfassungen vermieden werden. Gleichzeitig ermöglicht sie eine einheitliche Nutzung organisatorischer Strukturen über verschiedene Systeme hinweg [LL20, S. 243–248].

Ticketsysteme sind somit ein grundlegendes Werkzeug zur strukturierten Organisation und Dokumentation von Arbeitsprozessen in Unternehmen. Die Kombination aus formalisierter Datenerfassung, Statusabbildung, Rollensteuerung und Protokollierung unterstützt die effiziente und nachvollziehbare Durchführung betrieblicher Abläufe [Axe19, S. 28–34; Wes19, S. 11–15].

2.2 Architekturen webbasierter Unternehmensanwendungen

Webbasierte Unternehmensanwendungen sind Informationssysteme, die über das Netzwerk bereitgestellt werden und üblicherweise nach dem Client-Server-Prinzip aufgebaut sind. Dabei übernimmt der Client – meist ein Webbrowser – die Anzeige und die Interaktion mit dem Benutzer, während der Server für die Verarbeitung von Anfragen, die Geschäftslogik sowie den Zugriff auf persistente Daten verantwortlich ist. Diese Aufteilung ermöglicht eine zentrale Verwaltung der Anwendung und vereinfacht die Wartung, Erweiterung und den Betrieb in unternehmensweiten Umgebungen [LL20, S. 202–204].

Ein zentrales Architekturprinzip von Webanwendungen ist die Aufgabenteilung innerhalb des Systems. Das Ziel besteht darin, unterschiedliche Aufgabenbereiche wie Darstellung, Geschäftslogik und Datenhaltung klar voneinander abzugrenzen. Dank dieser Struktur können einzelne Komponenten unabhängig voneinander entwickelt, getestet und angepasst werden. In der Softwarearchitektur wird dieses Prinzip häufig mithilfe etablierter Architekturmuster umgesetzt, die eine klare Rollenverteilung innerhalb der Anwendung vorsehen.

Ein weit verbreitetes Architekturmuster für Webanwendungen ist das Model-View-Controller-Modell (MVC). Die Anwendung ist in drei logisch getrennte Komponenten unterteilt. Das Modell repräsentiert die Fachdaten und Zustände der Anwendung. Die Ansicht ist für die Darstellung dieser Daten und die Interaktion mit dem Benutzer zuständig. Der Controller nimmt dabei die Funktion einer Vermittlungsinstanz ein, verarbeitet die Benutzereingaben, führt die Anwendungslogik aus und übergibt die entsprechenden Daten an die Ansicht. Aufgrund dieser Aufteilung wird eine schwache Kopplung zwischen den einzelnen Komponenten erreicht, was die Wartbarkeit und Erweiterbarkeit der Anwendung verbessert [Wal18, S. 31–38].

Neben der strukturellen Aufteilung spielt auch die Modularität eine zentrale Rolle in der Architektur von Unternehmensanwendungen. Module fassen miteinander verbundene Funktionen zu eigenständigen Einheiten zusammen. Dadurch lässt sich die Anwendung schrittweise erweitern, ohne die bestehenden Funktionen wesentlich zu verändern. Diese Eigenschaft erweist sich insbesondere in Unternehmensumgebungen als relevant, da sich die technischen Anforderungen oft über einen längeren Zeitraum hinweg weiterentwickeln [LL20, S. 371–373].

Zusammenfassend bilden klare Architekturprinzipien, die Trennung von Zuständigkeiten und modulare Strukturen die Grundlage moderner webbasierter Unternehmensanwendungen. Architekturmuster wie MVC unterstützen die klare Strukturierung komplexer Anwendungssysteme und gewährleisten deren langfristige Wartbarkeit und Erweiterbarkeit [Wal18; LL20].

2.3 Persistenz und relationale Datenhaltung

Die nachfolgenden Erläuterungen zu den Themen Persistenz und relationale Datenhaltung basieren auf den grundlegenden Prinzipien von Datenbank-Systemen nach Connolly und Begg [CB15].

Persistenz bezeichnet die dauerhafte Speicherung von Daten während der gesamten Laufzeit eines Programms. In betrieblichen Informationssystemen ist dies eine grundlegende Voraussetzung, da Geschäftsobjekte langfristig verfügbar, konsistent und nachvollziehbar gespeichert werden müssen. Persistente Daten bilden die Grundlage für die Abbildung, Steuerung und Auswertung betrieblicher Prozesse [CB15, S. 671].

In den meisten Unternehmensanwendungen erfolgt die Datenspeicherung mithilfe relationaler Datenbanksysteme. Diese organisieren Informationen in Tabellen, die durch Beziehungen miteinander verknüpft sind. Jede Tabelle repräsentiert dabei eine bestimmte Entität, und die Beziehungen zwischen den Entitäten werden mithilfe von Schlüsselmechanismen dargestellt. Dieses Modell hat sich im Kontext von Unternehmensanwendungen aufgrund seiner klaren Struktur und Zuverlässigkeit bewährt [CB15, S. 150–157].

Zentrale Elemente relationaler Datenbanken sind Primär- und Fremdschlüssel. Sie ermöglichen die eindeutige Identifizierung von Datensätzen sowie die Modellierung von Beziehungen zwischen verschiedenen Tabellen. Durch diese Vorgehensweise ist es möglich, verknüpfte Informationen in strukturierter Form ohne Datenredundanz zu speichern. Dies trägt zur Gewährleistung ihrer Konsistenz bei [CB15, S. 158–163].

Zur Sicherstellung einer zuverlässigen Datenverarbeitung bieten relationale Datenbanksysteme Unterstützung für Transaktionen. Transaktionen fassen mehrere Datenbankoperationen zu einer logischen Einheit zusammen und gewährleisten, dass Änderungen entweder vollständig oder gar nicht übernommen werden. Integritätsmechanismen stellen ergänzend dazu sicher, dass gespeicherte Daten bestimmten technischen und strukturellen Regeln entsprechen [CB15][S. 670–671][S. 161–163].

Für den zuverlässigen Betrieb von operativen Informationssystemen bilden Persistenz und die relationale Datenspeicherung die zentrale Grundlage.

2.4 Webtechnologien zur Gestaltung der Präsentationsschicht

Webbasierte Anwendungen nutzen verschiedene Webtechnologien zur Abbildung der Darstellungsschicht. Der Einsatz dieser Technologien dient der strukturierten Darstellung von Inhalten, der visuellen Gestaltung der Benutzeroberfläche sowie der Realisierung von Interaktivität. Das Ziel besteht darin, den Benutzern über einen Webbrowser Zugriff auf die Funktionen des Informationssystems zu ermöglichen.

Serverseitige Technologien zur Generierung dynamischer Webseiten wie JSP erlauben die Erzeugung von HTML-Inhalten auf dem Server, noch bevor diese an den Client übertragen werden. Dabei können Daten aus der Anwendungslogik in die Darstellung eingebunden und während der Ausführung verarbeitet werden. Solche Technologien werden häufig eingesetzt, um Inhalte abhängig vom Kontext des Benutzers oder vom aktuellen Systemzustand dynamisch bereitzustellen [Ora20; W3S].

Die grundlegende Struktur der Webschnittstelle wird mithilfe von Auszeichnungssprachen wie HTML beschrieben. Diese definieren den inhaltlichen Aufbau einer Seite, während Gestaltungs- und Layoutaspekte durch ergänzende Mechanismen getrennt davon festgelegt werden. Durch diese Trennung von Struktur und Gestaltung lassen sich Benutzeroberflächen übersichtlich aufbauen und flexibel an unterschiedliche Darstellungsanforderungen anpassen [W3S].

Zur Erweiterung der Benutzeroberfläche um interaktive Elemente kommen skriptbasierte Sprachen wie JavaScript zum Einsatz. Sie gestatten es, auf Benutzereingaben zu reagieren, Oberflächenelemente dynamisch zu verändern oder Eingaben vor der weiteren Verarbeitung zu überprüfen. Dadurch kann die Benutzerfreundlichkeit von Webanwendungen

verbessert und eine unmittelbare Rückmeldung an den Benutzer bereitgestellt werden [Moz].

Die Kombination aus serverseitiger Inhaltserzeugung, strukturierter Darstellung und clientseitiger Interaktivität bildet die technische Grundlage moderner Weboberflächen. Diese Technologien tragen wesentlich dazu bei, dass webbasierte Anwendungen sowohl funktional als auch benutzerfreundlich gestaltet werden können.

2.5 Benutzerfreundlichkeit, Human-Computer Interaction und UI-Patterns

Die Entwicklung benutzerfreundlicher Informationssysteme ist ein zentrales Forschungsgebiet der Human-Computer Interaction (HCI). Das Ziel von HCI ist es, die Interaktion zwischen Mensch und System so zu gestalten, dass Aufgaben effizient, verständlich und mit minimaler kognitiver Belastung ausgeführt werden können [ND86, S. 1–4; Nor13, S. 21–26]. Neben allgemeinen Gestaltungsprinzipien spielen wiederkehrende Lösungsansätze für typische Interaktionsprobleme eine wichtige Rolle. In der HCI werden diese häufig in Form von sogenannten Interface-Patterns oder UI-Patterns beschrieben [Nie].

Um unterschiedliche Interface-Patterns systematisch beschreiben und einordnen zu können, ist zunächst die Definition geeigneter Bewertungskriterien erforderlich. Dieser Ansatz ermöglicht einen strukturierenden Vergleich der Stärken und Schwächen einzelner Patterns, unabhängig von der konkreten Anwendung.

2.5.1 Kriterien zur Bewertung von Interface-Patterns

Auf der Grundlage von Untersuchungen im Bereich UX und HCI [Nor13, S. 21–27; Nie] wurde ein Kriterienkatalog erstellt, der bei der Gestaltung von Benutzerschnittstellen eine wichtige Rolle spielt.

1. Übersichtlichkeit und Informationsdichte

Dieser Aspekt beschreibt, in welchem Umfang eine Benutzerschnittstelle einen Überblick über zusammenhängende Informationsbereiche vermittelt [Nor13, S. 21–23].

2. Kognitive Belastung

Hierbei wird betrachtet, welche geistige Anstrengung erforderlich ist, um Inhalte zu erfassen und Interaktionen auszuführen [Nor13, S. 21–27].

3. Unterstützung rollenabhängiger UI-Logik

Dieser Punkt bezieht sich auf die Möglichkeit, Benutzerschnittstellen abhängig von Rollen oder Berechtigungen unterschiedlich darzustellen [Nie16].

4. **Mobile Anpassungsfähigkeit**

Die mobile Nutzbarkeit beschreibt, inwieweit sich eine Benutzerschnittstelle an unterschiedliche Endgeräte und Bildschirmgrößen anpassen lässt [Bud16].

5. **Technische Umsetzbarkeit im Rahmen bestehender Systemarchitekturen**

Dieser Aspekt betrachtet, wie sich Interface-Strukturen grundsätzlich in bestehende Architekturen integrieren lassen [PR24].

6. **Erweiterbarkeit und Wartbarkeit**

Hierbei wird berücksichtigt, in welchem Maße sich Benutzeroberflächen strukturell erweitern und langfristig pflegen lassen.

Dieser Kriterienkatalog dient als theoretisches Ordnungssystem für die Beschreibung unterschiedlicher Interface-Strukturen.

2.5.2 **One-Page Form**

Die One-Page Form [Pop25] stellt ein klassisches Muster für komplexe Formulare dar, in denen Nutzer große Mengen zusammenhängender Informationen zu einer einzigen Entität erfassen oder bearbeiten müssen. Tidwell führt aus, dass die Fragmentierung zusammenhängender Informationen die kognitive Belastung erhöht und Entscheidungsprozesse erschwert [Tid20, S. 133–135].

Die unten stehende Abbildung 2.1 illustriert ein klassisches Beispiel für die Struktur eines One-Pager-Formulars, das für die Eingabe großer Datenmengen vorgesehen ist. Das Hauptmerkmal besteht darin, dass alle erforderlichen Eingabefelder auf einer einzigen Seite vollständig angezeigt werden, ohne dass der Benutzer zwischen verschiedenen Schritten oder Unterseiten wechseln muss.

Billing Information - Exactly as it appears on your Credit Card

Company Name:		Optional
Cardholder First Name:		Required
Cardholder Middle Initial:		Optional
Cardholder Last Name:		Required
Suffix (ie: Jr,Sr,II):		Optional
Billing Address:		Required
Apartment or Suite:		Optional
Billing City:		Required
Billing State or Province:		Required
Billing Postal/Zip Code:		Required
Billing Country:	United States	Required
Day Time Phone Number:		Required
Mobile Phone Number:		Optional
Credit Card#:		Required
Expiration Date:	January 2011	Required
Card Verification Value:	What's This?	Optional
E-Mail Address:		Required
Special Instructions/Notes:		Optional

Maximum of 70 Characters:

Shipping Information - If you wish to ship to an address other than the address listed above, uncheck the box below and complete the additional information.

Ship to Billing Address: ☒ Yes, Ship To Address Above

Company:		Optional
Ship To / Recipient:		Optional
Shipping Street Address:		Optional
Shipping Apt or Suite:		Optional
Shipping City:		Optional
Shipping State or Province:		Optional
Shipping Postal/Zip Code:		Optional
Shipping Country:	United States	Optional

Abbildung 2.1: One-Page Form Darstellung (Quelle zur Abb. [Lea])

Vorteile

- **Ganzheitliche Wahrnehmung der Informationen** Alle Felder sind gleichzeitig sichtbar, was die Bildung eines konsistenten mentalen Modells erleichtert.
- **Minimierung von Kontextwechseln** Nach der Nielsen Norman Group verursachen Seitenwechsel einen erhöhten *Interaction Cost* [Nie; Bud13]. Die One-Page Form eliminiert diese Wechsel vollständig.
- Unterstützung komplexer Abhängigkeiten zwischen Eingabefeldern

Nachteile

- mögliche visuelle Überfrachtung bei sehr umfangreichen Formularen,
- erhöhter Aufwand für responsives Design,
- lange Scrollwege auf mobilen Geräten.

Diese Nachteile lassen sich allerdings durch UI-Maßnahmen wie Gruppierungen, Zwischenüberschriften oder fixierte Bereiche abmildern.

2.5.3 Wizard / Step-by-Step Form

Wizard-Formulare werden aufgrund ihrer Fähigkeit geschätzt, den Eingabeprozess in einzelne Schritte zu unterteilen und somit die kognitive Belastung der Nutzer zu reduzieren. Die Nielsen Norman Group betont jedoch, dass Wizard-Formulare nur dann geeignet sind, wenn die einzelnen Schritte weitgehend unabhängig voneinander sind und keine durchgehende Gesamtübersicht erforderlich ist [Nie; Bud17].

In Abbildung 2.2 wird ein von der Nielsen Norman Group dokumentiertes Beispiel eines klassischen Wizard-Flows dargestellt. Der Nutzer wird schrittweise durch den Prozess geleitet und in jedem Schritt werden ihm ausschließlich kontextrelevante Eingabeelemente vorgestellt. In Abhängigkeit von der getroffenen Auswahl im ersten Schritt («Are you already a Fidelity customer?») verzweigt sich der Ablauf in unterschiedliche Folgeansichten. Auf diese Weise wird die typische Struktur eines Wizard-Designs verdeutlicht [Bud17].

The image displays three sequential screenshots of the Fidelity website's 'Open a Roth IRA' wizard, illustrating a step-by-step form design.

Step 1: The page features the Fidelity logo at the top. Below it, the heading 'Open a Roth IRA' is followed by a subtext: 'Opening a new account takes just a few minutes. Before you begin, find out [what information you'll need to provide](#).' A note states 'All fields required unless otherwise noted.' The main question is 'Are you already a Fidelity customer?' with two radio buttons: 'Yes' and 'No'. A small note below the buttons says 'Select Yes if you have a brokerage, IRA, 401(k), or other Fidelity account.' At the bottom, there are 'Exit' and 'Get Started' links.

Step 2: This screen is identical to Step 1, but the 'Yes' radio button is selected, indicating the user is an existing customer.

Step 3: The heading remains 'Open a Roth IRA', but the subtext changes to 'Once you log in, we can prefill your application with information from your existing Fidelity accounts.' The question is 'Please log in'. Below it, there are input fields for 'Username' and 'Password'. A link 'Need help logging in?' is next to the password field. At the bottom, there are 'Exit' and 'Get Started' links.

Step 4: This screen is identical to Step 3, but the 'Get Started' button is highlighted, indicating the user is ready to proceed.

Abbildung 2.2: Wizard / Step-by-Step Form Darstellung

Vorteile

- Reduktion der Belastung des Arbeitsgedächtnisses,
- hohe Klarheit des Eingabeprozesses.

Nachteile

- **Verlust des Gesamtkontextes** Die Aufteilung in Schritte führt zu Informationsverlust und widerspricht dem HCI-Prinzip *Context Preservation* [Nie07; Tid20, S. 87].
- Erhöhter Aufwand für die Verwaltung von Zwischenschritten und Zuständen
- Zusätzliche Wechselwirkungen zwischen einzelnen Prozessschritten

2.5.4 Accordion Form

Die Accordion Form stellt einen Mittelweg zwischen langen Formularseiten und schritt-basierten Strukturen dar. Sie wird häufig für umfangreiche Inhalte genutzt und dient dazu, die sichtbare Informationsmenge zu reduzieren, indem einzelne Bereiche ein- und ausgeklappt werden können [Nie14].

Die Abbildung 2.3 illustriert die grundlegende Struktur eines Accordion-Elements, wie sie in einem Artikel der Nielsen Norman Group beschrieben wird. Dabei ist jeweils nur der aktuell geöffnete Abschnitt vollständig sichtbar, während alle anderen Bereiche eingeklappt bleiben [Nie14].

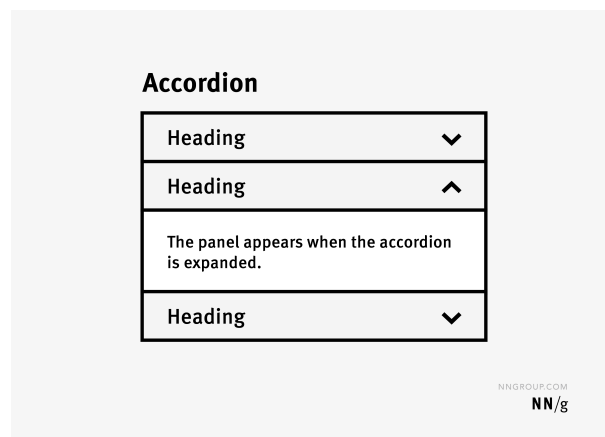


Abbildung 2.3: Accordion Form Darstellung

Vorteile

- **Reduzierung der wahrgenommenen Formularlänge** Durch das Einklappen nicht benötigter Bereiche wirkt das Formular kompakter. Im Artikel der Nielsen Norman Group ist es ebenfalls als zentraler Vorteil hervorgehoben [Nie14].
- **Thematische Gruppierung von Feldern** Die Bündelung von ähnlichen Inhalten nach logischen Gesichtspunkten erleichtert die Orientierung und trägt in umfangreichen Formularen zu einer optimierten Struktur bei.

Nachteile

- **Eingeschränkte Übersichtlichkeit** Nur ein Abschnitt ist sichtbar.

Ein Beispiel eines Checkout-Prozesses [Nie15], der Accordion-Strukturen verwendet, ist in der Abbildung 2.4 vorgesehen. Die Darstellung verdeutlicht, dass Nutzer immer nur einen Abschnitt gleichzeitig sehen und dadurch den Gesamtkontext nicht

vollständig erfassen können, da ihre Aufmerksamkeit auf einen begrenzten Bereich gerichtet ist.

