

# **Todo list**

Überprüfen, ob Prozessmodellierung und Anforderungen tauschen, ggf auch Ausarbeitung User-Workflow noch vor Anforderungen . . . . .	11
Umformulieren und richtig anordnen . . . . .	80
Kapitel Fazit schreiben . . . . .	84
Komplette Arbeit nochmal lesen . . . . .	85



# **Cloud-Computing - Translator Web-App**

**Cloud Computing 2 Projekt**

**Bachelor of Science**

des Studiengangs IT-Automotive

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

**David Raisch**

07.04.2025

**Bearbeitungszeitraum**  
**Matrikelnummer, Kurs**  
**Dozent**

24.02.2025 - 07.04.2025  
3768392, TINF22ITA  
Nils Riekers

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>III</b>
<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>Code-Abschnitte</b>	<b>VII</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Aufgabenstellung</b>	<b>2</b>
<b>3 Theoretische Grundlagen</b>	<b>3</b>
3.1 Anforderungen . . . . .	3
3.1.1 Anforderungsanalyse . . . . .	3
3.1.2 Funktionale Anforderungen . . . . .	4
3.1.3 Nicht-funktionale Anforderungen . . . . .	6
3.2 Prozessmodellierung . . . . .	7
3.2.1 Ereignisgesteuerte Prozesskette . . . . .	7
3.2.2 Business Process Model and Notation . . . . .	9
3.3 Ausarbeitung User-Workflow . . . . .	12
3.3.1 Grob-Ablauf . . . . .	12
3.3.2 Rollenmanagement . . . . .	21
3.3.3 Löschen von Prozessen . . . . .	22
3.3.4 Benachrichtigungen . . . . .	24
3.4 Wahl der Anwendungsart . . . . .	25
3.5 Wahl der Applikation-Technologie . . . . .	28
3.5.1 Frontend-Technologien . . . . .	28
3.5.2 Backend-Technologien . . . . .	29
3.5.3 Datenbank-Technologie . . . . .	29
3.5.4 Fazit . . . . .	30
<b>4 Implementierung</b>	<b>31</b>
4.1 Überblick und Kontext der Implementierung . . . . .	31
4.2 Login . . . . .	32
4.3 Start-Seite des Process-Managers . . . . .	38
4.4 Funktionen des Admins . . . . .	40
4.4.1 Verwaltung der Projekte . . . . .	40
4.4.2 Verwaltung der Benutzer . . . . .	45
4.5 Funktionen des Benutzers . . . . .	51
4.5.1 Benutzerinformationen . . . . .	51

4.5.2 Benachrichtigungen . . . . .	55
4.5.3 Prozesse verwalten . . . . .	59
4.5.4 Prozesse ausführen . . . . .	73
4.5.5 Docker-Bereitstellung der Projekt-Komponenten . . . . .	82
<b>5 Fazit und Ausblick</b>	<b>84</b>
<b>Anhang</b>	<b>A</b>
<b>Glossar</b>	<b>P</b>

# Abkürzungsverzeichnis

<b>BPMN</b>	Buisness Process Model and Notation
<b>EPK</b>	Ereignisgesteuerte Prozesskette
<b>IEEE</b>	Institute of Electrical and Electronic Engineers
<b>IT</b>	Informations-Technologie
<b>ID</b>	Identifikationsnummer
<b>OMG</b>	Object Management Group
<b>RBAC</b>	Role-based Access Control
<b>XOR</b>	exklusives Oder

# Abbildungsverzeichnis

3.1	Beispiel für EPK-Aufbau . . . . .	8
3.2	Beispiel für BPMN-Aufbau . . . . .	10
3.3	Login-Fenster des ProcessManagers . . . . .	13
3.4	Startseite des ProcessManagers . . . . .	13
3.5	Drag-and-Drop- und Code-basierte Methode . . . . .	14
3.6	Interface für das Auswählen eines Prozesses . . . . .	17
3.7	Interface bei einem gestarteten Prozess . . . . .	17
3.8	Zusatzinforamtionen eines Events . . . . .	18
3.9	Prozess ausführen mit Zugriffsmanagement . . . . .	18
3.10	Interface für das Bearbeiten eines Prozesses . . . . .	19
3.11	Abfrage für Löschen eines Prozesses . . . . .	19
3.12	Szenarien beim Löschen von Prozessen . . . . .	20
4.1	Die Projektstruktur des Process-Managers . . . . .	31
4.2	Die Login-Seite des Process-Managers . . . . .	33
4.3	Die Überprüfung der Get-Abfrage der Benutzer mit Postman . . . . .	38
4.4	Die Benutzer Start-Seite . . . . .	39
4.5	Die Navigationsleiste von Administratoren . . . . .	40
4.6	Das Formular zur Erstellung neuer Projekte . . . . .	41
4.7	Die Fehlernachricht des Projekt-Erstellungsprozess bei ungültiger Eingabe .	41
4.8	Die Tabelle der erstellten Projekten . . . . .	42
4.9	Das seperate Fenster mit Projekt-Informationen . . . . .	43
4.10	Das Formular zum Bearbeiten eines erstellten Projekts . . . . .	43
4.11	Das Formular zur Erstellung neuer Benutzer . . . . .	45
4.12	Die Fehlernachricht des Benutzer-Erstllungsprozess bei ungültiger Eingabe	46
4.13	Die Tabelle der erstellten Benutzer . . . . .	48
4.14	Das Formular zum Bearbeiten eines erstellten Benutzers . . . . .	49
4.15	Das sepereate Eingabefeld zum Password ändern . . . . .	50
4.16	Die Navigationsleiste von Benutzer . . . . .	51
4.17	Das Dropdown-Feld mit den Benutzerinformationen . . . . .	51
4.18	Die Passwortänderungsseite für den Benutzer . . . . .	52
4.19	Die Benachrichtigung-Seite für alle Benutzer . . . . .	55
4.20	Eine Benachrichtigung des Typs Anfrage . . . . .	56

4.21 Eine Benachrichtigung des Typs Anfrage . . . . .	56
4.22 Übersicht über die ManageProcess-Seite . . . . .	59
4.23 BPMN-Editor mit geladenem Prozess . . . . .	61
4.24 Das Prozess-Informationsfenster eines Managers . . . . .	61
4.25 Das Fenster für die Zuweisung von Role und Beschreibung eines Elements .	62
4.26 Die Fehlernachricht bei fehlenden Rollen und Beschreibungen . . . . .	63
4.27 Die Übersicht der Änderungen im Modal . . . . .	64
4.28 Die Benachrichtigung für den Änderungsantrag . . . . .	65
4.29 Der Änderungsantrag aus Sicht des Managers . . . . .	65
4.30 Der Änderungsantrag bei Erstellen eines neuen Prozesses . . . . .	66
4.31 Die negative Rückmeldung des Änderungsantrags . . . . .	66
4.32 Die Prozess-Liste aus Sicht eines Managers . . . . .	67
4.33 Die Bestätigungsfrage vor dem Löschen eines Prozesses . . . . .	68
4.34 Die Prozess-Liste aus Sicht eines Employee . . . . .	69
4.35 Die Anfrage für das Löschen des Prozesses . . . . .	69
4.36 Das Modal-Fenster für die Anfrage zum Löschen . . . . .	70
4.37 Das Prozess-Informations Feld für die Anfrage zum Löschen . . . . .	70
4.38 Die postive Rückmeldung nach Anfrage zum Löschen . . . . .	71
4.39 Die vollständige Execute Process-Seite . . . . .	73
4.40 Die Execute Process-Seite mit angeklickenden Prozess . . . . .	74
4.41 Die Execute Process-Seite mit angeklickender Instanz . . . . .	75
4.42 Die Ansicht eines Employee für eine Genehmigungsanfrage . . . . .	77
4.43 Die Benachrichtigung für die Genehmigungsanfrage . . . . .	77
4.44 Die Ansicht der Genehmigungsentscheidung . . . . .	78
4.45 Die Rückmeldung auf die Genehmigungsanfrage . . . . .	78
4.46 Das Element Informations-Feld . . . . .	79
4.47 Die Seite mit allen archivierten Instanzen . . . . .	80
4.48 Der Mini-Kalender zur Datum-Auswahl . . . . .	81
4.49 Hinweis auf aktive Instanzen am Filter-Symbol . . . . .	81
4.50 Hinweis auf keine passende Instanzen mit den gewählten Filter . . . . .	82
A1 Fehlernachricht beim Login – leeres Login-Feld . . . . .	A
A2 Fehlernachricht beim Login – falsche Anmeldedaten . . . . .	A
A3 Die Admin Start-Seite . . . . .	D
A4 Die vollständige Projektverwaltungs-Seite . . . . .	E
A5 Sicherheitsabfrage vor dem Löschen eines erstellten Projekts . . . . .	E
A6 Die vollständige Benutzerverwaltungs-Seite . . . . .	F
A7 Sicherheitsabfrage vor dem Löschen eines erstellten Benutzers . . . . .	F

A8 Fehlernachricht aufgrund eines falschen aktuellen Passwort . . . . .	G
A9 Fehlernachricht aufgrund nicht übereinstimmende neue Passwörter . . . . .	H
A10 Erfolgreiche Änderung des Passworts durch den Benutzer . . . . .	I
A11 Das Prozess-Informationsfenster eines Benutzers . . . . .	M
A12 Das Prozess-Informationsfenster eines Managers mit Änderungen . . . . .	M
A13 Die Fehlernachricht bei fehlendem Prozess-Namen . . . . .	M
A14 Die Fehlernachricht bei fehlendem Projekt . . . . .	N
A15 Die positive Rückmeldung des Änderungsantrags . . . . .	N
A16 Die negative Rückmeldung nach Anfrage zum Löschen . . . . .	N

# Code-Abschnitte

4.1	Verbindungs-Herstellung zwischen MongoDB und Backend . . . . .	32
4.2	Ein Beispiel für eine PrivateRoute des Process-Managers . . . . .	34
4.3	Die Nutzer-Sammlung in der Datenbank . . . . .	34
4.4	Der Login-Endpunkt im Backend des Process-Managers . . . . .	35
4.5	Die Get-Abfrage für die Benutzer . . . . .	37
4.6	Die Funktion zur Erstellung neuer Projekte . . . . .	41
4.7	Der Put-Befehl zum Bearbeiten der Projekte . . . . .	44
4.8	Die Funktion zur Erstellung neuer Benutzer . . . . .	47
4.9	Der Put-Befehl zum Bearbeiten der Benutzer . . . . .	49
4.10	Die Put-Abfrage zum Ändern des Benutzer Password . . . . .	53
4.11	Das Datenbank-Schema für die Benachrichtigungen . . . . .	57
A1	Das vollständige Routing im Process-Manager . . . . .	B
A2	Die Datei customRules . . . . .	K
A3	Die Datei customExtensions . . . . .	L

# 1 Einleitung

## Motivation

Das ist die Motivation

## Problemstellung

Das ist die Problemstellung

## 2 Aufgabenstellung

Das ist die Aufgabenstellung

# 3 Theoretische Grundlagen

## 3.1 Anforderungen

In diesem Kapitel werden die Anforderungen an das Projekt analysiert, definiert und strukturiert. Dabei wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden, die in den Abschnitten Funktionale Anforderungen und Nicht-funktionale Anforderungen näher erläutert werden.

### 3.1.1 Anforderungsanalyse

Die Anforderungsanalyse ist ein zentraler Prozess in der Systementwicklung, der sich mit der Analyse, Dokumentation und Verwaltung der Anforderungen an ein System beschäftigt. Sie umfasst die Identifizierung der Bedürfnisse und Erwartungen der Stakeholder, die Definition der Anforderungen an das System und die Sicherstellung, dass diese Anforderungen verstanden und umgesetzt werden können. Laut der Definition des Institute of Electrical and Electronic Engineers (IEEE) beinhaltet die Anforderungsanalyse zwei Hauptprozesse: das Studium der Benutzerbedürfnisse, um eine klare Definition der System-, Hardware- oder Softwareanforderungen zu erstellen, sowie die Verfeinerung und Detaillierung dieser Anforderungen im Verlauf des Projekts. Eine kontinuierliche Anforderungsanalyse ist notwendig, da Anforderungen sich während des Projektlebenszyklus ändern und diese Änderungen identifiziert, überprüft und integriert werden müssen.

Die Bedeutung der Anforderungsanalyse ergibt sich aus mehreren zentralen Aspekten. Erstens stellt die Anforderungsanalyse sicher, dass die Wünsche und Bedürfnisse der Stakeholder erfasst, dokumentiert und spezifiziert werden. Dies minimiert das Risiko, dass das entwickelte System nicht den Erwartungen der Stakeholder entspricht. Zweitens kann die Anforderungsspezifikation als Teil des Vertrags über die Entwicklung eines IT-Systems dienen. Sie fungiert als verbindliches Dokument, das Missverständnisse reduziert, indem es klar definiert, was benötigt wird, und somit unnötige Mehrarbeit und Frustration vermeidet. Ein weiterer wichtiger Aspekt der Anforderungsanalyse ist die Unterstützung bei Entscheidungsprozessen. Klar formulierte und gut durchdachte Anforderungen helfen bei der Auswahl technischer Lösungen, die diese Anforderungen bestmöglich erfüllen. Zudem trägt die Anforderungsanalyse zur Komplexitätskontrolle bei, indem sie das System

in verständliche Teilmodelle zerlegt, was das Verständnis und die Nutzung des Systems erleichtert.

Zusammenfassend ist die Anforderungsanalyse ein unverzichtbarer Bestandteil des Projektmanagements, der die Grundlage für die erfolgreiche Entwicklung und Implementierung eines Systems bildet. Sie gewährleistet, dass die entwickelten Lösungen den Bedürfnissen der Benutzer entsprechen, und trägt wesentlich zur Qualität des gesamten Entwicklungsprozesses bei.

### **3.1.2 Funktionale Anforderungen**

#### **Muss-Anforderungen**

- M1.** Das System muss eine Funktion zum Erstellen neuer Prozesse bieten.
- M2.** Das System muss eine Möglichkeit zur Bearbeitung bestehender Prozesse bereitstellen.
- M3.** Das System muss eine Funktion zur Ausführung von Prozessen implementieren.
- M4.** Das System muss eine Liste aller gespeicherten Prozesse anzeigen können.
- M5.** Das System muss die Möglichkeit bieten, Prozesse zu speichern und zu löschen.
- M6.** Das System muss die Möglichkeit bieten, jedem Event im Prozess eine spezifische Rolle zuzuweisen.
- M7.** Das System muss sicherstellen, dass nur Benutzer mit der zugewiesenen Rolle ein Event ausführen können.
- M8.** Das System muss einen Benachrichtigungsmechanismus implementieren, um andere Rollen bei Bedarf zu informieren.
- M9.** Das System muss die Ausführung paralleler Geschäftsprozesse ermöglichen, einschließlich der Möglichkeit, zwischen parallelen Prozessen zu wechseln.
- M10.** Das System muss eine Bestätigungsabfrage beim Löschen von Prozessen implementieren.
- M11.** Das System muss einen Löschmechanismus mit Berechtigungsprüfung implementieren, sodass nicht jeder Benutzer Prozesse löschen kann.

- M12.** Das System muss verhindern, dass laufende Prozesse gelöscht werden, und das Löschen erst nach Abschluss des aktuellen Prozesses erlauben.

## **Soll-Anforderungen**

- S1.** Das System sollte eine Vorschau des Prozessablaufs während der Erstellung, Bearbeitung und vor der Ausführung anzeigen.
- S2.** Das System sollte die Möglichkeit bieten, Prozesse in verschiedene Kategorien oder Typen einzuteilen.
- S3.** Das System sollte eine Suchfunktion für gespeicherte Prozesse bereitstellen.
- S4.** Das System sollte die Möglichkeit bieten, Benutzerrollen und -berechtigungen zu verwalten.

## **Kann-Anforderungen**

- K1.** Das System kann eine Funktion zur Prozessanalyse und -optimierung anbieten.
- K2.** Das System kann eine mobile Version oder App für den Zugriff von unterwegs anbieten.
- K3.** Das System kann eine Funktion zur automatischen Dokumentation von Prozessausführungen implementieren.
- K4.** Das System kann eine Kolaborationsfunktion für die gemeinsame Arbeit an Prozessen bieten.
- K5.** Das System kann eine detaillierte Protokollierung von Prozessänderungen und -lösungen bereitstellen.

### 3.1.3 Nicht-funktionale Anforderungen

#### Muss-Anforderungen

- NM1.** Das System muss eine Benutzeroberfläche mit Drag-and-Drop-Funktionalität oder einer ähnlich intuitive Funktion für die Prozesserstellung bieten.
- NM2.** Das System muss ein robustes Rollen- und Berechtigungsmanagement implementieren, um die Sicherheit und Integrität der Prozesse zu gewährleisten.

#### Soll-Anforderungen

- NS1.** Das System sollte eine hohe Benutzerfreundlichkeit aufweisen, um die Effizienz bei der Prozesserstellung und -ausführung zu gewährleisten.
- NS2.** Das System sollte eine angemessene Leistung und Reaktionszeit bei der Verarbeitung von Prozessen bieten.
- NS3.** Das System sollte eine hohe Verfügbarkeit aufweisen, um parallele Prozessausführungen zuverlässig zu unterstützen.

#### Kann-Anforderungen

- NK1.** Das System kann eine mehrsprachige Benutzeroberfläche anbieten, um internationale Nutzung zu unterstützen.
- NK2.** Das System kann eine hohe Skalierbarkeit aufweisen, um mit wachsender Anzahl von Prozessen und Benutzern umgehen zu können.

## 3.2 Prozessmodellierung

Die Modellierung von Geschäftsprozessen stellt einen zentralen Aspekt des Prozessmanagements dar. Zwei bedeutende Notationen, die in diesem Kontext eine große Rolle spielen, sind die Ereignisgesteuerte Prozesskette (EPK) und die Business Process Model and Notation (BPMN). Beide Methoden dienen der grafischen Darstellung und Analyse von Geschäftsprozessen, unterscheiden sich jedoch in ihrer Entstehung, Anwendung und Komplexität.

### 3.2.1 Ereignisgesteuerte Prozesskette

Die Ereignisgesteuerte Prozesskette ist 1992 entwickelt worden und findet vor allem im deutschsprachigen Raum Anwendung. Die EPK basiert auf einer klaren Struktur aus drei Grundelementen: Ereignissen, Funktionen und logische Operatoren.

**Ereignisse** beschreiben eingetretene Zustände dar. Sie sind passive Elemente und werden typischerweise durch ein sechseckiges Symbol dargestellt.

**Funktionen** repräsentieren Aktivitäten oder Aufgaben, die durch ein Ereignis ausgelöst werden und selbst wiederum ein neues Ereignis auslösen können. Funktionen werden üblicherweise mit einem Rechteck mit abgerundeten Ecken dargestellt. Funktionen sind aktive Elemente des Prozesses und beschreiben was getan werden muss.

**Logische Operatoren** dienen dazu, Ereignisse und Funktionen miteinander zu verbinden und den Kontrollfluss zu steuern. Dabei gibt es drei Arten von Operatoren: UND, ODER und exklusives Oder (XOR). Diese Operatoren erlauben es, komplexe Prozessstrukturen wie Verzweigungen, Zusammenführungen und Schleifen abzubilden.

Die Struktur der EPK basiert auf festen Regeln: Jede EPK beginnt und endet mit einem Ereignis. Funktionen und Ereignisse wechseln sich ab. Die Modellierung erfolgt von oben nach unten.

EPKs finden breite Anwendung in der Prozessdokumentation, der Zuweisung von Zuständigkeiten, sowie der Analyse und Optimierung von Geschäftsprozessen. Ihre Stärke liegt in der intuitiven Verständlichkeit, die sie auch für Anfänger zugänglich macht.

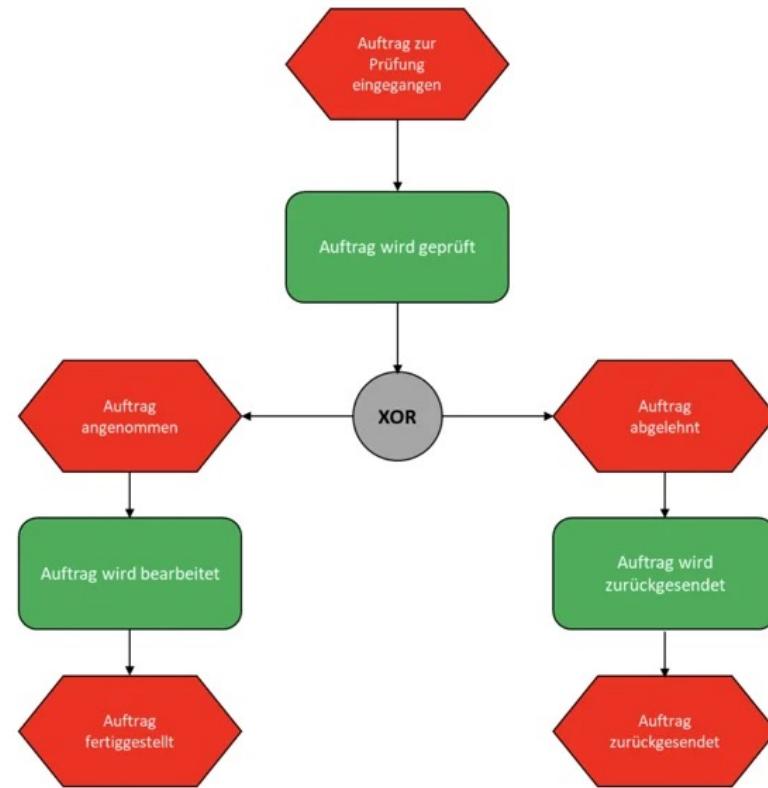


Abbildung 3.1: Beispiel für EPK-Aufbau

Die Abbildung 3.1 dient als Beispiel, um den Ablauf einer EPK zu veranschaulichen. Es zeigt den Prozess eines Auftrags, der mit dem Ereignis „Auftrag zur Prüfung eingegangen“ beginnt. Darauf folgt die Funktion „Auftrag wird geprüft“. Anschließend wird über einen logischen Operator, dargestellt durch ein XOR-Gatter, der weitere Verlauf bestimmt. Es gibt zwei mögliche Wege: Wird der Auftrag angenommen, erfolgt die Funktion „Auftrag wird bearbeitet“, gefolgt vom Abschluss durch das Ereignis „Auftrag fertiggestellt“. Wird der Auftrag abgelehnt, wird der Auftrag zurückgesendet, was mit dem Ereignis „Auftrag zurückgesendet“ endet. Das Diagramm zeigt den typischen Wechsel zwischen Ereignissen (rot) und Funktionen (grün) sowie die strukturierte Modellierung von oben nach unten.

### 3.2.2 Business Process Model and Notation

Die BPMN ist eine standardisierte grafische Darstellungsmethode zur Modellierung von Geschäftsprozessen. Sie ermöglicht es, komplexe Abläufe übersichtlich und verständlich darzustellen. BPMN gliedert sich in vier Hauptkategorien von Elementtypen: Flussobjekte, Verbindungsobjekte, Pools und Lanes sowie Artefakte.

**Flussobjekte** bilden die Kernelemente eines BPMN-Diagramms und umfassen Ereignisse, Aktivitäten und Gateways. Ereignisse sind durch Kreise dargestellt, und können Start-, Zwischen- oder Endereignisse sein. Aktivitäten repräsentieren die Aufgaben oder Arbeiten, die im Rahmen des Prozesses durchgeführt werden müssen, und werden durch abgerundete Rechtecke symbolisiert. Gateways, visualisiert durch Rauten, dienen als Entscheidungs- oder Verzweigungspunkt im Prozessfluss, die unterschiedliche Pfade basierend auf Bedingungen ermöglichen.

**Verbindungsobjekte** sind entscheidend für die Strukturierung des Prozesses und verbinden die Flussobjekte miteinander. Sie umfassen Sequenzflüsse, die die Reihenfolge der Aktivitäten darstellen; Nachrichtenflüsse, die den Informationsaustausch zwischen verschiedenen Teilnehmern zeigen; und Assoziationen, die zusätzliche Informationen zu Flussobjekten bereitstellen.

**Pools und Lanes** organisieren die Verantwortlichkeiten innerhalb eines BPMN-Diagramms. Ein Pool repräsentiert einen übergeordneten Prozessbeteiligten oder eine Organisationseinheit und definiert den Rahmen eines Ablaufs. Innerhalb dieses Rahmens unterteilen Lanes die Verantwortlichkeiten weiter, indem sie spezifische Rollen oder Abteilungen darstellen. Diese visuelle Trennung erleichtert das Verständnis der Beziehungen zwischen den verschiedenen Prozessteilnehmern.

**Artefakte** bieten zusätzliche Informationen über den Prozess und kategorisieren Aktivitäten. Dazu gehören Datenobjekte, die notwendige Informationen für den Prozess darstellen; Gruppen, die verschiedene Aktivitäten zusammenfassen; und Textanmerkungen, die erläuternde Informationen liefern.

Insgesamt ermöglicht BPMN durch diese vier Hauptkategorien eine klare und strukturierte Darstellung von Geschäftsprozessen, was zu einer verbesserten Kommunikation zwischen den beteiligten Akteuren führt und die Analyse sowie Optimierung von Abläufen unterstützt.

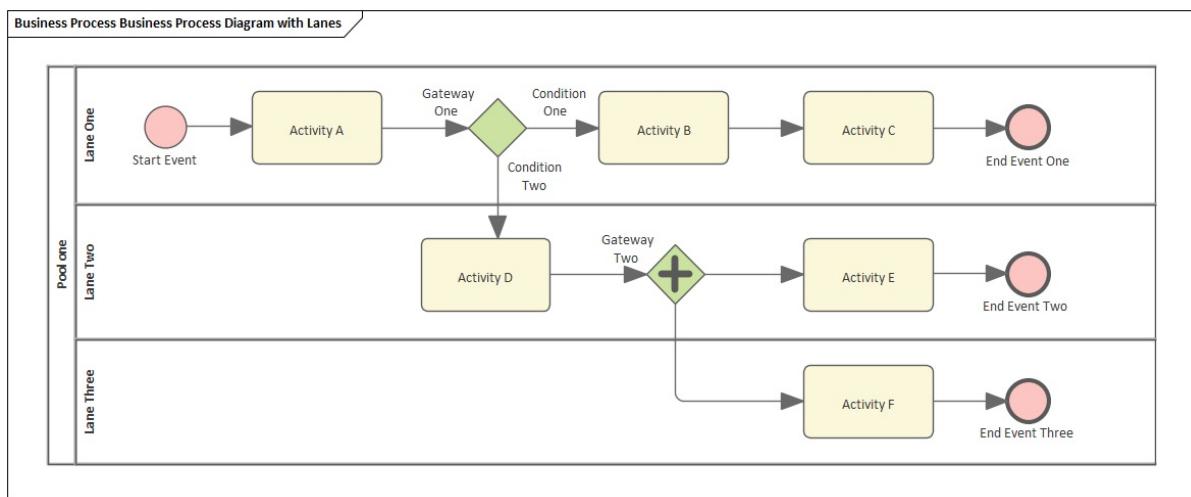


Abbildung 3.2: Beispiel für BPMN-Aufbau

Die Abbildung 3.2 stellt einen Geschäftsprozess in BPMN-Notation dar und nutzt Lanes, um die Zuständigkeiten der einzelnen Prozessschritte zu strukturieren. Der Prozess beginnt in „Lane One“ mit einem Startereignis und führt zunächst zur Aktivität „Activity A“. Anschließend wird an „Gateway One“ eine Entscheidung getroffen, die den Prozess in zwei mögliche Wege verzweigt: Im ersten Weg, der durch „Condition One“ definiert ist, werden die Aktivitäten „Activity B“ und „Activity C“ ausgeführt, bevor der Prozess mit „End Event One“ abgeschlossen wird. Der zweite Weg, der durch „Condition Two“ bestimmt ist, wechselt in „Lane Two“ zur Aktivität „Activity D“. Von dort erfolgt eine weitere Verzweigung an „Gateway Two“, die den Prozess in zwei weitere Richtungen leitet: Entweder wird „Activity E“ ausgeführt, bevor der Prozess mit „End Event Two“ endet, oder es wird in „Lane Three“ zur Aktivität „Activity F“ übergegangen, die mit „End Event Three“ abgeschlossen wird. Dieses Diagramm veranschaulicht die strukturierte Darstellung von Prozessen, die mithilfe von Lanes Verantwortlichkeiten abbilden und durch Gateways Entscheidungen und Parallelverzweigungen modellieren.

## Begründung der Standardisierung auf BPMN

Im Rahmen dieses Projekts ist es aufgrund begrenzter Ressourcen notwendig, sich auf einen einzigen Standard zur Modellierung von Geschäftsprozessen zu konzentrieren. Die parallele Umsetzung mehrerer Modellierungsstandards würde die Komplexität deutlich erhöhen und den Projektumfang überschreiten. Daher muss eine fundierte Auswahlentscheidung getroffen werden, welcher Standard den Anforderungen des Projekts am besten gerecht wird.

Die Entscheidung fällt auf die BPMN, da sie in mehreren zentralen Aspekten Vorteile gegenüber EPK bietet:

**1. Internationale Verbreitung und Standardisierung:**

BPMN ist ein von der Object Management Group (OMG) standardisiertes Notationssystem, das international weit verbreitet ist. Diese breite Akzeptanz gewährleistet eine hohe Kompatibilität mit modernen Informations-Technologie (IT)-Systemen und Geschäftsprozessmanagement-Tools, was insbesondere im Hinblick auf langfristige Wartbarkeit und Interoperabilität von Vorteil ist.

**2. Hohe Ausdrucksstärke und Flexibilität:**

Die Notation ermöglicht durch ihre differenzierte Symbolik – etwa durch die Nutzung von Pools, Lanes sowie diversen Ereignis- und Aufgabenarten – eine präzise und strukturierte Abbildung selbst komplexer, unternehmensübergreifender Prozesse. Diese Ausdrucksstärke unterstützt die klare Kommunikation zwischen fachlichen und technischen Stakeholdern und trägt somit zur Qualität der Prozessdokumentation bei.

**3. Technische Umsetzbarkeit durch verfügbare Bibliotheken:**

Für BPMN existieren etablierte Softwarebibliotheken, die eine Drag-and-Drop-Umsetzung sowie die Einhaltung der BPMN-Spezifikation ermöglichen. Diese Bibliotheken erleichtern die technische Implementierung erheblich, da sie eine robuste Grundlage für die Modellierung und Validierung von Prozessen bieten. Dadurch kann der Entwicklungsaufwand deutlich reduziert und die Umsetzung effizient gestaltet werden.

Aufgrund dieser drei Faktoren wird BPMN als einziger Modellierungsstandard in diesem Projekt eingesetzt. Eine detaillierte, wissenschaftliche Analyse der verwendeten BPMN-Bibliothek und ihrer technischen Eigenschaften erfolgt in Unterabschnitt 4.5.3.