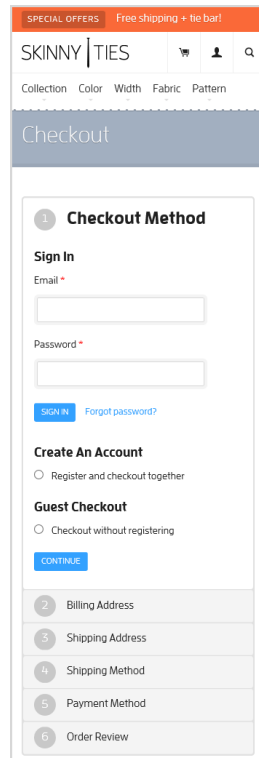
The image shows a mobile application interface for a checkout process. At the top, there is a header with 'SPECIAL OFFERS' and 'Free shipping + tie bar!'. Below this is a navigation bar with the brand name 'SKINNY TIES' and icons for a shopping cart, user profile, and search. A filter bar below the navigation lists 'Collection', 'Color', 'Width', 'Fabric', and 'Pattern'. The main section is titled 'Checkout' and contains a vertical list of steps: 1. Checkout Method, 2. Billing Address, 3. Shipping Address, 4. Shipping Method, 5. Payment Method, and 6. Order Review. Step 1 is currently active and expanded, showing a 'Sign In' section with fields for 'Email' and 'Password', a 'SIGN IN' button, and a 'Forgot password?' link. Below this is a 'Create An Account' section with a radio button for 'Register and checkout together'. Underneath is a 'Guest Checkout' section with a radio button for 'Checkout without registering' and a 'CONTINUE' button. The other steps in the list are collapsed and shown as greyed-out boxes.

Abbildung 2.4: Accordion-basierter Checkout-Prozess

- **Limitierte mobile Nutzbarkeit** potenzielle Einschränkungen auf mobilen Geräten [Nie15].
- Erhöhte Komplexität bei abhängigen Informationsfeldern

2.5.5 Card-Based Layout

Das Card-Based Layout spielt eine zentrale Rolle in modernen Webdesign-Systemen (z. B. Material Design). Die Nielsen Norman Group beschreibt Cards als UI-Komponente zur Darstellung kurzer, zusammenhängender Inhaltseinheiten, häufig als verlinkte «Snapshots» [Lau16]. Für umfangreiche Formulare kann dieses Pattern jedoch zu einer ungünstigen Fragmentierung zusammenhängender Eingaben führen [Lau16; Nie25; Nie23].

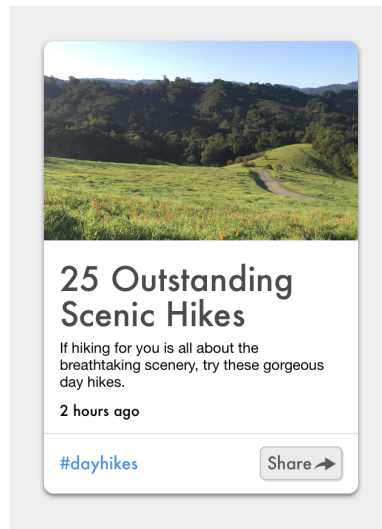


Abbildung 2.5: Card-based Layout Darstellung

Vorteile

- moderne und klare Struktur,
- hohe Modularität der dargestellten Inhalte.

Nachteile

- **Fragmentierung der Daten** Karten trennen logische Zusammenhänge, was das mentale Modell stört.
- **Erhöhte kognitive Belastung** Der Fokus wechselt zwischen zusammenhängenden Informationseinheiten.
- **Ineffizienz bei datenintensiven Aufgaben auf mobilen Geräten** Auf kleinen Displays benötigen Karten vergleichsweise viel Raum, was zu einer Fragmentierung der Eingabe führt.

3 Ist-Analyse der bestehenden Webanwendung

In diesem Abschnitt wird der aktuelle Stand der bestehenden Webanwendung, die im Unternehmen zur Unterstützung verschiedener ERP-Prozesse eingesetzt wird, analysiert. Die vorliegende Ist-Analyse beschreibt den aktuellen technischen, funktionalen und organisatorischen Aufbau des Systems und stellt die verwendeten Technologien sowie die zentralen Komponenten vor.

Der Schwerpunkt liegt dabei ausschließlich auf der Beschreibung des bestehenden Systems. Weiterführende Betrachtungen zu Anforderungen und Lösungsansätzen erfolgen in nachfolgenden Kapiteln.

3.1 Gesamtüberblick über die bestehende Anwendung

Die bestehende Anwendung wurde als webbasierte Unternehmensanwendung implementiert und funktioniert nach dem Client-Server-Prinzip. Der Zugriff erfolgt über einen Webbrowser, während die Verarbeitung der Anfragen auf der Serverseite stattfindet. Die Anwendung ist in die bestehende IT-Infrastruktur des Unternehmens integriert und unterstützt verschiedene operative Geschäftsprozesse.

Die Systemarchitektur basiert auf dem Model-View-Controller-Ansatz (MVC). Dieser unterteilt die Anwendung in klar getrennte Bereiche für die Darstellung, die Geschäftslogik und den Datenzugriff. Die Anwendung zeichnet sich durch einen modularen Aufbau aus, bei dem einzelne Funktionsbereiche in eigenständigen Komponenten organisiert sind.

3.2 Serverseitige Architektur und Anwendungslogik

Die serverseitige Implementierung der Anwendung erfolgt in der Programmiersprache Java. Als zentrales Framework findet Spring MVC Anwendung, welches die Umsetzung des Architekturmusters MVC unterstützt. Der Controller ist für die Verarbeitung eingehender HTTP-Anfragen zuständig und koordiniert den Ablauf der Anwendungslogik.

Die Geschäftslogik ist in Servicekomponenten gekapselt, die unabhängig von der Darstellungsschicht arbeiten. Diese Struktur ermöglicht eine klare Verteilung der Verantwortlichkeiten innerhalb der Anwendung und trägt zur Übersichtlichkeit des Systems bei.

3.3 Datenzugriff und Persistenz

Für die Speicherung von Betriebsdaten wird eine relationale Datenbank verwendet. Der Zugriff auf die Daten erfolgt über eine JDBC-basierte Anbindung. Die Formulierung von

SQL-Abfragen erfolgt unmittelbar und deren Ausführung findet innerhalb spezifischer Komponenten statt.

Das Basisdatenmodell bildet die zentralen Geschäftsobjekte und ihre Beziehungen untereinander ab. Transaktionen dienen der Sicherstellung der Konsistenz von Daten während der Verarbeitung. Die Persistenzschicht ist klar von der Geschäftslogik getrennt.

3.4 Sicherheits- und Rollenmodell

Die vorhandene Anwendung verfügt über ein integriertes Sicherheitskonzept auf Basis von Spring Security. Die Authentifizierung der Benutzer erfolgt zentral, während die Autorisierung durch ein rollenbasiertes Zugriffskonzept gesteuert wird.

Die Rollen legen fest, welche Funktionen in der Anwendung genutzt werden können und welche Daten für die Benutzer sichtbar sind. Die Zugriffskontrolle wird sowohl auf der Ebene der Controller als auch innerhalb der Präsentationsschicht umgesetzt. Die Implementierung dieser Maßnahme gewährleistet, dass die Benutzer lediglich auf die für sie vorgesehenen Funktionen Zugriff erhalten.

Das Rollenmodell wurde systemweit einheitlich implementiert und ist Bestandteil der zentralen Sicherheitsarchitektur.

3.5 Präsentationsschicht und Benutzeroberfläche

Die Benutzeroberfläche der Anwendung wird serverseitig generiert und dem Client im HTML-Format bereitgestellt. Als Anzeigetechnologie wird JavaServer Pages (JSP) verwendet, um dynamische Inhalte auf der Grundlage von Daten aus der serverseitigen Anwendungslogik zu erzeugen. Die Integration der vom Controller bereitgestellten Daten erfolgt im Rahmen von JSP-Vorlagen, die die Ausgabe von HTML-Seiten steuern. Zur Strukturierung der Seiten wird HTML verwendet, für das Layout und Design CSS. Zusätzlich kommt das CSS-Framework Bootstrap zum Einsatz, um ein einheitliches Design zu gewährleisten.

Zur Unterstützung verschiedener Endgeräte werden Medienanfragen (Media CSS) verwendet, sodass sich die Darstellung an unterschiedliche Bildschirmgrößen anpasst. Interaktive Funktionen in der Benutzeroberfläche werden mit JavaScript realisiert, beispielsweise zur Validierung eingegebener Daten oder zur dynamischen Anpassung einzelner Oberflächenelemente.

3.6 Organisatorischer Rahmen

Die Webanwendung ist seit zwei Jahren produktiv im Betrieb und fest in die organisatorischen Prozesse des Unternehmens eingebunden. Die Durchführung des Betriebs, der

Wartung sowie der Weiterentwicklung erfolgt im Rahmen der bestehenden organisatorischen Strukturen und Prozesse.

Die Anwendung ist an zentrale Systeme, wie die Benutzer- und Rollenverwaltung, angebunden und bildet einen integralen Bestandteil der bestehenden Systemlandschaft.

4 Anforderungen an das Projektticket-Modul

In diesem Kapitel werden die Anforderungen an das zu entwickelnde Projektticket-Modul beschrieben. Die Anforderungen definieren die funktionale und nicht-funktionale Ausgestaltung des Systems, um den Bedürfnissen der Nutzer und der Organisation gerecht zu werden.

Die Anforderungen basieren auf:

- der Analyse des bestehenden Systems (Kapitel 3),
- Gesprächen mit Kollegen, die die Anwendung zukünftig nutzen werden,
- organisatorischen Rahmenbedingungen des Unternehmens,
- sowie praktischen Anforderungen aus dem Arbeitsumfeld von Kollegen, die außerhalb des Firmennetzwerks tätig sind.

4.1 Herkunft und Rahmenbedingungen der Anforderungen

Die Anforderungen an das Projektticket-Modul ergeben sich aus einer konkreten betrieblichen Problemstellung. Im Unternehmen gibt es Mitarbeitende, die dauerhaft oder zeitweise bei externen Partnern tätig sind. Aufgrund der dortigen Netzwerkstruktur ist kein Zugriff auf das interne Unternehmensnetzwerk der IAMT-Gruppe über die Arbeitsplatzrechner vor Ort möglich.

Um dennoch einen sicheren Zugriff auf interne Systeme zu gewährleisten, wurden den betroffenen Mitarbeitenden mobile Endgeräte zur Verfügung gestellt, über die ein VPN-Tunnel in das Firmennetzwerk aufgebaut werden kann. Obwohl auf diese Weise ein technischer Zugang zum internen Netzwerk möglich ist, kann das bislang eingesetzte, desktopbasierte ERP-System nicht genutzt werden. Das vorhandene ERP-System ist unabhängig von der in Kapitel 3 beschriebenen Webanwendung und basiert auf der Programmiersprache Java.

Vor diesem Hintergrund wurde der Beschluss gefasst, die bereits vorhandene Webanwendung des Unternehmens zu erweitern. Diese Anwendung ist in Betrieb, technisch stabil und in die bestehende IT-Infrastruktur integriert. Ziel ist es, innerhalb dieses Systems eine neue Funktionalität für die Bearbeitung von Projekttickets bereitzustellen. Die Anforderungen wurden gemeinsam mit Kollegen aus Wolfsburg sowie der IT-Abteilung am Standort Weischlitz erarbeitet.

4.2 Liste der funktionalen und nicht-funktionalen Anforderungen

Zur übersichtlichen Darstellung werden die Anforderungen in funktionale (F) und nicht-funktionale (NF) Anforderungen unterteilt. Die funktionalen Anforderungen beschreiben die technisch erforderlichen Funktionen des Projektticket-Moduls, während die nicht-funktionalen Anforderungen die Qualitätsmerkmale und Randbedingungen festlegen (z. B. mobile Nutzung, Benachrichtigungen und technische Integration).

F1 Projekttickets können angelegt, angezeigt, gesucht und bearbeitet werden (Übersicht, Suche, Detailansicht, Bearbeitung).

F2 Rollenmodell mit Ersteller, Teamleiter, Projektleiter und Sachbearbeiter.

F3 Projektleiter kann Tickets annehmen oder ablehnen; bei Ablehnung ist eine Begründung verpflichtend.

F4 Rollenabhängige Feldsteuerung: Felder sind je nach Rolle verpflichtend/optional sowie schreibgeschützt/ausgeblendet.

F5 Abrechnungsmeilensteine: pro Ticket mehrere Meilensteine; Summe der Prozente muss 100% ergeben.

NF1 Mobile Nutzung (Smartphone) unter VPN-Bedingungen: Oberfläche muss responsiv und bedienbar sein.

NF2 Benachrichtigungen per E-Mail bei definierten Statusänderungen im Ticket-Lebenszyklus.

NF3 Umsetzung als Erweiterung ohne Umbau der bestehenden Systemarchitektur (bestehender Technologie-Stack).

4.3 Funktionale Anforderungen an Projekttickets

4.3.1 Grundlegende Ticketfunktionalitäten

Zuordnung: F1 (Grundfunktionen und Lebenszyklus)

Das Projektticket-Modul soll folgende grundlegende Funktionen bereitstellen:

- eine eigenständige Ansichtssseite zur Anzeige aller vorhandenen Projekttickets,
- eine Suchfunktion, um Tickets anhand verschiedener Kriterien zu finden,
- ein Formular zur Erstellung neuer Projekttickets,

- eine Detailansicht zur Anzeige bestehender Projekttickets,
- eine Bearbeitungssicht zur Bearbeitung von Informationen des Projekttickets.

Projekttickets sollen eine eindeutige Identifizierbarkeit aufweisen und einen klar definierten Lebenszyklus besitzen.

4.3.2 Ticketinformationen und Datenfelder

Zuordnung: F1 (Ticketdaten im Formular), ggf. ergänzend F4 (rollenabhängige Pflege)

Ein Projektticket soll verschiedene Informationsfelder enthalten, darunter unter anderem eine eindeutige Projektticket_ID, Angaben zum Erstellungsdatum und Ersteller, einen Titel sowie eine Beschreibung in Freitextform. Weitere Felder betreffen die Projektzuordnung, zuständige Sachbearbeiter, kunden- und auftragsbezogene Informationen, den geschätzten Aufwand, abrechnungsrelevante Angaben sowie den Freigabestatus des Projekttickets.

Zu den im Projektticket vorgesehenen Feldern zählen außerdem Beschreibung, Titel, Stunden, Team, Projekt, Sachbearbeiter, Parent-Ticket, Auftraggeber, Kundenabteilung, geforderter Liefertermin, Kundenabruf, Kundenauftrag, Fahrzeugprojekt, Entwicklungsauftragsnummer, Freigabe-Status, Meilensteintabelle, Ticketpreis, abgerechneter Betrag, offener Betrag, Bemerkungen sowie der Abschlusstermin.

Ein Teil dieser Felder wird automatisch durch das System gesetzt, während andere durch Benutzer gepflegt werden.

4.3.3 Rollen und Berechtigungen

Zuordnung: F2 (Rollenmodell), F3 (Projektleiterentscheidung inkl. Begründung), F4 (rollenabhängige Rechte auf Felder)

Das Projektticket-Modul soll ein rollenbasiertes Zugriffskonzept unterstützen. Vorgesehen sind die Rollen Ersteller, Teamleiter, Projektleiter sowie Sachbearbeiter.

Der Ersteller soll berechtigt sein, neue Projekttickets anzulegen, wobei er nur eine begrenzte Anzahl von Feldern ausfüllen darf. Er soll ausschließlich Projekttickets bearbeiten dürfen, die er selbst erstellt hat. Bei der Ticketerstellung soll das zugehörige Team automatisch vorausgewählt sein, sofern eine entsprechende Zuordnung besteht. Der Ersteller darf die Basisfelder Beschreibung, Titel (Pflichtfeld), Stunden (Pflichtfeld), Team (Pflichtfeld), Projekt, Sachbearbeiter, Parent-Ticket, Auftraggeber, Kundenabteilung sowie den geforderten Liefertermin erfassen.

Teamleiter sollen über die gleichen Berechtigungen wie Ersteller verfügen. Zusätzlich soll

bei der Ticketerstellung für Teamleiter die Zuordnung des Projekttickets zu einem Projekt verpflichtend sein, dem ein Projektleiter zugeordnet ist. Auch für Teamleiter sind Titel, Stunden und Team verpflichtende Pflichtfelder.

Projektleiter sollen Projekttickets vollständig bearbeiten können, sofern sie dem zugehörigen Projekt zugeordnet sind. Sie sollen berechtigt sein, alle projekt-, kunden- und abrechnungsrelevanten Informationen zu erfassen. Zusätzlich soll für Projektleiter das Feld Sachbearbeiter verpflichtend sein, um eine eindeutige operative Zuständigkeit sicherzustellen. Für Projektleiter sind Titel, Stunden und Team verpflichtende Pflichtfelder; zusätzlich sind Projekt und Sachbearbeiter verpflichtend. Darüber hinaus sollen sie berechtigt sein, Projekttickets freizugeben oder abzulehnen. Im Falle einer Ablehnung ist eine Begründung anzugeben.

Der Sachbearbeiter ist der Benutzer, der für die operative Durchführung der Arbeitsschritte verantwortlich ist, die durch ein genehmigtes Projektticket initiiert werden. Für den Sachbearbeiter ist keine inhaltliche Bearbeitung der Projektticketdaten vorgesehen. Es ist sicherzustellen, dass der Sachbearbeiter über alle statusrelevanten Änderungen des Projekttickets informiert wird. Dies umfasst insbesondere Freigaben, Ablehnungen sowie Änderungen des Freigabestatus (beispielsweise «unbestätigt», «wartet», «genehmigt», «abgelehnt»).

4.3.4 Dynamische und rollenabhängige Felder

Zuordnung: F4 (rollenabhängige Feldsteuerung), F5 (Meilensteine = 100 %)

Das Projektticket-Modul soll dynamische Formularfelder unterstützen, deren Verhalten von der jeweiligen Benutzerrolle gesteuert wird. In Abhängigkeit von den zugewiesenen Berechtigungen sollen einzelne Felder entweder verpflichtend oder optional sowie schreibgeschützt oder vollständig ausgeblendet sein. Diese Vorgehensweise stellt sicher, dass Benutzer ausschließlich die für ihre Rolle relevanten Informationen erfassen oder bearbeiten können und die Dateneingabe strukturiert erfolgt.

Außerdem ist bei der Eingabe der Abrechnungsmeilensteine sicherzustellen, dass die prozentuale Verteilung der einzelnen Meilensteine in ihrer Gesamtheit den Wert 100 % ergibt.

4.4 Nicht-funktionale Anforderungen

4.4.1 Benutzerfreundlichkeit und mobile Nutzung

Zuordnung: NF1 (mobile Nutzbarkeit unter VPN)

Die Benutzeroberfläche des Projektticket-Moduls soll so gestaltet sein, dass auch eine Verwendung auf mobilen Geräten möglich ist. Smartphones spielen dabei eine besonders

wichtige Rolle, da der Zugriff häufig über mobile Geräte mit VPN-Tunneln erfolgt. Die Oberfläche soll eine übersichtliche Struktur aufweisen, sich an verschiedene Bildschirmgrößen anpassen und die Eingabe von Daten in Formulare auch auf kleinen Displays ermöglichen.