Überprüfen ob Prozessmodellierung und Anforderungen tauschen, ggf auch Ausarbeitung User-Workflow noch vor Anforde-

## 3.3 Ausarbeitung User-Workflow

Dieses Kapitel dient der konzeptionellen Ausarbeitung des Workflows des Process-Managers. Es stellt keine konkrete Implementierung dar, sondern fokussiert sich auf die theoretische und wissenschaftlich fundierte Entwicklung einer geeigneten Struktur. Ziel dieser Ausarbeitungen ist es, eine wissenschaftlich fundierte Grundlage zu schaffen, auf der die spätere technische Entwicklung systematisch und nachvollziehbar aufbauen kann. Die dargestellten Inhalte umfassen erste konzeptionelle Entwürfe sowie Skizzen der Benutzeroberfläche, in denen zentrale Funktionalitäten und Eigenschaften des Process-Managers veranschaulicht werden. Diese Skizzen sind als Gestaltungsrahmen zu verstehen, die im weiteren Verlauf der Entwicklung als Orientierung dienen. Im Zuge der späteren Implementierung können die hier entwickelten Strukturen und Darstellungen angepasst und weiterentwickelt werden, um spezifischen Anforderungen und praktischen Gegebenheiten gerecht zu werden.

### 3.3.1 Grob-Ablauf

Der Grobablauf des Process-Managers dient der abstrakten Darstellung des typischen Workflows eines Benutzers. Dieser Ablauf gliedert sich in vier zentrale Bestandteile: den Anmeldevorgang sowie die drei Hauptfunktionen des Process-Managers – das Erstellen, Ausführen und Bearbeiten von Prozessen. In diesem Abschnitt werden diese Bestandteile strukturiert beschrieben und durch geeignete Visualisierungen verständlich veranschaulicht.

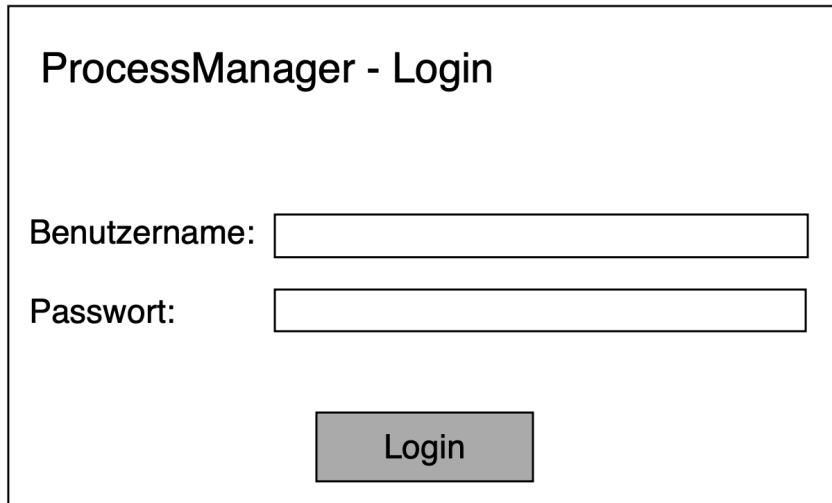
#### Login

Nach dem Starten des ProcessManager-Systems öffnet sich ein Login-Fenster (Abbildung 3.3), das die gesamte Benutzeroberfläche abdeckt, bevor die Startseite geladen wird. Dieses Fenster ist als strukturiertes Dialogfeld mit der Überschrift „ProcessManager - Login“ gestaltet. Es enthält zwei zentrale Eingabefelder: eines für den Benutzernamen und eines für das Passwort. Diese Felder dienen der Benutzerauthentifizierung und tragen zur Sicherheit des Systems bei.

Zur Interaktion mit dem Login-Fenster steht den Benutzern eine „Login“-Schaltfläche zur Verfügung. Diese ermöglicht die Übermittlung der eingegebenen Anmeldeinformationen zur Überprüfung und gewährt bei erfolgreicher Authentifizierung den Zugriff auf die Startseite des Systems.

Die Implementierung dieses Login-Mechanismus ist ein grundlegender Bestandteil der

Zugangskontrolle und Sicherheitsarchitektur des ProcessManager-Systems. Er stellt sicher, dass ausschließlich autorisierte Benutzer Zugriff auf die Systemfunktionen und Daten erhalten. Die Eingabe individueller Anmeldedaten dient nicht nur der Identitätsprüfung, sondern bildet auch die Basis für ein rollenbasiertes Zugriffsrechtesystem, das die Berechtigungen und Aktionen innerhalb des Systems reguliert.



The image shows a login interface titled "ProcessManager - Login". It contains two input fields: "Benutzername:" and "Passwort:", each with a corresponding text input box. Below these fields is a large, dark grey rectangular button labeled "Login".

Abbildung 3.3: Login-Fenster des ProcessManagers

Nach erfolgreicher Authentifizierung wird die Startseite geladen, wie in Abbildung 3.4 dargestellt. Auf dieser Seite stehen drei Hauptfunktionen zur Verfügung: Prozesse erstellen, ausführen und bearbeiten. Diese Funktionen werden im Folgenden näher erläutert.

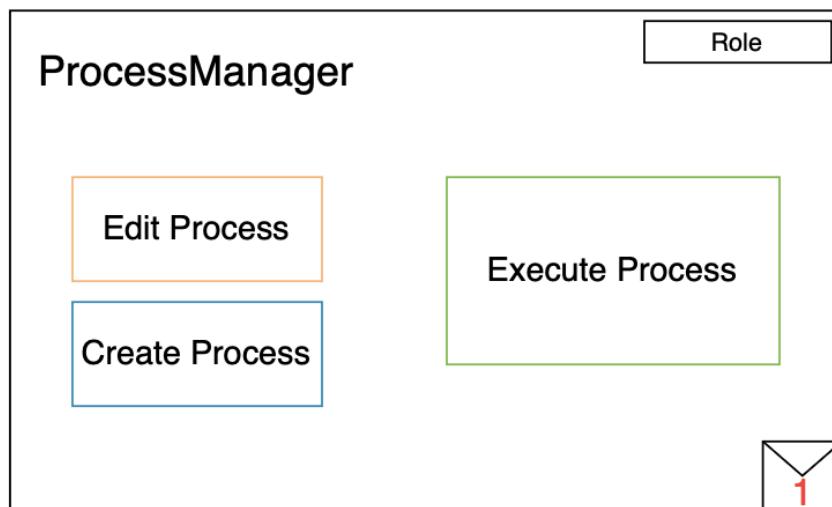


Abbildung 3.4: Startseite des ProcessManagers

Die Anzeigen in den unteren und oberen rechten Bereichen (Benachrichtigungen und Rol-

lenmanagement) der Benutzeroberfläche werden zu einem späteren Zeitpunkt thematisiert (Siehe Unterabschnitt 3.3.2 und Unterabschnitt 3.3.4).

## Prozess erstellen

Die Funktion zur Erstellung von Prozessen im ProcessManager-System wird durch den „Create Process“-Knopf initiiert und bietet zwei unterschiedliche Methoden zur Prozessdefinition: Drag-and-Drop und Coding.

Die **Drag-and-Drop Methode** (links in Abbildung 3.5) zur Prozesserstellung nutzt eine grafische Oberfläche, die es Benutzern ermöglicht, Prozesselemente wie „Ereignis XY“ und „Aktivität XYA“ visuell zu platzieren und miteinander zu verknüpfen. Diese intuitive Herangehensweise erleichtert es auch weniger technisch versierten Anwendern, komplexe Prozessabläufe zu modellieren, ohne direkt mit Code interagieren zu müssen.

Alternative ist eine **Code-basierte Methode** (rechts in Abbildung 3.5) zur Prozessdarstellung möglich. In diesem Modus wird die Prozesslogik mithilfe einer spezifischen Syntax oder Programmiersprache definiert. Die Anweisung „Process is illustrated by code“ weist darauf hin, dass die grafische Darstellung des Prozesses rechts neben dem Code daraus automatisch generiert wird. Diese Methode ermöglicht eine präzise und flexible Definition komplexer Prozesslogiken.

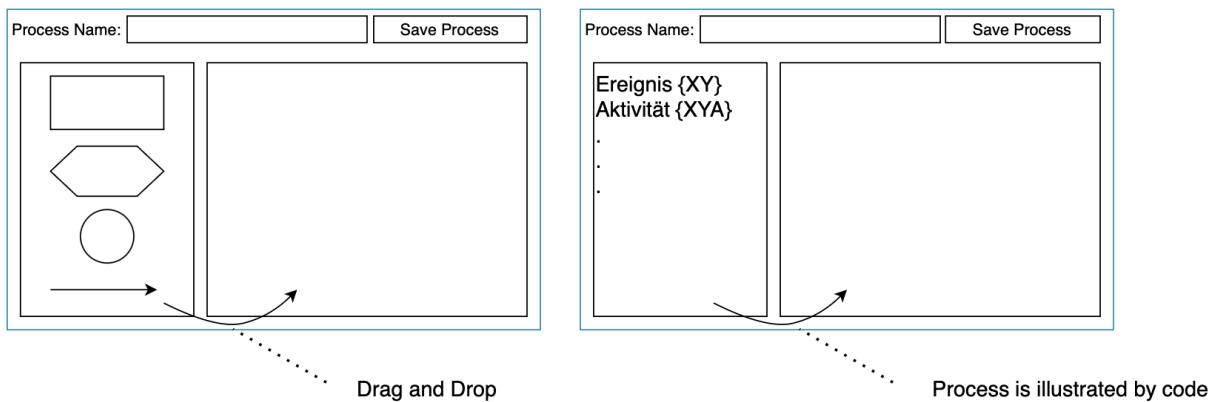


Abbildung 3.5: Drag-and-Drop- und Code-basierte Methode

Die Drag-and-Drop-Methode und die Code-basierte Methode unterscheiden sich deutlich in mehreren Aspekten und eignen sich je nach Kontext unterschiedlich gut. Im Folgenden sollen die beiden Methoden hinsichtlich einiger Faktoren verglichen werden, um die beste Methode für dieses Projekt finden zu können.

Bezüglich der **Benutzerfreundlichkeit** bietet die Drag-and-Drop-Methode klare Vorteile, da sie keine Programmierkenntnisse erfordert und durch ihre intuitive Bedienbarkeit insbesondere für Endnutzer geeignet ist. Die Code-basierte Methode hingegen setzt technischen Wissen voraus, was sie für weniger erfahrene Anwender weniger zugänglich macht.

Bei der **Effizienz** zeigt sich, dass Drag-and-Drop für die schnelle Umsetzung einfacher Prozesse ideal ist, da Nutzer Prozesse visuell und ohne großen Aufwand erstellen können. Allerdings kann diese Methode bei komplexeren Prozessen an ihre Grenzen stoßen. Die Code-basierte Methode benötigt mehr Zeit für die initiale Entwicklung, ist jedoch bei wiederholbaren oder komplexen Prozessen langfristig effizienter, da Anpassungen und Erweiterungen besser integrierbar sind.

In Bezug auf **Flexibilität** ist die Code-basierte Methode überlegen, da sie nahezu unbegrenzte Anpassungsmöglichkeiten bietet. Im Gegensatz dazu ist die Drag-and-Drop-Methode durch die vorgegebenen Werkzeuge eingeschränkt, was sie weniger geeignet für spezifische Anforderungen macht.

Auch in der **Fehleranfälligkeit** gibt es Unterschiede: Während Drag-and-Drop die Gefahr von Syntaxfehlern minimiert, können fehlerhafte Logiken durch Missverständnisse in der Prozessstruktur entstehen. Bei der Code-basierten Methode sind Syntax- und Logikfehler zwar häufiger, können jedoch mithilfe von Debugging-Tools effizient behoben werden.

Die **Wartbarkeit** ist ein weiterer entscheidender Faktor. Prozesse sind mit der Drag-and-Drop-Methode leicht wartbar, da Änderungen intuitiv vorgenommen werden können. Die Code-basierte Methode erfordert dagegen spezielles Wissen, ist jedoch langfristig besonders für größere und skalierbare Systeme wartungsfreundlicher.

Dies zeigt sich auch in der **Skalierbarkeit**, wo die Code-basierte Entwicklung klar im Vorteil ist, da sie sich ideal für komplexe und erweiterbare Anwendungen eignet. Die Drag-and-Drop-Methode stößt hingegen bei umfangreichen Projekten schnell an ihre Grenzen, da die Übersichtlichkeit und Anpassungsfähigkeit begrenzt sind.

Die Drag-and-Drop-Methode ist die optimale Wahl für dieses Projekt, da sie die Anforderungen besser erfüllt als die Code-basierte Methode. Ihre hohe Benutzerfreundlichkeit ermöglicht es, Prozesse ohne Programmierkenntnisse zu erstellen, während die Code-basierte Methode aufgrund technischer Kenntnisse eine Hürde darstellt. In puncto Effizienz überzeugt Drag-and-Drop durch die schnelle Erstellung einfacher Abläufe, während die Code-basierte Methode unnötig aufwendig wäre. Die geringere Flexibilität der Drag-and-Drop-Methode ist unproblematisch, da keine individuellen Anpassungen erforderlich sind –

die Anforderungen von EPK und BPMN definieren bereits, welche Werkzeuge für die Prozesse genutzt werden dürfen. Die visuelle Steuerung der Drag-and-Drop-Methode verringert die Fehleranfälligkeit, während die Code-basierte Methode trotz Debugging-Tools anfälliger für Syntax- und Logikfehler ist. Auch bei der Wartbarkeit punktet Drag-and-Drop, da die Prozesse einfach und intuitiv zu pflegen sind, während die Code-basierte Methode mehr Aufwand erfordert. Da die Skalierbarkeit keine große Rolle spielt, erfüllt Drag-and-Drop die Anforderungen problemlos, ohne die Komplexität der Code-basierten Methode. Insgesamt ist Drag-and-Drop aufgrund seiner Benutzerfreundlichkeit, Effizienz, Fehlerreduktion und Wartungsfreundlichkeit die bessere Wahl.

Die Möglichkeit einer Hybrid-Lösung, die die Vorteile beider Methoden kombiniert, könnte in manchen Fällen noch besser geeignet sein, insbesondere wenn eine gewisse Flexibilität und erweiterte Anpassbarkeit erforderlich wäre. Eine solche Lösung würde es ermöglichen, sowohl die Benutzerfreundlichkeit von Drag-and-Drop als auch die Flexibilität und Funktionalität der Code-basierten Methode zu nutzen. Allerdings würde die Umsetzung einer solchen Hybrid-Lösung in der Komplexität den Rahmen dieser Arbeit sprengen, da sie zusätzliche Entwicklungsressourcen und einen höheren Wartungsaufwand erfordern würde. Daher bleibt die Drag-and-Drop-Methode für die vorliegenden Anforderungen die praktikabelste Wahl.

## Prozesse ausführen

Die Funktion zur Ausführung von Prozessen im ProcessManager-System wird durch den „Execute Process“-Knopf initiiert und umfasst mehrere Tätigkeiten. Dazu zählen unter anderem die Prozessauswahl, das Abrufen von weiteren Informationen eines Ereignisses, Überprüfung von Zugriffsrechten, sowie das allgemeine Ausführen von Prozessen.

Die **Auswahl eines Prozesses** erfolgt aus einer übersichtlichen Liste verfügbarer Prozesse, die im Interface des ProcessManagers angezeigt wird (3.6). Neben der Möglichkeit, durch die Liste zu scrollen, steht eine Suchleiste zur Verfügung, um gezielt nach Prozessen zu suchen. Dies erleichtert insbesondere in umfangreichen Prozesssammlungen das schnelle Auffinden des gewünschten Eintrags. Um die Wahl des korrekten Prozesses zu unterstützen, wird rechts neben der Liste eine Vorschau des ausgewählten Prozesses angezeigt. Diese Vorschau liefert eine Zusammenfassung der wichtigsten Informationen und ermöglicht eine visuelle Überprüfung, bevor der Prozess gestartet wird.

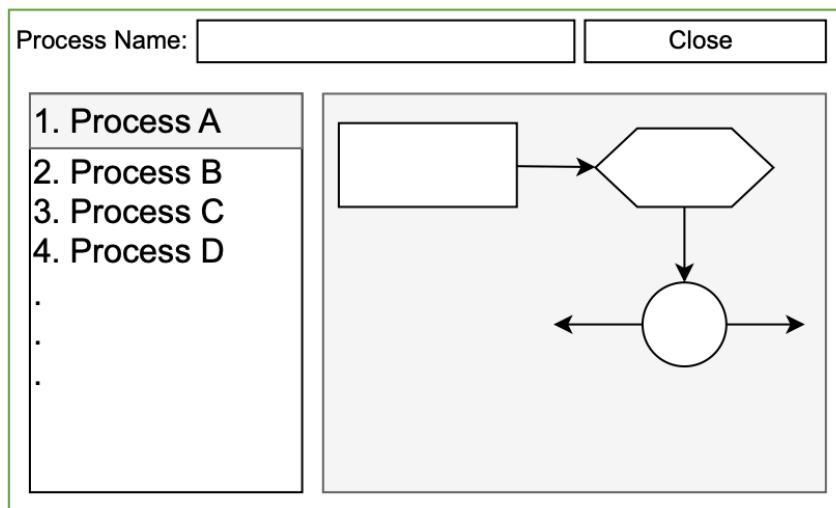


Abbildung 3.6: Interface für das Auswählen eines Prozesses

Der **Start des Prozesses** könnte entweder durch einen Doppelklick auf den entsprechenden Eintrag in der Liste oder durch die Auswahl des Prozesses mit anschließendem Klick auf einen Start-Knopf erfolgen. Diese intuitive Interaktionsmöglichkeit erleichtert die Bedienung und sorgt für eine effiziente Prozessausführung. Das Starten eines Prozesses und dessen anschließende Darstellung sind in Abbildung 3.7 dargestellt.

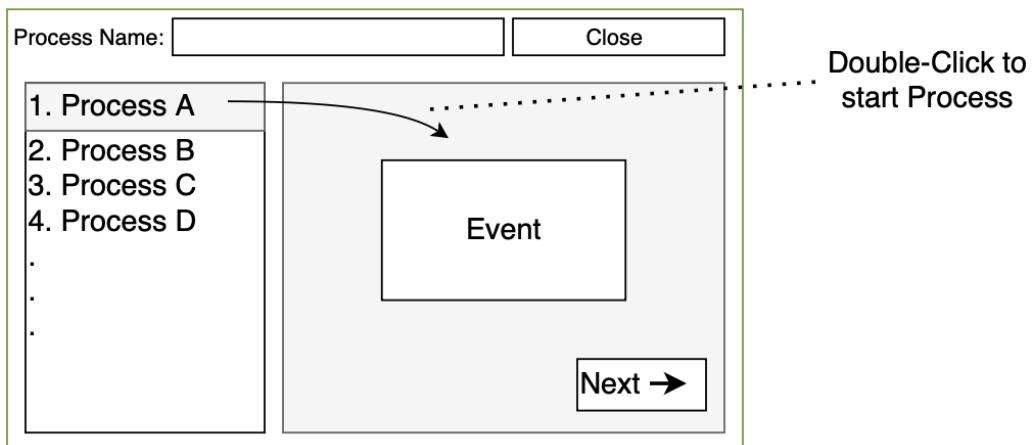


Abbildung 3.7: Interface bei einem gestarteten Prozess

Um **detaillierte Informationen** zu einem bestimmten Ereignis eines Prozesses abzurufen, bietet das System die Möglichkeit, durch einen Doppelklick auf das jeweilige Ereignis entsprechende Daten anzuzeigen. Zusätzlich wird auch die benötigte Rolle angezeigt, die erforderlich ist, um das Ereignis auszuführen. Eine beispielhafte Darstellung dessen ist in

Abbildung 3.8 zu sehen.

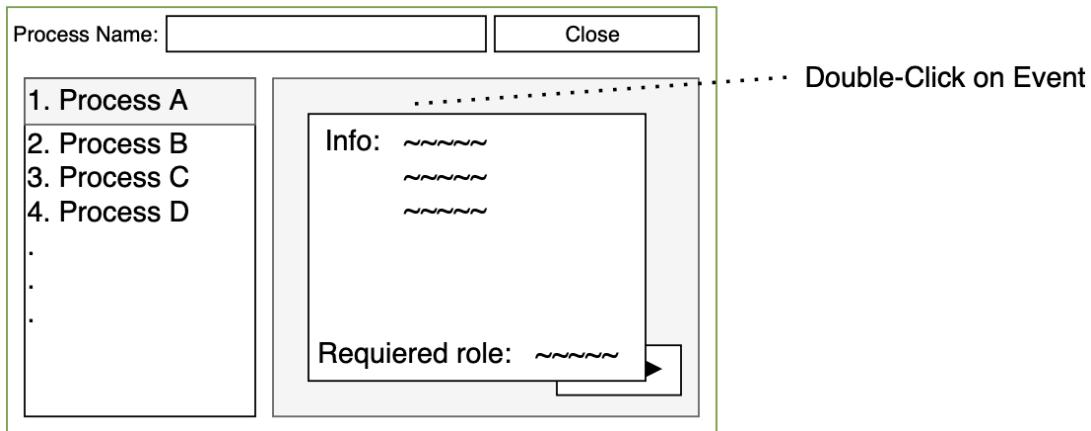


Abbildung 3.8: Zusatzinforamtionen eines Events

Vor der Ausführung eines Prozesses wird überprüft, ob der Nutzer über die **erforderlichen Rechte** verfügt. Falls diese fehlen, erhält der Nutzer eine entsprechende Benachrichtigung und kann gegebenenfalls eine Anfrage zur Freigabe an einen Vorgesetzten senden. Diese Zugriffsprüfung stellt sicher, dass Prozesse nur von berechtigten Nutzern ausgeführt werden können. Dies gewährleistet, dass in jedem Prozess regelmäßig überprüft wird, ob Fehler auftreten, wodurch die Qualität des Prozesses auf höchstem Niveau gehalten wird.

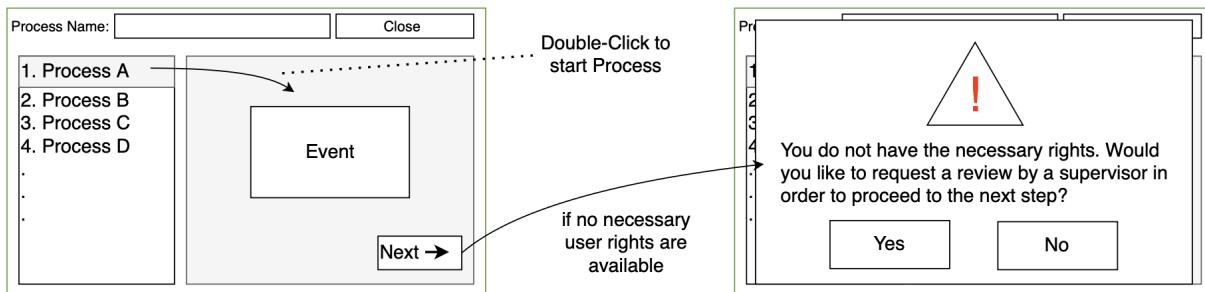


Abbildung 3.9: Prozess ausführen mit Zugriffsmanagement

## Prozesse bearbeiten

Die Funktion zum Bearbeiten von Prozessen im ProcessManager-System wird durch den „Edit Process“-Knopf initiiert und bietet zwei hauptsächliche Funktionen: das Bearbeiten und das Löschen von Prozessen. Dabei ist das Interface ähnlich aufgebaut wie im Kapitel „Prozess ausführen“, wobei links eine Liste aller Prozesse und rechts eine Vorschau des ausgewählten Prozesses angezeigt wird. Zusätzlich werden die beiden Knöpfe „Edit Process“

und „Delete Process“ eingeführt.

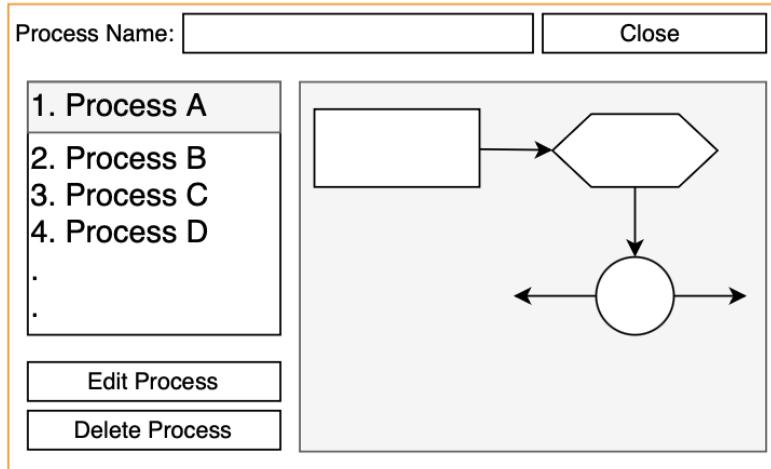


Abbildung 3.10: Interface für das Bearbeiten eines Prozesses

Das Bearbeiten oder Löschen von Prozessen erfolgt, indem ein Prozess aus der Liste auf der linken Seite ausgewählt wird und anschließend der entsprechende Knopf für Bearbeiten oder Löschen betätigt wird.

Wird der **Bearbeiten-Knopf** betätigt, wird der Ablauf wie beim Erstellen eines Prozesses (siehe Abbildung 3.5) aufgerufen. Der bestehende Prozess ist bereits geladen und kann entsprechend angepasst werden.

Wird der **Löschen-Knopf** betätigt, wird der Nutzer zunächst gefragt, ob er sicher ist, dass er den Prozess löschen möchte. Diese Aktion muss bestätigt oder abgelehnt werden, um versehentliches Löschen zu vermeiden.

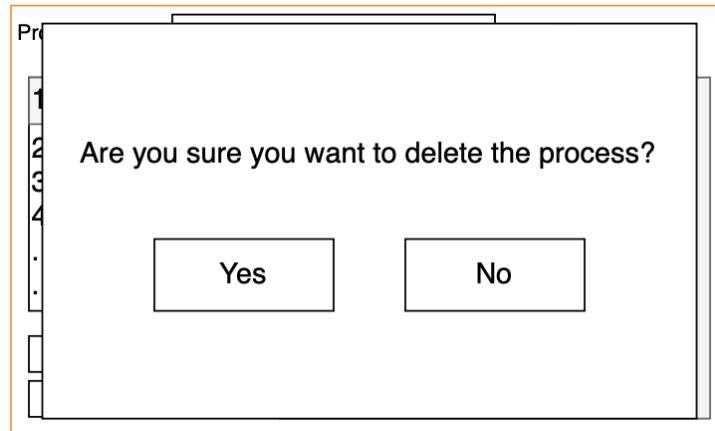


Abbildung 3.11: Abfrage für Löschen eines Prozesses

Die Logik hinter dem Löschen-Knopf ist je nach Rolle des Nutzers unterschiedlich, sodass es zwei Szenarien gibt:

**Szenario 1:** Der Nutzer hat keine Rechte, Prozesse zu löschen. In diesem Fall wird der Nutzer benachrichtigt, dass eine Anfrage an den Vorgesetzten zur Löschung des Prozesses gesendet worden ist. Bestätigt der Vorgesetzte das Löschen, erhält der Nutzer eine Benachrichtigung, dass der Prozess gelöscht worden ist. (siehe Abbildung 3.12 oben)

**Szenario 2:** Der Nutzer hat die erforderlichen Rechte, um Prozesse zu löschen. In diesem Fall wird der Nutzer benachrichtigt, dass der Prozess erfolgreich gelöscht worden ist. (siehe Abbildung 3.12 unten)

Unabhängig vom Szenario wird beim Löschen eines Prozesses nicht direkt alle aktiven Instanzen gelöscht. Lediglich die Möglichkeit, neue Prozesse dieser Art zu starten, wird entfernt. Der Prozess wird aus der Liste ausführbarer Prozesse gelöscht, bereits gestartete Prozesse können jedoch weiterhin abgeschlossen werden. Neue Instanzen können nicht mehr gestartet werden.

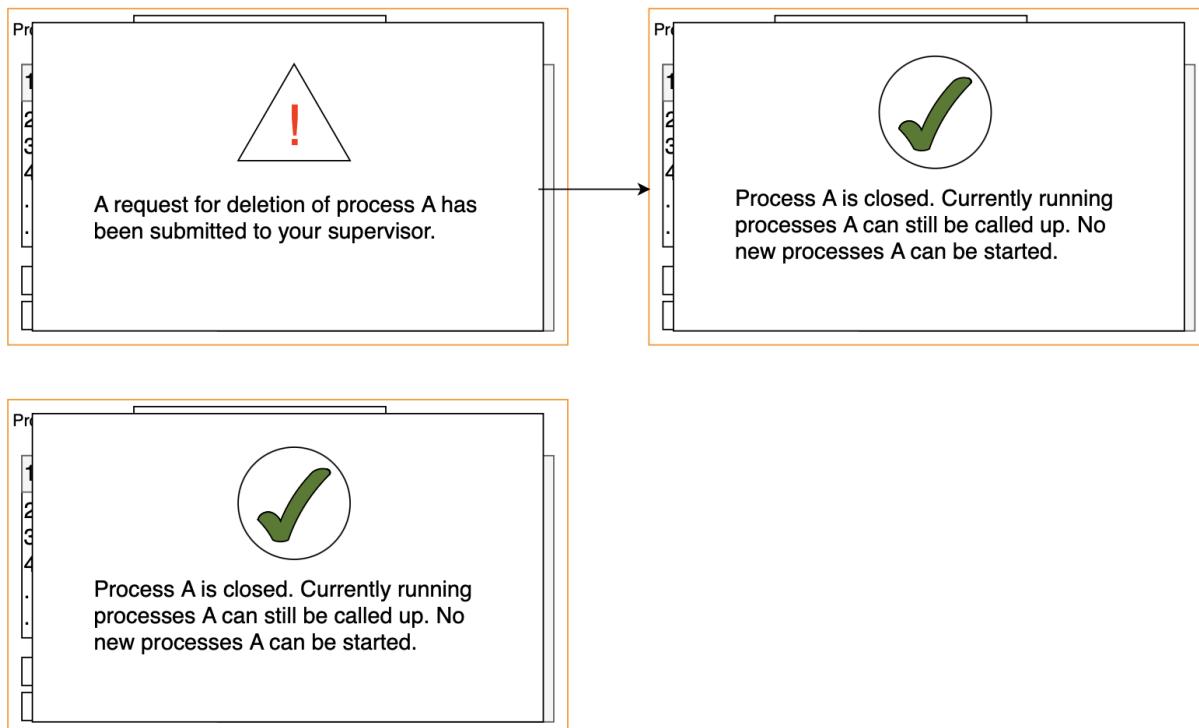


Abbildung 3.12: Szenarien beim Löschen von Prozessen

### 3.3.2 Rollenmanagement

Das Rollenmanagement ist ein essenzieller Bestandteil moderner Organisationsstrukturen und IT-Systeme, da es sicherstellt, dass nur autorisierte Personen Zugriff auf sensible Daten und Prozesse haben und dass Verantwortlichkeiten klar zugewiesen werden. Ein fundiertes Rollenmanagement trägt zur Verbesserung der Sicherheit, Effizienz und Nachvollziehbarkeit von Arbeitsprozessen bei.