4.4.2 Benachrichtigungen

Zuordnung: NF2 (E-Mail bei Statusänderungen)

Das Projektticket-Modul soll Benachrichtigungen über relevante Statusänderungen von Projekttickets bereitstellen. Das Ziel besteht darin, die beteiligten Rollen über die erforderlichen Bearbeitungsschritte zu informieren und den Bearbeitungsfortschritt transparent zu machen.

Benachrichtigungen sollen an definierte Statusänderungen im Lebenszyklus eines Projekttickets gekoppelt sein und in Form von E-Mails erfolgen, um die Beteiligten zeitnah über erforderliche Bearbeitungsschritte zu informieren.

4.4.3 Technische und organisatorische Randbedingungen

Zuordnung: NF3 (Erweiterung ohne Architekturumbau)

Die Einführung des Projektticket-Moduls soll ohne Änderungen an der bestehenden Systemarchitektur und auf Basis des vorhandenen Technologie-Stacks erfolgen. Zudem ist eine möglichst schnelle Umsetzbarkeit vorgesehen, insbesondere unter Berücksichtigung der Testphase. Ein grundlegender Umbau der Architektur ist aufgrund des zeitlichen und organisatorischen Aufwands nicht vorgesehen.

4.5 Abgrenzung der Anforderungen

Die Anforderungen umfassen keine vollständige Neuentwicklung der Anwendung, den Ersatz bestehender Technologien sowie umfangreiche funktionale Erweiterungen außerhalb des Projektticket-Kontexts. Der Fokus liegt ausschließlich auf der Erweiterung der bestehenden Anwendung um die Funktionalität des Projekttickets.

4.6 Zusammenfassung der Anforderungen

Die Anforderungen legen den gewünschten Funktionsumfang, die Rollenlogik sowie die organisatorischen und technischen Rahmenbedingungen fest. Die hier beschriebenen Anforderungen bilden die Grundlage für die Konzeption und Implementierung des Projektticket-Moduls, die in den folgenden Kapiteln entwickelt werden.

5 Konzeption des Projektticket-Moduls

Das Konzept des Projektticket-Moduls beschreibt die Umsetzung der in Kapitel 4 definierten Anforderungen in der bestehenden Webanwendung. Das Ziel dieses Kapitels besteht darin, die konzeptionellen Entscheidungen darzulegen, die als Grundlage für die spätere Implementierung dienen sollen.

Das Konzept orientiert sich an den in Kapitel 3 erläuterten Analysen der bestehenden Anwendung sowie an den in Kapitel 4 beschriebenen funktionalen und nicht-funktionalen Anforderungen. Im Rahmen dessen erfolgt die Festlegung der Integration des Projektticket-Moduls in die vorhandene Systemarchitektur, der konzeptionellen Strukturen sowie der Organisation zentraler Aspekte wie der Benutzerinteraktion, der Rollenlogik und der Ticketabläufe.

In diesem Kapitel wird ausschließlich die konzeptionelle Ebene der Lösung erörtert. Die konkreten technischen Details sowie die Implementierungsentscheidungen werden im nachfolgenden Kapiteln behandelt.

5.1 Einordnung des Projektticket-Moduls in die bestehende Systemarchitektur

Die vorliegende Webanwendung basiert auf der klassischen Model-View-Controller-Architektur (MVC), wobei das Spring-Framework zum Einsatz kommt. Es wird darauf hingewiesen, dass die vorliegende Architektur für das Projektticket-Modul unverändert bestehen bleibt. Damit wird die nicht-funktionale Anforderung NF3 erfüllt. Das neue Modul integriert sich in die bestehende Struktur, indem zusätzliche Controller, Services und Persistenzkomponenten ergänzt werden, die ausschließlich für die Verarbeitung von Projekttickets zuständig sind. Die bestehenden Komponenten der Anwendung werden dabei nicht modifiziert.

Die Datenhaltung des Projektticket-Moduls erfolgt über zwei neu eingeführte Datenbanktabellen: Projektticket und Projektticket_Meilenstein. Diese Tabellen stellen eine Erweiterung des bestehenden Datenmodells der Anwendung dar und dienen der Abbildung projektticketbezogener Informationen. Die Integration in das bestehende Datenmodell erfolgt über definierte Fremdschlüsselbeziehungen zu bereits vorhandenen Entitäten, insbesondere zu Projekten, Teams und Benutzern.

Die Verwendung expliziter Fremdschlüssel gewährleistet eine konsistente fachliche und

technische Einbindung des Projektticket-Moduls in die bestehende Systemlandschaft. Gleichzeitig bleibt die bestehende Datenstruktur unverändert, da keine Anpassungen an den vorhandenen Tabellen erforderlich sind. Das Datenmodell wird somit gezielt erweitert, ohne dass bestehende Abhängigkeiten oder Funktionalitäten beeinträchtigt werden.

Die Präsentationsschicht des Projektticket-Moduls wird in die vorhandene JSP-basierte Benutzeroberfläche integriert. Die Steuerung der Zugriffsrechte und Rollen erfolgt weiterhin serverseitig über Spring Security. Die Implementierung rollenabhängiger Einschränkungen der Benutzeroberfläche, wie beispielsweise die Sichtbarkeit, die Aktivierung oder die Pflichtausfüllung einzelner Felder, erfolgt auf der Ebene der Darstellungsschicht. Die konkrete technische Umsetzung der UI-Steuerung und Zugriffssicherung wird in Kapitel 6 (insbesondere Abschnitte 6.4 und 6.7) detailliert beschrieben.

5.2 Konzeption zur Benutzerinteraktion und Formularstruktur

Die Konzeption der Benutzerinteraktion bildet eine zentrale Grundlage für die Entwicklung des Projektticket-Moduls. Das Ziel besteht darin, eine Benutzerführung zu definieren, die den in Kapitel 4 beschriebenen Anforderungen an mobile Nutzung (NF1), an die Reduktion von Kontextwechseln sowie an die rollenabhängige Feldlogik (F4) und die grundlegenden Ticketfunktionen (F1) entspricht und gleichzeitig eine effiziente und fehlerarme Bearbeitung der Projekttickets ermöglicht.

Da Projekttickets eine Vielzahl unterschiedlicher Informationen enthalten, darunter organisatorische, zeitliche und abrechnungsrelevante Daten, ist eine strukturierte und konsistente Benutzerinteraktion erforderlich. Die Konzeption der Benutzeroberfläche hat dabei nicht die visuelle Gestaltung zum Ziel, sondern legt fest, wie Benutzer mit dem System interagieren sollen und wie die Formularstruktur aufgebaut ist, um die in Kapitel 4 beschriebenen Vorgaben umzusetzen.

Ein besonderer Schwerpunkt liegt auf der rollenabhängigen Nutzung des Moduls, der Unterstützung mobiler Endgeräte sowie der Verringerung unnötiger Interaktions- und Kontextwechsel während der Ticketbearbeitung.

5.2.1 Ableitung konzeptioneller Prinzipien für die Benutzerinteraktion

Aus den in Kapitel 4 beschriebenen Anforderungen ergeben sich mehrere konzeptionelle Leitlinien für die Benutzerinteraktion des Projektticket-Moduls.

Zunächst ist zu berücksichtigen, dass ein Projektticket eine große Anzahl an Datenfeldern enthält, die unterschiedliche fachliche Aspekte abdecken. Daher muss die Interaktion mit dem Benutzer so gestaltet sein, dass diese Informationen strukturiert erfasst und übersichtlich dargestellt werden können, ohne den Benutzer mit unnötiger Komplexität zu überfordern.

Darüber hinaus ist für bestimmte Rollen (Ersteller, Teamleiter, Projektleiter, Sachbearbeiter) eine rollenabhängige Interaktion mit dem Formular erforderlich. Abhängig von der jeweiligen Rolle müssen bestimmte Felder verpflichtend, optional, schreibgeschützt oder unsichtbar sein. Die Benutzerinteraktion muss diese Unterschiede klar abbilden und den Benutzer eindeutig durch die für ihn relevanten Bearbeitungsschritte führen. Die konkreten Berechtigungen je Rolle (Ersteller, Teamleiter, Projektleiter, Sachbearbeiter) sind in Kapitel 4.3.3 (F2–F3) festgelegt; die daraus abgeleitete Feldsteuerung ist als Anforderung F4 formuliert.

Ein weiterer wesentlicher Aspekt betrifft die mobile Nutzung des Projektticket-Moduls. Da der Zugriff in der Regel über mobile Endgeräte erfolgt, muss die Interaktion so gestaltet sein, dass eine vollständige und komfortable Bearbeitung auch auf kleinen Bildschirmen möglich ist (NF1). Dies erfordert eine klare Struktur der Eingabeelemente sowie eine Reduzierung paralleler Interaktionen.

Aus den Anforderungen an die mobile Nutzung sowie die rollenabhängige Bearbeitung ergibt sich konzeptionell die Notwendigkeit, unnötige Kontextwechsel während der Ticketbearbeitung zu vermeiden. Benutzer sollen möglichst ohne Seitenwechsel oder Medienbrüche durch den Prozess der Ticketerstellung und -bearbeitung geführt werden. Alle für die jeweilige Rolle relevanten Informationen und Eingabedaten sollen in einem einheitlichen Interaktionskontext verfügbar sein.

Zusammenfassend ergeben sich für die Interaktion mit dem Benutzer folgende konzeptionelle Leitlinien:

- strukturierte Erfassung einer großen Anzahl fachlicher Daten (F1, F2),
- klare Abbildung rollenabhängiger Zuständigkeiten und Eingaberechte (F2, F4),
- Unterstützung mobiler Nutzungsszenarien (NF1),
- Reduktion von Kontextwechseln und unnötigen Interaktionsschritten (NF1),
- transparente und nachvollziehbare Führung durch den Ticketlebenszyklus (F3, NF2).

Diese Leitlinien bilden die Grundlage für die nachfolgende konzeptionelle Auswahl und Begründung der Struktur der Benutzeroberfläche.

5.2.2 Vorgehen zur Auswahl eines Interface-Patterns

Auf der Grundlage der in Abschnitt 5.2.1 abgeleiteten konzeptionellen Leitlinien ergibt sich die Frage, wie diese Anforderungen an die Benutzerinteraktion in einer geeigneten Struktur der Benutzeroberfläche umgesetzt werden können. Zur systematischen Beantwortung dieser Frage werden sogenannte Interface-Patterns herangezogen.

Interface-Patterns sind ein Mittel der Beschreibung wiederkehrender und bewährter Lösungsansätze für typische Probleme bei der Gestaltung der Mensch-Computer-Interaktion (HCI) [Tid20, S. 29–35; TK24]. Es handelt sich um abstrakte Entwurfsprinzipien, die unabhängig von konkreten Technologien angewendet werden können und die Strukturierung komplexer Benutzeroberflächen ermöglichen. Die theoretischen Grundlagen zu Interface-Patterns sowie deren Relevanz für die Gestaltung benutzerfreundlicher Interaktionen wurden bereits in Kapitel 2.5 vorgestellt.

Das methodische Vorgehen für das Konzept des Projektticket-Moduls sieht eine nachvollziehbare und an den Anforderungen orientierte Auswahl eines geeigneten Interface-Patterns vor. Das Ziel dieses Vorgehens besteht nicht in der Bewertung von Designpräferenzen, sondern darin, die Eignung unterschiedlicher Patterns unter Berücksichtigung der spezifischen Anforderungen des Projektticket-Moduls zu untersuchen.

Das Vorgehen ist in mehrere Schritte unterteilt. Zunächst werden geeignete Interface-Patterns identifiziert, die für formularbasierte Anwendungen mit umfangreicher Dateneingabe geeignet sind. Die Auswahl der betrachteten Patterns erfolgt auf Basis einschlägiger Literatur aus dem Bereich Human-Computer-Interaction sowie etablierter UX-Quellen [Tid20].

Die Bewertung der identifizierten Interface-Patterns erfolgt auf der Grundlage des in Abschnitt 2.5.1 entwickelten Kriterienkatalogs. Die dort definierten Kriterien – Übersichtlichkeit und Informationsdichte, kognitive Belastung, Unterstützung rollenabhängiger UI-Logik, mobile Anpassungsfähigkeit, technische Umsetzbarkeit innerhalb bestehender Systemarchitekturen sowie Erweiterbarkeit und Wartbarkeit – werden auf den konkreten Anwendungskontext des Projektticket-Moduls übertragen und für den Vergleich herangezogen.

Auf Basis des vorliegenden Kriterienkatalogs erfolgt ein qualitativer Vergleich der analysierten Interface-Patterns. Die Bewertung orientiert sich an der Erfüllung der in Kapitel 4 formulierten Anforderungen sowie der in Abschnitt 5.2.1 abgeleiteten konzeptionellen Leitlinien. Die Entscheidung für ein Interface-Pattern erfolgt erst nach Abschluss dieses Vergleichs.

5.2.3 Vergleich relevanter UI-Patterns

Auf Grundlage des in Abschnitt 5.2.2 beschriebenen Vorgehens werden im Folgenden mehrere etablierte Interface-Patterns betrachtet, die grundsätzlich für die Umsetzung formularbasierter Anwendungen mit umfangreicher Dateneingabe geeignet sind. In diesem Abschnitt erfolgt eine Gegenüberstellung der ausgewählten Patterns, die anhand des in Abschnitt 2.5.1 definierten Kriterienkatalogs erfolgt. Zudem erfolgt eine Einordnung der Patterns in den Kontext der Anforderungen des Projektticket-Moduls.

Da die grundlegenden Eigenschaften der betrachteten Patterns bereits in Kapitel 2.5 erläutert wurden, erfolgt an dieser Stelle keine erneute ausführliche Beschreibung. Stattdessen werden die Patterns insbesondere im Hinblick auf die Anforderungen F4 (rollenabhängige Feldlogik) und NF1 (mobile Nutzung, inkl. Zugriff über VPN) sowie anhand der Bewertungskriterien aus Abschnitt 2.5.1 vergleichend bewertet.

Die Auswahl der betrachteten Interface-Patterns orientiert sich an gängigen Entwurfsmustern aus der Literatur zur Mensch-Computer-Interaktion sowie an deren praktischer Relevanz für komplexe, datenintensive Formulare (vgl. Kapitel 2.5) [Tid20; TK24].

- One-Page Form (Langformular)
- Wizard / Step-by-Step-Form (Mehrschrittformular)
- Accordion
- Card-Based Layout

Diese Patterns unterscheiden sich insbesondere in ihrer Informationsstruktur, der Führung des Benutzers sowie der Verarbeitung umfangreicher Formulardaten.

Vergleich anhand definierter Bewertungskriterien

Der Vergleich der Interface-Patterns erfolgt anhand der in Abschnitt 2.5.1 definierten Bewertungskriterien. Die Bewertung erfolgt unter anderem unter Berücksichtigung von Aspekten wie Übersichtlichkeit, kognitive Belastung, Unterstützung rollenabhängiger UI-Logik, technische Umsetzbarkeit innerhalb der bestehenden JSP/Spring-MVC-Architektur sowie Eignung für die mobile Nutzung und den Zugriff über VPN-Verbindungen.

Die Ergebnisse dieses Vergleichs sind in Tabelle 5.1 zusammengefasst. Die Tabelle stellt eine qualitative Gegenüberstellung der betrachteten Interface-Patterns dar und dient als Grundlage für die im nächsten Abschnitt getroffene konzeptionelle Entscheidung.

Bewertungsskala und Einordnung der qualitativen Begriffe

Die Zuordnung der qualitativen Bewertungen in Tabelle 5.1 erfolgt auf Grundlage einer konzeptionellen Auswertung der jeweiligen Interface-Patterns. Diese stellt eine eigenständige konzeptionelle Einschätzung dar und basiert auf dem in Abschnitt 2.5.1 definierten Kriterienkatalog sowie dem Abgleich mit den in Kapitel 4 formulierten Anforderungen. Die vorliegenden Bewertungen spiegeln wider, in welchem Maß die einzelnen Patterns die definierten Kriterien im konkreten Nutzungsszenario des Projektticket-Moduls erfüllen.

Die in Tabelle 5.1 verwendeten Qualitätsbewertungen (z. B. niedrig, mittel, gut, einfach, eingeschränkt) dienen der vergleichenden Einordnung der Interface-Patterns. Es handelt sich dabei nicht um messbare Kennzahlen, sondern um qualitative Bewertung auf Basis der in Abschnitt 2.5.1 definierten Kriterien. Die Gesamtbewertung ergibt sich aus einer

zusammenfassenden Betrachtung aller Einzelkriterien.

Die verwendeten Begriffe sind wie folgt zu verstehen:

- **Niedrig / Schlecht:** bezeichnet eine geringe Eignung des Patterns für das jeweilige Kriterium oder einen deutlichen Zielkonflikt mit den in Kapitel 4 formulierten Anforderungen.
- **Mittel / Teilweise:** beschreibt eine grundsätzliche Eignung, die jedoch mit funktionalen Einschränkungen oder zusätzlichem konzeptionellem beziehungsweise technischem Aufwand verbunden ist.
- **Gut / Vollständig:** kennzeichnet eine hohe Übereinstimmung mit den konzeptionellen Leitlinien aus Abschnitt 5.2.1 und eine direkte Unterstützung der Anforderungen.
- **Einfach / Komplex (Umsetzung):** bezieht sich auf den erwarteten Implementierungsaufwand innerhalb der bestehenden serverseitigen JSP/MVC-Architektur unter Berücksichtigung rollenabhängiger Logik und Wartbarkeit.

Diese qualitative Bewertungsform ist in der frühen Konzeptionsphase üblich und ermöglicht einen strukturierten Vergleich alternativer Lösungsansätze, ohne dabei zu sehr ins technische Detail zu gehen.

Kriterium	One-Page Form	Wizard	Accordion	Card-Based
Übersichtlichkeit	Vollständig	Niedrig	Mittel	Niedrig
Kognitive Belastung	Mittel	Niedrig	Mittel	Hoch
UI-Logik Unterstützung	Gut	Komplex	Mittel	Komplex
Umsetzung in JSP/MVC	Einfach	Komplex	Mittel	Komplex
Mobile Anpassungsfähigkeit	Gut	Schlecht	Schlecht	Schlecht
Performance bei VPN	Gut	Niedrig	Mittel	Mittel
Eignung für Unternehmensanforderungen	Gut	Niedrig	Mittel	Niedrig
Eignung für große Formulare	Ja	Teilweise	Teilweise	Nein
Gesamtbewertung	Gut geeignet	Eingeschränkt geeignet	Bedingt geeignet	Nicht geeignet

Tabelle 5.1: Vergleich der UI-Patterns anhand definierter Bewertungskriterien

In der Bewertung wurden insbesondere jene Kriterien stärker berücksichtigt, die für das Projektticket-Modul als kritisch identifiziert wurden. Hierzu zählen die Reduktion von

Kontextwechseln, die konsistente Umsetzung rollenabhängiger Formularlogik sowie die stabile Nutzung unter mobilen VPN-Bedingungen.

5.2.4 Begründung des ausgewählten Interface-Patterns

Auf der Grundlage eines Vergleichs der entsprechenden Interface-Patterns, die in Abschnitt 5.2.3 vorgestellt wurden, wird in diesem Abschnitt eine begründete Auswahl der Struktur der Benutzeroberfläche für das Projektticket-Modul getroffen.

Das Ziel der Auswahl besteht darin, ein Interface-Pattern zu bestimmen, das die in Kapitel 4 formulierten funktionalen und nicht-funktionalen Anforderungen sowie die in Abschnitt 5.2.1 abgeleiteten konzeptionellen Leitlinien am besten unterstützt.

Die Auswertung ergibt, dass das One-Page-Formular im Vergleich zu den betrachteten Alternativen (Wizard, Accordion, Card-Based-Layout) den Anforderungen des Projektticket-Moduls am besten entspricht.