#### Zugang

Ein zentraler Aspekt des Rollenmanagements ist die Regelung des Zugangs zu vertraulichen Prozessen und Informationen. Unberechtigte Personen sollten keinen Zugriff erhalten, um Sicherheitsrisiken wie Datenlecks oder unbefugte Manipulation zu minimieren. Dies wird durch die Implementierung von rollenbasierten Zugriffsmechanismen (RBAC) erreicht.

Role-based Access Control (RBAC) basiert auf der Zuweisung von Rollen, die spezifische Rechte und Pflichten definieren. Eine Person kann je nach Funktion eine oder mehrere Rollen übernehmen. Diese Rollen beschränken den Zugang zu Ressourcen und Prozessen auf das notwendige Minimum, was nicht nur die Sicherheit erhöht, sondern auch die Einhaltung gesetzlicher und organisatorischer Vorschriften erleichtert, wie beispielsweise das Datenschutz-Gesetz.

#### Kontrolle über Prozess-Aktivitäten

##### Rollenspezifische Aktivitäten

Die Zuweisung von Prozess-Aktivitäten an spezifische Rollen gewährleistet, dass nur qualifizierte und autorisierte Personen bestimmte Aufgaben übernehmen. Jede Aktivität innerhalb eines Prozesses wird einer verantwortlichen Rolle zugeordnet, die entweder die Aufgabe eigenständig durchführt oder deren Ausführung überwacht und die Verantwortung übernimmt. Diese klare Rollenverteilung fördert Transparenz und Rückverfolgbarkeit, wodurch Sicherheitsvorfälle oder ineffiziente Abläufe effizient adressiert werden können. Insbesondere in stark regulierten Sektoren wie dem Finanzwesen, dem Gesundheitssektor oder der Automobilindustrie ist diese Nachvollziehbarkeit unverzichtbar.

##### Löschen von Prozessen

Das Löschen von Prozessen ist eine kritische Aktion, die einer strengen Kontrolle unterliegt. Nur Rollen mit spezifischen Berechtigungen dürfen diese Aufgabe ausführen, um das Risiko von Datenverlusten oder unautorisierter Löschung sensibler Informationen zu minimieren.

Ein mehrstufiger Genehmigungsprozess, der eine unabhängige Überprüfung vorsieht, bietet zusätzliche Sicherheit. Solche Mechanismen senken das Risiko von Fehlern oder Missbrauch und gewährleisten, dass alle Löschvorgänge dokumentiert werden und so nachvollziehbar bleiben.

## Fazit des Rollenmanagements

Das Rollenmanagement stellt sicher, dass Zugang und Kontrolle über Prozess-Aktivitäten klar geregelt sind. Durch den Einsatz von RBAC und anderen Kontrollmechanismen werden Sicherheit und Effizienz verbessert, während die Einhaltung gesetzlicher und organisatorischer Vorschriften gewährleistet wird. Die sorgfältige Gestaltung und Implementierung solcher Systeme sind daher unerlässlich für jede Organisation.

### 3.3.3 Löschen von Prozessen

Das Löschen von Prozessen stellt eine kritische Operation dar, die sowohl technische als auch organisatorische Herausforderungen mit sich bringt. Um Datenverlust, Sicherheitsrisiken und regulatorische Verstöße zu vermeiden, ist eine strukturierte und regelkonforme Vorgehensweise unerlässlich. In diesem Unterkapitel wird dargelegt, wie Löschprozesse effizient und sicher gestaltet werden können.

### Grundprinzipien beim Löschen von Prozessen

Das Löschen von Prozessen erfordert eine sorgfältige Abwägung und Kontrolle. Dabei sollten die im Folgenden behandelten Prinzipien berücksichtigt werden.

Das Löschen von Prozessen erfordert eine klare **Regelung von Berechtigungen und Verantwortlichkeiten**. Hierfür bietet sich ein rollenbasiertes Zugriffskontrollsysteem, wie in Kapitel Rollenmanagement beschrieben, an. Diese Regelung sorgt dafür, dass nur qualifizierte und berechtigte Akteure in der Lage sind Prozesse zu löschen, was die Sicherheit erhöht. Durch die strikte Zuordnung der Verantwortlichkeiten können außerdem unbefugte und fehlerhafte Eingriffe minimiert oder komplett verhindert werden.

Die Löschung von Prozessen erfordert eine sorgfältige **Prüfung des Status**. Ein Prozess kann entweder als angelegte, aber nicht gestartete Instanz oder als aktive, laufende Instanz vorliegen.

Angelegte Instanzen stellen keine kritischen Anforderungen, da sie noch nicht aktiv sind

und keine laufenden Aufgaben oder Abhängigkeiten aufweisen. Diese Instanzen können direkt gelöscht werden.

Aktive Instanzen müssen sorgfältig beendet werden, um Datenverlust und Störungen zu vermeiden. Der Prozess muss kontrolliert abgeschlossen werden, wobei alle relevanten Zwischenergebnisse gespeichert und Abhängigkeiten berücksichtigt werden. Nach Abschluss kann die Instanz gelöscht werden. Eine umfassende Protokollierung des Löschvorgangs gewährleistet Nachvollziehbarkeit und unterstützt die Einhaltung regulatorischer Vorgaben.

Die **Nachvollziehbarkeit von Löschvorgängen** ist ein essenzieller Bestandteil eines sicheren Prozessmanagements. Durch eine lückenlose Protokollierung aller Löschaktionen wird Transparenz geschaffen und die Rückverfolgbarkeit gewährleistet. Ein Löschprotokoll sollte Informationen wie den Zeitpunkt des Vorgangs, die verantwortliche Rolle oder Person, den Grund für das Löschen und die Identifikationsmerkmale des betroffenen Prozesses enthalten. Diese Daten sind nicht nur für interne Analysen hilfreich, sondern auch für externe Audits oder rechtliche Nachweise erforderlich. Die Protokollierung erhöht die Kontrolle und ermöglicht die Rekonstruktion von Prozessen, falls dies erforderlich wird.

Um Risiken beim Löschen von Prozessen zu minimieren, sind **Sicherheitsmaßnahmen und Backup-Strategien** unverzichtbar. Vor einem Löschvorgang sollte eine Sicherungskopie des Prozesses erstellt werden, die im Bedarfsfall eine Wiederherstellung ermöglicht. Zusätzlich sollten Mechanismen wie ein mehrstufiger Genehmigungsprozess eingesetzt werden, um versehentliche Löschungen zu verhindern. Diese Sicherheitsvorkehrungen schützen vor unbeabsichtigten Datenverlusten.

## **Umsetzung eines effizienten Löschvorgangs**

Für eine effiziente Umsetzung eines Löschvorgangs müssen zunächst die Berechtigungen und Verantwortlichkeiten geklärt werden. Danach erfolgt eine Prüfung des Prozessstatus, um zwischen angelegten und aktiven Instanzen zu unterscheiden. Um Nachvollziehbarkeit und Sicherheit zu gewährleisten, ist eine lückenlose Protokollierung sowie ein mehrstufiger Genehmigungsprozess erforderlich. Vor der Löschung sollte zudem ein vollständiges Backup erstellt werden. Nach Abschluss des Löschvorgangs ist eine Überprüfung der Systemintegrität durchzuführen, und der gesamte Prozess sollte dokumentiert werden.

## Schlussfolgerung

Dieser Ablauf stellt den optimalen Fall eines Löschganges dar. In der tatsächlichen Umsetzung können jedoch aufgrund des begrenzten Entwicklungszeitraums und des umfangreichen Projektumfangs Abstriche erforderlich sein. Um den Prioritäten gerecht zu werden, müssen möglicherweise einige Schritte wie etwa detaillierte Protokollierung oder umfangreiche Sicherheitsvorkehrungen, angepasst oder vereinfacht werden, um eine effiziente Durchführung zu gewährleisten.

### 3.3.4 Benachrichtigungen

Benachrichtigungen spielen eine zentrale Rolle im ProcessManager-System und sind essenziell für die effektive Realisierung des gesamten Projekts. Sie dienen als Kommunikationsmechanismus, der Benutzer über wichtige Ereignisse, Statusänderungen und erforderliche Aktionen innerhalb des Prozessablaufs informiert. Die Implementierung eines robusten Benachrichtigungssystems erfüllt die Muss-Anforderung **M8.**, die explizit einen Mechanismus zur Information anderer Rollen bei Bedarf vorsieht.

Im Kontext des Process-Managers sind Benachrichtigungen besonders wichtig für die Koordination rollenbasierter Aktionen. Da das System jedem Event im Prozess eine spezifische Rolle zuweist (**M6.**) und sicherstellt, dass nur Benutzer mit der entsprechenden Rolle ein Event ausführen können (**M7.**), sind Benachrichtigungen unerlässlich, um die beteiligten Akteure über ihre Aufgaben und Verantwortlichkeiten zu informieren.

Die Umsetzung des Benachrichtigungssystems erfolgt durch Integration in die Benutzeroberfläche, wie in Abbildung 3.4 ersichtlich. Hier ist ein spezieller Bereich für Benachrichtigungen in der unteren rechten Ecke der Startseite vorgesehen. Diese Position gewährleistet, dass Benutzer wichtige Mitteilungen leicht wahrnehmen können, ohne den Arbeitsfluss zu unterbrechen.

Die Implementierung eines gut durchdachten Benachrichtigungssystems ist entscheidend für die Realisierung eines effektiven, rollenbasierten Prozessmanagements. Es verbessert nicht nur die Kommunikation zwischen den Beteiligten, sondern trägt auch zur Einhaltung von Fristen, zur Qualitätssicherung und zur allgemeinen Effizienz des Prozessablaufs bei.

## 3.4 Wahl der Anwendungsart

Bei der Entwicklung eines Prozessmanagement-Tools mit den in Kapitel 3.1 genannten Anforderungen stehen grundsätzlich zwei technologische Ansätze zur Auswahl: webbasierte Anwendungen und Desktop-Anwendungen. Beide Optionen bieten spezifische Vor- und Nachteile, die im Kontext der Projektanforderungen abgewogen werden müssen.

### Web-Anwendungen

Web-Anwendungen sind Softwarelösungen, die über einen Webbrowser zugänglich sind und auf einem Server ausgeführt werden. Sie erfordern keine lokale Installation und sind plattformunabhängig, da sie auf allen Geräten mit einem modernen Browser genutzt werden können. Nutzer greifen über das Internet auf die Anwendung zu, wobei die Daten auf Servern gespeichert und verarbeitet werden. Ein wesentlicher Vorteil von Webanwendungen ist ihre einfache Wartung und Aktualisierung, da Änderungen zentral vorgenommen werden und automatisch allen Nutzern zur Verfügung stehen. Webanwendungen bieten zudem die Möglichkeit zur Integration von Echtzeit-Kommunikationsmechanismen, was sie besonders für kollaborative und dynamische Anwendungen geeignet macht.

### Desktop-Anwendungen

Desktop-Anwendungen sind Softwareprogramme, die lokal auf einem Computer installiert und dort ausgeführt werden. Sie sind an das Betriebssystem des jeweiligen Geräts gebunden. Desktop-Anwendungen bieten in der Regel eine höhere Performance, da sie direkt auf die Ressourcen des Computers zugreifen können, und sind häufig für rechenintensive Aufgaben oder Anwendungen ohne permanente Internetverbindung konzipiert. Die Benutzer können mit Desktop-Anwendungen auch dann arbeiten, wenn keine Internetverbindung besteht, was sie in Umgebungen mit instabiler oder fehlender Netzwerkverbindung besonders nützlich macht.

### Wissenschaftliche Analyse der möglichen Anwendungsarten

Ein zentraler Faktor ist die **Flexibilität und Zugänglichkeit**. Webanwendungen bieten den Vorteil, plattformunabhängig zu sein und lediglich einen Browser zu benötigen. Im Gegensatz dazu sind Desktop-Anwendungen auf spezifische Betriebssysteme beschränkt,

können jedoch ohne Internetverbindung genutzt werden. Für Anwendungen, die breite Zugänglichkeit und flexible Einsatzmöglichkeiten erfordern, bieten Webanwendungen klare Vorteile.

Ein weiteres Kriterium ist die **Benutzerfreundlichkeit und Akzeptanz**. Webanwendungen zeichnen sich dadurch aus, dass sie gut für einen breiten Nutzerkreis mit unterschiedlichen Fachkenntnissen und Qualifikationen geeignet sind. Die intuitive Bedienung und moderne UI/UX-Designs fördern die Akzeptanz bei diversen Anwendergruppen. Darüber hinaus können Änderungen oder Aktualisierungen einfach und zentral vorgenommen und dann breitflächig integriert werden, was insbesondere in dynamischen Umgebungen von Vorteil ist. Desktop-Anwendungen hingegen bieten eine stärkere Anpassbarkeit an spezifische Benutzeranforderungen, erfordern jedoch häufig mehr Aufwand bei der Verteilung von Änderungen oder Updates. Angesichts der Projekterfordernisse, die eine hohe Zugänglichkeit und einfache Wartung voraussetzen, ist eine Web-Anwendung in diesem Kriterium vorzuziehen.

Die **Sicherheit und Datenschutz** stellen ein zentrales Anliegen dar. Web-Anwendungen profitieren von zentralisierten Sicherheitsmaßnahmen, die eine konsistente Implementierung und Wartung ermöglichen. Allerdings sind sie potenziell anfälliger für Online-Bedrohungen, da die Daten über Netzwerke übertragen und in zentralen Systemen gespeichert werden. Desktop-Anwendungen hingegen bieten durch die lokale Speicherung eine bessere Kontrolle über sensiblen Daten. Diese Dezentralisierung macht sie jedoch abhängig von individuellen Sicherheitsmaßnahmen auf jedem Gerät, was zu uneinheitlichen Schutzniveaus führen kann.

Auch **Performance und Offline-Funktionalität** sind entscheidende Faktoren. Desktop-Anwendungen bieten durch stärkere Hardwareintegration eine bessere Performance, insbesondere bei rechenintensiven Prozessen. Sie sind außerdem vollständig offline nutzbar. Webanwendungen hingegen sind von der Internetverbindung sowie der Serverleistung abhängig und besitzen in der Regel eingeschränkte Offline-Funktionalität. Für Szenarien mit hohem Leistungsbedarf oder ohne durchgängige Internetverbindung ist daher eine Desktop-Anwendung zu bevorzugen.

Die **projektspezifischen Anforderungen** umfassen Aspekte wie parallele Prozessausführung, Rollenzuweisung, Benachrichtigungen sowie die Unterstützung für gemeinsames und zeitgleiches Arbeiten innerhalb der Anwendung. Webanwendungen ermöglichen durch Echtzeit-Kommunikationstechnologien eine effektive Umsetzung von Rollenzuweisungen und Benachrichtigungen. Darüber hinaus erleichtert die zentrale Infrastruktur einer Web-Anwendung die gleichzeitige Zusammenarbeit mehrerer Nutzer, da Änderungen in Echtzeit

synchronisiert und allen Beteiligten direkt zugänglich gemacht werden können. Desktop-Anwendungen bieten hingegen eine stärkere Kontrolle über die Nutzung lokaler Ressourcen, sind jedoch weniger geeignet für das simultane Arbeiten, da sie typischerweise keine native Unterstützung für Echtzeit-Synchronisation oder Kollaboration bieten. Die zentrale Aktualisierung und die Fähigkeit zur nahtlosen Zusammenarbeit sprechen daher für die Wahl einer Web-Anwendung, insbesondere in Projekten, die auf dynamische Prozesse angewiesen sind.

## Schlussfolgerung

Nach Abwägung aller Kriterien erscheint eine Web-Anwendung als die am besten geeignete Wahl für das vorliegende Projekt. Sie bietet Vorteile hinsichtlich Zugänglichkeit, Benutzerfreundlichkeit, Wartbarkeit und den projektspezifischen Anforderungen. Desktop-Anwendungen wären nur dann vorzuziehen, wenn die Anforderungen an Offline-Funktionalität oder Performance eine zentrale Rolle spielen. Die Projekt-Anforderungen legen jedoch nahe, dass diese Aspekte im vorliegenden Fall keine Priorität haben, wodurch die Vorteile einer Web-Anwendung überwiegen.

## 3.5 Wahl der Applikation-Technologie

Basierend auf den Anforderungen des ProcessManager-Projekts und einer detaillierten Analyse relevanter Technologien werden im Folgenden die jeweils besten Optionen für Frontend, Backend und Datenbank evaluiert. Die Bewertung erfolgt anhand von Kriterien wie Performanz, Skalierbarkeit, Wartbarkeit, Entwicklungseffizienz und der Unterstützung durch die Community.

### 3.5.1 Frontend-Technologien

**React** bietet eine hervorragende Performanz, insbesondere bei komplexen Benutzeroberflächen, da der Virtual DOM eine effiziente Aktualisierung von UI-Elementen ermöglicht. Als Bibliothek statt Framework gewährt React den Entwicklern große Flexibilität in der Strukturierung und Integration von Anwendungen. Die flache Lernkurve ermöglicht eine schnelle Einarbeitung neuer Teammitglieder, und die modulare Struktur von React-Komponenten erleichtert Wartung und Erweiterung. Mit einer großen Community und einem reichhaltigen Ökosystem bietet React einfachen Zugang zu Bibliotheken und Tools, was die Entwicklung effizienter macht.

**Angular**, ein Framework, zeichnet sich durch eine umfassende Struktur aus, die für große Unternehmensanwendungen geeignet ist. Es bietet Optimierungen wie Tree-Shaking und Ahead-of-Time Compilation, was die Performanz verbessert. Jedoch ist die Lernkurve steil, und die Komplexität der Architektur kann die Entwicklungszeit verlängern. Angular überzeugt bei der Wartbarkeit durch starke Typisierung und integrierte Tools.

**Vue.js** punktet mit einer intuitiven Lernkurve und einer effizienten Rendering-Engine. Es ist skalierbar und einfach zu implementieren, eignet sich jedoch weniger für sehr große Anwendungen aufgrund der begrenzten Flexibilität und einer kleineren Community im Vergleich zu React.

**Empfehlung:** React wird für das Frontend des ProcessManager-Projekts empfohlen, da es eine ideale Balance zwischen Flexibilität, Performanz und Entwicklungseffizienz bietet. Die große Community-Unterstützung erleichtert zudem die langfristige Wartung und Erweiterung.

### 3.5.2 Backend-Technologien

Node.js bietet eine hohe Performanz, insbesondere bei I/O-intensiven Anwendungen, dank seiner ereignisgesteuerten, asynchronen Architektur. Die einheitliche Nutzung von JavaScript im Frontend und Backend reduziert den Entwicklungsaufwand erheblich. Eine große und aktive Community sowie eine Vielzahl von Bibliotheken ermöglichen die schnelle Implementierung von Funktionalitäten. Node.js zeichnet sich zudem durch seine Skalierbarkeit aus, was für das ProcessManager-Projekt essenziell ist.

Spring Boot, basierend auf Java, liefert hervorragende Performance für rechenintensive Aufgaben und eine robuste Architektur. Allerdings ist die Komplexität höher, was die Entwicklungszeit und den Schulungsaufwand für neue Entwickler erhöht. Spring Boot ist besonders für Unternehmensanwendungen geeignet, die strenge Typisierung und ausfeilte Sicherheitsmechanismen benötigen.

Django, ein Framework auf Basis von Python, bietet dank seiner "Batteries included"-Philosophie eine hohe Entwicklungseffizienz. Es ist gut strukturiert und eignet sich für mittlere bis große Anwendungen. Im Bereich der Skalierbarkeit bleibt es jedoch hinter Node.js zurück.

Empfehlung: Node.js ist die beste Wahl für das Backend des ProcessManager-Projekts, da es eine hohe Performanz, Skalierbarkeit und Flexibilität bietet. Die einheitliche JavaScript-Umgebung erleichtert die Zusammenarbeit zwischen Frontend- und Backend-Teams.

### 3.5.3 Datenbank-Technologie

MongoDB ist eine NoSQL-Datenbank, die sich durch ein dynamisches Schema und eine hervorragende horizontale Skalierbarkeit auszeichnet. Sie ist besonders geeignet für Anwendungen mit dynamischen Datenmodellen wie im ProcessManager-Projekt. MongoDB ermöglicht schnelle Anpassungen und bietet eine hohe Performanz bei Lese- und Schreiboperationen. Die Community und das Ökosystem sind umfangreich, was die Integration in bestehende Systeme erleichtert.

Relationale Datenbanken wie MySQL oder PostgreSQL bieten strenge ACID-Eigenschaften und eignen sich besonders für Anwendungen mit komplexen Abfragen und hohem Bedarf an Datenintegrität. Sie sind jedoch durch ihr festes Schema weniger flexibel und primär vertikal skalierbar, was sie weniger geeignet für dynamische und skalierbare Anwendungen macht.

Oracle bietet höchste Performanz und strenge Datenkonsistenz, ist jedoch mit hohen Lizenzkosten und einem hohen Wartungsaufwand verbunden. Für kleinere und dynamische Anwendungen wie den ProcessManager ist Oracle überdimensioniert.

Empfehlung: MongoDB wird als optimale Wahl empfohlen, da es die Flexibilität und Skalierbarkeit bietet, die für die dynamischen Prozessmodelle des Projekts erforderlich sind. Die einfache Integration und hohe Performanz machen es zur idealen Datenbank-Technologie.

### 3.5.4 Fazit

Nach eingehender Analyse und Bewertung der technologischen Anforderungen des ProcessManager-Projekts ergibt sich eine klare Empfehlung für die verwendeten Technologien:

Das Frontend sollte mit React entwickelt werden, da diese Bibliothek durch ihre Flexibilität, hohe Performance und die starke Unterstützung einer breiten Community überzeugt. Für das Backend ist Node.js die bevorzugte Wahl, da es eine einheitliche JavaScript-Umgebung bietet und durch hohe Performanz sowie ausgezeichnete Skalierbarkeit punktet. Als Datenbank empfiehlt sich MongoDB, deren dynamische Struktur, Flexibilität und Skalierbarkeit ideal auf die Bedürfnisse des Projekts zugeschnitten sind.

Diese Technologie-Kombination bietet eine optimale Balance zwischen Entwicklungseffizienz, Leistung und Wartbarkeit. Die Komponenten harmonieren nahtlos miteinander und gewährleisten eine umfassende Erfüllung der spezifischen Projektanforderungen. Mit dieser Auswahl wird eine skalierbare, leistungsstarke und zukunftssichere Implementierung des ProcessManager-Projekts ermöglicht.

# 4 Implementierung

In diesem Kapitel wird die tatsächliche Umsetzung und Implementierung des geplanten Process-Managers aus Kapitel 3 thematisiert. Dabei wird die Projektstruktur erläutert und alle wichtigen Funktionen des Process-Managers erklärt und anschaulich gezeigt.

## 4.1 Überblick und Kontext der Implementierung

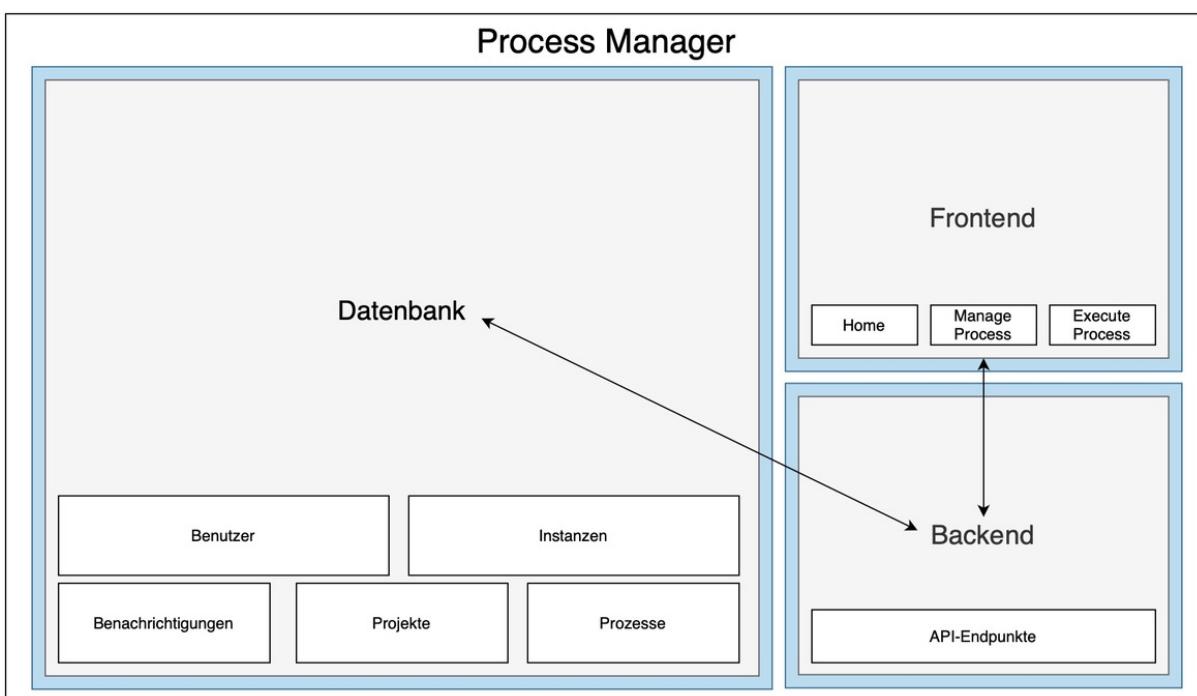


Abbildung 4.1: Die Projektstruktur des Process-Managers

Das Projekt gliedert sich in drei zentrale Komponenten: das Frontend, das Backend und die Datenbank. Eine vereinfachte Darstellung der Projektstruktur ist in Abbildung 4.1 zu sehen. Die drei grau hinterlegten Rechtecke repräsentieren die genannten Hauptkomponenten, während die blaue Umrandung die Bereitstellung dieser Komponenten mittels Docker symbolisiert. Eine detaillierte Auseinandersetzung mit der Docker-basierten Bereitstellung erfolgt in Unterabschnitt 4.5.5. Mit Hilfe des Frontends wird den Benutzern Funktionen wie das Verwalten oder das Ausführen von Prozessen ermöglicht. Dabei kommuniziert das Frontend mit dem Backend, um Daten aus der Datenbank zu erhalten. Das Backend kommuniziert sowohl mit dem Frontend als auch der Datenbank und enthält so den

wichtigsten Teil des Projekts für die Kommunikation, die API-Endpunkte. So kann sicher gestellt werden, dass die verwalteten Daten in der Datenbank anschaulich dem Benutzer auf dem Frontend zur Verfügung gestellt werden. In der Datenbank werden wichtige Daten-Sammlungen wie Benutzer, Projekte, Prozesse, Instanzen oder Benachrichtigungen verwaltet.

Die Verbindung zwischen Backend und Datenbank erfolgt über die Node.js-Bibliothek Mongoose, welche die Kommunikation mit einer MongoDB-Instanz ermöglicht. Der folgende Code-Ausschnitt 4.3 zeigt die konkrete Implementierung dieser Verbindung innerhalb des Backends:

```
1  mongoose
2    .connect(process.env.MONGODB_URI || 'mongodb://localhost:27017/
  bpmn', {
3      useNewUrlParser: true,
4      useUnifiedTopology: true
5    })
6    .then(() => {
7      console.log('MongoDB connected');
8    })
9    .catch(err => console.error(err));
```

Code 4.1: Verbindungs-Herstellung zwischen MongoDB und Backend

Dabei wird über die Umgebungsvariable „MONGODB\_URI“ die Adresse der Datenbank definiert. Falls diese Variable nicht gesetzt ist, wird auf eine lokale Instanz zurückgegriffen. Die Parameter „useNewUrlParser“ und „useUnifiedTopology“ dienen der Kompatibilität mit aktuellen MongoDB-Treibern. Die erfolgreiche Verbindung wird über eine Konsolen ausgabe bestätigt, während potenzielle Fehler im Fehlerfall protokolliert werden.

Nur durch eine gute Zusammenarbeit dieser drei Komponenten, kann der Process-Manager vollständig und effizient funktionieren. Die einzelnen Bestandteile und deren Zusammenspiel wird im Laufe dieses Kapitels noch genauer, anhand der wichtigsten Funktionen, beschrieben.

## 4.2 Login

Der Login des Process-Managers ist klassisch. Der Benutzer muss seinen Benutzernamen und das passende Passwort eingeben, bevor er auf die Startseite des Process-Managers weitergeleitet wird. Diese Benutzeroberfläche ist in Abbildung 4.2 zu sehen.

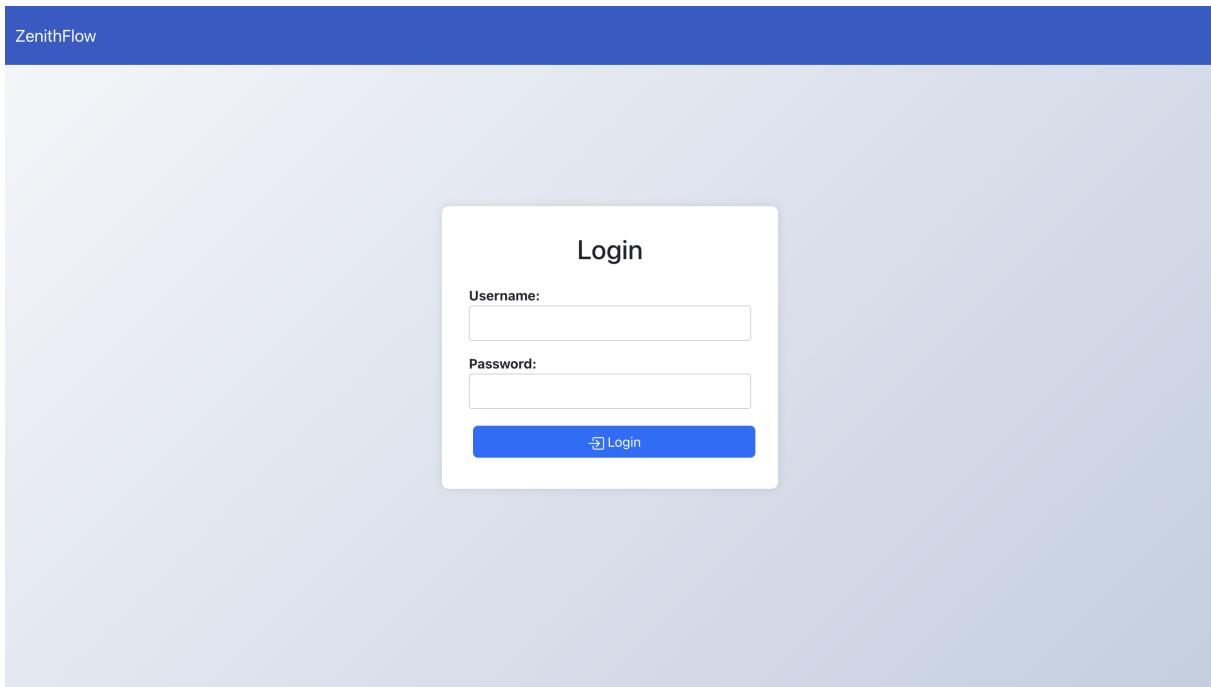


Abbildung 4.2: Die Login-Seite des Process-Managers

Dabei müssen drei Fälle beachtet werden. Der erste Fall ist der Benutzer gibt einen vorhandenen Benutzernamen mit passendem Passwort ein. In diesem Fall wird der Benutzer auf die Startseite des Process-Managers weitergeleitet. Allerdings könnte der Benutzer auch ein nicht passendes Passwort zu dem angegebenen Benutzer angeben oder eins der beiden Felder komplett leer lassen. Diese beiden Bilder sind im Anhang in Abbildung A1 und Abbildung A2 zu sehen.