Unterstützung komplexer und umfangreicher Formulare

Ein wesentliches Merkmal des Projekttickets ist die große Menge an Informationen, die eingegeben werden müssen. Dazu zählen organisatorische, zeitliche, kundenbezogene und abrechnungsrelevante Aspekte. Wie in Tabelle 5.1 dargestellt, eignet sich das One-Page-Formular im Gegensatz zu anderen Patterns ohne Einschränkungen für große und komplexe Formulare.

Im Vergleich zum Wizard-Ansatz ist festzustellen, dass die Aufteilung der Dateneingabe in mehrere Schritte zu häufigen Kontextwechseln führen würde. Diese Vorgehensweise steht im Widerspruch zu der in Abschnitt 5.2.1 formulierten Leitlinie, den Benutzer möglichst ohne Seitenwechsel durch den Ticketprozess zu führen [Tid20, S. 86–88; Bud17].

Rollenabhängige UI-Logik

Das Projektticket-Modul erfordert eine ausgeprägte, rollenabhängige Benutzerinteraktion. Je nach Rolle des registrierten Benutzers (Ersteller, Teamleiter, Projektleiter, Sachbearbeiter) müssen die Felder des Formulars verpflichtend, optional, schreibgeschützt oder unsichtbar sein.

Das One-Page-Formular erlaubt eine zentrale und konsistente Umsetzung dieser UI-Logik innerhalb eines einheitlichen Interaktionskontexts. Im Gegensatz dazu erfordern Wizard- oder Card-Ansätze eine wesentlich komplexere Verwaltung der Rollenlogik über mehrere Teilansichten hinweg, was sowohl die Implementierung als auch die Wartung erschwert. Dies adressiert insbesondere die Anforderungen F2 (Rollenmodell) und F4 (rollenabhängige Feldsteuerung).

Mobile Nutzung und VPN-Bedingungen

Ein weiterer wesentlicher Aspekt, der bei der Entscheidungsfindung zu berücksichtigen ist, ist die mobile Nutzung des Projektticket-Moduls über VPN-Verbindungen. Unter diesen Voraussetzungen ist eine stabile und leistungsstarke Benutzerinteraktion erforderlich.

Das One-Page-Formular reduziert die Anzahl der erforderlichen Serveranfragen und Seitenwechsel und ist daher bei einer instabilen Netzwerkverbindung zuverlässiger [Bud13; W3C25; Web]. Mehrseitige Interaktionsmodelle, wie etwa Wizards oder Card-Based Interfaces, weisen in diesem Zusammenhang eine erhöhte Anfälligkeit für Verbindungsabbrüche und Ladeverzögerungen auf. Damit wird insbesondere NF1 (mobile Nutzung unter VPN-Bedingungen) erfüllt.

Integration in die bestehende Systemarchitektur

Die bestehende Webanwendung basiert auf einer serverseitigen JSP/MVC-Architektur. Das One-Page-Formular kann ohne strukturelle Änderungen in die vorliegende Architektur integriert werden und ermöglicht eine vergleichsweise einfache Umsetzung der erforderlichen Formular- und Rollenlogik mit geringem zusätzlichem Implementierungsaufwand.

Im Vergleich dazu erfordern alternative Interface-Patterns einen höheren Implementierungsaufwand, insbesondere in Bezug auf die Zustandsverwaltung, die Navigation zwischen den Ansichten und die Synchronisierung der Formular-Daten. Dies entspricht der Randbedingung NF3 (Erweiterung ohne Umbau des bestehenden Technologie-Stacks).

Zusammenfassende Bewertung

Das One-Page-Formular erfüllt die zentralen Anforderungen des Projektticket-Moduls in höherem Maße als die betrachteten Alternativen. Insbesondere unterstützt dieses Pattern die funktionalen Anforderungen F1 (Erstellen, Anzeigen, Bearbeiten von Projekttickets) und F4 (rollenabhängige Feldsteuerung innerhalb eines einheitlichen Formulars) in einer konsistenten Oberflächenstruktur. Des Weiteren leistet es einen Beitrag zur Erfüllung der nicht-funktionalen Anforderung NF1 (mobile Nutzbarkeit unter VPN-Bedingungen), da Seitenwechsel und zusätzliche Ladeprozesse reduziert werden. Durch die Umsetzung innerhalb der bestehenden JSP/Spring-MVC-Architektur wird außerdem die Randbedingung NF3 (Integration ohne grundlegenden Umbau der Systemarchitektur) eingehalten.

5.3 Konzeption der Rollen- und Berechtigungslogik

Die Logik der Rollen und Zugriffsrechte des Projektticket-Moduls ist so konzipiert, dass sie die organisatorischen Strukturen des Unternehmens flexibel abbildet und gleichzeitig einen klaren, nachvollziehbaren Ticketbearbeitungsprozess unterstützt. Ziel ist es, Zustän-

digkeiten nicht anhand eines starren Rollenmodells festzulegen, sondern sie dynamisch aus der organisatorischen Einbindung des Benutzers und der bei der Erstellung des Tickets getroffenen Auswahl abzuleiten. Die nachfolgende Rollenlogik konkretisiert die in Kapitel 4.3.3 definierten Anforderungen an Rollen und Berechtigungen (F2, F3) sowie die rollenabhängige Feldsteuerung (F4).

Im Rahmen der Erstellung eines Projekttickets nimmt jeder Benutzer zunächst die Rolle des Erstellers ein. Diese Rolle bildet das Fundament für alle weiteren Zugriffsberechtigungen. Abhängig von der organisatorischen Zuordnung des Benutzers zu Teams und Projekten kann der Ersteller zusätzlich die Rolle des Teamleiters, des Projektleiters oder beide Rollen gleichzeitig übernehmen. Damit sind die tatsächlichen Rechte des Benutzers abhängig vom Kontext und werden während der Interaktion mit dem Projektticket-Modul bestimmt.

In der Rolle des Erstellers ist der Benutzer dazu berechtigt, neue Projekttickets zu erstellen und grundlegende Informationen einzugeben. Dabei ist vorgesehen, dass der Ersteller nur die Projekttickets bearbeiten kann, die er selbst erstellt hat, und dass nur eine begrenzte Anzahl von Feldern zur Eingabe oder Änderung verfügbar ist. Bei der Erstellung eines Tickets erfolgt die Auswahl des Teams, dem der Benutzer zugeordnet ist, sofern eine solche Zuordnung besteht. Auf dieser Grundlage wird überprüft, ob der Benutzer auch Teamleiter ist.

Wenn der Ersteller der Leiter des ausgewählten Teams ist, übernimmt er automatisch die Rolle des Teamleiters. In diesem Fall ist eine Erweiterung des Verantwortungsbereichs des Benutzers zu erwarten. Konzeptionell ist vorgesehen, dass für Teamleiter die Projektzuweisung verpflichtend ist und damit die Ernennung eines verantwortlichen Projektleiters für das ausgewählte Projekt sicherstellt. Somit nimmt der Teamleiter eine koordinierende Funktion ein, die sich auf den Prozess zwischen der ursprünglichen Erstellung des Tickets und der technischen Genehmigung auf Projektebene bezieht.

Sofern der Ersteller zusätzlich Leiter des ausgewählten Projekts ist, nimmt er gleichzeitig auch die Rolle des Projektleiters ein. In diesem Fall verfügt der Benutzer über umfassende Bearbeitungsrechte innerhalb des Projekttickets. Alle projektbezogenen, kundenbezogenen und abrechnungsrelevanten Felder werden in diesem Szenario aktiviert und sind für den Benutzer bearbeitbar, während sie für andere Rollen schreibgeschützt bleiben. In diesem Szenario ist kein zusätzlicher Freigabeprozess erforderlich, sodass das Projektticket nach der Erstellung automatisch freigegeben wird.

Wenn der Ersteller der Teamleiter, jedoch nicht Projektleiter des zugeordneten Projekts ist, wird das Projektticket nach der Erstellung in einen Freigabeprozess überführt. Der zuständige Projektleiter wird über die neue Anfrage in Kenntnis gesetzt und ist befugt, diese aus technischer Sicht zu überprüfen. Der Projektleiter ist dazu befugt, das Ticket

entweder zu genehmigen oder abzulehnen. Im Falle einer Ablehnung ist eine Begründung zu verfassen, um die fachliche Entscheidung nachvollziehbar zu dokumentieren. Die Ablehnung des Projekttickets wird als fachliche Abschlussfolgerung betrachtet. Eine weitere inhaltliche Bearbeitung des Projekttickets ist in diesem Zustand nicht mehr vorgesehen.

Der Sachbearbeiter wird über den Status der Ablehnung informiert und erhält somit die Möglichkeit, das Ticket zu Dokumentations- und Nachverfolgungszwecken einzusehen.

Neben den Rollen des Erstellers, Teamleiters und Projektleiters ist konzeptionell auch die Rolle des Sachbearbeiters vorgesehen. Der Sachbearbeiter trägt die Verantwortung für die operative Durchführung der Arbeitsschritte, die durch ein genehmigtes Projektticket initiiert werden. Es besteht keine Berechtigung zur inhaltlichen Bearbeitung der Projektticketdaten. Dem Sachbearbeiter werden jedoch Benachrichtigungen über alle statusrelevanten Änderungen des Projekttickets, insbesondere über Freigaben, Ablehnungen sowie Änderungen des Freigabestatus, übermittelt. Dies ermöglicht eine transparente Nachverfolgung des Bearbeitungsfortschritts, ohne die Konsistenz der Ticketdaten zu beeinträchtigen.

Die Berechtigungen eines Benutzers werden demnach durch die Kombination seiner organisatorischen Zuordnung zu Teams und Projekten, der bei der Erstellung des Tickets getroffenen Auswahl sowie des aktuellen Status des Projekttickets bestimmt. Dieses dynamische Rollenmodell ermöglicht eine realistische Abbildung der Arbeitsprozesse und vermeidet gleichzeitig die Komplexität eines starren, systemischen Rollenmodells.

5.4 Konzeption des Ticket-Lebenszyklus und der Benachrichtigungen

Das sogenannte Ticket-Lebenszykluskonzept beschreibt den technischen Prozess eines Projekttickets von der Erstellung bis zur Freigabe oder Ablehnung. Das Ziel besteht darin, einen klar strukturierten, nachvollziehbaren und rollenbasierten Prozess zu definieren, der die Zusammenarbeit zwischen Ersteller, Teamleiter und Projektleiter unterstützt und den in Kapitel 4 beschriebenen Anforderungen entspricht. Die Konzeption orientiert sich insbesondere an der Anforderung zur Projektleiterentscheidung inkl. Ablehnungsbegründung (F3) sowie an der Anforderung zur Benachrichtigung bei statusrelevanten Änderungen per E-Mail (NF2).

Der Lebenszyklus eines Tickets ist eng mit der Rollenlogik des Projektticket-Moduls verknüpft. Abhängig von der Zuordnung des Benutzers können einzelne Bearbeitungsschritte automatisch erfolgen oder eine zusätzliche Freigabe erfordern.

5.4.1 Zustände eines Projekttickets

Nach der Erstellung eines Projekttickets befindet sich dieses zunächst in einem initialen Ausgangszustand. In dieser Phase existiert weder eine fachliche Annahme noch eine Ablehnung. Die Weiterleitung des Tickets erfolgt in Abhängigkeit von der Organisationsstruktur entweder an den zuständigen Teamleiter oder direkt an den Projektleiter. Erst im Rahmen dieser Weiterleitung wird der formale Entscheidungsprozess angestoßen.

Die initiale explizite Zustandsfestlegung wird durch den Projektleiter vorgenommen. Dieser trifft eine fachliche Entscheidung darüber, ob das Projektticket angenommen oder abgelehnt wird. Diese Entscheidung bildet eine eigenständige Statusdimension und dient ausschließlich der initialen Freigabepfung. Im Falle einer Ablehnung wird der Lebenszyklus des Tickets an dieser Stelle beendet. Es ist eine Begründung für die Entscheidung anzugeben, um die Nachvollziehbarkeit der Entscheidung zu gewährleisten. Damit wird die in Kapitel 4 formulierte Anforderung erfüllt, dass eine Ablehnung nur mit verpflichtender Begründung erfolgen darf (F3).

Im Falle der Annahme des Projekttickets durch den Projektleiter wird dieses in den eigentlichen Bearbeitungslebenszyklus überführt. Der Freigabestatus kann bereits während der fachlichen Prüfung gesetzt werden, entfaltet jedoch seine operative Bedeutung erst nach der Annahme des Projekttickets. Ab diesem Zeitpunkt wird ein separater Freigabestatus geführt, der den aktuellen Bearbeitungsstand des Tickets abbildet. Für diesen Freigabestatus sind die Zustände «Unbestätigt», «Wartet», «Genehmigt» und «Abgelehnt» vorgesehen. Im weiteren Verlauf können diese Zustände durch den Projektleiter angepasst werden. Sie dienen der operativen Steuerung und Nachverfolgung des Tickets. Konzeptionell werden damit zwei Statusdimensionen geführt: die initiale Projektleiterentscheidung (Annahme/Ablehnung) sowie der nachgelagerte operative Freigabestatus zur weiteren Bearbeitungssteuerung. Die Trennung von Projektleiterentscheidung und operativem Freigabestatus unterstützt zudem die strukturierte Bearbeitung des Tickets im weiteren Verlauf (F1).

Die konzeptionelle Trennung zwischen der Projektleiterentscheidung und dem nachgelagerten Freigabestatus führt zur Erreichung einer klaren und nachvollziehbaren Zustandslogik.

5.4.2 Konzeption der Benachrichtigungen

Die Benachrichtigungen stellen einen integralen Bestandteil des Projektticket-Moduls dar und unterstützen einen transparenten, rollenbasierten Ticketprozess. Das Ziel der hier dargestellten Konzeption der Benachrichtigung besteht in der Sicherstellung, dass alle beteiligten Benutzer zeitnah über statusrelevante Ereignisse informiert werden und erforderliche Bearbeitungsschritte eindeutig erkennbar sind.

Die Auslösung von Benachrichtigungen ist eng an den in Abschnitt 5.4.1 beschriebenen Ticket-Lebenszyklus gekoppelt. Benachrichtigungen erfolgen ausschließlich bei definierten Zustandsänderungen eines Projekttickets, bei denen eine Aktion oder Kenntnisnahme durch einen verantwortlichen Benutzer erforderlich ist. Diese Benachrichtigungslogik basiert auf der in Kapitel 4 definierten nicht-funktionalen Anforderung, statusrelevante Änderungen automatisch per E-Mail zu kommunizieren (NF2).

Nach der Erstellung eines Projekttickets erfolgt zunächst die Benachrichtigung des zuständigen Teamleiters, sofern der Ersteller diese Rolle nicht selbst einnimmt. Die Benachrichtigung dient der Mitteilung, dass ein neues Projektticket angelegt wurde und eine Zuordnung zu einem Projekt erforderlich ist.

Nach der Zuordnung eines Projekts wird der verantwortliche Projektleiter automatisch informiert, da eine fachliche Prüfung sowie eine Entscheidung über Annahme oder Ablehnung des Projekttickets erforderlich ist.

Im Falle einer Entscheidung über Annahme oder Ablehnung durch den Projektleiter erfolgt eine entsprechende Benachrichtigung des für das Projektticket zuständigen Sachbearbeiters. Im Falle einer Ablehnung des Projekttickets erfolgt eine zusätzliche Übermittlung einer Begründung, um die Entscheidung nachvollziehbar zu dokumentieren. In einem solchen Fall wird das Projektticket als geschlossen betrachtet.

Nach der Annahme eines Projekttickets wird dieses in den operativen Bearbeitungsprozess überführt, in dem der Projektleiter den Freigabestatus des Projekttickets weiterführend setzen oder ändern kann. Für diesen operativen Freigabestatus sind die Zustände «unbestätigt», «wartet», «genehmigt» und «abgelehnt» vorgesehen. Änderungen des Freigabestatus werden als fachlich relevante Ereignisse klassifiziert und lösen jeweils eine automatische Benachrichtigung an den Sachbearbeiter aus.

Die Benachrichtigungen werden konzeptionell in Form von E-Mails übermittelt, um eine zuverlässige Information der Benutzer auch bei mobiler Nutzung sicherzustellen. Die Wahl des E-Mail-Kanals unterstützt zudem die mobile Nutzung unter VPN-Bedingungen (NF1)

5.5 Konzeption der Datenstruktur

Die Konzeption der Datenstruktur des Projektticket-Moduls orientiert sich an den in Kapitel 4 beschriebenen funktionalen Anforderungen sowie an dem in Kapitel 5 definierten Ablauf der Ticketbearbeitung. Das Ziel besteht darin, die fachlichen Informationen eines Projekttickets in einer strukturierten, konsistenten und erweiterbaren Form darzustellen, ohne dabei unnötige Abhängigkeiten zur bestehenden Datenstruktur der Anwendung zu erzeugen. Die Struktur adressiert insbesondere die Anforderungen an die Abbildung der Ticketinformationen und Bearbeitung (F1) sowie die Abrechnungsmeilensteine inklusive prozentualer Gesamtlogik (F5).

Das Projektticket stellt das zentrale fachliche Objekt dar. Es fasst alle projektbezogenen, organisatorischen, zeitlichen und abrechnungsrelevanten Informationen eines Tickets zusammen. In der Konzeption wird eine Differenzierung zwischen Projektticket und zugehörigen Abrechnungsmeilensteinen vorgenommen, da ein Projektticket potenziell mehrere abrechnungsrelevante Teilschritte umfassen kann. Ein Projektticket kann demnach aus mehreren Meilensteinen bestehen, während jeder Meilenstein eindeutig einem Projektticket zugeordnet ist.

Diese Trennung ermöglicht eine flexible Abbildung unterschiedlicher Abrechnungsmodelle und unterstützt die in Kapitel 4 formulierte Anforderung, dass die prozentuale Verteilung der Meilensteine insgesamt 100% ergeben muss (F5). Die Datenstruktur bleibt in dieser Konzeption übersichtlich und erweiterbar, beispielsweise für zusätzliche Meilensteine oder weiterführende, abrechnungsbezogene Informationen.

Die konkrete technische Ausgestaltung der Datenstruktur sowie deren Implementierung in der bestehenden Systemarchitektur werden in Kapitel 6 detailliert beschrieben.

5.6 Zusammenfassung der Konzeption

In diesem Kapitel wurde die Konzeption des Projektticket-Moduls auf konzeptioneller Ebene dargelegt. Ausgehend von der bestehenden Systemarchitektur wurden zentrale Entwurfsentscheidungen zur Benutzerinteraktion, zur Auswahl des Interface-Patterns, zur Rollen- und Berechtigungslogik sowie zum Ticket-Lebenszyklus und zur Datenstruktur hergeleitet.

Die Konzeption stellt sicher, dass die in Kapitel 4 definierten funktionalen und nicht-funktionalen Anforderungen vollständig berücksichtigt werden und bildet eine konsistente Grundlage für die anschließende technische Umsetzung. Das nachfolgende Kapitel widmet sich daher der konkreten Implementierung der konzipierten Lösung innerhalb der bestehenden Webanwendung.

6 Implementierung der Lösung

In diesem Kapitel wird die technische Umsetzung des in Kapitel 5 beschriebenen Projektticket-Moduls beschrieben. Ziel ist es, die in diesem Kapitel entwickelten konzeptionellen Entscheidungen in der bestehenden Webanwendung umzusetzen. Der Fokus liegt auf den wesentlichen Implementierungsentscheidungen sowie deren technischer Ausgestaltung.

Die Implementierung erfolgt unter Berücksichtigung der bestehenden Systemarchitektur und unter Verwendung des bestehenden Technologie-Stacks der Anwendung. Strukturelle Änderungen am gesamten System wurden bewusst vermieden. Stattdessen wurde das Projektticket-Modul als eigenständige funktionale Erweiterung integriert, die sich nahtlos in die bestehende Architektur einfügt.

6.1 Technologische Grundlagen und Rahmenbedingungen

Die Implementierung des Projektticket-Moduls erfolgte vollständig innerhalb der bestehenden technischen Struktur der Webanwendung. Infolgedessen konnte das neue Modul ohne zusätzliche Änderungen an der Infrastruktur oder Architektur in den laufenden Betrieb integriert werden.

Als Programmiersprache findet Java Anwendung. Die Logik der Anwendung auf der Serverseite basiert auf dem Spring-Framework, insbesondere auf Spring MVC zur Umsetzung des Architekturmodells Model-View-Controller sowie auf Spring Security zum Schutz der Anwendung. Es konnte festgestellt werden, dass diese Technologien bereits Teil der bestehenden Webanwendung waren und konsequent weiterverwendet wurden.

Für die Präsentationsschicht werden JavaServer Pages (JSP) verwendet. Durch den Einsatz von JSP wird die Generierung dynamischer HTML-Seiten auf der Serverseite ermöglicht und eine enge Integration der Präsentationslogik mit den vom Controller bereitgestellten Daten gewährleistet. Dieser Ansatz ist mit der bestehenden Architektur kompatibel und unterstützt insbesondere die Umsetzung der Rollenlogik für die Anzeige und Eingabe.

Die Speicherung der Daten erfolgt mittels einer relationalen Datenbank, die auf dem Microsoft SQL Server basiert. Die Verwaltung und Pflege der Datenbankstrukturen erfolgt über das SQL Server Management Studio (SSMS). Auch in diesem Fall wurde bewusst entschieden, den bereits installierten Datenbanktyp zu verwenden, um die Konsistenz innerhalb der bestehenden Systemumgebung sicherzustellen.

6.2 Entwurf der Datenstruktur des Projekttickets

Die Implementierung der Datenstruktur des Projektticket-Moduls basiert auf der in Kapitel 5.5 entwickelten konzeptionellen Modellierung. Ziel der vorliegenden Umsetzung ist die Abbildung der fachlichen Anforderungen an Projekttickets in einer klar strukturierten und erweiterbaren relationalen Datenbank, ohne dabei die bestehenden Datenstrukturen der Webanwendung zu verändern.

Für das Projektticket-Modul wurden zwei neue Tabellen eingeführt: «Projektticket» und «Projektticket_Meilenstein». Die Tabelle «Projektticket» stellt die zentrale technische Einheit des Moduls dar und beinhaltet alle projektbezogenen, organisatorischen, zeitlichen sowie abrechnungsrelevanten Informationen eines Projekttickets. Zu den relevanten Metadaten zählen dabei insbesondere jene zur Erstellung, Verweise auf einbezogene Organisationseinheiten, Statusinformationen sowie Finanzkennzahlen.

Die Tabelle «Projektticket» ist über definierte Fremdschlüsselbeziehungen in das bestehende Datenmodell integriert. Die fachliche Zuordnung erfolgt über die Attribute Ersteller_ID und Hauptbearbeiter_ID zu der Tabelle «IAMT_Nutzer», über Projekt_ID zu der Tabelle «Projekt» sowie über Team_ID zu der Tabelle «Team». Die vorliegende Tabelle beinhaltet zudem das Attribut Parent_ID, das als selbstreferenzierender Fremdschlüssel auf die Tabelle «Projektticket» verweist. Dadurch wird die fachliche Verknüpfung von Projekttickets ermöglicht, beispielsweise zur Abbildung abhängiger oder hierarchischer Ticketstrukturen. Auf diese Weise wird eine eindeutige Zuordnung eines Projekttickets zu seinem Ersteller, dem verantwortlichen Team, dem zuständigen Projektleiter sowie dem Sachbearbeiter sichergestellt, ohne bestehende Datenstrukturen strukturell zu verändern.

Die Tabelle «Projektticket_Meilenstein» dient der Abbildung abrechnungsrelevanter Teilabschnitte eines Projekttickets. Es besteht eine 1:n-Beziehung zwischen den beiden Tabellen, da einem Projektticket mehrere Meilensteine zugeordnet sein können. Diese Struktur ermöglicht eine flexible Aufteilung des Ticketpreises in einzelne Abrechnungsphasen und unterstützt die in den Anforderungen definierte prozentuale Gewichtung der Meilensteine.

Zur Steuerung des Ticket-Lebenszyklus werden in der Tabelle «Projektticket» zwei logisch getrennte Statusinformationen geführt: die Projektleiterentscheidung (Annahme oder Ablehnung) sowie der Freigabestatus des Tickets. Die Projektleiterentscheidung stellt dabei die fachliche Freigabe dar, wohingegen der Freigabestatus den operativen Bearbeitungsstand abbildet und nachgelagerte Prozesse wie Bearbeitungsschritte und Benachrichtigungen steuert. Die operative Relevanz des Freigabestatus entfaltet sich erst nach erfolgter fachlicher Annahme des Projekttickets.

Die Abbildung 6.1 zeigt das relationale Datenmodell des Projektticket-Moduls mit den beiden zentralen Entitäten «Projektticket» und «Projektticket_Meilenstein» sowie deren

Beziehungen zu bestehenden Tabellen der Anwendung.

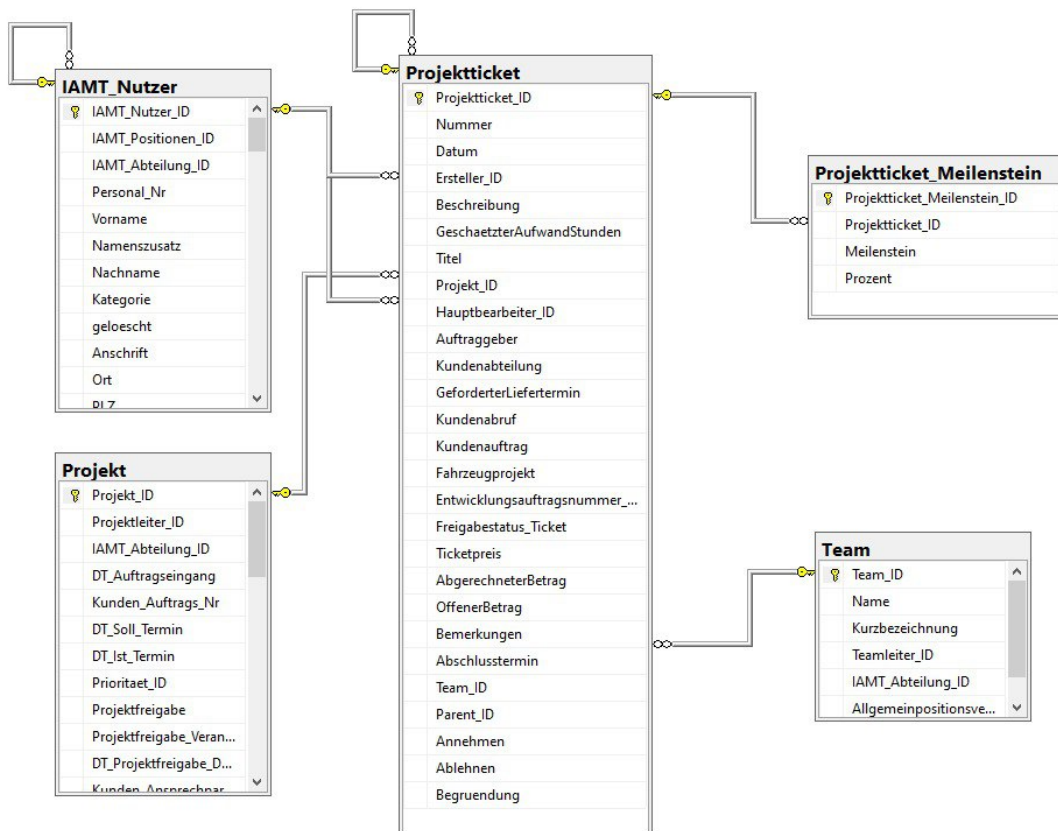


Abbildung 6.1: Projektticket, Projektticket_Meilenstein

6.3 Implementierung der Geschäftslogik

In Anlehnung an die in Kapitel 5.3 dargestellte Konzeption der Rollen- und Berechtigungslogik erfolgt die technische Umsetzung der zentralen Fachlogik innerhalb einer spezifischen Service-Schicht. Die vorliegende Schicht dient als vermittelnde Instanz zwischen der Controller-Schicht und der Datenzugriffsschicht (DAO) und stellt sicher, dass die fachliche Verarbeitungslogik konsistent und wiederverwendbar umgesetzt wird.

Die Ermittlung des Bearbeitungskontexts sowie die Durchsetzung rollenabhängiger Einschränkungen erfolgen überwiegend in der Controller- und Präsentationsschicht. Die Service-Schicht stellt demgegenüber eine konsistente Verarbeitung der ausgelösten fachlichen Aktionen sicher und kapselt wiederverwendbare Verarbeitungslogik, ohne an konkrete Benutzerinteraktionen gebunden zu sein. Somit bleibt die fachliche Verarbeitungsebene klar von UI-Logik und Persistenzdetails getrennt.

6.3.1 Ermittlung des Bearbeitungskontexts und der effektiven Benutzerrolle

Die Zuordnung der effektiven Benutzerrolle innerhalb des Projektticket-Moduls erfolgt nicht anhand eines statischen Rollenmodells. Stattdessen wird der Bearbeitungskontext zur Laufzeit aus einer Kombination von Benutzeridentität, organisatorischer Zuordnung zu Teams und Projekten sowie den Ticketstammdaten abgeleitet. Grundlage hierfür bildet der aus dem Spring-Security-Kontext ermittelte Benutzer (z. B. über den *SecurityContextHolder*) sowie die Zuordnung des Benutzers zu Teamleiter- und Projektleiterrollen anhand der zugehörigen Team- und Projektdaten.

Die hierfür erforderlichen Informationen werden in der Controller-Schicht aus dem Sicherheitskontext sowie aus den geladenen Ticket-, Projekt- und Teamdaten bestimmt. Auf dieser Grundlage wird die fachliche Rolle des angemeldeten Benutzers im Kontext eines spezifischen Projekttickets bestimmt, beispielsweise als Ersteller, Teamleiter oder Projektleiter. Abhängig von dieser Zuordnung werden serverseitig Steuerinformationen an die Präsentationsschicht übergeben (z.B. Flags wie *isErsteller*, *isTeamleiter*, *isProjektleiter*), welche die Sichtbarkeit und Editierbarkeit einzelner Formularfelder festlegen.

Die Service-Schicht führt keine eigenständige Ableitung oder Bewertung von Benutzerrollen durch. Sie verarbeitet ausschließlich jene Vorgänge, die durch den Controller unter Berücksichtigung des ermittelten Bearbeitungskontexts ausgelöst werden. Mittels dieser Aufgabenteilung wird sichergestellt, dass der Bearbeitungskontext eines Benutzers bei der Ausführung fachlicher Aktionen berücksichtigt wird, ohne dass innerhalb der Service-Schicht eine explizite Rollenverwaltung implementiert werden muss.

6.3.2 Validierung von Bearbeitungsschritten und Zustandsübergängen

Die Validierung von Bearbeitungsschritten und Zustandsübergängen erfolgt in dieser Implementierung nicht über eine zentrale fachliche Prüfungslogik innerhalb der Service-Schicht. Demgegenüber erfolgt die Sicherstellung der fachlichen Konsistenz der Bearbeitung mittels einer Kombination aus rollenabhängiger Steuerung im Controller, klar getrennten Bearbeitungswege und gezielten Persistenzoperationen.

Die Controller-Schicht ist so ausgelegt, dass ausschließlich jene Verarbeitungsschritte ausgelöst werden, die im aktuellen Bearbeitungskontext fachlich zulässig sind. Zustandsändernde Aktionen wie die Annahme oder Ablehnung eines Projekttickets, werden demnach nicht über allgemeine Aktualisierungsoperationen ausgeführt, sondern über dedizierte Service-Methoden (*acceptTicket(...)*, *declineTicket(...)*). Diese leiten die entsprechenden Vorgänge gezielt an die Persistenzschicht weiter.

Die Service-Methoden delegieren die eigentliche Zustandsänderung an die Datenzugriffsschicht, welche die dafür vorgesehenen Statusfelder des Tickets aktualisiert. Im Falle einer

Ablehnung wird das Projektticket serverseitig als abgelehnt gekennzeichnet und zusätzlich eine Begründung persistiert. Die bereits erfassten Ticketinhalte bleiben hierbei unverändert erhalten, sodass die Entscheidung auch nach Abschluss des Vorgangs nachvollziehbar dokumentiert ist.

Die Verhinderung unzulässiger Bearbeitungsschritte erfolgt in dieser Implementierung primär durch getrennte, serverseitige Zustandsaktionen (z. B. Annahme/Ablehnung) sowie ergänzend durch eine statusabhängige Einschränkung in der Benutzeroberfläche. Dadurch werden «begleitende» Zustandsaktionen bei einer allgemeinen Formularaktualisierung vermieden. Darüber hinaus wird der endgültige Ticketzustand (z. B. abgelehnt) in der Oberfläche konsequent als schreibgeschützt geführt. Dies reduziert fehlerhafte Bearbeitungsversuche im UI und unterstützt die konsistente Umsetzung des fachlichen Ticketabschlusses.

6.3.3 Verarbeitung fachlicher Aktionen im Ticketprozess

Die im Projektticket-Modul vorgesehenen fachlichen Aktionen werden über klar abgegrenzte Methoden innerhalb der Service-Schicht umgesetzt. Hierzu zählen insbesondere das Anlegen neuer Projekttickets (*saveTicket(...)*), die Aktualisierung bestehender Ticketdaten (*updateTicket(...)*) sowie zustandsverändernde Aktionen wie die fachliche Freigabe oder Ablehnung eines Projekttickets (*acceptTicket(...)*, *declineTicket(...)*).

Jede dieser Aktionen wird als fachlich eigenständiger Verarbeitungsschritt betrachtet und durch die Controller-Schicht explizit ausgelöst. Die Service-Schicht ist für die fachliche Verarbeitung der jeweiligen Aktion zuständig und koordiniert die notwendigen Verarbeitungsschritte. Dabei ist zu beachten, dass die Service-Schicht über keine Kenntnisse bezüglich der konkreten Darstellung oder des Ablaufs der Benutzerinteraktion verfügt.

Im Rahmen der Verarbeitung übernimmt die Service-Schicht zusätzlich eine zentrale Aufbereitung der übergebenen Ticketdaten, bevor diese an die Persistenzschicht weitergeleitet werden. Ziel dieser Vorverarbeitung ist die Sicherstellung einer konsistenten und technisch verwertbaren Datenstruktur, um fehlerhafte oder widersprüchliche Zustände in der Datenbank zu vermeiden. Zu diesem Zweck werden die Eingaben aus den Formularfeldern bereinigt, wobei überflüssige Leerzeichen entfernt und inhaltlich leere Werte als nicht gesetzt interpretiert werden. Im vorliegenden Kontext ist es erforderlich, dass solche Felder konsequent als *NULL* behandelt werden. Nur durch die einheitliche Anwendung dieses Vorgehens kann eine einheitliche Datenhaltung sichergestellt werden. Darüber hinaus werden Datumsangaben geprüft und ungültige bzw. systembedingt auftretende Platzhalterwerte entfernt, sofern kein fachlich relevanter Termin vorliegt. Durch diese systematische Datenaufbereitung wird gewährleistet, dass formularbasierte Eingaben unabhängig vom jeweiligen Bearbeitungskontext zuverlässig, nachvollziehbar und konsistent verarbeitet werden.

6.3.4 Abgrenzung zur Präsentations- und Persistenzschicht

Die Service-Schicht des Projektticket-Moduls ist ausschließlich für die Umsetzung fachlicher Logik zuständig und bewusst von darstellungs- sowie persistenzbezogenen Aspekten getrennt. Entscheidungen über die Darstellung, Sichtbarkeit oder Bearbeitbarkeit einzelner Eingabefelder werden demnach nicht innerhalb der Service-Schicht getroffen, sondern in der Präsentationsschicht umgesetzt. Diese umfasst sowohl die serverseitig gerenderten JSP-Seiten als auch die clientseitige JavaScript-Logik, welche die rollen- und statusabhängige Benutzerinteraktion steuert.

Ebenso verbleiben alle persistenzspezifischen Details vollständig in der Datenzugriffsschicht. Die Service-Schicht greift ausschließlich über klar definierte Schnittstellen auf diese Schicht zu und enthält weder SQL-Anweisungen noch Kenntnisse über konkrete Tabellenstrukturen oder Datenbanktechnologien. Die Persistenzoperationen werden über *JdbcTemplate*-basierte DAO-Implementierungen umgesetzt, wobei die Zuordnung relationaler Daten auf Domänenobjekte über dedizierte RowMapper erfolgt.

Durch die klare Trennung der Verantwortlichkeiten wird sichergestellt, dass die fachliche Logik des Projektticket-Moduls unabhängig von der Benutzeroberfläche und der eingesetzten Persistenztechnologie umgesetzt werden kann. Gleichzeitig wird eine hohe Wartbarkeit und Erweiterbarkeit des Moduls innerhalb der bestehenden Systemarchitektur gewährleistet.

6.4 Implementierung der Benutzerinteraktion

Dieser Abschnitt beschreibt die technische Umsetzung der in Kapitel 5.2 beschriebenen Benutzerinteraktion. Die Interaktion erfolgt mittels eines One-Page-Formulars, welches sowohl für die Erstellung als auch für die Anzeige und Bearbeitung eines Projekttickets verwendet wird. Rollen- und statusabhängige Unterschiede werden nicht durch separate Seiten implementiert, sondern durch eine dynamische Steuerung der Eingabefelder und Aktionsschaltflächen innerhalb derselben Oberfläche.

6.4.1 Umsetzung des One-Page-Formulars

Die Erstellung eines Projekttickets erfolgt über den Endpunkt `/addTicketPage`. Beim Aufruf der Seite wird ein neues Ticketobjekt serverseitig vorbereitet und anschließend gemeinsam mit den erforderlichen Stammdaten an die View übergeben. Dazu gehören insbesondere die verfügbaren Projekte, Mitarbeiter, Teams sowie bestehende Tickets, die als Parent-Tickets referenziert werden können.

Zusätzlich wird eine feste Anzahl an Meilensteinzeilen initialisiert, um unabhängig von der späteren Nutzung eine konsistente Tabellenstruktur im Formular bereitzustellen. Die

Initialisierung der One-Page-Form erfolgt über die folgende Controller-Methode.

```
1 @GetMapping("/addTicketPage")
2 public String addTicket(Model model) {
3     Ticket ticket = new Ticket();
4     List<Projekt> projekte = projektService.loadProjekte();
5     List<User> mitarbeiter = userService.loadUsers();
6     List<Teamlead> teamlead = teamleadService.loadTeams();
7     List<Ticket> alleTickets = ticketService.loadParentTickets();
8
9     String today = new SimpleDateFormat("dd.MM.yyyy")
10         .format(new java.util.Date());
11
12     int userId = loginService.getUserByUsername(
13         ((UserDetails) SecurityContextHolder.getContext()
14             .getAuthentication().getPrincipal()).getUsername()
15     ).getIAMT_Nutzer_ID();
16
17     ticket.setDatum(today);
18
19     for (int i = 0; i < 10; i++) {
20         ticket.getMeilensteine().add(new Projektticket_Meilenstein());
21     }
22
23     ticket.setErsteller_ID(userId);
24
25     model.addAttribute("ticketAdd", ticket);
26     model.addAttribute("mitarbeiter", mitarbeiter);
27     model.addAttribute("projekte", projekte);
28     model.addAttribute("teamlead", teamlead);
29     model.addAttribute("alleTickets", alleTickets);
30     model.addAttribute("loggedInUserId", userId);
31
32     return "addTicket";
33 }
```

Listing 6.1: Controller – Initialisierung der Ticketerstellung (GET /addTicketPage)

Die Initialisierung, die in der vorliegenden Arbeit dargestellt wird, bildet die Grundlage dafür, dass die Benutzeroberfläche ein einheitliches Formular anzeigen kann – und das unabhängig vom Endgerät oder Bearbeitungskontext. Gleichzeitig werden durch die Übergabe von Referenzdaten alle Auswahlfelder unmittelbar beim Laden der Seite befüllt.