Sofern eines der beiden Eingabefelder nicht ausgefüllt wird, erfolgt eine Validierung durch die im Browser integrierte Standardfunktion. Für den seltenen Fall, dass der verwendete Browser keine derartige Validierung unterstützt, ist zusätzlich eine serverseitige Fehlerbehandlung implementiert, die den Benutzer bei unvollständiger Eingabe entsprechend informiert. Dieser Mechanismus ist in Code 4.4, Zeile 3–4 dargestellt. Eine detaillierte Betrachtung der Fehlerbehandlung bei ungültigen Login-Daten erfolgt im Abschnitt zur Backend-Logik.

### Private Routing

Es ist sicherzustellen, dass auf sämtliche Bereiche des Process-Managers – mit Ausnahme der Login-Seite – ausschließlich nach erfolgreichem Login zugegriffen werden kann. Ein direkter Aufruf von Seiten über die URL ohne vorherige Authentifizierung muss unterbunden werden. Dies wird durch den Einsatz von Private Routing gewährleistet. Dies ist im

Code 4.2 verkürzt veranschaulich und in Code A1 im Anhang ist das komplette Routing des Process-Managers zu sehen.

```

1 <Route
2   path="/start"
3   element={
4     <PrivateRoute>
5       <StartPage />
6     </PrivateRoute>
7   }
8 />
```

Code 4.2: Ein Beispiel für eine PrivateRoute des Process-Managers

Private Routing stellt sicher, dass alle geschützten Seiten des Process-Managers bei fehlender Authentifizierung automatisch zur Login-Seite umleiten. Die Login-Seite selbst ist die einzige Seite, die nicht über das PrivateRoute-Tag geschützt ist und somit auch für nicht angemeldete Benutzer zugänglich bleibt.

### Nutzer-Sammlung in Datenbank

Für den Login-Vorgang wird eine Datenbank mit einer Sammlung von Nutzerinformationen benötigt. Diese Datenbank enthält alle relevanten Angaben zu den Benutzern, darunter: Benutzername, Passwort, Rolle sowie zugewiesene Projekte (siehe Code 4.3). Für die Authentifizierung beim Login sind jedoch ausschließlich der Benutzername und das verschlüsselte Passwort erforderlich.

```

1 const userSchema = new mongoose.Schema({
2   username: { type: String, required: true, unique: true },
3   password: { type: String, required: true },
4   role: { type: String, required: true, enum: ['Admin', 'Manager',
5     'Employee'] },
6   projects: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Project'
7     }],
8 });
9 const User = mongoose.model('User', userSchema);
```

Code 4.3: Die Nutzer-Sammlung in der Datenbank

### Wieso sollte das Passwort unbedingt verschlüsselt gespeichert werden?

Die Verschlüsselung bzw. sichere Hashing des Passworts schützt die Daten der Nutzer bei einem möglichen unautorisierten Zugriff auf die Datenbank. Ohne Verschlüsselung könnten

Angreifer Passwörter im Klartext auslesen und missbrauchen. Durch die Verwendung von Einweg-Hashfunktionen mit Salt wird sichergestellt, dass selbst bei gleichen Passwörtern unterschiedliche Hashwerte entstehen, was die Sicherheit zusätzlich erhöht. Damit wird ein grundlegender Datenschutzstandard erfüllt und das Risiko von Identitätsdiebstahl reduziert.

### Login-Endpunkt im Backend

Die Verarbeitung der Login-Logik, die hier im Frontend grafisch dargestellt ist, wird im Backend implementiert. Um diese nachvollziehen zu können, muss der Code in Code 4.4 genauer betrachtet werden.

```

1 app.post('/api/login', async (req, res) => {
2   const { username, password } = req.body;
3   if (!username || !password) {
4     return res.status(400).json({ error: 'Username and password
5       are required.' });
6   }
7   try {
8     const user = await User.findOne({ username });
9     if (!user) return res.status(401).json({ error: 'Invalid
10      credentials.' });
11    const isMatch = await bcrypt.compare(password, user.password);
12    if (!isMatch) return res.status(401).json({ error: 'Invalid
13      credentials.' });
14    return res.json({
15      message: 'Login successful',
16      user: { _id: user._id, username: user.username, role: user.
17        role },
18    });
19  } catch (err) {
20    return res.status(500).json({ error: 'Server error' });
21  }
22});
```

Code 4.4: Der Login-Endpunkt im Backend des Process-Managers

Der Login-Endpunkt dient der Authentifizierung von Benutzern über eine HTTP-POST-Anfrage. Nachdem sichergestellt wurde, dass sowohl Benutzername als auch Passwort übermittelt worden sind, beginnt in Zeile 5 der asynchrone Verarbeitungsprozess innerhalb eines try-catch-Blocks.

Zunächst wird in Zeile 6 durch den Aufruf von „User.findOne( username )“ versucht, ein Benutzerobjekt in der Datenbank zu finden, dessen Benutzername mit dem übermittelten Wert übereinstimmt. Dabei handelt es sich um eine asynchrone Datenbankabfrage (siehe Glossareintrag Asynchrone Programmierung), die den Programmfluss nicht blockiert, sondern auf das Ergebnis wartet, ohne andere Prozesse zu unterbrechen. Die Eindeutigkeit des Benutzernamens wird dabei durch das „unique-Flag“ im Schema sichergestellt, wie in der Definition des Mongoose-Modells (siehe Code 4.3) ersichtlich. Dadurch ist gewährleistet, dass jeder Benutzername nur einmal in der Datenbank vorkommen kann, was die Zuverlässigkeit und Effizienz der Suche verbessert.

Falls kein passender Benutzer gefunden wird, wird in Zeile 7 mit dem HTTP-Statuscode „401 (Unauthorized)“ eine Fehlermeldung zurückgegeben, die auf ungültige Anmeldedaten hinweist.

Im Anschluss erfolgt in Zeile 8 der Vergleich des übermittelten Passworts mit dem in der Datenbank gespeicherten Passwort-Hash. Hierzu wird die Bibliothek bcrypt verwendet, die für sichere Passwortverarbeitung in webbasierten Anwendungen eingesetzt wird. bcrypt erzeugt aus dem Klartextpasswort mittels eines sogenannten Salt-Verfahrens einen Hash-Wert, der in der Datenbank gespeichert wird. Beim Login prüft „bcrypt.compare()“, ob das eingegebene Passwort mit dem gespeicherten Hash übereinstimmt, ohne das ursprüngliche Passwort selbst rekonstruieren zu müssen. Diese Methode schützt effektiv vor dem unbefugten Zugriff auf Passwörter, selbst im Falle eines Datenlecks.

Falls der Vergleich fehlschlägt, wird erneut mit Statuscode 401 eine entsprechende Fehlermeldung ausgegeben (Zeile 9). Andernfalls wird in Zeile 10 eine Erfolgsmeldung übermittelt, die zusätzlich ein Objekt mit ausgewählten Benutzerdaten enthält: der Benutzer-ID (`_id`), dem Benutzernamen sowie der Benutzerrolle. Diese Informationen sind für die Authentifizierung und rollenbasierte Autorisierung in der Anwendung von Bedeutung.

Sollte es beim Zugriff auf die Datenbank oder während der Verarbeitung zu einem unerwarteten Fehler kommen, greift der catch-Block in Zeile 12, der mit Statuscode „500 (Internal Server Error)“ eine serverseitige Fehlermeldung an den Client zurückliefert. Damit wird eine klare Trennung zwischen benutzerbedingten Fehlern und systembedingten Ausfällen sichergestellt.

Durch diese strukturierte und fehlertolerante Umsetzung erfüllt der Endpunkt grundlegende Anforderungen an eine sichere Authentifizierungsschnittstelle im Backend eines webbasierten Systems.

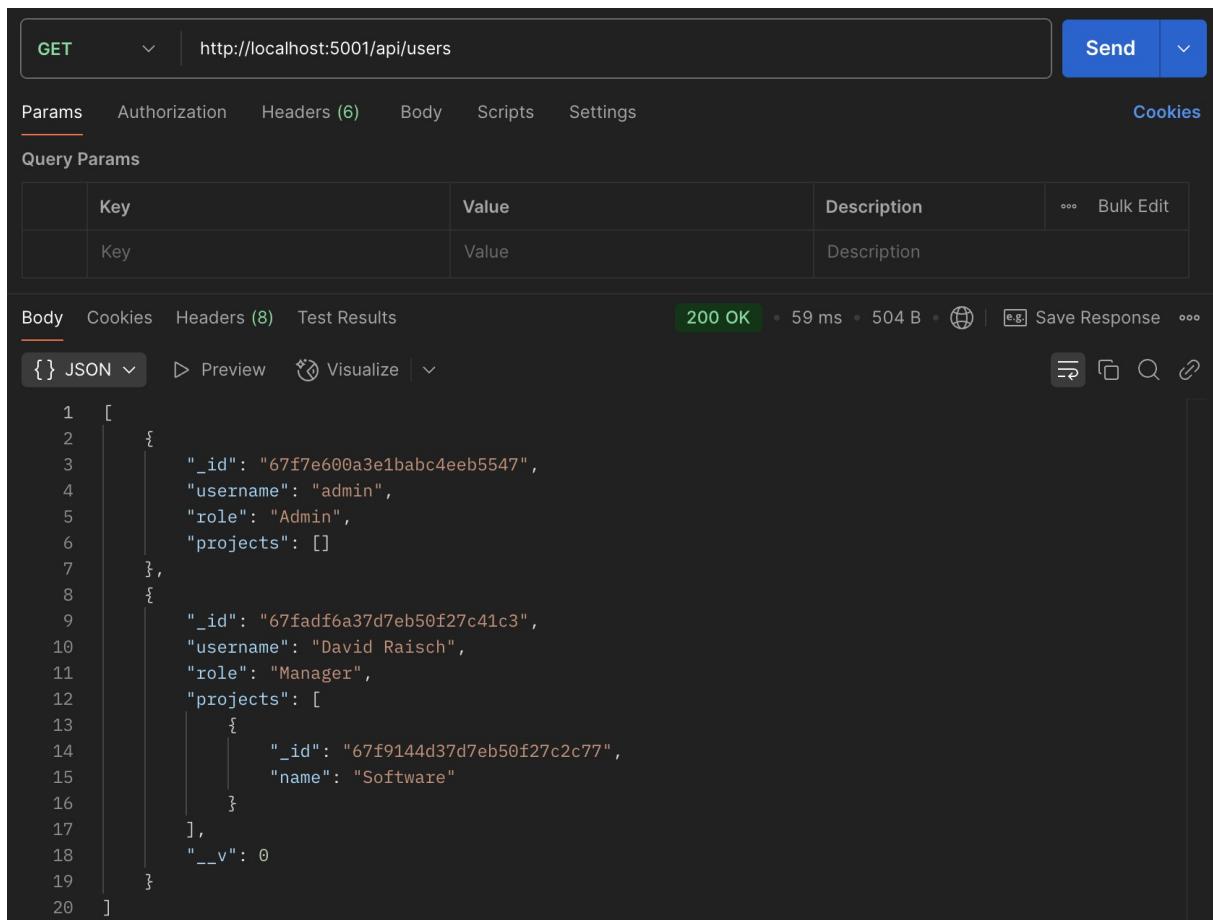
## Sicherstellung der Passwort-Ausblendung bei Benutzerabfragen

Neben der sicheren Speicherung ist es ebenso essenziell, dass Passwörter bei der Abfrage von Benutzerdaten nicht versehentlich mitgeliefert werden. Hierzu wurde ein Backend-Endpunkt implementiert, der beim Abruf aller Benutzer das Passwortfeld explizit ausblendet. Dies wird durch den Aufruf „User.find(, password: 0 )“ erreicht, wie in Code 4.5 dargestellt. Die Abfrage nutzt das sogenannte Projections-Feature von Mongoose, wobei password: 0 bedeutet, dass das Feld „password“ nicht in das Resultat aufgenommen wird.

```
1 app.get('/api/users', async (req, res) => {
2   try {
3     const users = await User.find({}, { password: 0 }).populate(
4       'projects', 'name');
5     res.json(users);
6   } catch (err) {
7     res.status(500).json({ error: 'Error retrieving users' });
8   }
9 });
```

Code 4.5: Die Get-Abfrage für die Benutzer

Eine praktische Überprüfung erfolgte mittels einer GET-Anfrage über Postman. Die Rückgabe zeigte erwartungsgemäß nur die öffentlichen Felder der Benutzer – ohne Passwort (siehe Abbildung 4.3). Dies stellt sicher, dass selbst bei internen Abfragen keine sensiblen Informationen unbeabsichtigt preisgegeben werden und unterstützt somit die Prinzipien von Datensparsamkeit und Privacy-by-Design.



The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: http://localhost:5001/api/users
- Headers: (6)
- Body: (JSON) - Contains the following JSON response:

```

1  [
2   {
3     "_id": "67f7e600a3e1bab4eeb5547",
4     "username": "admin",
5     "role": "Admin",
6     "projects": []
7   },
8   {
9     "_id": "67fadf6a37d7eb50f27c41c3",
10    "username": "David Raisch",
11    "role": "Manager",
12    "projects": [
13      {
14        "_id": "67f9144d37d7eb50f27c2c77",
15        "name": "Software"
16      }
17    ],
18    "__v": 0
19  }
20 ]
  
```

Test Results: 200 OK • 59 ms • 504 B • [Save Response](#)

Abbildung 4.3: Die Überprüfung der Get-Abfrage der Benutzer mit Postman

## 4.3 Start-Seite des Process-Managers

Nach erfolgreichem Login gelangen Benutzer auf die Start-Seite des Process-Managers, die als zentrale Anlaufstelle für sämtliche Interaktionen innerhalb der Anwendung dient. Ein exemplarischer Screenshot dieser Home-Seite aus Sicht eines regulären Benutzers ist in Abbildung 4.4 dargestellt.

Die Benutzeroberfläche ist klar strukturiert und enthält alle wesentlichen Funktionen, die zur Nutzung des Systems erforderlich sind. Im oberen Bereich befindet sich eine Navigationsleiste, über die alle Hauptfunktionen des Process-Managers direkt erreichbar sind. Diese Navigation umfasst unter anderem:

- den Namen des Systems (Process-Manager),
- benutzerspezifische oder administratorbezogene Funktionen,
- eine Benachrichtigungszentrale zur Anzeige wichtiger Systemmeldungen,

- Informationen zum aktuell angemeldeten Benutzer,
- sowie die Möglichkeit zum Logout.

Die Benutzerinformationen umfassen den Benutzernamen, die Rolle innerhalb des Systems sowie eine Übersicht der zugewiesenen Projekte. Neben der Logout-Funktion steht den Nutzern auch eine Option zur Verfügung, ihr Passwort zu ändern.

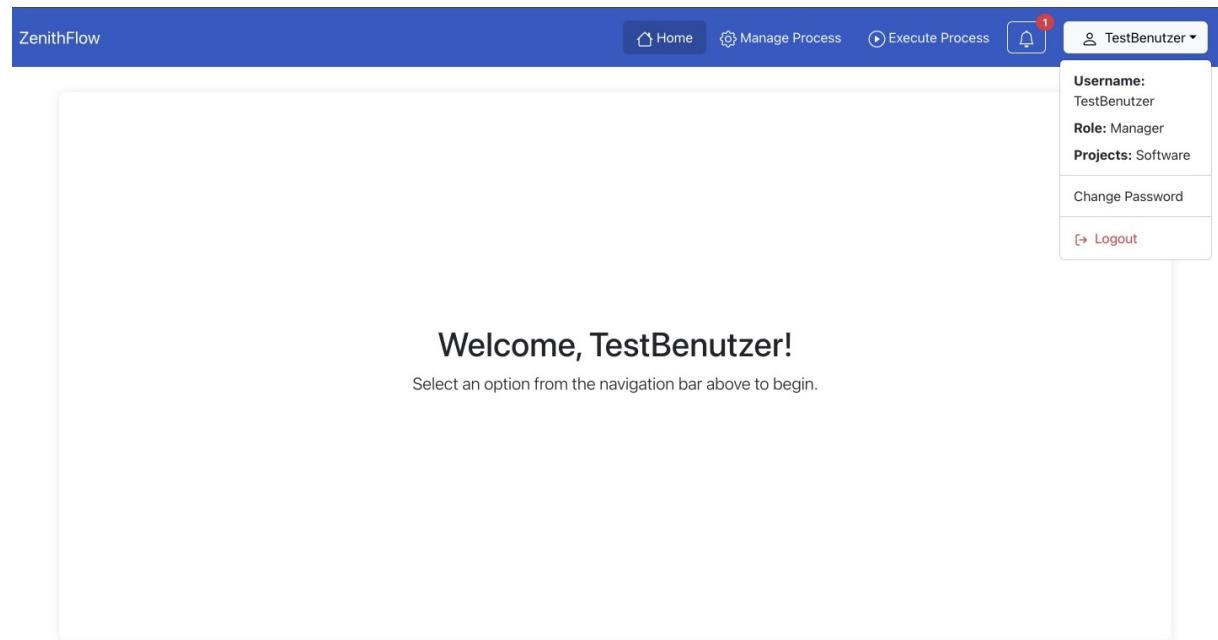


Abbildung 4.4: Die Benutzer Start-Seite

Die genaue Ausgestaltung der Funktionen innerhalb der Navigationsleiste variiert in Abhängigkeit von der Rolle des angemeldeten Nutzers. Reguläre Benutzer erhalten Zugriff auf Funktionen zum Verwalten und Ausführen von Prozessen, um ihre projektbezogenen Aufgaben effizient umzusetzen. Administratoren hingegen haben zusätzlich die Möglichkeit, Nutzer- und Projektverwaltung durchzuführen. Dadurch wird eine klare Trennung zwischen operativen Tätigkeiten und administrativen Aufgaben gewährleistet.

Ein Vergleich der Start-Seite aus Sicht eines Administrators ist im Anhang in Abbildung A3 dargestellt. Die detaillierte Beschreibung der jeweiligen Rollenrechte und Funktionsunterschiede erfolgt in nachfolgenden Kapiteln 4.4 und 4.5.

## 4.4 Funktionen des Admins

In diesem Kapitel werden die Funktionen von Administratoren erläutert. Meldet sich ein Benutzer mit der Rolle „Admin“ an, wird eine spezifische Navigationsleiste angezeigt (siehe Abbildung 4.5), die ausschließlich Administratoren vorbehalten ist.



Abbildung 4.5: Die Navigationsleiste von Administratoren

Diese enthält Bereiche zur Verwaltung von Projekten und Benutzern innerhalb der Process-Manager-Umgebung. Die entsprechenden Seiten werden im Folgenden näher beschrieben.

### 4.4.1 Verwaltung der Projekte

Die Projektverwaltungsseite dient der systematischen Erfassung und Pflege projektbezogener Informationen. Sie ermöglicht es Administratoren, neue Projekte zu erstellen, bestehende Projekte übersichtlich darzustellen und deren Inhalte bei Bedarf anzupassen oder zu löschen. Ziel ist eine einfache, intuitive Verwaltung aller Projektstrukturen innerhalb der Anwendung. Alle Interaktionen erfolgen auf derselben Seite, ohne dass ein Seitenwechsel notwendig ist. Dadurch wird eine effiziente und konsistente Arbeitsweise unterstützt. Die gesamte Projektverwaltungs-Seite ist im Anhang zu sehen (siehe Abbildung A4).

#### Erstellen von Projekten

Im oberen Bereich der Projektverwaltungsseite befindet sich ein Formular zur Neuanlage von Projekten (siehe Abbildung 4.6).

Es umfasst zwei Eingabefelder:

- Projektnamen
- Kurzbeschreibung



Create New Project

Project Name

Project Description

Create Project

Abbildung 4.6: Das Formular zur Erstellung neuer Projekte

Die Schaltfläche „Create Project“ bleibt solange deaktiviert, bis beide Felder mit gültigen Werten ausgefüllt sind. Bei einem Versuch, das Formular mit unvollständigen Angaben abzusenden, erscheint eine Fehlermeldung mit dem Hinweis:

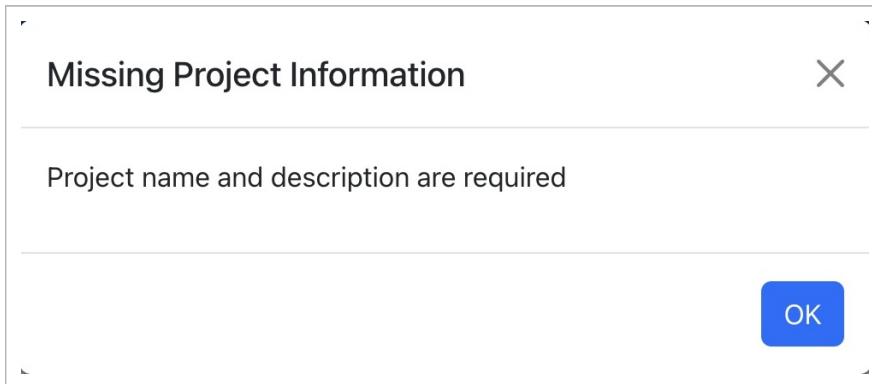


Abbildung 4.7: Die Fehlermeldung des Projekt-Erstellungsprozess bei ungültiger Eingabe

Sobald beide Felder gültige Eingaben enthalten, wird die Schaltfläche aktiv. Ein Klick löst die Methode `handleCreateProject` (siehe Code 4.6) aus, die die eingegebenen Daten validiert und anschließend einen `POST`-Request an die API sendet. Das neue Projekt wird daraufhin in der Datenbank gespeichert. Nach erfolgreichem Abschluss wird die Projektliste automatisch aktualisiert, das Formular zurückgesetzt und eine Bestätigungsmeldung angezeigt. Im Fehlerfall wird eine entsprechende Rückmeldung ausgegeben, wobei unerwartete Probleme im `catch`-Block behandelt werden.

```

1 const handleCreateProject = async () => {
2   if (!newProject.name || !newProject.description) {
3     triggerAlert('Missing Project Information', 'Project name and
4       description are required');
5   }
6   try {
7     await axios.post('http://localhost:5001/api/projects',
8       newProject);
9     fetchProjects();
10    setNewProject({ name: '', description: '' });

```

```

10      triggerAlert('Success', 'Project created successfully');
11  } catch (err) {
12      console.error('Error creating project:', err);
13      triggerAlert('Error', 'Error creating project');
14  }
15 };

```

Code 4.6: Die Funktion zur Erstellung neuer Projekte

## Darstellung erstellter Projekte

Darunter folgt eine Liste aller bestehenden Projekte in einer Tabelle (siehe Abbildung 4.8). Jeder Eintrag enthält:

- **Name** (klickbar für Detailanzeige)
- Drei Aktionsschaltflächen:
  - **Info:** öffnet ein Modal mit der Projektbeschreibung
  - **Edit:** zeigt das Inline-Bearbeitungsformular
  - **Delete:** entfernt das Projekt nach Bestätigung

Existing Projects		
Software	<a href="#">Info</a>	<a href="#">Edit</a>
Engineering	<a href="#">Info</a>	<a href="#">Edit</a>

Abbildung 4.8: Die Tabelle der erstellten Projekten

Diese Darstellung ermöglicht eine schnelle Übersicht und direkte Interaktion mit den einzelnen Projekten, ohne die Seite verlassen zu müssen.

## Verwalten erstellter Projekte

Zur Verwaltung bereits angelegter Projekte stehen in der Projektliste drei Aktionsschaltflächen zur Verfügung: Info, Edit und Delete (siehe ??).

### Projektinformationen anzeigen:

Über die Schaltfläche Info oder alternativ durch einen Klick auf den Projektnamen selbst kann ein modulares Informationsfenster geöffnet werden, das detaillierte Informationen zum

Projekt bereithält. Während in der tabellarischen Übersicht ausschließlich der Projektname aufgeführt ist, zeigt dieses Fenster zusätzlich auch die zugehörige Projektbeschreibung an. So können auch umfassendere Inhalte zum Projektkontext komfortabel abgerufen werden, ohne die Übersicht zu verlassen (siehe Abbildung 4.9).

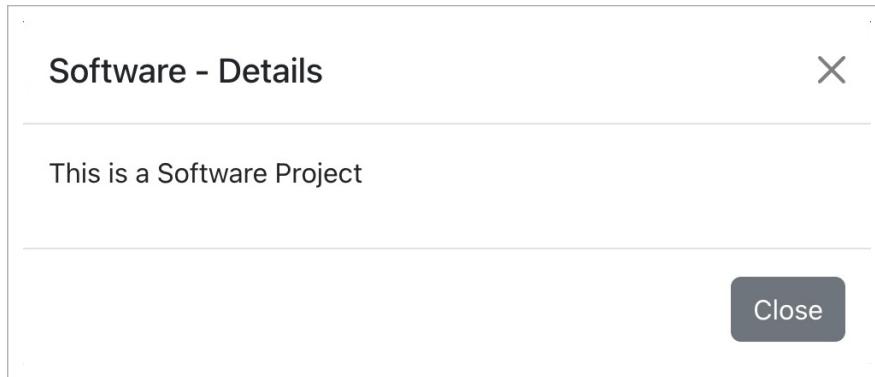
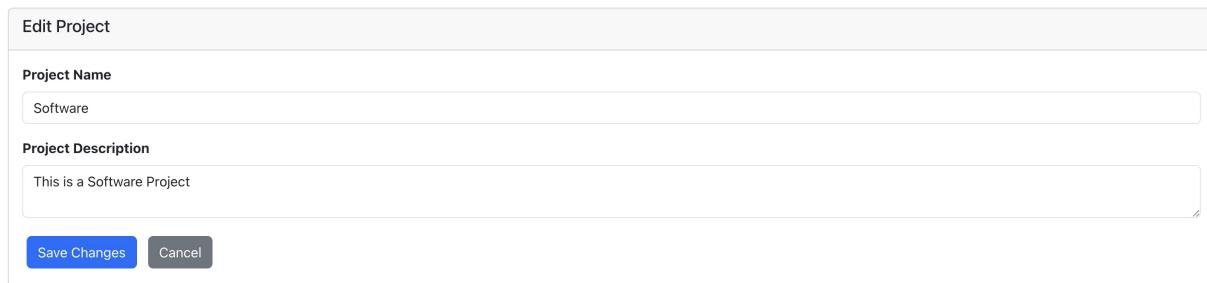


Abbildung 4.9: Das separate Fenster mit Projekt-Informationen

### Projekt bearbeiten:

Ein Klick auf **Edit** öffnet ein Inline-Bearbeitungsformular direkt oberhalb der Projekt-tabelle. Diese Darstellung fügt sich nahtlos in das bestehende Layout ein und bietet den Vorteil, dass der Nutzer weiterhin die Kontextinformationen aller anderen Projekte im Blick behalten kann. Zudem entfällt das Umschalten zwischen verschiedenen Ansichten, was den Bearbeitungsprozess beschleunigt und intuitiver macht (siehe Abbildung 4.10).



Edit Project	
<b>Project Name</b>	Software
<b>Project Description</b>	This is a Software Project
<input type="button" value="Save Changes"/> <input type="button" value="Cancel"/>	

Abbildung 4.10: Das Formular zum Bearbeiten eines erstellten Projekts

Das Formular enthält zwei Pflichtfelder: **Name** und **Description**. Solange eines dieser Felder leer ist, bleibt die Schaltfläche **Save Changes** ausgegraut. Ein Versuch, die Änderungen trotzdem zu speichern, führt zur Anzeige der Meldung wie in Abbildung 4.7

Erfolgt eine gültige Eingabe, wird über die Funktion `handleSaveProject` ein PUT-Request an die API gesendet, um die Änderungen direkt in der Datenbank zu speichern. Der entsprechende API-Aufruf lautet:

```
1 await axios.put(  
2   'http://localhost:5001/api/projects/${editProject._id}',  
3   { name: editProject.name, description: editProject.description }  
4 );
```

Code 4.7: Der Put-Befehl zum Bearbeiten der Projekte

Im Vergleich zur Projektneuanlage, bei der ein POST-Request mit denselben Attributen verwendet wird, unterscheidet sich hier lediglich die HTTP-Methode und das Ziel der Anfrage: Statt ein neues Projekt anzulegen, wird ein bestehendes anhand seiner ID aktualisiert.

### Projekt löschen:

Das Entfernen eines Projekts aus der Datenbank erfolgt über die `Delete`-Schaltfläche. Zur Sicherheit erscheint vor dem Löschvorgang ein Bestätigungsdialog, um unbeabsichtigte Aktionen zu vermeiden. Die genaue Darstellung des Löschdialogs ist im Anhang dokumentiert (siehe Abbildung A5).

Alle drei Verwaltungsaktionen – Informationen einsehen, Bearbeiten und Löschen – greifen auf eine konsistente und reaktive Benutzeroberfläche zurück, die ohne Unterbrechung der Arbeitsumgebung eine vollständige Projektverwaltung erlaubt. Änderungen werden direkt in der Datenbank gespeichert und die Benutzeroberfläche automatisch aktualisiert, wodurch jederzeit eine aktuelle Übersicht über die Projekte innerhalb des Process-Managers gewährleistet ist.

## 4.4.2 Verwaltung der Benutzer

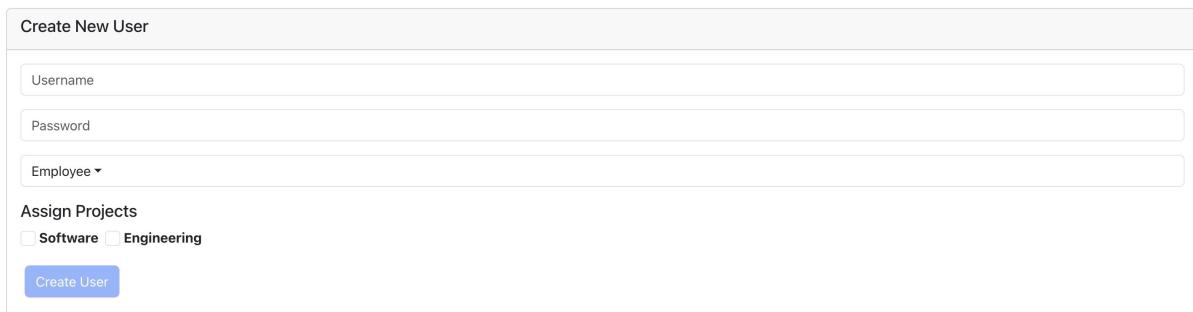
Die Benutzeroberfläche stellt ein zentrales Werkzeug zur Administration von Systemnutzern dar. Sie vereint Funktionalitäten zur Erstellung neuer Benutzer, zur Darstellung und Interaktion mit bestehenden Benutzerkonten sowie zur Durchführung administrativer Aufgaben wie Passwortänderungen, Bearbeitungen und Löschvorgänge von bestehenden Benutzern. Die gesamte Benutzeroberfläche ist im Anhang zu sehen (siehe Abbildung A6).

### Erstellen von Benutzern

Im oberen Bereich der Benutzeroberfläche befindet sich ein Formular zur Erstellung neuer Benutzer (siehe Abbildung 4.11).

Dieses Formular besteht aus folgenden Eingabefeldern:

- Benutzername
- Passwort
- Rolle (Dropdown-Auswahlfeld)
- Projektzuweisung (Mehrfachauswahl)



The screenshot shows a user interface titled "Create New User". It includes input fields for "Username" and "Password". A dropdown menu labeled "Employee" is open. Below these, there is a section titled "Assign Projects" with checkboxes for "Software" and "Engineering". At the bottom is a blue "Create User" button.

Abbildung 4.11: Das Formular zur Erstellung neuer Benutzer

Das Rollenfeld ist als Dropdown-Menü realisiert und ermöglicht die Auswahl aus allen im System vordefinierten Rollen, beispielsweise Admin, Manager oder Employee. Falls keine Rolle explizit ausgewählt wird, greift automatisch der Standardwert „Employee“, wodurch sichergestellt ist, dass jeder neue Benutzer eine gültige Rollenberechtigung erhält.

Die Rolle ist ein zentraler Bestandteil der Benutzeroberfläche, da sie unmittelbar mit den Zugriffsrechten und Handlungsmöglichkeiten innerhalb des Systems verknüpft ist. So

bestimmt die Rolle unter anderem, welche Benutzeroberflächen und Funktionen sichtbar sind, welche Aktionen im System durchgeführt werden dürfen und in welcher Form Entscheidungsprozesse innerhalb des Process-Managers ablaufen. Dies bietet somit eine differenzierte Rechtevergabe bereits zum Zeitpunkt der Benutzeranlage und erlaubt es, die Systemnutzung präzise auf die jeweiligen Verantwortlichkeiten und Aufgabenbereiche abzustimmen. Die Rolle ist somit nicht nur eine organisatorische Einteilung, sondern essentiell für die korrekte technische Ausführung des Process-Managers. Sie steuert sicherheitsrelevante Aspekte ebenso wie die Ausführung logischer Prozesspfade. Die Bedeutung dieser Rollensteuerung wird im weiteren Verlauf der Arbeit – insbesondere bei den Funktionen der Benutzer in Abschnitt 4.5 – noch detaillierter deutlich.

Die Interaktion mit dem Formular ist dynamisch: Die Schaltfläche „Create User“ ist initial deaktiviert und wird nur aktiv, wenn alle Pflichtfelder gültig ausgefüllt sind – darunter zählen der Benutzername, das Password und die Auswahl von mindestens einem Projekt. Diese Validierungslogik wird auf der Client-Seite umgesetzt und bietet somit ein unmittelbares Feedback an den Benutzer. Besonders hervorzuheben ist das Verhalten bei inkorrekt Versuch, einen Benutzer zu erstellen: Sollte der Benutzer versuchen, auf die ausgegraute Schaltfläche zu klicken, wird eine explizite Fehlermeldung angezeigt:

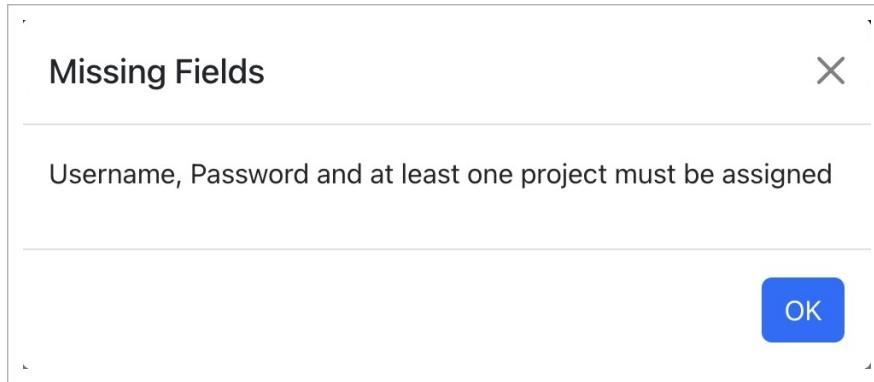


Abbildung 4.12: Die Fehlernachricht des Benutzer-Erstellungsprozess bei ungültiger Eingabe

Diese Maßnahme dient der zusätzlichen Absicherung und Nutzerführung, selbst bei Umgehung der UI-Vorgaben.

Die Validierung sowie die serverseitige Verarbeitung bei der Erstellung eines neuen Benutzers erfolgt durch die Funktion `handleCreateUser`, welche die gesamte Logik zur Prüfung und Übertragung der Eingabedaten implementiert:

```

1  const handleCreateUser = async () => {
2      if (!newUser.username || !newUser.password || newUser.projects
3          .length === 0) {
4          triggerAlert(
5              'Missing Fields',
6              'Username, Password and at least one project must be
7              assigned'
8          );
9          return;
10     }
11     const uniqueProjects = [...new Set(newUser.projects)];
12     try {
13         await axios.post('http://localhost:5001/api/users', { ...
14             newUser, projects: uniqueProjects });
15         fetchUsers();
16         setNewUser({ username: '', role: 'Employee', password: '',
17             projects: [] });
18         triggerAlert('Success', 'User created successfully');
19     } catch (error) {
20         console.error('Error creating user:', error);
21         triggerAlert('Error', 'Error creating user');
22     }
23 };

```

Code 4.8: Die Funktion zur Erstellung neuer Benutzer

Zu Beginn prüft die Funktion, ob alle notwendigen Eingabefelder (Benutzername, Passwort und mindestens ein Projekt) ausgefüllt wurden. Ist dies nicht der Fall, wird durch einen Aufruf der Methode `triggerAlert` eine Fehlermeldung generiert, wie sie in Abbildung Abbildung 4.12 dargestellt ist.

Erst bei vollständiger Eingabe erfolgt die Übergabe der Daten mittels POST-Anfrage an die API-Schnittstelle zur serverseitigen Speicherung in der Datenbank (siehe Code 4.8 in Zeile 11). Nach erfolgreicher Erstellung wird die Benutzerliste aktualisiert, die Eingabemaske geleert und eine Bestätigungsmeldung angezeigt.

Unerwartete Fehler im Übertragungsprozess – wie Netzwerkprobleme oder serverseitige Validierungsfehler – werden durch den `catch`-Block abgefangen und ebenfalls dem Benutzer über eine Meldung angezeigt. Damit wird sowohl eine robuste Fehlerbehandlung als auch ein konsistentes Feedback an den Benutzer sichergestellt.

## Darstellung erstellter Benutzer

Unterhalb des Benutzerformulars wird eine tabellarische Übersicht aller bereits angelegten Benutzer dargestellt. Die Tabelle bietet eine zentrale Verwaltungsansicht und zeigt folgende Informationen:

- **Username:** Der Name des Benutzers.
- **Role:** Die dem Benutzer zugewiesene Rolle (z. B. Admin, Manager oder Employee).
- **Projects:** Die zugeordneten Projekte.
- **Actions:** Schaltflächen zur Bearbeitung, Passwortänderung und zum Löschen.

Existing Users			
Username	Role	Projects	Actions
admin	Admin	None	<a href="#">Edit</a> <a href="#">Change Password</a> <a href="#">Delete</a>
TestBenutzer	Manager	Software	<a href="#">Edit</a> <a href="#">Change Password</a> <a href="#">Delete</a>

Abbildung 4.13: Die Tabelle der erstellten Benutzer

Diese kompakte Übersicht erlaubt eine schnelle Beurteilung und Verwaltung der Benutzerinformationen. Die Rollenanzeige hilft insbesondere dabei, die Aufgabenverteilung im System zu überblicken.

## Verwalten erstellter Benutzer

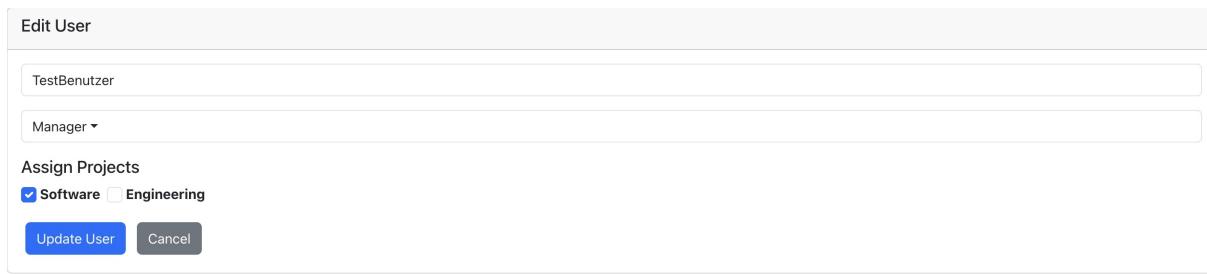
Neben jeder Benutzerzeile befinden sich drei Aktionsschaltflächen (siehe Abbildung 4.13): Bearbeiten, Password ändern und Löschen des Benutzers.

### Bearbeiten:

Wird die **Edit**-Schaltfläche angeklickt, erscheint oberhalb der Benutzertabelle ein Bearbeitungsformular. Anders als bei der Neuanlage ist dieses nicht direkt in der Tabelle eingebettet, sondern als eigenständiger Block angeordnet (siehe Abbildung 4.14).

Im Formular können folgende Felder bearbeitet werden:

- Benutzername (Textfeld)
- Rolle (Dropdown-Menü)
- Projekte (Mehrfachauswahl mit Checkboxen)



The screenshot shows a user editing interface. At the top, it says "Edit User". Below that is a text input field containing "TestBenutzer". Underneath is a dropdown menu set to "Manager". A section titled "Assign Projects" contains two checkboxes: "Software" (which is checked) and "Engineering" (which is unchecked). At the bottom of the form are two buttons: a blue "Update User" button and a grey "Cancel" button.

Abbildung 4.14: Das Formular zum Bearbeiten eines erstellten Benutzers

Nach dem Anpassen wird über die Schaltfläche `Update User` der aktualisierte Datensatz an den Server gesendet. Dazu wird die Funktion `handleUpdateUser` aufgerufen. Diese überprüft zunächst, ob mindestens ein Projekt ausgewählt wurde. Anschließend wird ein PUT-Request mit den aktualisierten Daten an die API gesendet, wodurch die Änderungen in der Datenbank gespeichert werden. Im Gegensatz zur Neuanlage (`handleCreateUser`, welche einen POST-Request verwendet), greift `handleUpdateUser` auf dieselbe Datenstruktur zurück, ersetzt jedoch lediglich vorhandene Werte – die Funktionalität bleibt in großen Teilen gleich.

Besonderheit beim Bearbeiten: Es können alle Benutzerattribute geändert werden – inklusive Benutzername, Rolle und zugewiesene Projekte –, mit Ausnahme des Passworts. Letzteres liegt in einem eigenen Formularbereich, sodass es nicht bei jeder kleineren Änderung (z.B. einer Projektzuweisung) erneut eingegeben werden muss. Das Bearbeiten aller Benutzer-Attribute funktioniert, da der Benutzer eindeutig über seine ID identifiziert wird. Auch der Benutzername kann dadurch ohne Probleme aktualisiert werden. Siehe hierzu den verwendeten PUT-Befehl im Code 4.9:

```

1 await axios.put(
2   'http://localhost:5001/api/users/${editingUser._id}',
3   { ...editingUser, projects: uniqueProjects }
4 );

```

Code 4.9: Der Put-Befehl zum Bearbeiten der Benutzer

### Passwort ändern:

Über die Schaltfläche `Change Password` kann ein neues Passwort vergeben werden. Dazu erscheint ein separates Eingabefeld für das neue Passwort (siehe Abbildung 4.15). Diese Funktion ist bewusst ausgelagert und nicht Teil der allgemeinen Bearbeitungsmaske.

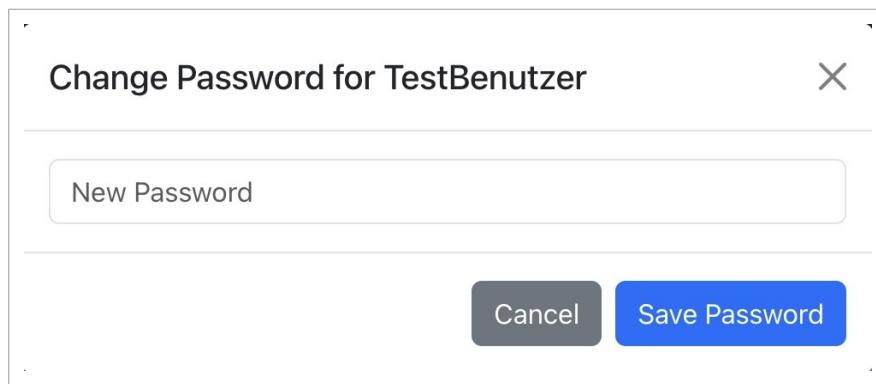


Abbildung 4.15: Das separate Eingabefeld zum Password ändern

### Löschen:

Beim Klick auf **Delete** wird der entsprechende Benutzer aus der Datenbank entfernt. Zum Schutz vor unbeabsichtigtem Löschen erfolgt vorab eine Sicherheitsabfrage (siehe Abbildung A7).

Die dargestellten Funktionen ermöglichen eine vollständige Benutzerverwaltung direkt aus der Hauptansicht, ohne dass zusätzliche Navigationsschritte erforderlich sind.

## Fazit

Die Funktionen zur Projekt- und Benutzerverwaltung bilden das Rückgrat der administrativen Steuerung im Process-Manager. Sie sichern eine klare Struktur, rollenbasierte Rechtevergabe und effiziente Prozessabläufe im System.

## 4.5 Funktionen des Benutzers

In diesem Kapitel werden die Funktionen der Benutzer erläutert. Für Benutzer der Rolle „Employee“ oder „Manager“ sieht die Navigationsleiste wie in Abbildung 4.16 aus.



Abbildung 4.16: Die Navigationsleiste von Benutzer

Diese enthält Bereiche für das Erstellen und Verwalten von Prozessen, sowie dem Ausführen dieser Prozesse. Außerdem Benachrichtigungen der Benutzer und spezifische Benutzerinformationen. Die entsprechenden Seiten werden im Folgenden näher beschrieben.

### 4.5.1 Benutzerinformationen

Im oberen rechten Bereich der Navigationsleiste befindet sich ein Dropdown-Menü, welches dem angemeldeten Benutzer eine kompakte Übersicht über seine persönlichen Informationen bietet. Im Dropdown-Knopf selbst wird der Benutzername angezeigt. Nach dem Öffnen erscheinen zusätzlich die Rolle des Benutzers sowie eine Liste der ihm zugewiesenen Projekte. Dadurch erhält der Benutzer unmittelbaren Zugriff auf seine kontextbezogenen Systemberechtigungen.

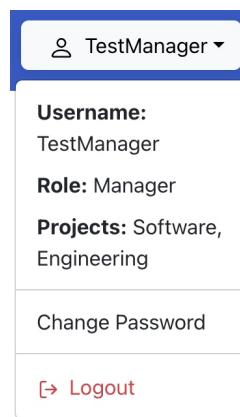
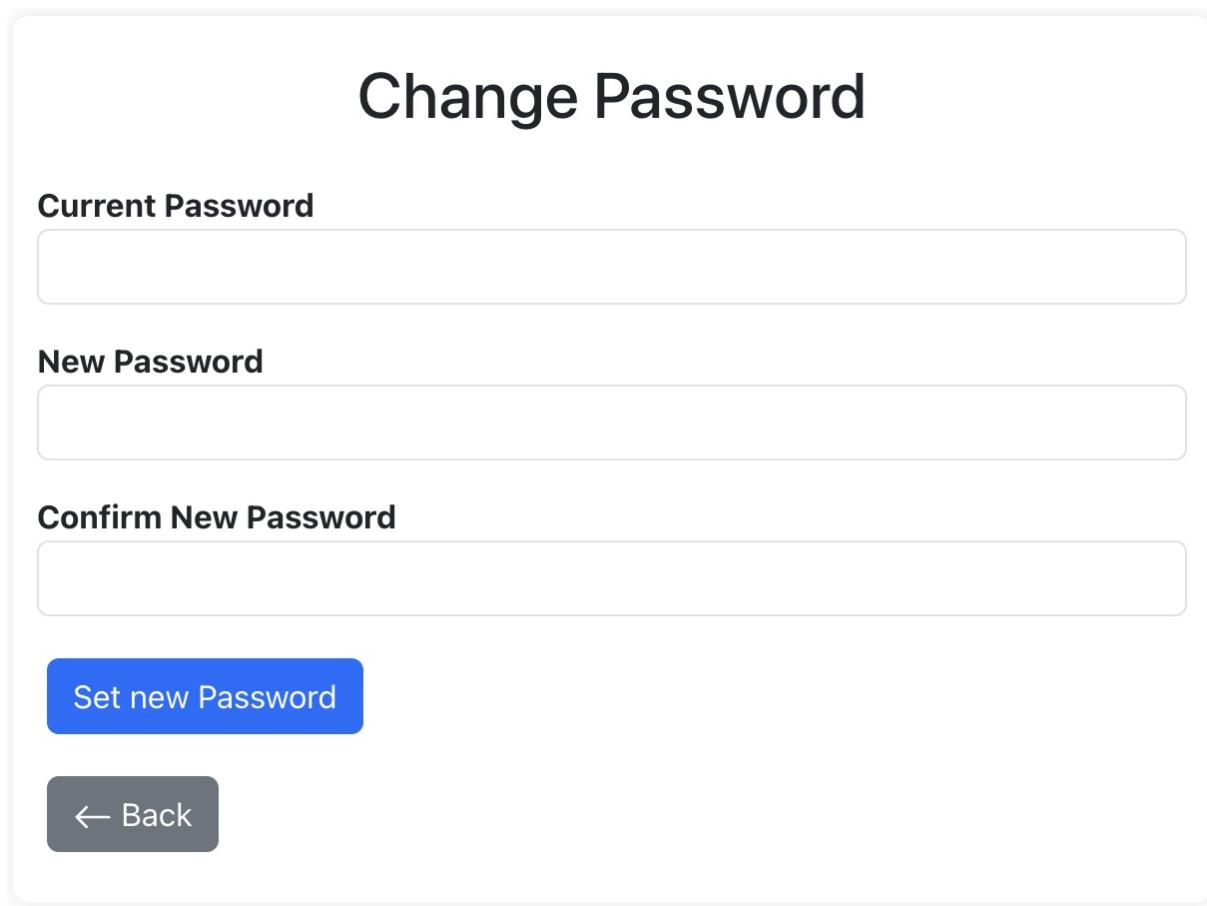


Abbildung 4.17: Das Dropdown-Feld mit den Benutzerinformationen

Zusätzlich enthält das Dropdown-Menü eine Option zur Passwortänderung („Change Password“) sowie einen Logout-Button, um die Sitzung sicher zu beenden. Diese Funktionen verbessern nicht nur die Benutzerfreundlichkeit, sondern fördern auch die Sicherheit und Selbstverwaltung im System.

## Passwortänderung durch den Benutzer

Die Möglichkeit für Benutzer, ihr Passwort selbstständig zu ändern, stellt eine wichtige Funktion zur Erhöhung der Systemsicherheit und zur Reduktion administrativer Aufwände dar. Die Anwendung bietet dafür eine separate Oberfläche, auf der der Benutzer sein aktuelles Passwort eingeben und anschließend ein neues Passwort definieren sowie bestätigen kann. Visuelle Rückmeldungen in Form von Erfolgs- oder Fehlermeldungen geben dem Benutzer jederzeit Auskunft über den Status seiner Eingabe. Zusätzlich ermöglicht ein Rückkehr-Button die einfache Navigation zurück zur vorherigen Ansicht.



**Change Password**

**Current Password**

**New Password**

**Confirm New Password**

**Set new Password**

**← Back**

Abbildung 4.18: Die Passwortänderungsseite für den Benutzer

Die Funktion stellt sicher, dass Passwörter nicht nur geändert, sondern auch verifiziert werden. Dies geschieht insbesondere durch die Prüfung, ob das neu eingegebene Passwort mit der Bestätigung übereinstimmt oder ob das aktuell eingegebene Passwort korrekt ist. Entsprechende Fehlermeldungen (siehe Abbildung A8 und Abbildung A9) verhindern dabei falsche oder inkonsistente Eingaben, bevor eine Kommunikation mit dem Server erfolgt.

Die Validierung der Passwortänderung erfolgt auf Serverseite über folgende REST-Endpunkt-Definition:

```

1 app.put('/api/users/:id/changePassword', async (req, res) => {
2   const { currentPassword, newPassword } = req.body;
3   try {
4     const user = await User.findById(req.params.id);
5     if (!user) return res.status(404).json({ error: 'User not
6       found.' });
7     // Compare the provided current password with the stored hash
8     const isMatch = await bcrypt.compare(currentPassword, user.
9       password);
10    if (!isMatch) return res.status(400).json({ error: 'Current
11      password is incorrect.' });
12    // Update the password
13    const salt = await bcrypt.genSalt(10);
14    user.password = await bcrypt.hash(newPassword, salt);
15    await user.save();
16    res.json({ message: 'Password updated successfully.' });
17  }
18});

```

Code 4.10: Die Put-Abfrage zum Ändern des Benutzer Password

Die PUT-Methode erwartet in der Anfrage den aktuellen sowie den neuen Passwortwert. Anschließend wird versucht, den Benutzer über die angegebene ID aus der Datenbank zu laden. Wird der Benutzer nicht gefunden, wird mit einem Statuscode 404 reagiert.

Im nächsten Schritt wird das aktuell angegebene Passwort mit dem in der Datenbank gespeicherten Passwort-Hash verglichen. Falls die Passwörter nicht übereinstimmen, erfolgt eine Rückmeldung mit einem Statuscode 400 und einer eindeutigen Fehlermeldung (siehe Abbildung A8).

Wurde das Passwort korrekt eingegeben, wird im nächsten Schritt das neue Passwort mithilfe von `bcrypt` gehasht und in der Datenbank gespeichert. Der Benutzer erhält daraufhin eine Bestätigung, dass das Passwort erfolgreich geändert wurde (siehe Abbildung A10).

Sollten während dieses Prozesses unerwartete Fehler auftreten, etwa bei der Kommunikation mit der Datenbank, so werden diese serverseitig protokolliert und dem Benutzer wird eine generische Fehlermeldung mit Statuscode 500 zurückgegeben.

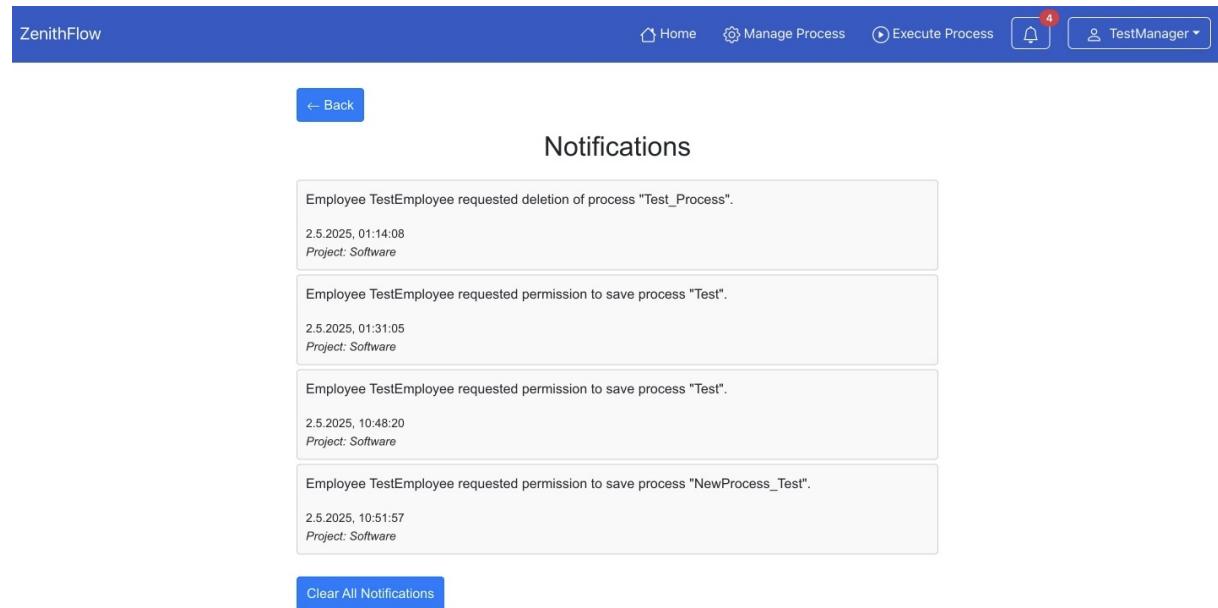
Die Kombination aus nutzerfreundlichem Frontend und sicherem Backend gewährleistet eine effektive und gleichzeitig geschützte Passwortverwaltung innerhalb der Anwendung. Die konkrete Fehlerbehandlung – sowohl bei nicht übereinstimmenden neuen Passwörtern im Frontend als auch bei falschen aktuellen Passwörtern im Backend – erhöht zusätzlich die Robustheit der Funktion.

## 4.5.2 Benachrichtigungen

Ein zentraler Bestandteil der Funktionen „Manage Process“ und „Execute Process“ des Process Managers ist die Kommunikation zwischen den beteiligten Benutzerrollen, insbesondere zwischen Manager und Employee. Um diesen Informationsaustausch zu unterstützen, wurde ein integriertes Benachrichtigungssystem implementiert.

Die Benutzeroberfläche der Benachrichtigungsfunktion ist übersichtlich gestaltet. Der Zugriff erfolgt über ein Glockensymbol in der Navigationsleiste (vgl. Abbildung 4.16), welches zusätzlich die Anzahl Benachrichtigungen anzeigt.

Im Mittelpunkt der Benachrichtigungsseite steht eine Liste mit allen Benachrichtigungen, die einem Benutzer zugewiesen sind. Ergänzt wird die Seite durch Interaktionselemente wie einen „Back“-Button zur Navigation sowie einen Button „Clear all Notifications“, mit dem sämtliche Einträge auf einmal entfernt werden können.



The screenshot shows the 'Notifications' page of the ZenithFlow application. At the top, there is a blue header bar with the 'ZenithFlow' logo and navigation links for 'Home', 'Manage Process', 'Execute Process', and 'TestManager'. The 'TestManager' link has a red notification badge with the number '1'. Below the header, there is a back button labeled '← Back' and a title 'Notifications'. The main content area displays four notifications in a list:

- Employee TestEmployee requested deletion of process "Test\_Process".  
2.5.2025, 01:14:08  
Project: Software
- Employee TestEmployee requested permission to save process "Test".  
2.5.2025, 01:31:05  
Project: Software
- Employee TestEmployee requested permission to save process "Test".  
2.5.2025, 10:48:20  
Project: Software
- Employee TestEmployee requested permission to save process "NewProcess\_Test".  
2.5.2025, 10:51:57  
Project: Software

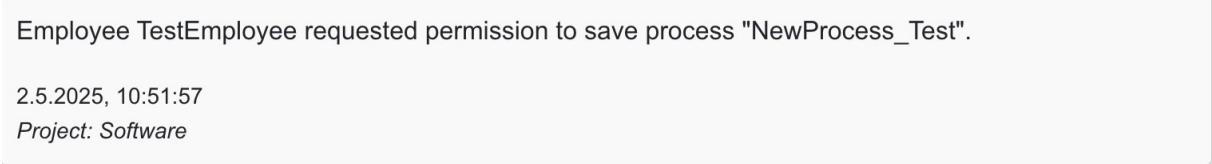
At the bottom of the list is a blue button labeled 'Clear All Notifications'.

Abbildung 4.19: Die Benachrichtigung-Seite für alle Benutzer

Es werden zwei Typen von Benachrichtigungen unterschieden:

- **Anfragen:** Diese dienen primär der Information und leiten den Benutzer bei Auswahl direkt an die entsprechende Stelle im System weiter. Dort kann beispielsweise automatisch der relevante Prozess oder die zugehörige Instanz vorausgewählt sein, sodass unmittelbar mit der Bearbeitung begonnen werden kann. Eine separate Interaktion

innerhalb der Benachrichtigung selbst ist nicht erforderlich. Nach Abschluss der zugehörigen Aktion wird die Benachrichtigung automatisch entfernt.



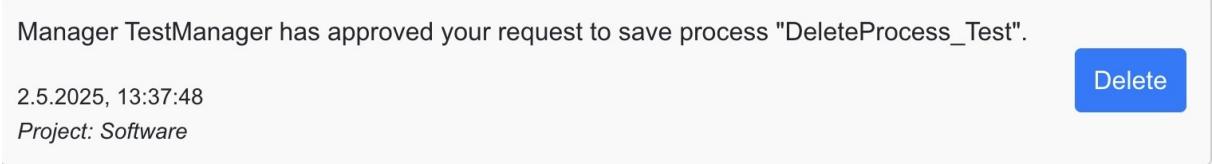
Employee TestEmployee requested permission to save process "NewProcess\_Test".

2.5.2025, 10:51:57

Project: Software

Abbildung 4.20: Eine Benachrichtigung des Typs Anfrage

- **Rückmeldungen:** Diese besitzen informativen Charakter und beinhalten eine Schaltfläche zum Löschen. Auch hier ist eine Weiterleitung durch Anklicken möglich, allerdings folgt daraus keine weitere verpflichtende Aktion im System.