6.4.2 Dynamische Formularbereiche (Meilensteine)

Ein Projektticket kann mehrere abrechnungsrelevante Meilensteine enthalten, welche im Formular in tabellarischer Form abgebildet werden. Im Zuge der Implementierung wird sichergestellt, dass nur vollständig ausgefüllte Meilensteinzeilen gespeichert werden. Eine

Zeile wird als vollständig betrachtet, wenn sowohl die Bezeichnung des Meilensteins als auch der Prozentwert gesetzt wurden.

Beim Absenden des Formulars werden die Ticketdaten zunächst gespeichert. Im Anschluss erfolgt die Verarbeitung der übergebenen Meilensteine. Die zugehörige Persistierung wird über die Meilenstein-Service- und DAO-Schicht umgesetzt.

```
1 @PostMapping("/addTicketPage")
2 public String saveTicket(Model model, @ModelAttribute("ticketAdd")
   Ticket ticket) {
3
4     String today = new SimpleDateFormat("dd.MM.yyyy").format(new java.
       util.Date());
5     //int userId = loginService.getUserByUsername(...);
6
7
8     ticket.setErsteller_ID(userId);
9     ticket.setDatum(today);
10
11     ticketService.saveTicket(ticket);
12
13     if(ticket.getMeilensteine() != null) {
14         for(Projektticket_Meilenstein m : ticket.getMeilensteine()) {
15             // Only save if both fields are filled
16             if (m.getMeilenstein() != null && !m.getMeilenstein().
               isEmpty()
17                 && m.getProzent() != null) {
18                 m.setProjektticket_ID(ticket.getProjektticket_ID());
19                 meilensteinService.saveMeilenstein(m);
20             }
21         }
22     }
23
24     boolean isProjektleiter = projekt.stream()
25         .anyMatch(p -> p.getProject_ID() == ticket.getProject_ID()
26             && p.getProjektleiter_ID() == userId);
27
28     // If user has authority -> auto-accept ticket
29     if (isProjektleiter) {
30         ticketService.acceptTicket(ticket.getProjectticket_ID());
31     }
32
33     // Return the view name
34     return "redirect:/tickets";
35 }
```

Listing 6.2: Controller – Speichern eines Tickets inkl. Meilensteine (POST /addTicketPage)

Durch diese Vorgehensweise wird verhindert, dass leere oder unvollständige Meilensteinzeilen als Datensätze persistiert werden. Gleichzeitig bleibt die Oberfläche in der Lage, eine feste Tabellenstruktur darzustellen, ohne dass jede Zeile zwingend gespeichert werden muss.

6.4.3 Rollen- und statusabhängige Feldsteuerung

Die rollen- und statusabhängige Steuerung der Bearbeitbarkeit erfolgt in der vorliegenden Implementierung überwiegend in der Präsentationsschicht. Im Controller werden dazu relevante Kontextinformationen ermittelt und an die JSP-Seite übergeben. Im Rahmen der Prüfung wird insbesondere analysiert, ob der angemeldete Benutzer als Ersteller des Tickets, als Teamleiter des ausgewählten Teams oder als Projektleiter des ausgewählten Projekts zu interpretieren ist.

Diese Informationen werden als Flags an die View übertragen und dort sowohl serverseitig (JSP) als auch clientseitig (JavaScript) zur Steuerung der Eingabefelder genutzt.

```
1 @GetMapping("/ticketInfo")
2 public String ticketInfoPage(@RequestParam("projektticket_ID") int id,
   Model model) {
3
4     List<Projekt> projekte = projektService.loadProjekte();
5     List<Teamlead> teamlead = teamleadService.loadTeams();
6
7     Ticket myTicket = ticketService.getTicket(id);
8
9     int userId = loginService.getUserByUsername(
10         ((UserDetails) SecurityContextHolder.getContext().
11         getAuthentication().getPrincipal()).getUsername()
12         ).getIAMT_Nutzer_ID();
13
14     boolean isProjektleiterFürDiesesTicket = projekte.stream()
15         .anyMatch(p -> p.getProjektleiter_ID() == userId && p.
16         getProject_ID() == myTicket.getProject_ID());
17
18     boolean isTeamleiterFürDiesesTicket = teamlead.stream()
19         .anyMatch(t -> t.getTeam_ID() == myTicket.getTeam_ID() && t.
20         getTeamleiter_ID() == userId);
21
22     boolean isErsteller = myTicket.getErsteller_ID() == userId;
23
24     List<Projektticket_Meilenstein> meilensteine = meilensteinService.
25     loadMeilensteine(id);
26     myTicket.setMeilensteine(meilensteine);
27
28     int maxRows = 10;
```

```

25     for (int i = meilensteine.size(); i < maxRows; i++) {
26         myTicket.getMeilensteine().add(new Projektticket_Meilenstein());
27     }
28
29     model.addAttribute("tickets", myTicket);
30     model.addAttribute("projekte", projekte);
31     model.addAttribute("teamlead", teamlead);
32     model.addAttribute("isProjektleiter", isProjektleiterFürDiesesTicket
33 );
34     model.addAttribute("isTeamleiter", isTeamleiterFürDiesesTicket);
35     model.addAttribute("isErsteller", isErsteller);
36     model.addAttribute("loggedInUserId", userId);
37
38     return "ticketInfo";
39 }

```

Listing 6.3: Controller – Laden eines Tickets und Rollenflags (GET /ticketInfo)

Die eigentliche Aktivierung oder Deaktivierung von Eingabefeldern erfolgt clientseitig über JavaScript. In diesem Kontext wird insbesondere berücksichtigt, ob ein Ticket bereits angenommen oder abgelehnt wurde. Ein abgelehntes Ticket wird vollständig gesperrt, so dass weder Ticketdaten noch Meilensteine verändert werden können. Ist der angemeldete Benutzer Projektleiter des im Formular gewählten Projekts, werden dagegen auch kundenbezogene und abrechnungsrelevante Felder freigeschaltet. Zudem werden die Aktionen «Annehmen» und «Ablehnen» zur Verfügung bereitgestellt.

```

1 function updateFieldStates(dynamicProjektleiter = null) {
2     const accepted = ($('#annehmen').val() || '') === 'true';
3     const declined = ($('#ablehnen').val() || '') === 'true';
4     const selectedProjektleiterId = $("#projekt_ID option:selected").
5     data("projektleiter");
6     const selectedTeamleiterId = $("#team_ID option:selected").data("
7     teamleiter");
8
9     const isCurrentProjektleiter = (selectedProjektleiterId) ===
10     currentUserId;
11     const isCurrentTeamleiter = (selectedTeamleiterId) === currentUserId
12     ;
13
14     $("#acceptBtn, #rejectBtn").hide();
15
16     // 1) Abgelehnt -> komplett gesperrt
17     if (declined) {
18         lockFields(allFields, true);
19         lockMilestones(true);
20         if (submitBtn) submitBtn.disabled = true;
21         return;
22     }
23 }

```

```

18     }
19
20     // 2) Projektleiter des ausgewählten Projekts -> erweitertes
Editieren
21     if (isCurrentProjektleiter) {
22         unlockFields(allFields);
23         lockMilestones(false);
24         setRequiredFields(projektleiterRequiredFields);
25
26         if (submitBtn) submitBtn.disabled = (!accepted);
27
28         if (!accepted && !declined) {
29             $("#acceptBtn, #rejectBtn").show();
30         }
31         return;
32     }
33
34     // ... weitere Fälle (Ersteller/Teamleiter/Fallback)
35 }

```

Listing 6.4: JSP/JavaScript – Kernlogik zur Feldsperrung und Projektleiter-Modus (ticketInfo.jsp)

Die Kombination aus serverseitiger Kontextbestimmung und clientseitiger Reaktion auf Status- und Rollenbedingungen ermöglicht die flexible Transformation des Formulars in unterschiedliche Bearbeitungsmodi, ohne dass hierfür separate Ansichten erforderlich sind.

6.4.4 Mobile Nutzbarkeit und responsives Verhalten

Die Benutzeroberfläche des Projektticket-Moduls wurde so umgesetzt, dass eine vollständige Nutzung auch auf mobilen Endgeräten möglich ist. Da der Zugriff in der Praxis häufig über Smartphones und unter VPN-Bedingungen erfolgt, wurde bei der Implementierung insbesondere auf eine übersichtliche, zusammenhängende Formularstruktur geachtet.

Das One-Page-Formular wird auf kleineren Bildschirmen in eine vertikale Darstellung überführt, sodass Eingabefelder in einer klaren Reihenfolge untereinander angeordnet bleiben und ohne horizontales Scrollen bedient werden können. Dadurch wird die Erfassung umfangreicher Ticketdaten auch bei begrenzter Bildschirmgröße ermöglicht.

Die in Abbildung 6.2 dargestellte mobile Darstellung der Seite zur Ticketerstellung dient als exemplarisches Beispiel. Es wird deutlich, dass sowohl Pflichtfelder als auch ergänzende Ticketinformationen und die Aktionsschaltflächen (Add und Back) vollständig zugänglich bleiben.

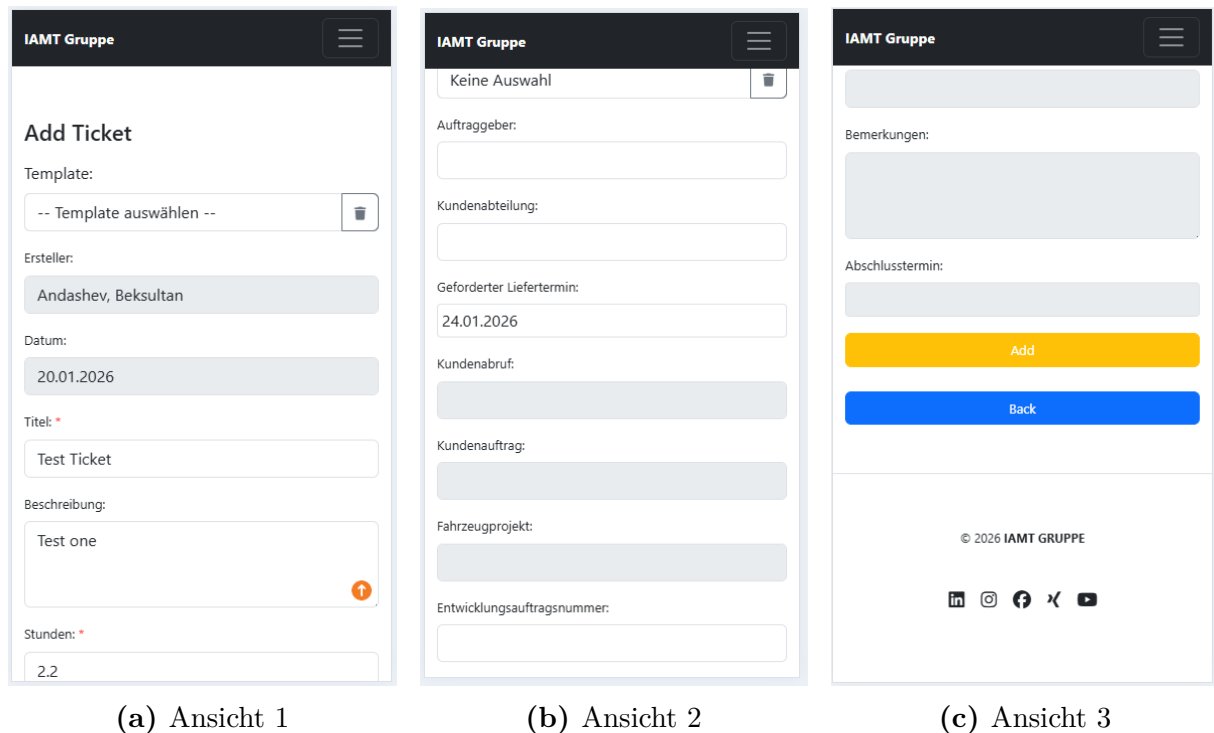


Abbildung 6.2: Add Ticket Formular (mobile Ansicht)

6.5 Implementierung des Ticket-Lebenszyklus und der Zustandslogik

In Anlehnung an Kapitel 5.4.1 wird der Lebenszyklus eines Projekttickets technisch über explizite Zustandsfelder im Ticketmodell sowie über dedizierte Verarbeitungsschritte im Controller und in der Persistenzschicht abgebildet. Die Implementierung trennt dabei zwischen der formularbasierten Aktualisierung der Ticketdaten und zustandsverändernden Entscheidungen (Annahme/Ablehnung). Dadurch wird sichergestellt, dass finale Entscheidungen technisch nachvollziehbar persistiert und nachgelagerte Bearbeitungsschritte kontrolliert eingeschränkt werden können.

6.5.1 Technische Abbildung der Zustände

Bezüglich der Datenhaltung wird der Ticketzustand nicht als einzelnes Statusobjekt modelliert, sondern über mehrere zustandsrelevante Attribute der Entität Ticket bzw. der Tabelle Projektticket abgebildet. Die Projektleiterentscheidung wird über die booleschen Felder «Annehmen» und «Ablehnen» gespeichert. Im Falle einer Ablehnung wird eine zusätzliche Begründung angefügt, um die Entscheidung nachvollziehbar zu dokumentieren. Zusätzlich existiert mit dem Statuswert «Freigabestatus_Ticket» ein weiterer Statuswert, der den Freigabestatus (z. B. unbestätigt, wartend, genehmigt, abgelehnt) abbildet. Dieses Feld wird in der Oberfläche angezeigt, ist in der aktuellen Implementierung jedoch schreibgeschützt geführt und erst im Projektleiter-Kontext zur Bearbeitung freigegeben.

Durch diese Kombination ist das System in der Lage, sowohl die finale Projektleiterentscheidung (angenommen/abgelehnt) als auch einen zusätzlichen Freigabestatus zu speichern und anzuzeigen. Die Zustände sind somit direkt im Ticketobjekt festgelegt und können in Oberfläche, Controller und Persistenzschicht konsistent ausgewertet werden.

6.5.2 Umsetzung der Zustandsübergänge

Zustandsübergänge werden in der Implementierung nicht implizit im Rahmen einer allgemeinen Ticketaktualisierung durchgeführt, sondern über explizite Zustandsaktionen ausgelöst. Die Unterscheidung erfolgt über den Request-Parameter `ticketAction`, der als Hidden-Field in das Formular integriert wird. Hierdurch wird eine klare Unterscheidung zwischen einer regulären Formularaktualisierung (Update) und finalen Zustandsentscheidungen (Annahme/Ablehnung) ermöglicht.

Die Vorbereitung der Zustandsaktionen erfolgt clientseitig über JavaScript. Durch das Betätigen der Schaltflächen «Annehmen» oder «Ablehnen» wird das Hidden-Feld (`ticketAction`) mit dem Wert `accept` bzw. `decline` gesetzt. Zusätzlich werden die Hidden-Felder `annehmen/ablehnen` aktualisiert, damit der aktuelle Zustand im Formular eindeutig abgebildet werden kann. Im Falle einer Ablehnung wird außerdem ein Pflichtdialog geöffnet, über den der Ablehnungsgrund erfasst und als Request-Parameter `reason` an den Server übergeben wird.

Ein wesentlicher Unterschied besteht im Auslösezeitpunkt der Persistierung:

- **accept:** Die Annahme wird im Formular vorgemerkt, indem der Status gesetzt und der Update-Button freigeschaltet wird. Es erfolgt jedoch kein automatisches Absenden des Formulars. Die Annahme wird daher erst serverseitig verarbeitet und gespeichert, wenn der Projektleiter anschließend bewusst die Funktion `Update` ausführt.
- **decline:** Nach Eingabe eines verpflichtenden Ablehnungsgrundes wird sofort ein Formular-Submit ausgelöst. Dadurch wird die Ablehnung unmittelbar serverseitig verarbeitet, inklusive Persistierung der Begründung, und anschließend direkt zur Ticketübersicht weitergeleitet.

Die serverseitige Verarbeitung erfolgt im POST-Endpunkt `/ticketInfo`. Dort wird anhand von `ticketAction` zwischen Annahme und Ablehnung unterschieden. Die fachliche Zustandsänderung wird über spezialisierte Service-Methoden (`acceptTicket(...)` bzw. `declineTicket(..., reason)`) ausgelöst und anschließend über `updateTicket(...)` konsistent in der Ticket-Entität gespeichert. Der folgende Codeausschnitt veranschaulicht die Verarbeitung der Ablehnung im Controller:

```
1 // accept / decline Aktionen (aus POST /ticketInfo)
2 if ("decline".equals(action)) {
```

```

3     System.out.println("Ticket " + id + " declined by user "
4                          + userId + " | Reason: " + reason);
5     myTicket.setAccept(false);
6     myTicket.setDecline(true);
7     myTicket.setReason(reason);
8
9     ticketService.declineTicket(id, reason);
10    ticketService.updateTicket(myTicket);
11    return "redirect:/tickets";
12 }

```

Listing 6.5: Controller – Verarbeitung der Projektleiterentscheidung (Ablehnung)

Die Persistierung der Ablehnung erfolgt zusätzlich gezielt in der Datenzugriffsschicht, indem die zustandsrelevanten Felder gesetzt und die Ablehnungsbegründung gespeichert werden. Der folgende Codeausschnitt aus `TicketDAOImpl.declineTicket(...)` belegt diese technische Umsetzung:

```

1 // Auszug aus TicketDAOImpl.declineTicket(int ticketId, String reason)
2 String sql = "UPDATE Projektticket "
3             + "SET Ablehnen = 1, Annehmen = 0, Begründung = ? "
4             + "WHERE Projektticket_ID = ?";
5
6 jdbcTemplate.update(sql, reason, ticketId);

```

Listing 6.6: DAO – Ablehnung des Tickets (`declineTicket()`)

Die Begründungspflicht ist zusätzlich auf UI-Ebene sichtbar, da der Ablehnungsprozess über einen Dialog mit Eingabefeld für den Grund ausgelöst wird (vgl. Abb. 6.3). Durch die Kombination aus clientseitiger Steuerung (`ticketAction`, Modal-Dialog) und serverseitiger Verarbeitung über dedizierte Methoden wird sichergestellt, dass Zustandsübergänge nachvollziehbar, kontrolliert und fachlich eindeutig umgesetzt sind.

(a) Ansicht: abgelehntes Ticket

(b) Dialog: Ablehnungsgrund

Abbildung 6.3: Ablehnungsprozess in der mobilen Ansicht

6.5.3 Verhinderung unzulässiger Bearbeitungsschritte

Nach einer finalen Entscheidung des Projektleiters erfolgt eine Einschränkung der weiteren Bearbeitung, um den fachlich abgeschlossenen Zustand konsistent beizubehalten. In der vorliegenden Implementierung wird das beschriebene Verhalten durch eine Kombination aus getrennten serverseitigen Verfahrensweisen sowie einer clientseitigen Sperrlogik umgesetzt.