Manager TestManager has approved your request to save process "DeleteProcess\_Test".

2.5.2025, 13:37:48

Project: Software

Delete

Abbildung 4.21: Eine Benachrichtigung des Typs Anfrage

## Struktur und Filtermechanismus der Benachrichtigungen

Die Zuweisung von Benachrichtigungen im Process Manager erfolgt gezielt anhand der Benutzerrolle sowie der Projektzugehörigkeit. Ein Benutzer erhält ausschließlich solche Benachrichtigungen, die sowohl für seine Rolle als auch für die ihm zugewiesenen Projekte bestimmt sind. Dieses Filterprinzip gewährleistet eine effektive Informationsverteilung an die relevanten Zielgruppen und verhindert zugleich eine unnötige Informationsflut. So kann beispielsweise durch eine Bestätigungsbenachrichtigung sichergestellt werden, dass alle betroffenen *Employees* eines Projekts zeitnah über den aktuellen Prozessstatus informiert werden und ihre weiteren Arbeitsschritte entsprechend anpassen können.

### Benötigte Attribute einer Benachrichtigung

Zur Umsetzung dieses selektiven Benachrichtigungsmechanismus ist jede Benachrichtigung mit spezifischen Attributen versehen:

- **message:** Der eigentliche Textinhalt der Benachrichtigung, der dem Benutzer angezeigt wird.

- **timestamp:** Zeitstempel der Benachrichtigung. Dieser ist wichtig, um den Zeitpunkt des Ereignisses nachvollziehen und die Informationen zeitlich einordnen zu können.
- **requestedBy:** Der Name oder die Identifikation des Benutzers, der die Benachrichtigung ausgelöst hat.
- **requestedById:** Die eindeutige Benutzer-ID des sendenden Benutzers zur internen Referenzierung und Rückverfolgbarkeit.
- **targetRole:** Die Zielrolle, für welche die Benachrichtigung bestimmt ist. Dies stellt sicher, dass nur berechtigte Benutzergruppen Zugriff auf die Information erhalten. Dadurch wird sowohl dem Datenschutz als auch der Übersichtlichkeit Rechnung getragen.
- **project:** Das zugehörige Projekt, auf das sich die Benachrichtigung bezieht. Auf diese Weise erhalten ausschließlich Projektbeteiligte Zugriff auf die entsprechende Mitteilung.

```

1 const notificationSchema = new mongoose.Schema({
2   message: { type: String, required: true },
3   timestamp: { type: Date, default: Date.now },
4   instanceId: { type: mongoose.Schema.Types.ObjectId, ref: 'Instance' },
5   requestType: { type: String, enum: ['save', 'delete'], default: 'save' },
6   requestedBy: { type: String, required: true },
7   requestedById: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
8   targetRole: { type: String, enum: ['Admin', 'Manager', 'Employee'], default: 'Manager' },
9   status: { type: String, enum: ['pending', 'approved', 'dismissed'], default: 'pending' },
10  project: { type: mongoose.Schema.Types.ObjectId, ref: 'Project', required: true },
11  processName: { type: String },
12  xml: { type: String },
13  processId: { type: mongoose.Schema.Types.ObjectId, ref: 'Process' }
14 });
15 const Notification = mongoose.model('Notification', notificationSchema);

```

Code 4.11: Das Datenbank-Schema für die Benachrichtigungen

Zwar existieren weitere Attribute im zugrunde liegenden Datenbankschema (vgl. Code 4.11), diese sind jedoch eher für spezifische Anwendungsfälle relevant und werden daher in späteren Kapiteln im Zusammenhang mit der konkreten Implementierung und Einbindung der Benachrichtigungsfunktion näher erläutert.

### **Filterung und Sichtbarkeit**

Die Zugriffskontrolle auf Benachrichtigungen erfolgt in zwei Stufen. Zunächst wird überprüft, ob die Zielrolle der Benachrichtigung mit der Rolle des aktuell angemeldeten Benutzers übereinstimmt. Anschließend erfolgt eine weitere Filterung basierend auf den Projekten, denen der Benutzer zugewiesen ist. Nur wenn beide Kriterien erfüllt sind, wird dem Benutzer die entsprechende Benachrichtigung angezeigt. Dieses Vorgehen stellt sicher, dass jede Benachrichtigung ausschließlich für die vorgesehenen Empfänger sichtbar und relevant ist.

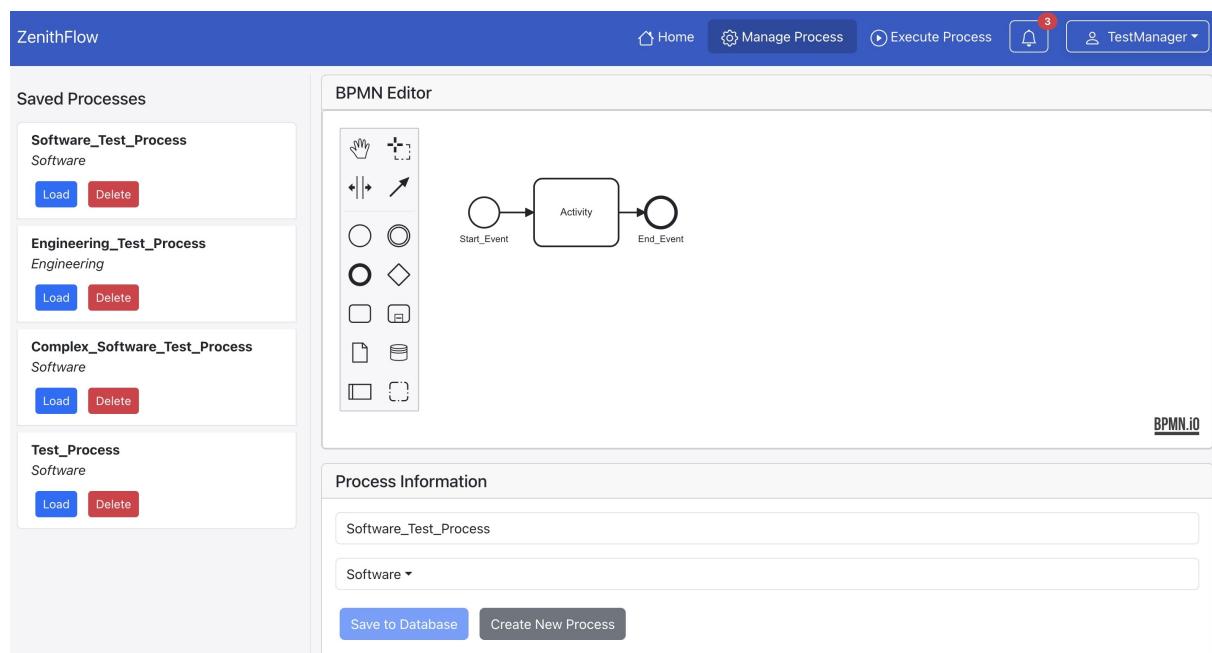
### **Fazit**

Das Benachrichtigungssystem stellt eine zentrale Komponente des Process Managers dar. Es unterstützt das Zusammenspiel zwischen *Manager* und *Employee* und bildet damit eine wesentliche Grundlage für die effiziente Umsetzung der Prozess-Funktionen.

### 4.5.3 Prozesse verwalten

Die Komponente **ManageProcess** dient der ganzheitlichen Verwaltung von BPMN-Prozessen und verfolgt das Ziel, Anwendern eine intuitive Oberfläche für Erstellung, Anpassung und Löschung von Workflows zu bieten, bei gleichzeitiger Sicherstellung der Datenintegrität und rollenbasierten Freigabeprozesse.

Die Benutzeroberfläche gliedert sich in zwei zentrale Bereiche: Rechts befindet sich der Diagrammeditor, der auf **bpmn-js** basiert. Links befindet sich eine einfache Liste aller gespeicherten Prozesse, die über Load- und Delete-Buttons bearbeitet werden kann.



The screenshot shows the 'Manage Process' section of the ZenithFlow application. On the left, there's a sidebar titled 'Saved Processes' listing four entries: 'Software\_Test\_Process' (Software), 'Engineering\_Test\_Process' (Engineering), 'Complex\_Software\_Test\_Process' (Software), and 'Test\_Process' (Software). Each entry has 'Load' and 'Delete' buttons. The main area is titled 'BPMN Editor' and contains a canvas with a simple process flow: a 'Start\_Event' (solid circle) connected to an 'Activity' (rectangle) which is then connected to an 'End\_Event' (solid circle). To the left of the canvas is a toolbar with icons for selection, creation, and deletion. Below the canvas is a 'Process Information' panel containing fields for 'Name' (set to 'Software\_Test\_Process') and 'Category' (set to 'Software'). At the bottom of this panel are 'Save to Database' and 'Create New Process' buttons. In the top right corner of the main area, there's a small 'BPMN.io' logo.

Abbildung 4.22: Übersicht über die ManageProcess-Seite

### Bibliothek bpmn-js

Die im Rahmen dieses Projekts eingesetzte BPMN-Bibliothek **bpmn-js** bildet die technologische Grundlage für die modellgetriebene, Drag-and-Drop-basierte Erstellung von Geschäftsprozessen. Sie stellt eine deklarative Canvas-Oberfläche bereit, die sämtliche Standard-BPMN-Elemente unterstützt und eine benutzerfreundliche Steuerung mittels Drag-and-Drop ermöglicht. Darüber hinaus verfügt die Bibliothek über ein modulares Erweiterungssystem, das über das Konstrukt **additionalModules** implementiert ist. Dieses erlaubt die funktionale Erweiterung und Individualisierung der Bibliothek, was eine

zentrale Voraussetzung für die effektive Integration in den entwickelten Process Manager darstellt.

Ein weiterer wesentlicher Bestandteil der Bibliothek ist die EventBus-Integration, über die Änderungen am Modell in Echtzeit erkannt und verarbeitet werden können. Dies erfolgt beispielsweise durch das Ereignis `commandStack.changed`, das bei jeder Modelländerung ausgelöst wird und so eine dynamische Synchronisation mit anderen Systemkomponenten ermöglicht.

Im vorliegenden Projekt werden zwei funktionale Erweiterungen über die Dateien „`customRules`“ und „`customExtensions`“ realisiert (siehe Code A2 und Code A3). Die erste Erweiterung betrifft die Modellkonsistenz: Das Start-Ereignis eines Prozesses kann nicht gelöscht werden. Dies stellt sicher, dass jedes Prozessmodell über einen eindeutig definierten Startpunkt verfügt – eine grundlegende Anforderung an jede BPMN-konforme Modellierung. Die zweite Erweiterung betrifft die semantische Anreicherung der Prozessmodelle. Hierzu werden zusätzliche benutzerdefinierte Attribute – namentlich „Rolle“ und „Beschreibung“ – für zentrale BPMN-Elemente (z. B. Events, Aktivitäten, Gateways) eingeführt. Die Bedeutung dieser zusätzlichen Informationen sowie deren Notwendigkeit im Kontext des Process Managers werden in einem späteren Abschnitt ausführlich erläutert.

Zusammenfassend zeigt sich, dass die Verwendung von `bpmn-js` sowohl aus technischer als auch aus wirtschaftlicher Perspektive sinnvoll ist. Zum einen reduziert sich der Entwicklungsaufwand erheblich, da eine etablierte Bibliothek mit bestehender BPMN-Regelunterstützung genutzt wird. Zum anderen bleibt die Bibliothek trotz ihrer Standardfunktionalitäten offen für individuelle Erweiterungen, etwa zur Sicherstellung eines festen Start-Ereignisses oder zur Ergänzung zusätzlicher Attribute. Ein entscheidender Vorteil liegt dabei in den benutzerspezifischen Erweiterungsmöglichkeiten, die eine nachhaltige und zukunftssichere Lösung ermöglichen: Neue Anforderungen oder Funktionen können flexibel integriert werden, ohne in bestehende Strukturen einzugreifen. Die Weiterentwicklung des Systems wird dadurch nicht eingeschränkt, sondern im Gegenteil deutlich erleichtert. Darüber hinaus überzeugt `bpmn-js` durch eine hohe Performance und plattformübergreifende Kompatibilität. Diese Qualitäten werden durch eine große und aktive Entwicklergemeinschaft sowie eine umfassende Dokumentation gestützt. `bpmn-js` erweist sich somit als geeignete, leistungsfähige und anpassbare Grundlage für die langfristige Prozessmodellierung im Rahmen des entwickelten Process Managers.

Wählt ein Benutzer einen Prozess aus der Liste aus und bestätigt dies über die Schaltfläche „Load“, wird die zugehörige BPMN-XML-Repräsentation in den Editor geladen (vgl. Abbildung 4.23).

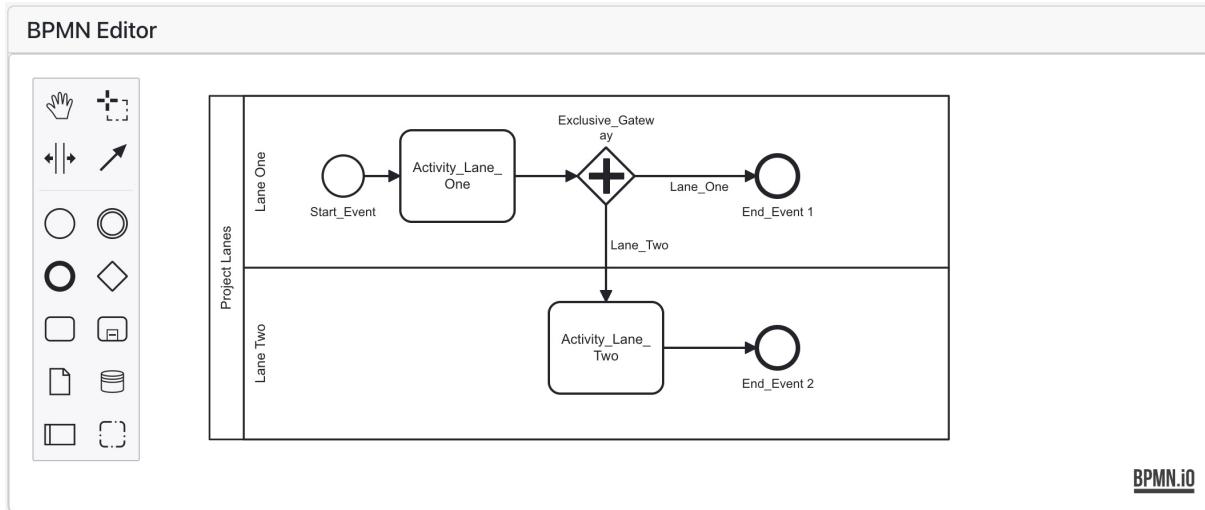
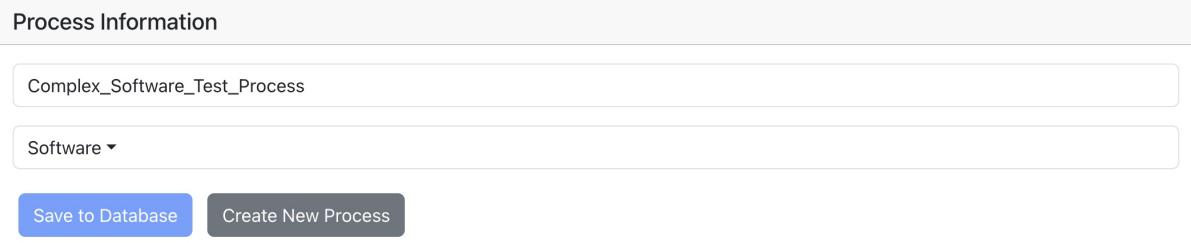


Abbildung 4.23: BPMN-Editor mit geladenem Prozess

Unterhalb des Editors befinden sich Eingabefelder zur Bearbeitung des Prozessnamens sowie zur Zuweisung zu einem Projekt. Die Schaltfläche „Save to Database“ ruft die Methode `handleSaveToDatabase()` auf. Diese implementiert eine rollenbasierte Verzweigung: Für Nutzer mit der Rolle „Manager“ erfolgt eine unmittelbare Persistierung des Prozesses in der Datenbank (siehe Abbildung 4.24). Nutzer mit der Rolle „Employee“ initiieren hingegen einen Genehmigungsworkflow, indem ein entsprechender Antrag erzeugt wird (vgl. Abbildung A11).



This screenshot shows the "Process Information" window. It contains a text input field with the value "Complex\_Software\_Test\_Process" and a dropdown menu set to "Software". At the bottom are two buttons: a blue "Save to Database" button and a dark grey "Create New Process" button.

Abbildung 4.24: Das Prozess-Informationsfenster eines Managers

Die Funktion `handleCreateNewProcess()` ermöglicht die Erstellung eines neuen Prozesses, indem ein leeres Diagramm initialisiert und sämtliche Zustände der Anwendung zurückgesetzt werden.

Ergänzend befindet sich unterhalb der Prozessinformationen ein weiteres Eingabefeld zur Zuweisung zusätzlicher Attribute für jedes BPMN-Element: Rolle und Beschreibung.

Diese benutzerspezifischen Erweiterungen wurden, wie bereits zuvor beschrieben, in die Modellstruktur integriert und dienen der Vorbereitung auf eine rollenbasierte Ausführung der Prozesse. Bereits während der Modellierung muss für jedes Element eine Berechtigung in Form einer zugewiesenen Rolle hinterlegt werden. Dies stellt sicher, dass in der späteren Prozessausführung eine differenzierte und autorisierte Bearbeitung erfolgen kann. Die konkrete funktionale Bedeutung dieser Attribute wird im Kapitel „Prozesse ausführen“ detaillierter erläutert. Die zugehörige Benutzeroberfläche ist in Abbildung 4.25 dargestellt.

Assign Role and Description	
Manager ▾	
This is the Start Event of the complex Software Test Process	
<b>Selected Element:</b> Start_Event	

Abbildung 4.25: Das Fenster für die Zuweisung von Role und Beschreibung eines Elements

Dem Benutzer wird dabei stets das aktuell ausgewählte BPMN-Element angezeigt. Sofern dem Element bereits ein Name zugewiesen wurde, wird dieser zur Identifikation verwendet; andernfalls erfolgt die Anzeige über die interne ID des Elements. Über ein Dropdown-Menü kann dem Element eine spezifische Rolle zugewiesen sowie eine erläuternde Beschreibung hinterlegt werden.

Zur Erkennung von Änderungen am Modell wird ein Command-Stack-Listener im Hintergrund ausgeführt. Dieser überprüft kontinuierlich, ob das aktuelle Modell von der ursprünglich geladenen Version abweicht. Nur bei tatsächlichen Änderungen werden die Speicherfunktionen (z. B. „Save to Database“ oder „Request Save“) aktiviert (vgl. Abbildung A12).

Wird die Schaltfläche „Save to Database“ oder „Request Save“ betätigt, erfolgt zunächst eine Validierung der Benutzereingaben. Dabei wird überprüft, ob alle für die Persistierung notwendigen Informationen vollständig vorliegen. Zu den verpflichtend anzugebenden Daten zählen:

- ein gültiger Prozessname,
- die Zuweisung des Prozesses zu einem Projekt,
- die Angabe einer Rolle für jedes im Modell enthaltene BPMN-Element,
- sowie eine erläuternde Beschreibung für jedes dieser Elemente.

Sollten eine oder mehrere dieser Angaben fehlen, wird der Benutzer durch eine gestaffelte Validierungslogik in absteigender Priorität auf die jeweiligen Unvollständigkeiten hingewiesen. Zunächst wird geprüft, ob ein Prozessname vergeben worden ist (vgl. Abbildung A13). Anschließend erfolgt die Überprüfung der Projektzuweisung (vgl. Abbildung A14). Zuletzt wird kontrolliert, ob für alle BPMN-Elemente sowohl eine Rolle als auch eine Beschreibung hinterlegt worden ist.

Im Rahmen dieser Validierung wird dem Benutzer jedes betroffene Element eindeutig angezeigt – bevorzugt mit dem zugewiesenen Namen, sofern vorhanden, andernfalls mit der internen ID. Eine beispielhafte Fehlermeldung, in der sowohl eine Rollenzuweisung als auch eine Beschreibung fehlen, ist in Abbildung 4.26 dargestellt.

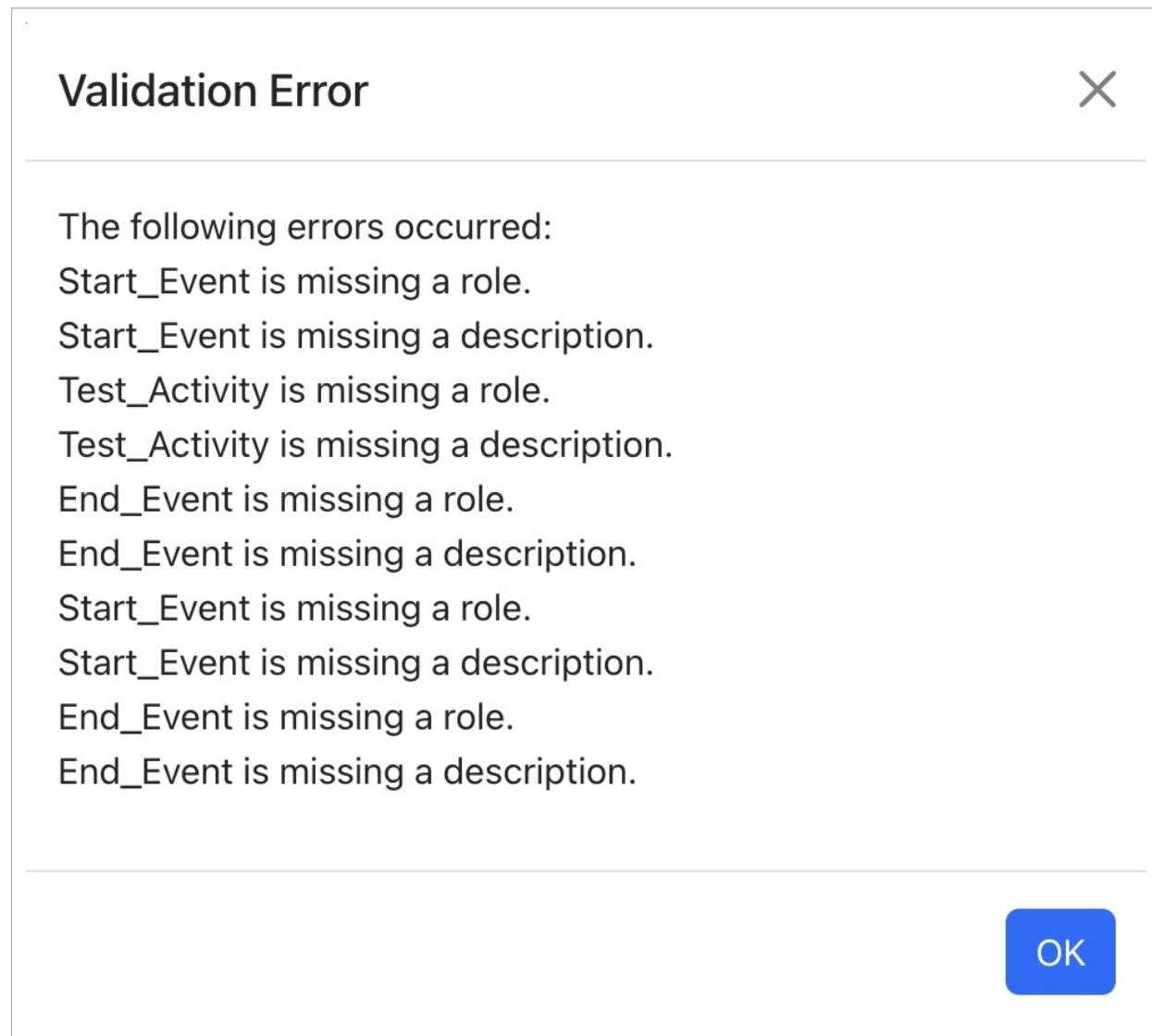


Abbildung 4.26: Die Fehlernachricht bei fehlenden Rollen und Beschreibungen

Durch diese strukturierte Überprüfung wird sichergestellt, dass ausschließlich vollständig definierte Prozesse gespeichert oder zur Freigabe eingereicht werden können. Dies gewährleistet eine konsistente Datenbasis und trägt zu einem fehlerfreien Ablauf in der anschließenden Prozessausführung bei.

## Prozessänderungen verwalten und freigeben

Bei der Bearbeitung eines bereits existierenden Prozesses wird sowohl dem Manager als auch dem Benutzer eine detaillierte Übersicht der vorgenommenen Änderungen in einem separaten Modal angezeigt (vgl. Abbildung 4.27). In dieser Übersicht werden alle Änderungen zusammengefasst und zusätzlich in der zugrunde liegenden BPMN-XML durch Inline-Markierungen hervorgehoben. Dies ermöglicht es, alle geplanten Anpassungen auf einen Blick zu erkennen, bevor sie in der Datenbank gespeichert oder eine Anfrage an den jeweiligen Manager gesendet wird. Diese zusätzliche Validierungsstufe trägt dazu bei, unnötige Fehler frühzeitig zu identifizieren und zu vermeiden, wodurch die Prozessqualität und -genauigkeit nachhaltig verbessert werden.

Preview Changes X

---

- bpmn:startEvent StartEvent\_1: attribute role:role changed: 'Manager' → 'Employee'
- bpmn:startEvent StartEvent\_1: attribute role:description changed: 'This is the Start Event of the complex Software Test Process' → 'This is the Start Event of the complex Software Test Process Changes'

```
<bpmn:flowNodeRef>Activity_1w/251v</bpmn:flowNodeRef>
<bpmn:flowNodeRef>Event_0tn9i45</bpmn:flowNodeRef> </bpmn:lane> </bpmn:laneSet>
<bpmn:sequenceFlow id="Flow_08cus1j" sourceRef="StartEvent_1" targetRef="Activity_194l754" />
<bpmn:sequenceFlow id="Flow_1jq2zx7" role:role="" role:description="" sourceRef="Activity_194l754"
targetRef="Gateway_18ahvh" /> <bpmn:startEvent id="StartEvent_1" role:role="Manager"
role:description="This is the Start Event of the complex Software Test Process" name="Start_Event">
<bpmn:startEvent id="StartEvent_1" role:role="Employee" role:description="This is the Start Event of
the complex Software Test Process Changes" name="Start_Event">
<bpmn:outgoing>Flow_08cus1j</bpmn:outgoing> </bpmn:startEvent> <bpmn:task
id="Activity_194l754" role:role="Employee" role:description="This is a Activity of the complex Software
Test Process" name="Activity_Lane_One"> <bpmn:incoming>Flow_08cus1j</bpmn:incoming>
<bpmn:outgoing>Flow_1jq2zx7</bpmn:outgoing> </bpmn:task> <bpmn:parallelGateway
id="Gateway_18ahvh" role:role="Manager" role:description="This is the exclusive Gateway of the
```

Cancel
Confirm

Abbildung 4.27: Die Übersicht der Änderungen im Modal

Im Rahmen der Änderungen trägt der Manager selbst die Verantwortung, jede Änderung eigenständig zu prüfen und abschließend zu bestätigen, um die Änderungen in die Datenbank zu übernehmen. Der Employee hingegen kann keine direkte Speicherung vornehmen. Deshalb wird eine Benachrichtigung für den Änderungsantrag automatisch an einen zuständigen Manager gesendet.

Employee TestEmployee requested permission to save process "Complex\_Software\_Test\_Process".

1.5.2025, 13:19:17

Project: Software

Abbildung 4.28: Die Benachrichtigung für den Änderungsantrag

Klickt der Manager auf die Benachrichtigung, wird ihm der bearbeitete Prozess samt aller vorgenommenen Änderungen präsentiert. Zur besseren Nachvollziehbarkeit erhält der Manager eine detaillierte Liste der Änderungen. Diese Auflistung erleichtert den Überblick und ermöglicht eine präzise Bewertung der Anpassungen. Zudem hat der Manager die Möglichkeit, die XML-Datei einzublenden, wobei auch hier die Änderungen durch Inline-Markierungen kenntlich gemacht werden.

**Process Information**

Complex_Software_Test_Process
Software ▾
<input style="background-color: #2e7131; color: white; border-radius: 5px; padding: 2px 10px; font-weight: bold; border: none; width: 150px; height: 25px; margin-right: 10px;" type="button" value="Approve Request"/> <input style="background-color: #dc3545; color: white; border-radius: 5px; padding: 2px 10px; font-weight: bold; border: none; width: 150px; height: 25px; margin-right: 10px;" type="button" value="Deny Request"/> <input style="background-color: #6c757d; color: white; border-radius: 5px; padding: 2px 10px; font-weight: bold; border: none; width: 150px; height: 25px; margin-right: 10px;" type="button" value="Create New Process"/> <input style="background-color: #f5f5f5; border-radius: 5px; padding: 2px 10px; font-weight: bold; border: none; width: 150px; height: 25px;" type="button" value="Hide XML"/>

- bpmn:startEvent StartEvent\_1: attribute role:role changed: 'Manager' → 'Employee'
- bpmn:startEvent StartEvent\_1: attribute role:description changed: 'This is the Start Event of the complex Software Test Process' → 'This is the Start Event of the complex Software Test Process Changes'

```

</bpmn:laneSet> <bpmn:sequenceFlow id="Flow_08cus1j" sourceRef="StartEvent_1" targetRef="Activity_194l754" />
<bpmn:sequenceFlow id="Flow_1jq2zx7" role:role="" role:description="" sourceRef="Activity_194l754" targetRef="Gateway_18ahvh" />
<bpmn:startEvent id="StartEvent_1" role:role="Manager" role:description="This is the Start Event of the complex Software Test Process" name="Start_Event"> <bpmn:startEvent id="StartEvent_1" role:role="Employee" role:description="This is the Start Event of the complex Software Test Process Changes" name="Start_Event"> <bpmn:outgoing>Flow_08cus1j</bpmn:outgoing> </bpmn:startEvent>
<bpmn:task id="Activity_194l754" role:role="Employee" role:description="This is a Activity of the complex Software Test Process" name="Activity_Lane_One"> <bpmn:incoming>Flow_08cus1j</bpmn:incoming> <bpmn:outgoing>Flow_1jq2zx7</bpmn:outgoing>
</bpmn:task> <bpmn:parallelGateway id="Gateway_18ahvh" role:role="Manager" role:description="This is the exclusive Gateway of the

```

Abbildung 4.29: Der Änderungsantrag aus Sicht des Managers

Handelt es sich bei der Anfrage nicht um die Änderung eines bestehenden Prozesses, sondern um die Erstellung eines neuen Prozesses, so wird dem zuständigen Manager anstelle einer Änderungsübersicht ein Hinweis angezeigt, der explizit auf die Neuerstellung

hinweist. Dieser Hinweis dient der klaren Einordnung des Vorgangs und wird im Rahmen des Genehmigungsprozesses bereitgestellt.

**Process Information**

NewProcess_Test
Software ▾

Approve Request
Deny Request
Create New Process

- New Process will be created

Abbildung 4.30: Der Änderungsantrag bei Erstellen eines neuen Prozesses

Der Manager hat nun die Option, die Änderungen entweder anzunehmen oder abzulehnen. Wird der Antrag angenommen, werden die Änderungen dauerhaft in der Datenbank gespeichert. Sollte der Manager die Änderungen ablehnen, erfolgt keine Speicherung und der Prozess bleibt unverändert.

Es ist darauf zu achten, dass Änderungen an Prozessen anhand ihrer eindeutigen Identifikationsnummer (ID) vorgenommen werden. Obwohl Namensduplikate technisch ausgeschlossen sind, würde eine Änderung des Prozessnamens in der Praxis dazu führen, dass der bestehende Prozess dupliziert und unter dem neuen Namen gespeichert wird. Um solche unerwünschten Duplizierungen zu vermeiden und eine konsistente sowie wartbare Systemstruktur sicherzustellen, ist es aus Gründen der Best Practice empfehlenswert, Prozessänderungen ausschließlich über die zugehörige Prozess-ID durchzuführen.

Unabhängig von der Entscheidung des Managers erhält der Employee, der die Anfrage gestellt hat, eine Benachrichtigung, die ihn über den aktuellen Status informiert. Diese Rückmeldung enthält die Information, ob die Änderungen angenommen (vgl. Abbildung A15) oder abgelehnt (vgl. Abbildung 4.31) wurden. Dieser Schritt gewährleistet eine transparente Kommunikation und stellt sicher, dass alle Beteiligten stets auf dem aktuellen Stand sind.

Manager TestManager has denied your request to save process  
 "Complex\_Software\_Test\_Process".

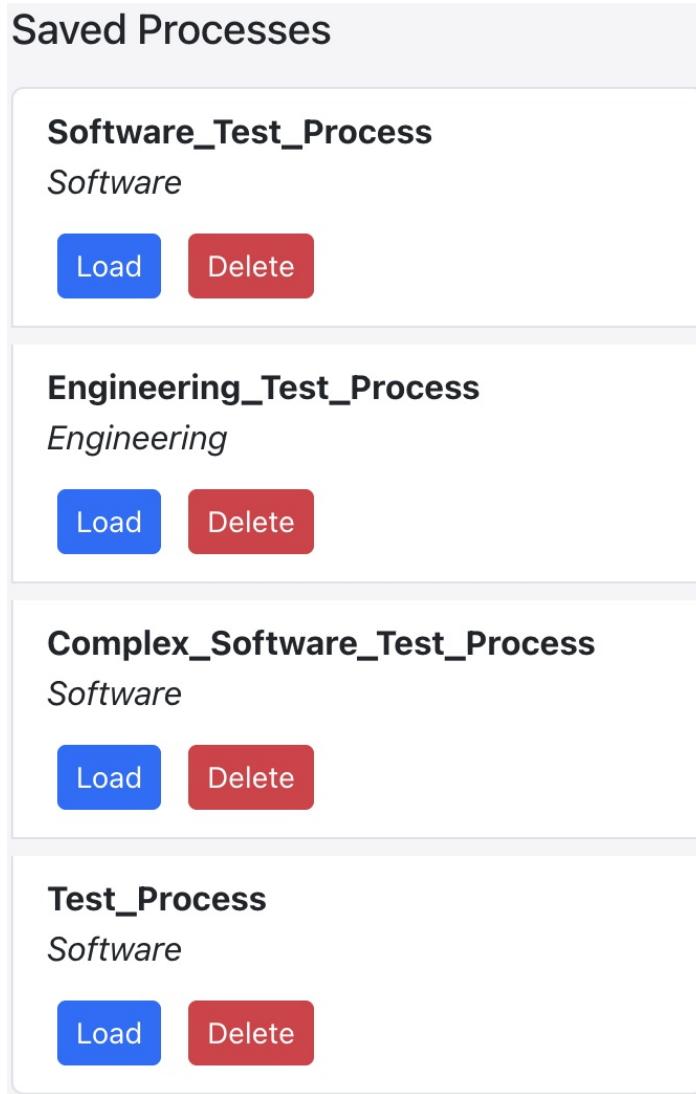
1.5.2025, 13:21:06
Project: Software
Delete

Abbildung 4.31: Die negative Rückmeldung des Änderungsantrags

## Löschen von Prozessen

Ein wesentlicher Bestandteil der Prozessverwaltung ist die Möglichkeit, Prozesse dauerhaft aus dem System zu entfernen. Die Funktion zum Löschen von Prozessen ist dabei rollenbasiert umgesetzt und unterscheidet sich hinsichtlich ihrer Verfügbarkeit und Verantwortlichkeiten zwischen Managern und Employees. Diese Differenzierung trägt zur Sicherstellung einer kontrollierten und nachvollziehbaren Löschpraxis bei.

Nutzer mit der Rolle Manager verfügen in der Prozessliste neben dem „Load“-Button über einen „Delete“-Button, mit dem ein Prozess direkt gelöscht werden kann. Eine exemplarische Darstellung der Prozessliste aus Sicht des Managers zeigt Abbildung 4.32.



**Saved Processes**

- Software\_Test\_Process**  
*Software*  
**Load**   **Delete**
- Engineering\_Test\_Process**  
*Engineering*  
**Load**   **Delete**
- Complex\_Software\_Test\_Process**  
*Software*  
**Load**   **Delete**
- Test\_Process**  
*Software*  
**Load**   **Delete**

Abbildung 4.32: Die Prozess-Liste aus Sicht eines Managers

Wird der „Delete“-Button betätigt, öffnet sich ein separates Bestätigungsfenster, in dem die Löschaktion nochmals explizit bestätigt werden muss (vgl. Abbildung 4.33). Dieses zusätzliche Dialogfeld dient der Vermeidung unbeabsichtigter Löschvorgänge. Die Verantwortung für die Durchführung und etwaige Konsequenzen des Löschvorgangs liegt dennoch ausschließlich beim Manager.

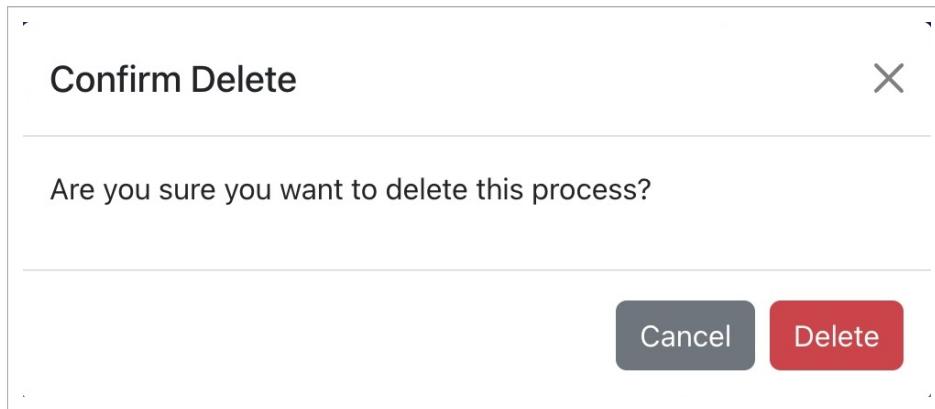


Abbildung 4.33: Die Bestätigungsfrage vor dem Löschen eines Prozesses

Im Gegensatz dazu besitzen Nutzer mit der Rolle Employee keine Berechtigung, Prozesse eigenständig zu löschen. Diese Einschränkung dient der Wahrung der Datenintegrität und stellt sicher, dass nur autorisierte Personen mit entsprechender Übersicht über die organisatorischen Abläufe über Löschvorgänge entscheiden können. Statt einer direkten Löschoption steht dem Employee in der Prozessliste ein Button mit der Bezeichnung „Request Deletion“ zur Verfügung. Diese Funktion ermöglicht es, einen Löschwunsch formal an den zuständigen Manager zu übermitteln. Die Darstellung der Prozess-Liste eines Employees ist in Abbildung 4.34 zu sehen.

## Saved Processes

**Software\_Test\_Process**  
*Software*

Load Request Deletion

**Engineering\_Test\_Process**  
*Engineering*

Load Request Deletion

**Complex\_Software\_Test\_Process**  
*Software*

Load Request Deletion

**Test\_Process**  
*Software*

Load Request Deletion

Abbildung 4.34: Die Prozess-Liste aus Sicht eines Employee

Durch Auslösen dieser Option wird eine Löschanfrage als Benachrichtigung an den zuständigen Manager gesendet.

Employee TestEmployee requested deletion of process "DeleteProcess\_Test".

2.5.2025, 13:37:57

Project: Software

Abbildung 4.35: Die Anfrage für das Löschen des Prozesses

Beim Öffnen dieser Benachrichtigung durch den Manager erscheint ein Modal, das zwei Handlungsoptionen bietet: Die Anfrage kann entweder angenommen oder abgelehnt werden. Diese Entscheidung obliegt allein dem Manager und sollte idealerweise auf einer inhaltlichen Prüfung des Prozesses basieren.

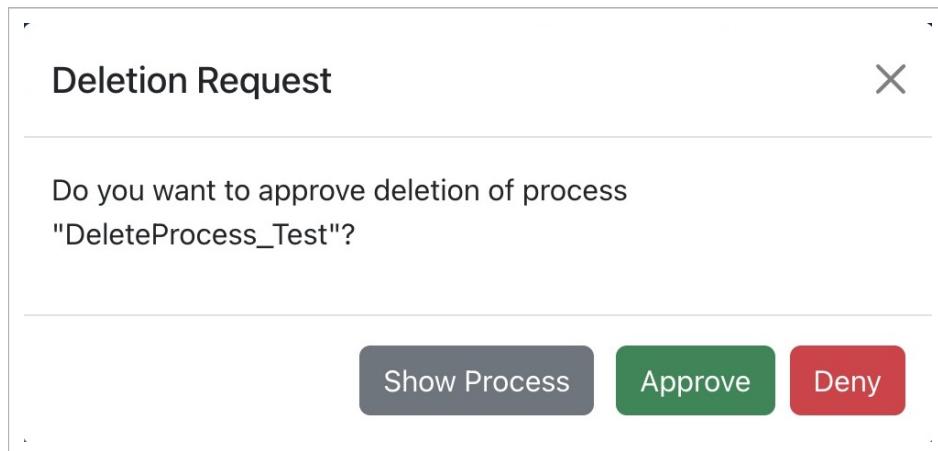


Abbildung 4.36: Das Modal-Fenster für die Anfrage zum Löschen

Zur Unterstützung dieser Entscheidung steht dem Manager zusätzlich der Button „Show Process“ zur Verfügung. Durch Betätigung dieser Schaltfläche wird eine Detailansicht des betreffenden Prozesses angezeigt. Diese Ansicht ermöglicht eine gezielte Überprüfung der gespeicherten Informationen und erlaubt dem Manager, den Prozess vor einer Entscheidung umfassend zu evaluieren. Die Darstellung in diesem Fenster ähnelt derjenigen, die bei der Bearbeitung und Speicherung von Prozessen verwendet wird, wodurch eine konsistente Benutzererfahrung gewährleistet ist.

**Process Information**

DeleteProcess\_Test

Software ▾

**Approve Deletion** **Deny Deletion** **Create New Process** **Show XML**

No changes detected.

Abbildung 4.37: Das Prozess-Informations Feld für die Anfrage zum Löschen

Entscheidet sich der Manager für die Annahme der Anfrage, wird der betreffende Prozess aus dem System entfernt. Anschließend wird die Prozessliste automatisch aktualisiert, sodass alle Benutzer sofort die aktualisierte Datenlage einsehen können. Dieser Schritt

stellt sicher, dass keine veralteten Informationen mehr angezeigt werden und reduziert vermeidbare Fehlerquellen in der weiteren Bearbeitung.

Nach Bearbeitung der Löschanfrage erhält der Employee, der den Antrag gestellt hat, eine Rückmeldung über den Status seiner Anfrage. Diese Rückmeldung kann entweder positiv (Anfrage wurde angenommen, vgl. Abbildung 4.38) oder negativ (Anfrage wurde abgelehnt, vgl. Abbildung A16) ausfallen. Auf diese Weise wird eine transparente Kommunikation zwischen den Rollen sichergestellt, und es besteht jederzeit Klarheit über den Stand der beantragten Änderungen.

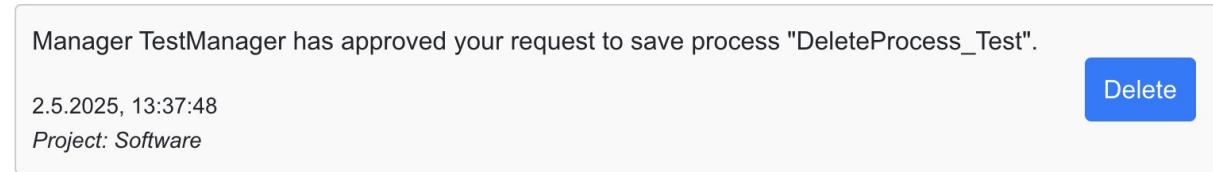


Abbildung 4.38: Die positive Rückmeldung nach Anfrage zum Löschen

## Fazit

Die Komponente **ManageProcess** stellt eine effiziente und benutzerfreundliche Lösung zur Verwaltung von BPMN-Prozessen dar. Durch die Verwendung der **bpmn-js**-Bibliothek wird eine solide technologische Grundlage geschaffen. Die Integration der Bibliothek ermöglicht eine skalierbare und erweiterbare Architektur, die den Anforderungen von Benutzern und Entwicklern gleichermaßen gerecht wird. Die Implementierung von benutzerspezifischen Erweiterungen bietet zusätzliche Flexibilität und unterstützt die langfristige Anpassbarkeit der Lösung.

Die Benutzeroberfläche und die Validierungslogik tragen zur Sicherstellung einer fehlerfreien Prozessmodellierung bei. Die Möglichkeit, Änderungen zu validieren und in einem klar strukturierten Genehmigungsworkflow freizugeben, stellt sicher, dass nur vollständig definierte und überprüfte Prozesse in die Datenbank übernommen werden. Darüber hinaus können bestehende Prozesse gezielt gelöscht werden. Dies ermöglicht es autorisierten Benutzern, veraltete oder fehlerhafte Prozessmodelle aus dem System zu entfernen. Die Löschfunktion ist in die Benutzeroberfläche integriert und folgt den gleichen Berechtigungsmechanismen wie andere Bearbeitungsfunktionen, um eine konsistente Zugriffskontrolle zu gewährleisten. Dieser Workflow fördert die Qualität und Genauigkeit der modellierten Prozesse und ermöglicht eine effiziente Zusammenarbeit zwischen den verschiedenen Benutzerrollen.

Insgesamt zeigt sich, dass die **ManageProcess**-Komponente eine robuste, skalierbare und benutzerfreundliche Lösung für die Verwaltung von BPMN-Prozessen bietet. Sie stellt sicher, dass die Modellierung, Bearbeitung und Freigabe von Prozessen sowohl technisch als auch organisatorisch gut unterstützt wird, und ermöglicht eine zukunftssichere Erweiterbarkeit zur Integration neuer Anforderungen.

#### 4.5.4 Prozesse ausführen

Die Komponente **ExecuteProcess** ist für die Ausführung zuvor modellierter BPMN-Prozesse zuständig. Ziel dieser Komponente ist es, Benutzern eine intuitive und strukturierte Ausführung der Prozesse zu ermöglichen.

Im linken Bereich der Benutzeroberfläche befindet sich eine Liste aller verfügbaren Prozesse, welche zuvor über die **ManageProcess**-Komponente erstellt worden sind (vgl. Unterabschnitt 4.5.3). Zusätzlich werden darunter die derzeit aktiven Instanzen dieser Prozesse angezeigt. Im rechten Bereich wird der Benutzer aufgefordert, einen Prozess oder eine Instanz auszuwählen, um die Ausführung zu starten.

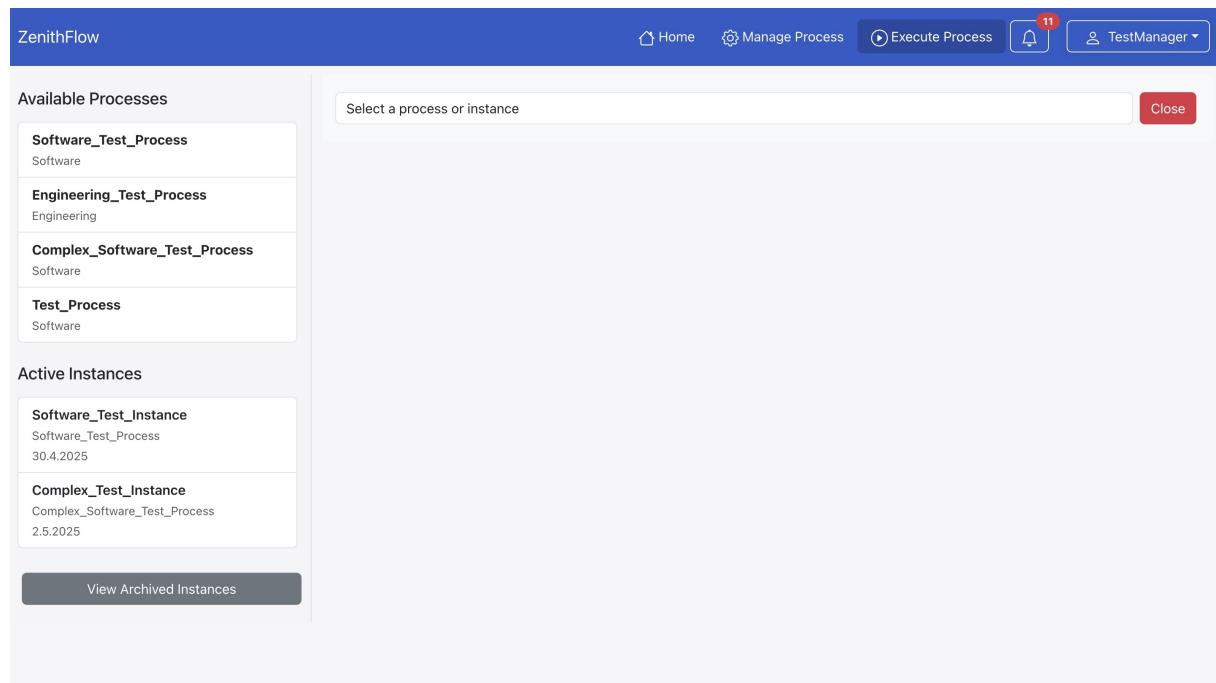


Abbildung 4.39: Die vollständige Execute Process-Seite

Wählt der Benutzer einen Prozess aus der Liste aus, wird das zugehörige Modell in einem schreibgeschützten BPMN-Viewer angezeigt. Im Gegensatz zur Komponente **ManageProcess** handelt es sich hierbei nicht um einen Editor, sodass keine Änderungen am Prozessmodell vorgenommen werden können. Dies stellt sicher, dass die Konsistenz des zugrundeliegenden Modells erhalten bleibt. Der Viewer dient ausschließlich der Visualisierung und ermöglicht dem Benutzer eine Vorschau des Prozesses vor dessen Ausführung.

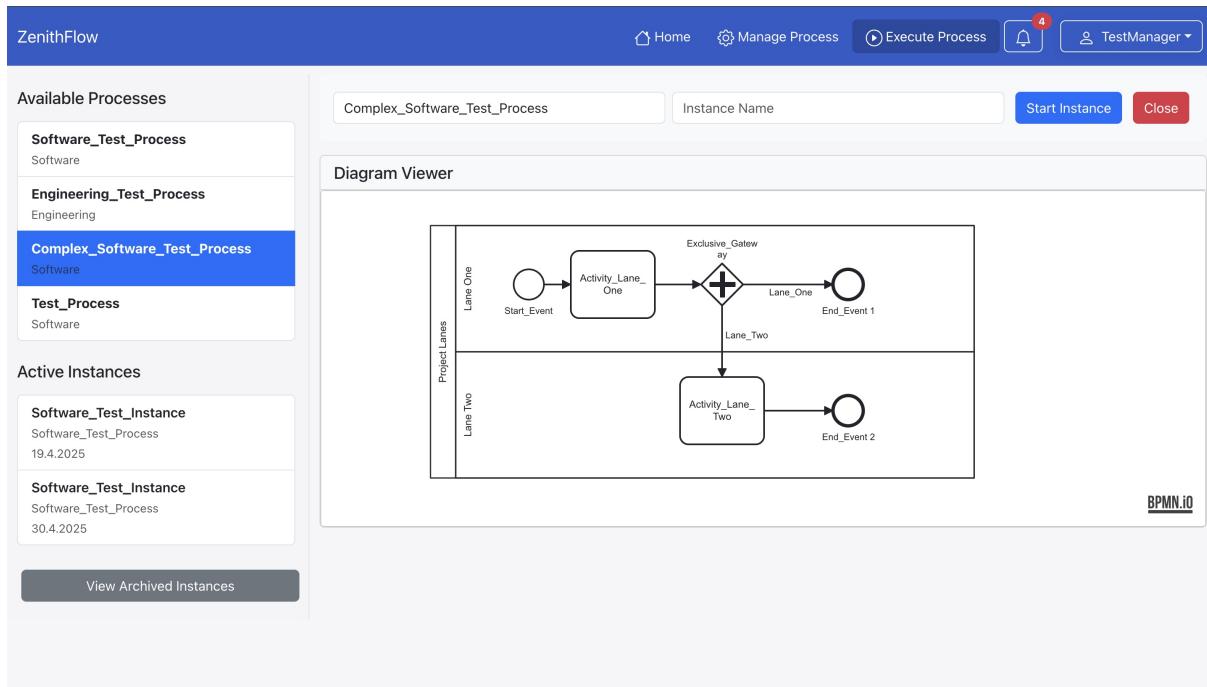
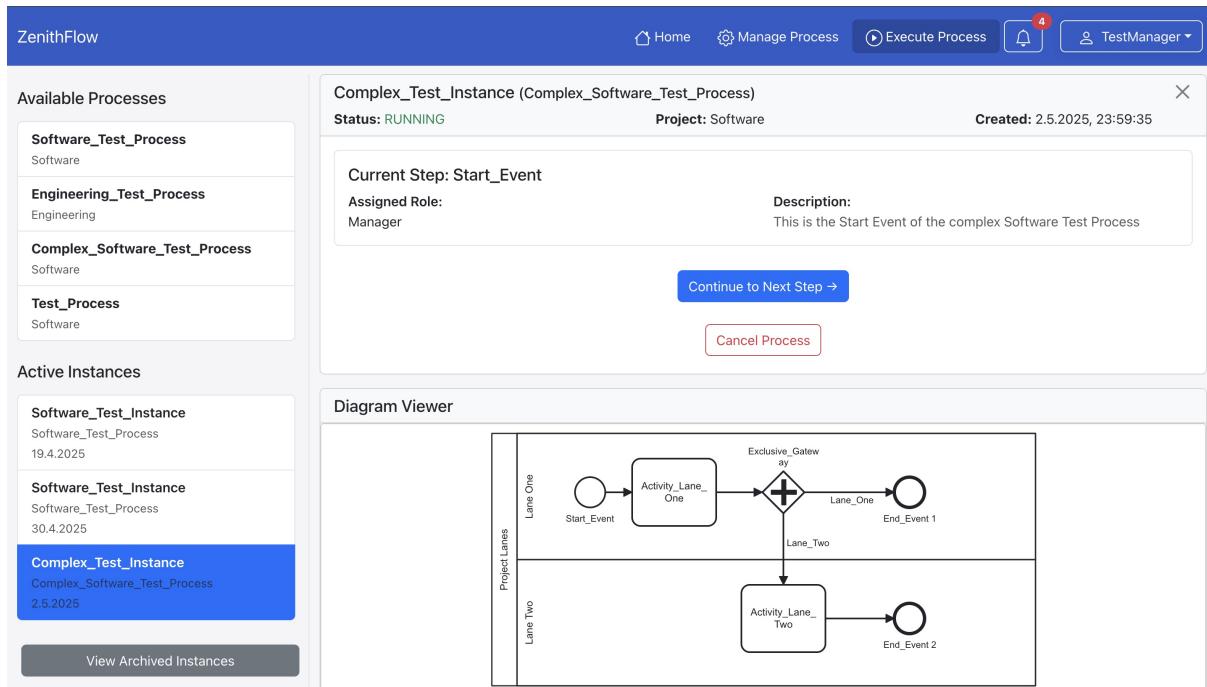


Abbildung 4.40: Die Execute Process-Seite mit angeklickten Prozess

Nach Auswahl eines Prozesses erhält der Benutzer die Möglichkeit, eine aktive Instanz dieses Prozesses zu starten. Hierfür ist zunächst die Eingabe eines eindeutigen Instanznamens erforderlich. Erst nach Angabe dieses Namens und dem anschließenden Betätigen der Schaltfläche Start Instance wird die neue Instanz erzeugt, in die Liste der aktiven Instanzen aufgenommen und am linken Rand der Benutzeroberfläche angezeigt.

Unter einer Instanz wird eine konkrete Ausführung des jeweiligen Prozesses verstanden. Jede erneute Ausführung führt zur Erzeugung einer separaten Instanz, die unabhängig gespeichert und verwaltet wird. Auf diese Weise lassen sich sowohl mehrere Ausführungen desselben Prozesses als auch unterschiedliche Prozesse parallel initiieren und steuern. Dadurch können beispielsweise verschiedene Teams denselben Prozess unabhängig voneinander ausführen oder ein einzelnes Team mehrere voneinander getrennte Prozessinstanzen verwalten. Die Möglichkeit zur parallelen Instanziierung stellt einen wesentlichen Aspekt für die effiziente und flexible Nutzung von Geschäftsprozessen dar.

Wird eine Instanz aus der Liste aktiver Instanzen ausgewählt, wird sie für die Ausführung vorbereitet. Der Benutzer kann die Instanz nutzen, um den zugehörigen Prozess schrittweise zu durchlaufen. Die Darstellung erfolgt im bereits bekannten *BPMN-Viewer*, der zuvor zur Prozessvorschau verwendet wurde. Ergänzt wird dieser nun durch ein zusätzliches Navigationsfenster auf der rechten Seite, das relevante Informationen zur ausgewählten Instanz bereitstellt.



The screenshot shows the ZenithFlow interface for executing processes. On the left, there's a sidebar with 'Available Processes' and 'Active Instances'. In the main area, a specific instance is selected: 'Complex\_Test\_Instance (Complex\_Software\_Test\_Process)' with status 'RUNNING', project 'Software', and creation date '2.5.2025, 23:59:35'. Below this, the 'Current Step: Start\_Event' is shown with assigned role 'Manager' and a description: 'This is the Start Event of the complex Software Test Process'. There are buttons for 'Continue to Next Step' and 'Cancel Process'. To the right, a 'Diagram Viewer' displays an BPMN diagram for the process. The diagram features two parallel lanes: 'Lane One' and 'Lane Two'. Lane One starts with a 'Start\_Event' leading to 'Activity\_Lane\_One', which then leads to an 'Exclusive\_Gateway' (indicated by a diamond with a plus sign). From the gateway, two paths emerge: one to 'Activity\_Lane\_Two' in Lane Two, and another to 'End\_Event\_1'. Lane Two starts with 'Activity\_Lane\_Two' leading to 'End\_Event\_2'. The diagram is labeled 'Project Lanes'.

Abbildung 4.41: Die Execute Process-Seite mit angeklickter Instanz

In der Kopfzeile dieses Bereichs befinden sich mehrere zentrale Attribute, die zur eindeutigen Identifikation der Instanz sowie zur Übersicht über deren Eigenschaften dienen:

- **Instanzname:** Bezeichnung der Instanz zur leichteren Identifikation.
- **Prozessname:** Gibt an, zu welchem Prozess die Instanz gehört bzw. welcher Prozess aktuell ausgeführt wird.
- **Aktueller Status:** Zeigt den Status der Instanz an. Mögliche Zustände sind *Running*, *Finished* oder *Canceled*. Diese Statusinformationen sind essenziell für die spätere Nachvollziehbarkeit, insbesondere zur Unterscheidung zwischen erfolgreich abgeschlossenen und abgebrochenen Prozessen.
- **Zugewiesenes Projekt:** Dient der Filterung und Übersichtlichkeit, indem nur relevante Instanzen innerhalb eines bestimmten Projekts angezeigt werden.
- **Erstellungsdatum:** Unterstützt die eindeutige Identifikation der Instanz und trägt zur besseren zeitlichen Einordnung sowie Nachverfolgbarkeit bei.

Während einige dieser Attribute primär der Übersicht und Benutzerorientierung dienen, besitzen andere auch funktionale Relevanz. So erlaubt beispielsweise das Attribut *Projekt* die projektbasierte Filterung von Prozessen und Instanzen. Der *Status* ist insbesondere für die spätere Archivierung und Prozessanalyse von Bedeutung. Das *Erstellungsdatum* ist vor

allem dann wichtig, wenn mehrere Instanzen denselben Namen tragen, was in bestimmten Anwendungsfällen ausdrücklich gewünscht ist.

Ein anschauliches Beispiel stammt aus dem Produktionsumfeld: Zwei Fahrzeuge desselben Modells sollen gleichzeitig durch denselben Fertigungsprozess geführt werden. Trotz identischer Prozessabläufe kann es zu zeitlichen Verschiebungen kommen. Eine identische Benennung der Instanzen ist in diesem Fall sinnvoll, da der Prozessinhalt gleich bleibt. Die eindeutige Unterscheidung erfolgt dann über das Erstellungsdatum, welches zusätzlich eine zeitliche Orientierung ermöglicht. Alternativ wäre auch eine Identifikation über eine interne Instanz-ID möglich. Das Erstellungsdatum bietet jedoch eine benutzerfreundlichere und zugleich funktional hilfreichere Darstellung für diesen Anwendungsfall.

Unterhalb der Kopfzeile befindet sich ein weiterer Informationsbereich, der Details zum aktuell aktiven Bearbeitungsschritt enthält. Folgende Attribute werden hier angezeigt:

- **Schrittnname:** Bezeichnung des Prozessschritts zur klaren Identifikation.
- **Zugewiesene Rolle:** Wesentlich für das Berechtigungsmanagement innerhalb des Prozesses.
- **Beschreibung:** Erläuterung des Bearbeitungsschritts. Diese sollte so formuliert sein, dass auch fachfremde Benutzer die Aufgabe nachvollziehen und korrekt ausführen können.

Besondere Bedeutung kommt der *Rolle* zu, da sie die Grundlage für das rollenbasierte Ausführungsmanagement bildet. Bestimmte Bearbeitungsschritte sind ausschließlich Benutzern mit einer entsprechend höheren Berechtigung, beispielsweise der *Manager*-Rolle, vorbehalten. Die Zuweisung von Rollen zu den jeweiligen Schritten erfolgt bereits im Rahmen der Prozessverwaltung (vgl. Unterabschnitt 4.5.3).