Auf Serverseite wird die Zustandsänderung (Annahme oder Ablehnung) nicht durch eine reine Feldaktualisierung durchgeführt, sondern ausschließlich über die Service-Methoden (`acceptTicket(...)`, `declineTicket(..., reason)`), wodurch sich zustandsverändernde Aktionen deutlich von gewöhnlichen Formularänderungen abgrenzen lassen. Im Falle einer Ablehnung wird der Zustand dauerhaft als abgelehnt persistiert, inklusive der Begründung, sodass das Ticket fachlich als abgeschlossen gilt und im Prozess nicht weitergeführt werden soll.

Zusätzlich wird eine Sperre auf Client-Seite implementiert, die den finalen Zustand unmittelbar in der Benutzeroberfläche darstellt und Eingaben verhindert. Der Status wird in der Ticket-Detailseite aus den Hidden-Fields `accept/decline` abgeleitet. Im Falle einer Ablehnung wird eine vollständige Sperrlogik aktiviert. Der folgende Auszug aus der JavaScript-Funktion `updateFieldStates()` veranschaulicht den zentralen Teil, der bei `declined == true` die Bearbeitung vollständig deaktiviert:

```

1 // Auszug aus updateFieldStates(): Abgelehnt -> komplett gesperrt
2 if (declined) {
3     lockFields(allFields, true, false);

```

```

4     lockMilestones(true);
5     if (submitBtn) submitBtn.disabled = true;
6     $("#acceptBtn, #rejectBtn").hide();
7     return;
8 }

```

Listing 6.7: ticketInfo.jsp – Clientseitige Sperrlogik bei abgelehnten Projekttickets

Durch diese Kombination aus getrennten serverseitigen Zustandsaktionen und einer UI-seitigen Sperrlogik bleibt der Ticket-Lebenszyklus technisch kontrolliert. Zustandswechsel erfolgen ausschließlich über vorgesehene Aktionen, die Ablehnung wird inklusive Begründung dauerhaft persistiert, und nach der finalen Entscheidung wird die Bearbeitung in der Oberfläche in einen schreibgeschützten Zustand überführt (vgl. Abb. 6.3).

6.6 Implementierung der Benachrichtigungen

In Anlehnung an Kapitel 5.4.2 wurde die Benachrichtigungslogik im Projektticket-Modul bewusst nicht in der Controller- oder Service-Schicht umgesetzt, sondern direkt in der Datenbank realisiert. Es wurde sichergestellt, dass Benachrichtigungen unabhängig von der Benutzeroberfläche ausgelöst werden und auch dann zuverlässig erfolgen, wenn Ticketänderungen nicht über die Webanwendung, sondern über andere technische Prozesse entstehen.

Die Implementierung erfolgt durch einen Trigger in Microsoft SQL Server, welcher nach dem Speichern von Ticketdaten automatisch ausgeführt wird. Der Trigger reagiert auf INSERT- und UPDATE-Operationen in der Tabelle Projektticket und prüft dabei, ob sich zustandsrelevante Werte verändert haben, die eine Benachrichtigung erfordern. Sobald eine solche Änderung erkannt wird, wird eine entsprechende E-Mail über die Datenbankfunktion `sp_send_dbmail` versendet.

6.6.1 Entscheidung für eine datenbankseitige Umsetzung

Die Entscheidung für eine datenbankseitige Lösung basiert auf dem Ziel, dass Benachrichtigungen nicht an bestimmte Controller-Methoden oder UI-Abläufe gebunden sind. Statusänderungen, wie beispielsweise «angenommen», «abgelehnt» oder eine Änderung des operativen Freigabestatus, sollten eine Benachrichtigung auslösen, unabhängig davon, wie diese Änderung technisch erfolgt ist. Mittels eines Triggers wird das Ereignis direkt dort verarbeitet, wo der Status dauerhaft gespeichert wird, nämlich in der Datenbank.

6.6.2 Auslöser der Benachrichtigungen

Der Trigger unterscheidet mehrere relevante Ereignisse, indem er den alten Zustand aus der virtuellen Tabelle `deleted` mit dem neuen Zustand aus `inserted` vergleicht. Dadurch

werden Benachrichtigungen ausschließlich bei tatsächlich zustandsrelevanten Änderungen ausgelöst. In der Folge können folgende Benachrichtigungen entstehen:

(1) Ticket wurde neu angelegt (INSERT) Beim Anlegen eines Projekttickets wird, abhängig von den gespeicherten Zuordnungen, entschieden, ob der zuständige Teamleiter oder Projektleiter informiert werden muss. In der praktischen Anwendung wird die jeweils zuständige Rolle – Teamleiter bei Neuanlage ohne Projektzuordnung bzw. Projektleiter nach erfolgter Projektzuordnung – unmittelbar über neu angelegte Tickets benachrichtigt.

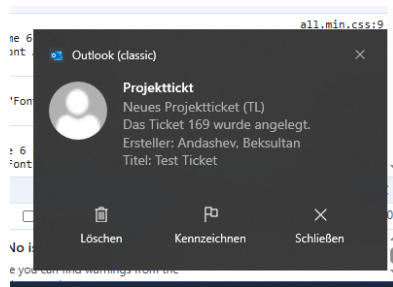
(2) Änderung der Projektleiterentscheidung (Annehmen / Ablehnen) Wenn sich der Status eines Tickets aufgrund einer Annahme oder Ablehnung ändert, wird dies als technisch relevante Statusänderung betrachtet. In diesem Fall versendet der Trigger eine Benachrichtigung an den zuständigen Sachbearbeiter, sodass der Bearbeitungs- bzw. Abschlussstatus unmittelbar erkennbar ist. Im Falle einer Ablehnung wird zusätzlich der gespeicherte Ablehnungsgrund übermittelt.

(3) Änderung des Freigabestatus (Freigabestatus_Ticket) Unabhängig von der Entscheidung des Projektleiters existiert ein operativer Freigabestatus, der den Bearbeitungsstand des Tickets abbildet. Im Falle einer Änderung dieses Statusfeldes wird ebenfalls eine Benachrichtigung ausgelöst. Infolgedessen wird der zuständige Sachbearbeiter automatisch über statusrelevante Änderungen informiert und kann den Fortschritt des Tickets nachvollziehen.

6.6.3 Ziel der Benachrichtigungen

Der Zweck der Benachrichtigungslogik besteht darin, dass die zuständigen Personen (z. B. Teamleiter, Projektleiter oder Sachbearbeiter) automatisch über relevante Änderungen informiert werden. Dadurch wird vermieden, dass neue Anträge oder Statusänderungen nur durch manuelle Überprüfung in der Übersicht entdeckt werden können. Die Kommunikation erfolgt unmittelbar per E-Mail, was eine zeitnahe Reaktion im Bearbeitungsprozess gewährleistet.

Die folgende Abbildung 6.4 veranschaulicht exemplarisch Benachrichtigungen, die während des Prozesses ausgelöst werden:



(a) Outlook-Benachrichtigung bei Ticketanlage (TL)

Neues Projektticket (SB)

Projektticket <projektticket@iamt-grupp
An Beksultan Andashev
Cc Andreas Jubelt

Das Ticket 185 wurde angenommen.
Ersteller: Andashev, Beksultan
Titel: Test Ticket
Beschreibung: Test one

(b) Ticket angenommen

Neues Projektticket (SB)

Projektticket <projektticket@iamt-grupp
An Beksultan Andashev
Cc Andreas Jubelt

Das Ticket 185 hat einen neuen Freigabe_Status.
Freigabestatus_Ticket: abgelehnt
Ersteller: Andashev, Beksultan
Titel: Test Ticket
Beschreibung: Test one

(c) Freigabestatus geändert

Neues Projektticket (PL)

Projektticket <projektticket@iamt-grupp
An Beksultan Andashev
Cc Andreas Jubelt

Das Ticket 185 wurde angelegt.
Ersteller: Andashev, Beksultan
Titel: Test Ticket
Beschreibung: Test one

(d) Ticketanlage (PL)

Projektticket <projektticket@iamt-grupp
An Beksultan Andashev
Cc Andreas Jubelt

Das Ticket 185 hat einen neuen Freigabe_Status.
Freigabestatus_Ticket: genehmigt
Ersteller: Andashev, Beksultan
Titel: Test Ticket
Beschreibung: Test one

(e) Freigabestatus genehmigt

Neues Projektticket (SB)

Projektticket <projektticket@iamt-grupp
An Beksultan Andashev
Cc Andreas Jubelt

Das Ticket 185 wurde abgelehnt.
Ersteller: Andashev, Beksultan
Titel:
Beschreibung:
Grund: Test Ablehnung

(f) Ticket abgelehnt mit Begründung

Abbildung 6.4: E-Mail-Benachrichtigungen im Ticketprozess (Neuanlage, Annahme, Statusänderung, Ablehnung)

6.6.4 Vorteile der Umsetzung

Durch die Trigger-basierte Umsetzung ergeben sich mehrere technische Vorteile:

Zunächst ist festzustellen, dass eine hohe Konsistenz erreicht wird. Dies ist dadurch bedingt, dass Benachrichtigungen direkt mit tatsächlichen Änderungen in der Datenbank verknüpft sind und nicht vom Verhalten der Benutzeroberfläche abhängen. Des Weiteren ist festzustellen, dass die Logik unabhängig von der Controller- und Service-Schicht bleibt. Dies hat zur Folge, dass das System auch bei zukünftigen Erweiterungen oder alternativen Zugriffsmethoden stabil bleibt. Die Gewährleistung einer hohen Zuverlässigkeit erfolgt durch eine automatisierte Überprüfung und Verarbeitung jeder gespeicherten Statusänderung.

Insgesamt ergänzt die datenbankseitige Benachrichtigungslogik den Ticket-Lebenszyklus um eine umfassende Kommunikationsebene und stellt sicher, dass zentrale Prozessschritte wie Freigabe, Ablehnung oder Statuswechsel technisch nachvollziehbar und automatisch begleitet werden.

6.7 Sicherheit und Zugriffskontrolle

Bezüglich des Abschnitts 5.3 wird die Zugriffskontrolle im Projektticket-Modul zweistufig umgesetzt. Die erste Stufe besteht aus dem Schutz des Zugriffs auf das Modul auf der Serverseite mithilfe von Spring Security. Die zweite Stufe betrifft die kontextbezogene

ne Bearbeitbarkeit einzelner Ticketdaten, die nicht als Sicherheitsregel modelliert wird, sondern sich aus der Beziehung des Benutzers zum Ticket (Ersteller/Teamleiter/Projektleiter) und dem Status des Tickets ergibt. Auf diese Weise wird klar zwischen «Zugriff auf das Modul» und «Bearbeitung innerhalb des Formulars» unterschieden.

6.7.1 Serverseitige Zugriffssicherung

Die Zugriffssicherung basiert auf Spring Security und setzt eine erfolgreiche Authentifizierung voraus, bevor Ticketseiten genutzt werden können. Innerhalb der Controller-Schicht wird der aktuell angemeldete Benutzer aus dem Security-Kontext ermittelt und anschließend zur Kontextbildung verwendet (z. B. zur Bestimmung, ob der Benutzer Ersteller, Teamleiter oder Projektleiter im Ticketkontext ist). Dadurch werden serverseitige Verarbeitungsschritte eindeutig einem authentifizierten Benutzer zugeordnet und können nicht anonym ausgelöst werden.

Der folgende Codeausschnitt zeigt die technische Ermittlung der Benutzeridentität über Spring Security:

```
1 int userId = loginService.getUserByUsername(  
2     ((UserDetails) SecurityContextHolder.getContext()  
3         .getAuthentication().getPrincipal()).getUsername()  
4 ).getIAMT_Nutzer_ID();
```

Listing 6.8: Controller - Ermittlung des eingeloggten Benutzers über Spring Security

Auf Basis dieser Identität werden die Ticketdaten geladen und der Bearbeitungskontext vorbereitet. Damit wird sichergestellt, dass serverseitige Verarbeitungsschritte stets einem authentifizierten Benutzer zugeordnet sind und nicht anonym ausgelöst werden können.

6.7.2 Abgrenzung: Sicherheit vs. UI-Steuerung

Die Sicherheit auf der Serverseite beantwortet die Frage, wer grundsätzlich Zugriff auf das Modul oder die Ticket-Seiten hat (Authentifizierung und Autorisierung auf Endpunkt-Ebene). Hierbei wird verhindert, dass nicht angemeldete oder nicht berechtigte Benutzer auf Ticketfunktionen zugreifen können.

Die rollen- und statusabhängige Feldsteuerung beantwortet hingegen die Frage, wie ein Benutzer innerhalb der One-Page-Form geführt wird (z. B. schreibgeschützt, editierbar oder ausgeblendet). Diese zweite Ebene stellt keine klassische Zugriffssicherheit dar, sondern dient der Benutzerführung und Prozesssteuerung innerhalb des Formulars. Sie ist daher Bestandteil der Benutzerinteraktion und wird in den Abschnitten zur Formularlogik und Statusverwaltung behandelt (siehe Kapitel 6.4 und 6.5).

Damit entsteht durch das Zusammenspiel beider Ebenen ein konsistentes Schutzkonzept: Spring Security verhindert unbefugten Zugriff auf das Modul, während die kontextbezo-

gene Steuerung innerhalb der Anwendung sicherstellt, dass Ticketinhalte je nach Rolle und Status nur in den vorgesehenen Bearbeitungssituationen geändert werden können.

6.8 Zusammenfassung der Implementierung

Die Implementierung des Projektticket-Moduls wurde als Erweiterung in die bestehende Systemarchitektur integriert und folgt dem vorgesehenen MVC-Ansatz mit klarer Trennung zwischen Präsentation, Controller, Service und Persistenzschicht. Die Datenhaltung wurde um eine Ticketstruktur sowie eine separate Meilensteinverwaltung erweitert, sodass sowohl Stammdaten als auch projektbezogene Abrechnungs- und Fortschrittsinformationen konsistent persistiert werden können.

Die Benutzerinteraktion wurde als formularbasierte One-Page-Oberfläche umgesetzt, wobei dynamische Bereiche wie die Meilenstein-Tabelle clientseitig gesteuert und serverseitig verarbeitet werden. Die rollen- und statusabhängige Benutzersteuerung wird durch eine Kombination aus serverseitig aufbereiteten Kontextinformationen und clientseitiger Feldsteuerung implementiert. Somit kann die Bearbeitbarkeit und Sichtbarkeit einzelner Bereiche situationsabhängig angepasst werden.

Der Ticket-Lebenszyklus wird technisch durch Statusfelder und eine klar getrennte Verarbeitungslogik abgebildet, wodurch Annahme- und Ablehnungsentscheidungen nachvollziehbar und kontrolliert verarbeitet werden. In Ergänzung dazu wurde die Implementierung von Benachrichtigungen auf der Datenbankseite vorgenommen. Dadurch ist eine automatisierte Kommunikation von statusrelevanten Änderungen an die zuständigen Rollen möglich, und dies unabhängig von der Benutzeroberfläche. Damit sind die in Kapitel 5 beschriebenen konzeptionellen Entscheidungen konsistent technisch umgesetzt. Das folgende Kapitel dient der Überprüfung und Bewertung der Lösung anhand von Tests und Ergebnissen.

7 Evaluierung und Ergebnisse

7.1 Evaluierung

In diesem Kapitel erfolgt eine Bewertung der entwickelten Erweiterung des Ticketsystems anhand zuvor festgelegter Anforderungen. Das Ziel besteht darin, die Umsetzung der funktionalen Anforderungen (F1–F5) sowie der nicht-funktionalen Anforderungen (NF1–NF3) einer systematischen Überprüfung zu unterziehen und die Ergebnisse in einer nachvollziehbaren Form darzustellen.

Die Evaluierung erfolgt auf Grundlage praktischer Testszenarien, welche typische Abläufe bei der Bearbeitung von Projekttickets abbilden. Dabei werden insbesondere die Ticketverwaltung, die Statusverwaltung, die Rollenlogik und die Meilensteinprüfung untersucht. Zusätzlich werden Auffälligkeiten und Hinweise aus einer internen Testphase dokumentiert, um mögliche Verbesserungspotenziale zu definieren.

7.1.1 Testaufbau und Vorgehensweise

Zur Bewertung des implementierten Moduls für Projekttickets wurde ein manueller Testansatz gewählt. Die Tests orientierten sich an den in Kapitel 4 definierten Anforderungen und wurden auf der Grundlage praktischer Anwendungsszenarien durchgeführt. Ziel war es, sowohl die Funktionalität als auch die Integration des Projektticket-Moduls in die bestehende Webanwendung zu überprüfen.

Die Evaluierung wurde sowohl auf Desktop- als auch auf mobilen Endgeräten durchgeführt. Für die mobile Nutzung wurden ein iPhone sowie ein Android-Smartphone verwendet. Für die Prüfung der Bedienbarkeit und Funktionalität auf unterschiedlichen Plattformen wurden die Browser Safari (iOS) und Google Chrome (Android/Windows) verwendet.

Die Testdurchführung erfolgte schrittweise anhand vordefinierter Testfälle, die aus den zuvor definierten Anforderungen abgeleitet wurden. Hierzu zählen insbesondere die Erstellung neuer Projekttickets, die Bearbeitung durch autorisierte Rollen sowie Statusänderungen, einschließlich Annahme und Ablehnung. Darüber hinaus wurden Negativtests durchgeführt, um sicherzustellen, dass nicht autorisierte Benutzerrollen keine unzulässigen Aktionen ausführen können.

Ein weiterer wichtiger Aspekt der Bewertung war die Überprüfung der rollenbasierten Zugriffskontrolle und der dynamischen Steuerung von Eingabefeldern. Dabei wurde die korrekte Aktivierung oder Deaktivierung bestimmter Felder in Abhängigkeit vom Status und der Rolle überprüft. Darüber hinaus wurde die korrekte Übermittlung von Benachrichti-

gungen bei entsprechenden Statusänderungen oder Annahme- und Ablehnungsprozessen überprüft.

7.1.2 Testergebnisse anhand der Anforderungen

Im Folgenden werden die Ergebnisse der Evaluierung anhand der in Kapitel 4 definierten funktionalen Anforderungen (F1–F5) sowie der nicht-funktionalen Anforderungen (NF1–NF3) dargestellt. Für jede Anforderung wurden praxisnahe Testfälle durchgeführt und das Ergebnis in «erfüllt» oder «teilweise erfüllt» klassifiziert. Die Einordnung wurde auf Grundlage der durchgeführten Tests vorgenommen und in Abstimmung mit dem Betreuer aus dem Unternehmen sowie potenziellen Anwendern der Webanwendung abgestimmt.

Anforderung	Testfall / Use-Case	Erwartetes Ergebnis	Ergebnis
F1 – Projektticket-Erstellung/-Bearbeitung	Projektticket erstellen, speichern und anschließend in der Übersicht suchen sowie erneut öffnen. Ticketdaten bearbeiten und Änderungen speichern.	Projekttickets können erstellt, angezeigt, gesucht und bearbeitet werden.	erfüllt
F2 – Rollenmodell / Berechtigungen	Tests mit unterschiedlichen Benutzerrollen (z.B. Ersteller, Sachbearbeiter, Projektleiter, Teamleiter) durchführen. Nicht berechnigte Rollen versuchen Aktionen wie Statuswechsel, Freigabe oder Ablehnung.	Berechtigungen sind rollenbasiert umgesetzt und unzulässige Aktionen werden zuverlässig verhindert.	erfüllt
F3 – Projektticket-Statusmanagement inkl. Ablehnung	Statuswechsel durchführen sowie Ticketfreigabe und Ticketablehnung testen. Ablehnungsprozess inkl. Erfassung und Anzeige der Ablehnungsinformation prüfen.	Statusmanagement funktioniert entsprechend dem vorgesehenen Workflow. Ablehnungen werden korrekt verarbeitet und nachvollziehbar dokumentiert.	erfüllt
F4 – Dynamische Feldsteuerung abhängig von Status/Rolle	Pflichtfelder und Feldlogik abhängig von Rolle und Status prüfen (z.B. aktiv/deaktiviert, sichtbare Felder).	Felder werden abhängig vom Status und der Rolle korrekt gesteuert.	erfüllt
F5 – Meilensteine / Abschluss bei 100%	Meilensteine erfassen und prüfen, ob die Abschlusslogik nur bei 100% Gesamtsumme zulässt.	Ein Abschluss ist nur möglich, wenn die Meilensteine insgesamt 100% ergeben.	erfüllt
NF1 – Mobile Nutzung (Smartphone / VPN-Szenario)	Zugriff über Smartphone-Browser testen und zentrale Funktionen (z.B. Ticket anlegen, Ticketübersicht) ausführen.	Das Modul ist mobil bedienbar und erfüllt die Anforderung der Nutzung über mobile Endgeräte.	teilweise erfüllt
NF2 – Benachrichtigungen bei Statusänderungen	Freigabe, Ablehnung und Statusänderungen durchführen und prüfen, ob Benachrichtigungen ausgelöst werden.	Benachrichtigungen werden bei relevanten Änderungen generiert und an die vorgesehenen Rollen weitergegeben.	erfüllt
NF3 – Integration ohne grundlegende Architekturänderung	Überprüfung der Systemintegration innerhalb der bestehenden Architektur (Spring MVC, bestehendes Rollenmodell, vorhandene Datenbankstruktur).	Das Modul ist ohne grundlegende Änderungen an der bestehenden Architektur integrierbar.	erfüllt

Tabelle 7.1: Übersicht der durchgeführten Tests und Ergebnisse

Neben der tabellarischen Auswertung wurden ausgewählte Testergebnisse exemplarisch anhand von Screenshots dokumentiert. Diese dienen als Nachweis für die korrekte Umsetzung zentraler Anforderungen.

Die Anforderung F5 wurde erfolgreich bestätigt. Im Test wurde geprüft, ob ein Ticket nur dann abgeschlossen werden kann, wenn die Summe der Meilenstein-Prozente exakt 100 % beträgt. Eine unvollständige oder fehlerhafte Meilensteinverteilung verhindert den Abschluss und es erfolgt eine entsprechende Hinweismeldung (vgl. Abbildung 7.1).

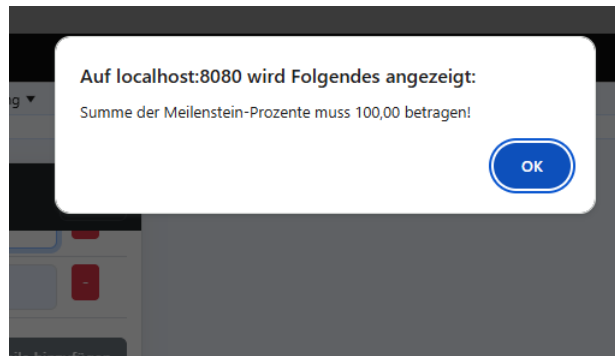


Abbildung 7.1: Validierung der Meilenstein-Prozente (100%-Prüfung)

Auch die Anforderung F3 konnte durch die Testausführung bestätigt werden. Im Falle einer Ablehnung wird diese im Ticket nachvollziehbar dargestellt, sodass der Status und der ablehnende Benutzer ersichtlich sind. Dadurch wird eine transparente Kommunikation innerhalb des Bearbeitungsprozesses unterstützt (vgl. Abbildung 7.2).

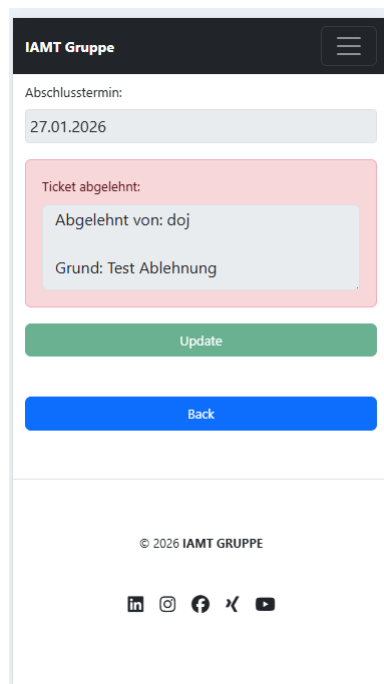


Abbildung 7.2: Darstellung einer Ticketablehnung (Status und Ablehnender sichtbar)

Die Evaluierung ergibt, dass alle funktionalen sowie der Großteil der nicht-funktionalen Anforderungen durch die implementierte Lösung erfüllt werden. Die Anforderung NF1 (mobile Nutzung) wurde als teilweise erfüllt bewertet, da die Anwendung grundsätzlich auf mobilen Geräten genutzt werden kann, jedoch je nach Plattform und Browser einzelne Unterschiede in der Darstellung oder Bedienung auftreten können. Im Rahmen der durchgeführten Tests wurde eine stabile Nutzung unter Android (Chrome) festgestellt, während unter iOS (Safari) in Einzelfällen Darstellungsabweichungen und Fehlfunktionen

einzelner Funktionen auftreten. Insgesamt ist die entwickelte Erweiterung geeignet, den Projektticket-Prozess innerhalb der bestehenden Webanwendung zu unterstützen und die mobile Nutzung sowie die rollenbasierte Bearbeitung zu ermöglichen.

7.1.3 Erweiterungsvorschläge und Auffälligkeiten aus der Testphase

Im Rahmen der Evaluierung wurde zusätzlich zur Anforderungsprüfung eine interne Testphase durchgeführt, in der Anwenderfeedback und Auffälligkeiten systematisch erfasst wurden. Ziel war es, mögliche Fehlerquellen und Optimierungspotenziale frühzeitig zu erkennen und die Bedienbarkeit des Projektticket-Moduls im praktischen Einsatz weiter zu verbessern. Die identifizierten Punkte wurden in einer Revisionsliste dokumentiert, die den Titel «Revision – Auffälligkeiten und Hinweise» trägt (vgl. Abbildung 7.3).

Revision 2 - Auffälligkeiten und Hinweise

1. Auswahlfelder funktionieren nicht mit allen Browsern/Telefonen - spez. iPhone plus Safari
2. nach Suche in Liste ist Kopfzeile verschwunden und bleibt verschwunden
3. Filterfunktion in Übersichtsliste fehlt
4. ~~Vorauswahl Projekte nicht umgesetzt~~
5. Projekttitle zu Projektnummer hinzufügen
6. Profilerstellung für Ersteller mit Vorauswahl umsetzen (Standardtickets die sich oft wiederholen)
 - a. Titel
 - b. Beschreibung
 - c. Stunden
 - d. Team
 - e. Projektnummer
7. Profilerstellung für Projektleiter mit restlichen Felder
8. Titel bitte nach oben schieben
9. ~~Teamauswahl optional mit Vorauswahl mit entsprechender Zuordnung der Teamzugehörigkeit~~
10. Bedienung mit iPhone schlecht - bitte bei [REDACTED] melden
11. ~~Ticketablehnung erst nach Projektzuordnung möglich, bitte auf direkte Ablehnung ändern~~
12. ~~Emails scheinen nicht flüssig generiert zu werden: bitte prüfen und ändern~~
13. IT intern intensiver vortesten
14. ~~kleineres Testteam bilden mit 2-SB + 2-PL und 2-TL + IT~~
15. ~~Ablehnungsgrund im Ticket dokumentieren, sichtbar machen und per Email versenden, inkl. Name des Abrechners~~
16. ~~automatische Enter-Belegung entfernen~~

Abbildung 7.3: Auffälligkeiten und Hinweise

Die Rückmeldungen bezogen sich hauptsächlich auf die Benutzeroberfläche, die Bedienung auf mobilen Endgeräten sowie kleinere funktionale Anpassungswünsche im Bereich der Ticketübersicht. Ein Großteil dieser Punkte konnte im weiteren Verlauf umgesetzt bzw. als behoben eingestuft werden. In der Folge wurde sowohl die Stabilität als auch die Nutzbarkeit des Moduls im Arbeitsalltag verbessert.

Neben diesen Korrekturen wurden auch Erweiterungsvorschläge identifiziert, die nicht zwingend für den initialen Funktionsumfang erforderlich sind, jedoch einen zusätzlichen Mehrwert für wiederkehrende Prozesse bieten können. Hierzu zählen insbesondere Profil-

bzw. Vorlagenfunktionen zur schnelleren Erstellung häufig vorkommender Projekttickets für bestimmte Benutzergruppen.

7.2 Fazit und Ausblick

Ziel dieser Arbeit war die Entwicklung und Integration eines Moduls zur Verwaltung von Projekttickets innerhalb der bestehenden Webanwendung. Den Mittelpunkt der Betrachtung bildeten insbesondere die rollenbasierte Bearbeitung, die strukturierte Abbildung von Status- und Genehmigungsprozessen sowie die Unterstützung der mobilen Nutzung. Im Rahmen der Implementierung umgesetzte Funktionen wurden anhand der in Kapitel 4 definierten Anforderungen evaluiert.

Die Ergebnisse der durchgeführten Tests zeigen, dass die funktionalen Anforderungen erfüllt wurden und die wesentlichen Prozesse im Projektticket-Workflow zuverlässig abgebildet werden können. Hierzu zählen das Erstellen und Bearbeiten von Projekttickets, die Nachverfolgbarkeit von Genehmigungs- und Ablehnungsprozessen sowie die Validierung von Meilensteinen. Im Bereich der nicht-funktionalen Anforderungen konnte die Lösung grundsätzlich bestätigt werden. Allerdings wurden Einschränkungen bei der mobilen Nutzung auf iOS-Geräten in Kombination mit Safari festgestellt, was dazu führte, dass die entsprechende Anforderung insgesamt als teilweise erfüllt bewertet wurde.

Zum Zeitpunkt der Abgabe befindet sich das Projektticket-Modul weiterhin in einer erweiterten Testphase. Die Einführung und Nutzung im Arbeitsalltag führt erfahrungsgemäß zu weiterem Anwenderfeedback, sodass mit zusätzlichen Optimierungswünschen sowie neuen Anforderungen zu rechnen ist. Aus diesem Grund sind fortlaufende Tests sowie eine iterative Weiterentwicklung vorgesehen, um die Stabilität, Benutzerfreundlichkeit und den Funktionsumfang zu erhöhen.

Für die zukünftige Weiterentwicklung ergeben sich mehrere sinnvolle Ansätze. Hierzu zählen insbesondere Verbesserungen der plattformübergreifenden Bedienbarkeit, insbesondere im Zusammenspiel mit iOS/Safari, sowie Erweiterungen der Ticketübersicht durch zusätzliche Filter- und Suchfunktionen. Darüber hinaus können technische Erweiterungen, wie etwa serverseitige Validierung und erweitertes Logging, zur weiteren Erhöhung der Systemqualität beitragen. Zusätzlich bieten Profil- bzw. Vorlagenfunktionen Potenzial, um wiederkehrende Ticketarten schneller zu erstellen und Arbeitsprozesse effizienter zu gestalten.

Insgesamt stellt die entwickelte Lösung eine funktionale und erweiterbare Basis dar, um Projekttickets strukturiert zu verwalten und den vorgesehenen Bearbeitungsprozess innerhalb der Webanwendung nachhaltig zu unterstützen.

Quellenverzeichnis

- [Axe19] Axelos. *ITIL Foundation – ITIL 4 Edition*. TSO (The Stationery Office), 2019. ISBN: 978-0-11-331607-6.
- [Bud13] Raluca Budiu: *Interaction Cost: Definition*, 2013, Online im Internet: <https://www.nngroup.com/articles/interaction-cost-definition/>, Zugriffen am 25.11.2025.
- [Bud16] Raluca Budiu: *Mobile-Dedicated, Responsive, Adaptive, or Desktop Site?*, 2016, Online im Internet: <https://www.nngroup.com/articles/mobile-vs-responsive/>, Zugriffen am 09.01.2026.
- [Bud17] Raluca Budiu: *Wizards: Definition and Design Recommendations*, 2017, Online im Internet: <https://www.nngroup.com/articles/wizards/>, Zugriffen am 25.11.2025.
- [CB15] Thomas Connolly und Carolyn Begg. *Database Systems: A Practical Approach to Design, Implementation and Management*. Pearson, 2015. ISBN: 978-0-13-294326-0.
- [Lau16] Page Laubheimer: *Cards: UI-Component Definition*, 2016, Online im Internet: <https://www.nngroup.com/articles/cards-component/>, Zugriffen am 26.11.2025.
- [Lea] LeadGen App: *Mehrstufige Formulare*, Online im Internet: <https://leadgenapp.io/de/mehrstufige-Formulare/>, Zugriffen am 23.11.2025.
- [LL20] Kenneth C. Laudon und Jane P. Laudon. *Management Information Systems: Managing the Digital Firm*. Pearson, 2020. ISBN: 978-1-292-40328-1.
- [Moz] Mozilla Foundation: *Dynamic scripting with JavaScript*, Online im Internet: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting, Zugriffen am 24.11.2025.
- [ND86] Donald A. Norman und Stephen W. Draper, Hrsg. *User Centered System Design: New Perspectives on Human-Computer Interaction*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1986. ISBN: 0-89859-781-1.
- [Nie] Nielsen Norman Group: *Articles on Form Design and Usability*, Online im Internet: <https://www.nngroup.com/articles/>, Zugriffen am 21.11.2025.
- [Nie07] Jakob Nielsen: *Complex Application Design*, 2007, Online im Internet: <https://www.nngroup.com/articles/complex-application-design/>, Zugriffen am 09.01.2026.

- [Nie14] Nielsen Norman Group: *Accordions on Desktop: When and How to Use*, 2014, Online im Internet: <https://www.nngroup.com/articles/accordions-on-desktop/>, Zugegriffen am 28.11.2025.
- [Nie15] Nielsen Norman Group: *Accordions on Mobile*, 2015, Online im Internet: <https://www.nngroup.com/articles/mobile-accordions/>, Zugegriffen am 25.11.2025.
- [Nie16] Nielsen Norman Group: *6 Tips for Successful Personalization*, 2016, Online im Internet: <https://www.nngroup.com/articles/personalization/>, Zugegriffen am 09.01.2026.
- [Nie23] Nielsen Norman Group: *The Negative Impact of Mobile-First Web Design on Desktop*, 2023, Online im Internet: <https://www.nngroup.com/articles/content-dispersion/>, Zugegriffen am 26.11.2025.
- [Nie25] Nielsen Norman Group: *4 Principles to Reduce Cognitive Load in Forms*, 2025, Online im Internet: <https://www.nngroup.com/articles/4-principles-reduce-cognitive-load/>, Zugegriffen am 26.11.2025.
- [Nor13] Don Norman. *The Design of Everyday Things*. Basic Books, 2013. ISBN: 978-0-465-07299-6.
- [Ora20] Oracle Corporation: *JavaServer Pages (JSP) Documentation*, 2020, Online im Internet: <https://docs.oracle.com/javaee/5/tutorial/doc/bnajo.html>, Zugegriffen am 21.11.2025.
- [Pop25] Diana Popa: *Single-page vs. Multi-step Forms: Which Strategy Boosts Conversion Rates?*, 2025, Online im Internet: <https://www.123formbuilder.com/blog/single-page-vs-multi-step-forms>, Zugegriffen am 25.11.2025.
- [PR24] Sara Paul und Maria Rosala: *The Wizard of Oz Method in UX*, 2024, Online im Internet: <https://www.nngroup.com/articles/wizard-of-oz/>, Zugegriffen am 09.01.2026.
- [Tid20] Jenifer Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. 3. Aufl. O'Reilly Media, 2020. ISBN: 978-1-492-05196-1.
- [TK24] Samhita Tankala und Alita Kendrick: *Design-Pattern Guidelines: Study Guide*, 2024, Online im Internet: <https://www.nngroup.com/articles/design-pattern-guidelines/>, Zugegriffen am 09.01.2026.
- [W3C25] W3C: *Mobile Web Best Practices*, 2025, Online im Internet: <https://www.w3.org/TR/mobile-bp/>, Zugegriffen am 23.11.2025.
- [W3S] W3Schools: *HTML Introduction*, Online im Internet: https://www.w3schools.com/html/html_intro.asp, Zugegriffen am 23.11.2025.
- [Wal18] Craig Walls. *Spring in Action*. Manning, 2018. ISBN: 978-1-61729-494-5.

- [Web] Web.dev: *Network connections are unreliable*, Online im Internet: <https://web.dev/articles/network-connections-unreliable>, Online; Zugriff am 11.01.2026.
- [Wes19] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2019. ISBN: 978-3-662-59431-5.

Übersicht verwendeter Hilfsmittel:

- ChatGPT (<https://chatgpt.com/>): Wurde zur Generierung von Ideen und zur präziseren Formulierung von Sätzen auf Basis des vom Autor bereitgestellten Textes verwendet.
- Gemini (<https://gemini.google.com/>): Wurde für die gleichen Aufgaben wie ChatGPT eingesetzt, also zur Ideenfindung, Formulierungshilfe und Grammatiküberprüfung.



Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig ☒
angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und
Hilfsmittel benutzt habe, und dass die Arbeit in gleicher oder ähnlicher Form in noch
keiner anderen Prüfung vorgelegen hat. Mir ist bewusst, dass ich Autor/in der
vorliegenden Arbeit bin und volle Verantwortung für den Text trage.

Ich erkläre, dass ich wörtlich oder sinngemäß aus anderen Werken – dazu gehören ☒
auch Internetquellen – übernommene Inhalte als solche kenntlich gemacht und die
entsprechenden Quellen angegeben habe.

Mir ist bewusst, dass meine Arbeit auf Plagiate überprüft werden kann. Mir ist ☒
bekannt, dass es sich bei der Abgabe eines Plagiats um ein schweres akademisches
Fehlverhalten handelt und dass Täuschungen nach der für mich gültigen
Prüfungsordnung geahndet werden.

Zusätzlich versichere ich, dass ich auf künstlicher Intelligenz (KI) basierende ☒
Werkzeuge nur in Absprache mit den Prüfern verwendet habe. Dabei stand meine
eigene geistige Leistung im Vordergrund, und ich habe jederzeit den Prozess
steuernd bearbeitet.

Diese Werkzeuge habe ich im Quellenverzeichnis in der Rubrik „Übersicht ☒
verwendeter Hilfsmittel“ mit ihrem Produktnamen und einer Übersicht des im
Rahmen dieser Prüfungs-/Studienarbeit genutzten Funktionsumfangs unter Angabe
der Textstelle in der Arbeit vollständig aufgeführt.

Ich versichere, dass ich keine KI-basierten Tools verwendet habe, deren Nutzung die ☒
Prüfer explizit schriftlich ausgeschlossen haben. Ich bin mir bewusst, dass die
Verwendung von Texten oder anderen Inhalten und Produkten, die durch KI-basierte
Tools generiert wurden, keine Garantie für deren Qualität darstellt.

Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter ☒
Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die
KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße
gegen das Datenschutz- und Urheberrecht oder Plagiate.

04.02.2026

Ort, Datum

Unterschrift