Während der Ausführung wird bei jedem Schritt überprüft, ob die Rolle des aktuellen Benutzers mit der für den jeweiligen Schritt definierten Rolle übereinstimmt. Ist dies der Fall, kann der Benutzer nach erfolgreicher Bearbeitung zum nächsten Schritt übergehen. Andernfalls wird der Zugriff verweigert. In diesem Fall muss eine Anfrage zur Freigabe an eine berechtigte Person – in der Regel einen Manager – gestellt werden. Diese Funktionalität ist in Abbildung 4.42 dargestellt.

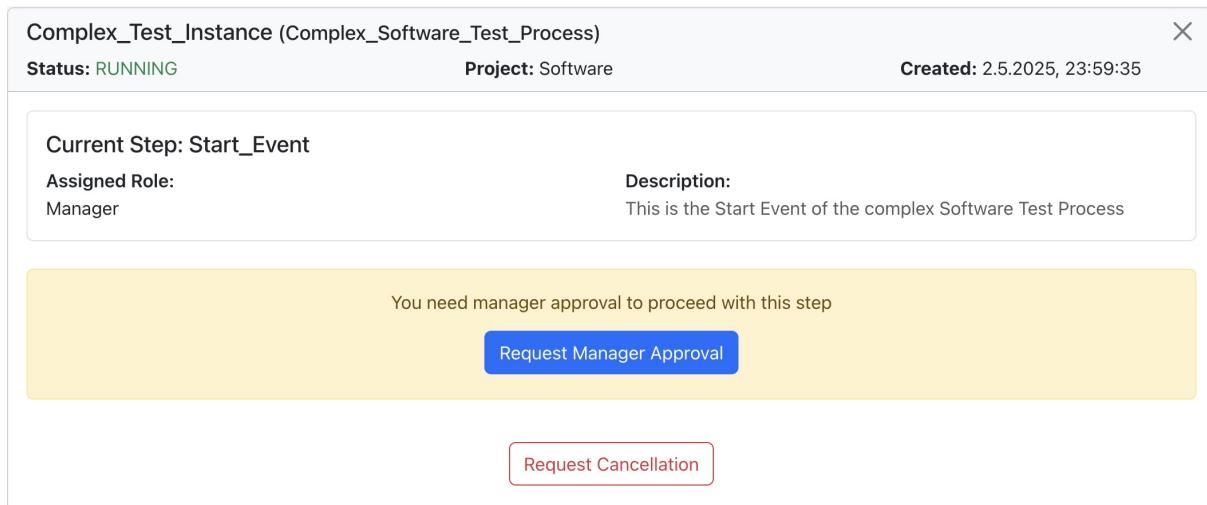


Abbildung 4.42: Die Ansicht eines Employee für eine Genehmigungsanfrage

Dadurch wird gewährleistet, dass die Bearbeitungsschritte des Prozesses während der Ausführung regelmäßig überprüft werden, um eine hohe Qualität sicherzustellen. Der Manager wird benachrichtigt und ist dazu angehalten, den aktuellen Bearbeitungsschritt sowie idealerweise die vorausgegangenen Schritte zu prüfen und anschließend zu genehmigen. Die entsprechende Benachrichtigung ist in Abbildung 4.43 dargestellt.

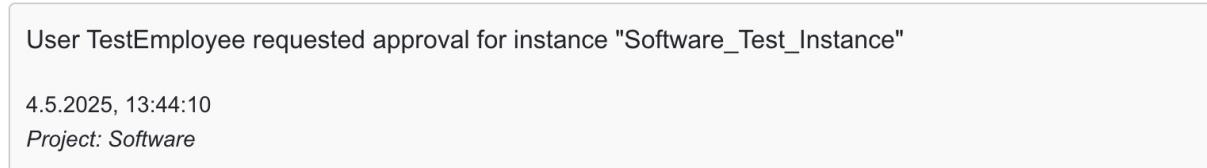


Abbildung 4.43: Die Benachrichtigung für die Genehmigungsanfrage

Erachtet der Manager die Ausführung der bisherigen Bearbeitungsschritte als angemessen und zufriedenstellend, kann er den nächsten Prozessschritt freigeben. In diesem Fall wird die Instanz zur weiteren Bearbeitung an den ursprünglichen Benutzer zurückgegeben, der den Prozess an der freigegebenen Stelle fortsetzen kann. Erkennt der Manager hingegen Mängel oder fehlerhafte Ausführungen, hat er die Möglichkeit, die Freigabe zu verweigern. Dadurch erhält der Benutzer eine Rückmeldung, dass der Bearbeitungsschritt nicht den geforderten Qualitätsstandards entspricht und eine Überarbeitung erforderlich ist.

Die Verantwortung für die inhaltliche Prüfung liegt in diesem Fall beim Manager, der die Freigabeentscheidung eigenständig trifft. Der Benutzer selbst kann damit die Verantwortung für die Bewertung des Bearbeitungsschritts an den Manager übergeben. Dies ermöglicht eine klare Trennung zwischen Durchführung und Freigabe einzelner Prozessschritte, was insbesondere in sicherheits- oder qualitätskritischen Anwendungen von Bedeutung ist.

Wie die Benutzeroberfläche für den Manager zur Entscheidung über die Freigabe aussieht, ist in Abbildung 4.44 dargestellt. Anstelle des üblichen Schalters „Continue to next Step“ stehen dem Manager dort zwei Optionen zur Verfügung: „Accept Request & Continue to next Step“ oder „Deny Request“. Dadurch wird dem Manager eine gezielte Entscheidungsmöglichkeit über den weiteren Verlauf der Instanz geboten.

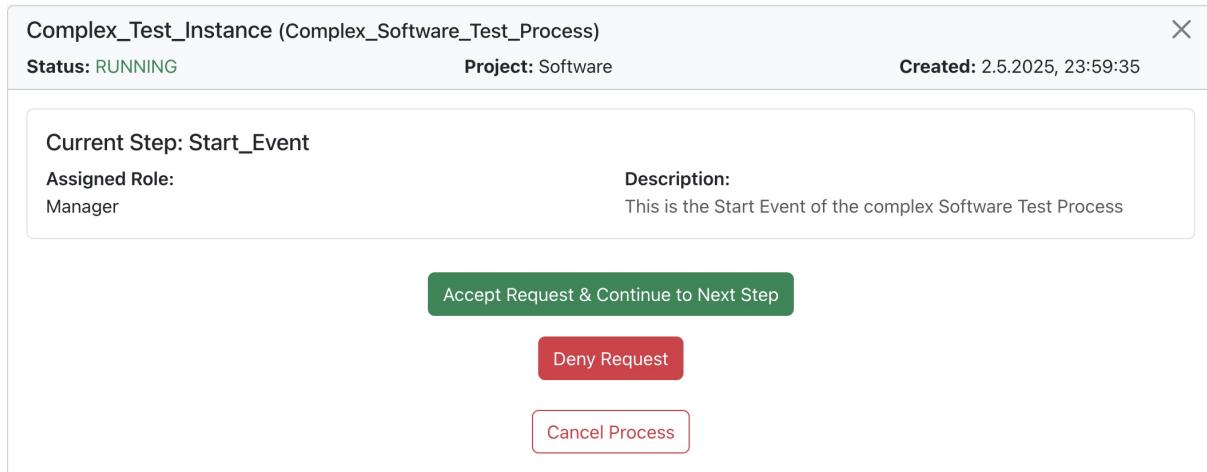


Abbildung 4.44: Die Ansicht der Genehmigungsentscheidung

Unabhängig vom Ergebnis der Freigabeentscheidung durch den Manager erhält der Benutzer im Anschluss eine entsprechende Rückmeldung. Diese Benachrichtigung kann im Fall einer erfolgreichen Freigabe positiv ausfallen, wie in Abbildung 4.45 dargestellt. Wird die Freigabe hingegen verweigert, erhält der Benutzer eine ablehnende Benachrichtigung, wie in Abbildung gezeigt.

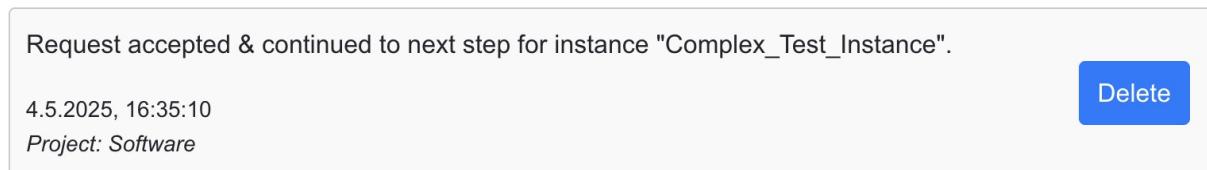


Abbildung 4.45: Die Rückmeldung auf die Genehmigungsanfrage

Der Benutzer kann auf die Benachrichtigung klicken und wird zur betreffenden Instanz weitergeleitet, dort kann er entweder direkt weiter arbeiten oder sich nochmal einen Überblick über den Prozess verschaffen und die damit möglicherweise mangelhafte Umsetzung der Bearbeitungsschritte.

Diese Art der Sicherstellung durch den Manager ist auch sehr gut geeignet, um zu entscheiden und sicherstellen, welchen Weg man nach einem Gateway im BPMN-Prozess gehen möchte. Dafür gibt es die Möglichkeit, wenn man eine Instanz ausgewählt hat,

ein beliebiges Element anzuklicken und alle notwendigen Informationen des gewählten Element zu sehen. Dafür erscheint ein weiteres Fenster zwischen dem Navigationsfenster und dem BPMN-Viewer. Dieses enthält alle Informationen über das gewählte Element, genau wie im Navigationsfenster auch. Vorteil ist hier nur es ist nicht das nächste, sondern man kann sich so über jedes Element im Prozess einen Überblick verschaffen, bevor man beispielsweise bei einem Gateway eine Entscheidung treffen muss.

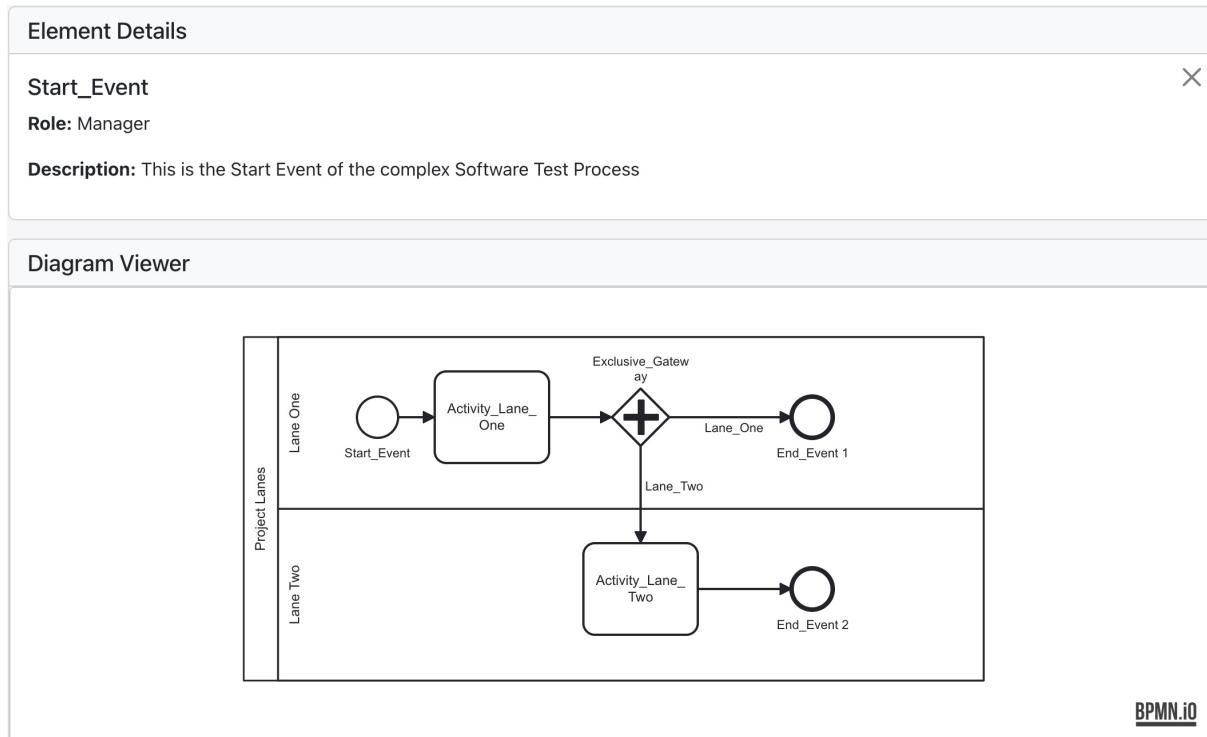


Abbildung 4.46: Das Element Informations-Feld

Unter der Liste der aktuellen verfügbaren Prozesse und den aktiven Instanzen ist noch ein Knopf "View Archived Instances" (vgl. Abbildung 43 unten links). Dieser bringt den Benutzer auf eine neue Seite, auf der alle archivierten Instanzen gespeichert und angezeigt werden. So kann jeder Benutzer vergangene Prozesse nachvollziehen und potentiell Schlüsse für die aktuelle Instanz ziehen oder auch im Falle eines Fehlers die Fehlerquelle besser identifizieren. Die vollständige Seite der archivierten Instanzen ist in Abbildung 45 zu sehen.

Um einen Überblick über alle möglichen Instanzen behalten zu können, ist in der Kopfzeile der Liste eine Filterfunktion integriert. So kann der Benutzer die vielen Instanzen filtern und so schneller die gewünschte Instanz finden oder anhand der Filter-Parameter identifizieren. Dabei stehen folgende Attribute für die Filterung zur Auswahl: - Projekte - Prozesse - Status - Erstellungsdatum - Abschließungsdatum

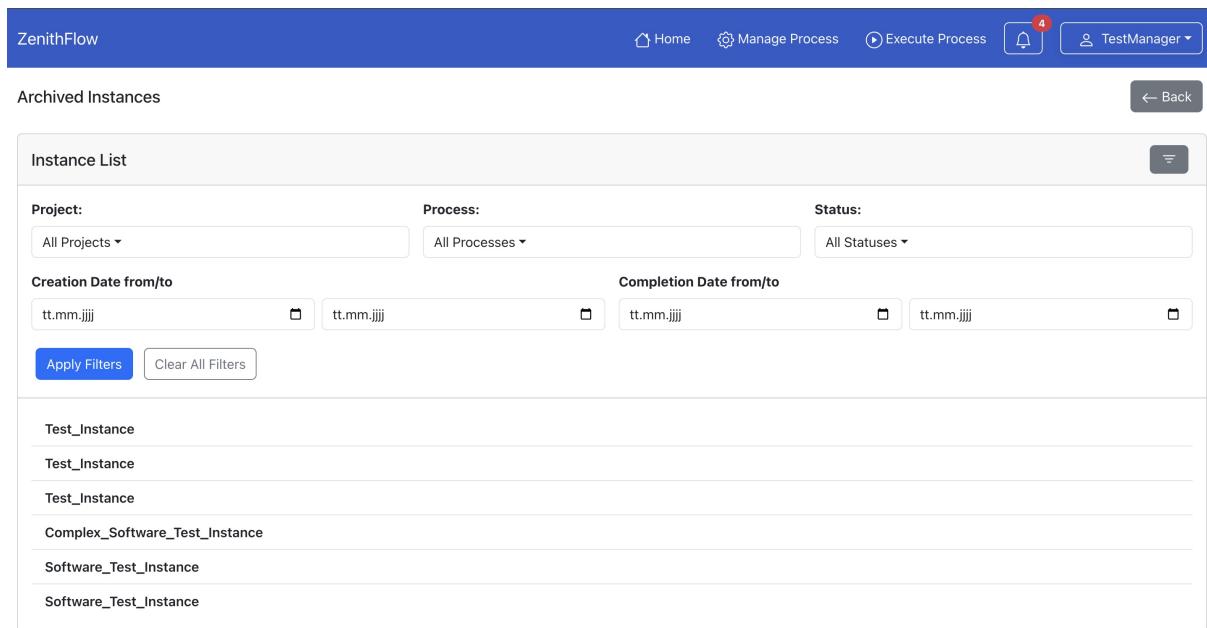


Abbildung 4.47: Die Seite mit allen archivierten Instanzen

Dabei haben alle Attribute eine innovative Darstellung, um den Benutzer die Nutzung so leicht und angenehm wie möglich zu machen. Beispielweise haben Projekte, Prozesse und Status ein Dropdown-Menü mit allen möglichen Eingabemöglichkeiten. Das Datum hingegen lässt sich aus einem sich öffneten Mini-Kalender auswählen (vgl. Abbildung Mini-Kalender).

Hat der Benutzer alle Filter-Attribute gewählt kann er mit dem Knopf Apply Filters seine Auswahl bestätigen und den Filter anwenden. Dabei wird in der Kopfzeile am Symbol für den Filter, die Anzahl aktiver Filter angezeigt, um dem Benutzer die Benutzung zu vereinfachen.

Dabei reduziert sich die Liste immer weiter auf die angegebenen Filter-Attribute bis keine passende Instanz mehr vorhanden ist, dann wird anstatt der Liste ein Hinweis angezeigt, der den Benutzer darauf hinweist, dass keine passende Instanz mit diesen Filter-Attributen gefunden werden konnte.

## Fazit

Die ExecuteProcess-Komponente stellt eine benutzerfreundliche, flexible und technisch robuste Lösung zur Die Komponente ExecuteProcess erfüllt eine zentrale Rolle innerhalb des Process-Managers, da sie die interaktive und nachvollziehbare Ausführung modellierter

Umformulieren und richtig anordnen

### Creation Date from/to

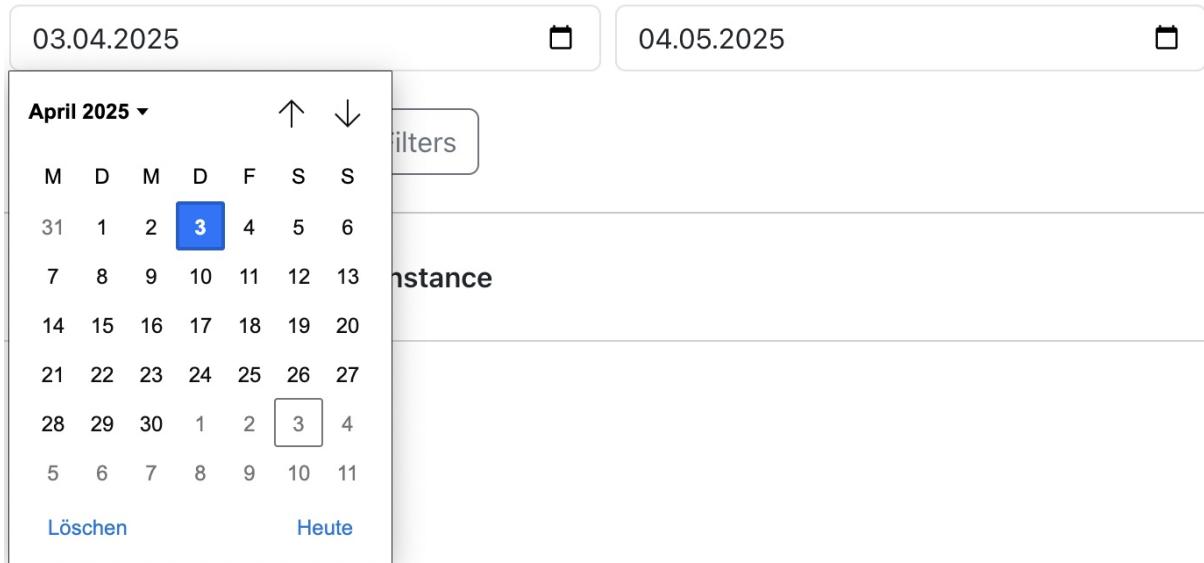


Abbildung 4.48: Der Mini-Kalender zur Datum-Auswahl

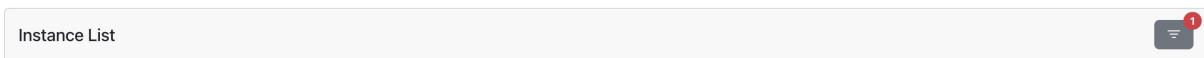


Abbildung 4.49: Hinweis auf aktive Instanzen am Filter-Symbol

BPMN-Prozesse ermöglicht. Dabei ist besonderes Augenmerk auf eine klare Trennung zwischen Modellierung und Ausführung gelegt worden: Die Prozesse werden in einem ausschließlich lesbaren Viewer dargestellt, wodurch eine unbeabsichtigte Veränderung des Modells während der Ausführung ausgeschlossen wird. Dies stellt sicher, dass ausschließlich validierte Prozessmodelle zur Anwendung kommen.

Ein wesentliches Merkmal der Komponente ist das Instanzkonzept. Für jeden Durchlauf eines Prozesses wird eine eigene Prozessinstanz erzeugt und separat verwaltet. Diese Herangehensweise erlaubt die gleichzeitige Ausführung mehrerer Instanzen desselben oder unterschiedlicher Prozesse. Dadurch wird nicht nur die parallele Bearbeitung ermöglicht, sondern auch die Nachvollziehbarkeit einzelner Abläufe erheblich verbessert.

Zusammenfassend kann festgestellt werden, dass die ExecuteProcess-Komponente die Anforderungen an eine benutzerfreundliche, parallele und transparente Prozessausführung erfüllt. Sie bietet eine robuste Grundlage für den praktischen Einsatz in geschäftlichen Anwendungsszenarien und stärkt die Trennung zwischen Design und Laufzeitumgebung – ein zentrales Prinzip moderner Prozessmanagementsysteme.

No archived instances found with current filters.

Abbildung 4.50: Hinweis auf keine passende Instanzen mit den gewählten Filtern

#### 4.5.5 Docker-Bereitstellung der Projekt-Komponenten

Zur Bereitstellung der entwickelten Anwendungskomponenten wird Docker als Containerplattform eingesetzt. Nähere Informationen zu Docker sind im Glossareintrag Docker zu finden. Ziel ist die Schaffung einer konsistenten und reproduzierbaren Ausführungsumgebung für Frontend, Backend und Datenbank, die unabhängig vom Zielsystem zuverlässig betrieben werden kann. Docker ermöglicht die Kapselung von Anwendungen samt aller Abhängigkeiten in isolierten Containern. Dadurch werden potenzielle Kompatibilitätsprobleme reduziert und manuelle Konfigurationsaufwände minimiert. Der Einsatz von Docker erweist sich insbesondere im Kontext von Entwicklung, Test und Deployment als vorteilhaft.

Die eingesetzte Architektur besteht aus drei logisch getrennten Containern (vgl. Abbildung 4.1): einem Container für die MongoDB-Datenbank, einem für das Node.js-Backend sowie einem weiteren für das statisch gebaute Frontend, das über einen Webserver ausgeliefert wird. Die Orchestrierung dieser Komponenten erfolgt mittels Docker Compose, wodurch der gesamte Containerverbund über einen einzigen Befehl gestartet, gestoppt oder aktualisiert werden kann. Dies bringt insbesondere im lokalen Entwicklungskontext sowie bei der Integration in CI/CD-Pipelines erhebliche Vorteile.

Die Datenbank wird auf Basis des offiziellen MongoDB-Images in der Version 4.4 in einem eigenen Container betrieben. Beim Start des Containers wird über ein Initialisierungsskript automatisch ein Benutzer mit dem Benutzernamen admin und dem Passwort admin angelegt. Dieser verfügt über alle notwendigen Rechte, um innerhalb des Process Managers neue Benutzer, Projekte und Entitäten anzulegen. Damit ist die Anwendung unmittelbar nach dem Start einsatzbereit, ohne dass zusätzliche manuelle Konfigurationen erforderlich sind. Diese automatisierte Initialisierung unterstützt insbesondere den Entwicklungsprozess, da beim Stoppen und Neustarten des Containers sämtliche Daten verloren gehen. Die Konfiguration ist bewusst so gewählt, da in der betrachteten Entwicklungsphase keine Persistenz im produktiven Sinne erforderlich ist. Für den späteren produktiven Einsatz ist jedoch eine dauerhafte Speicherung der Daten unabdingbar – dieser Aspekt wird im Ausblick erneut thematisiert.

Das Backend wird direkt aus dem lokalen Quellcodeverzeichnis erstellt und in einem dedizierten Container ausgeführt. Die Verbindung zur Datenbank erfolgt über definierte

Umgebungsvariablen unter Nutzung der zuvor genannten Zugangsdaten. Durch die Abhängigkeitsdefinitionen in Docker Compose wird sichergestellt, dass das Backend erst nach vollständiger Verfügbarkeit der Datenbank startet.

Das Frontend wird auf Grundlage eines zuvor erstellten statischen Builds in einem separaten Container über einen Webserver bereitgestellt. Die Orchestrierung über Docker Compose stellt sicher, dass der Start des Frontend-Containers erst erfolgt, nachdem das Backend erfolgreich gestartet wurde. Dadurch wird die korrekte Initialisierungsreihenfolge der Systemkomponenten gewährleistet.

Die gewählte Containerstruktur erhöht die Wartbarkeit und Skalierbarkeit der Anwendung. Änderungen an einzelnen Komponenten können unabhängig voneinander vorgenommen werden, ohne das Gesamtsystem direkt zu beeinflussen. Der deklarative Aufbau erlaubt zudem eine schnelle und konsistente Reproduktion der Umgebung auf unterschiedlichen Zielsystemen – sei es für Entwicklungs-, Test- oder Produktivzwecke. Insgesamt leistet die gewählte Konfiguration einen wesentlichen Beitrag zur Stabilität, Wiederverwendbarkeit und Betriebssicherheit der Anwendung.

# 5 Fazit und Ausblick

## Fazit

---

Kapitel  
Fazit  
schrei-  
ben

## Ausblick

Im Rahmen der vorliegenden Studienarbeit konnten aufgrund begrenzter zeitlicher und inhaltlicher Ressourcen nicht alle potenziellen Erweiterungen und Funktionalitäten umgesetzt werden. Während der Entwicklung wurde eine lokale Datenbank eingesetzt, da sie für den prototypischen Charakter des Projekts eine einfache und effiziente Lösung darstellt. Diese Datenbank ermöglicht jedoch keine persistente Speicherung über unterschiedliche Systemstarts oder Systemumgebungen hinweg und ist daher für den produktiven Einsatz nicht geeignet. Für den späteren produktiven Betrieb empfiehlt sich daher die Ablösung der lokalen Datenbank durch eine persistent verfügbare Lösung, beispielsweise eine Cloud-Datenbank oder eine zentral auf einem Firmenserver gehostete Datenbank. Welche Variante gewählt wird, hängt maßgeblich von zukünftigen Anforderungen und Rahmenbedingungen im produktiven Einsatz ab – insbesondere im Hinblick auf Skalierbarkeit, Zugriffssicherheit und Integration in bestehende Systemlandschaften. Da diese Anforderungen zum Zeitpunkt der Studienarbeit nicht definiert werden können, wird auf eine frühzeitige Integration einer solchen Lösung bewusst verzichtet. Die lokale Datenbank verbleibt somit als temporäre Lösung im Rahmen der Entwicklungsumgebung. Es ist ausdrücklich darauf hinzuweisen, dass diese nicht für den produktiven Betrieb vorgesehen ist.

Darüber hinaus sind weitere Funktionalitäten identifiziert worden, die potenziell einen Mehrwert für die Benutzerfreundlichkeit und Erweiterbarkeit des Systems bieten würden. So könnte beispielsweise eine Funktion implementiert werden, die es dem Benutzer erlaubt, neue Prozesse über die grafische Benutzeroberfläche zu laden. Dabei würde die zugehörige XML-Datei direkt in die Datenbank übernommen und der neue Prozess würde anschließend zur Bearbeitung zur Verfügung stehen.

Ein weiteres Beispiel ist die Möglichkeit, den Drag-and-Drop-Arbeitsbereich auf Knopfdruck zu vergrößern, sodass der Nutzer den gesamten Bildschirm zur Bearbeitung von Prozessen nutzen kann. Solche Erweiterungen würden die Interaktion mit dem System verbessern und die Effizienz bei der Prozessmodellierung erhöhen.

Diese Beispiele verdeutlichen, dass Softwareprojekte grundsätzlich einem iterativen Entwicklungsprozess unterliegen. Sie werden durch Rückmeldungen von Anwendern und sich ändernde Anforderungen kontinuierlich weiterentwickelt. Auch das hier vorgestellte System bietet Potenzial für zukünftige Funktionserweiterungen, die im Rahmen weiterer Arbeiten oder eines produktiven Einsatzes realisiert werden könnten.

Komplette  
Arbeit  
nochmal  
lesen

# Anhang

## Login

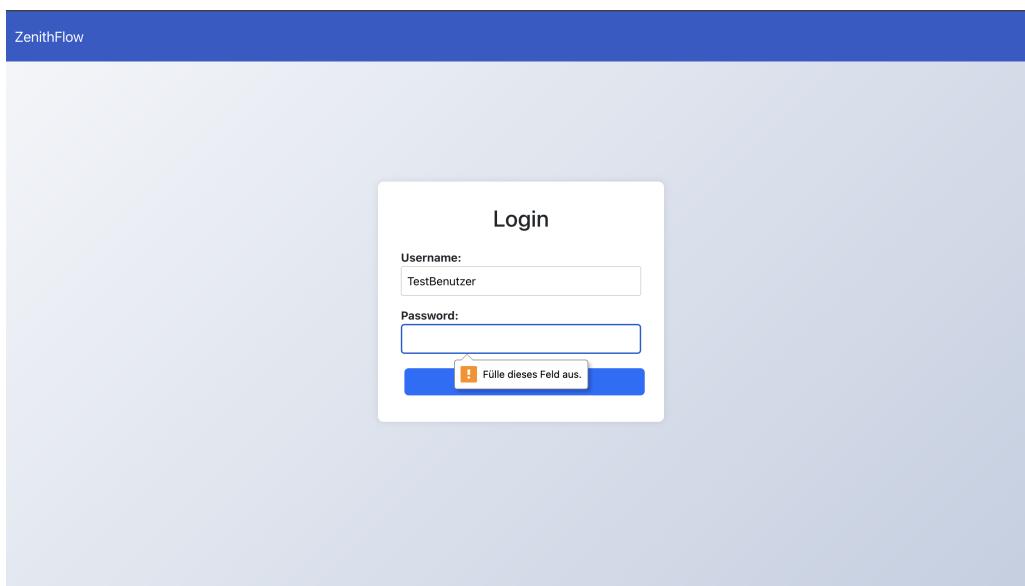


Abbildung A1: Fehlernachricht beim Login – leeres Login-Feld

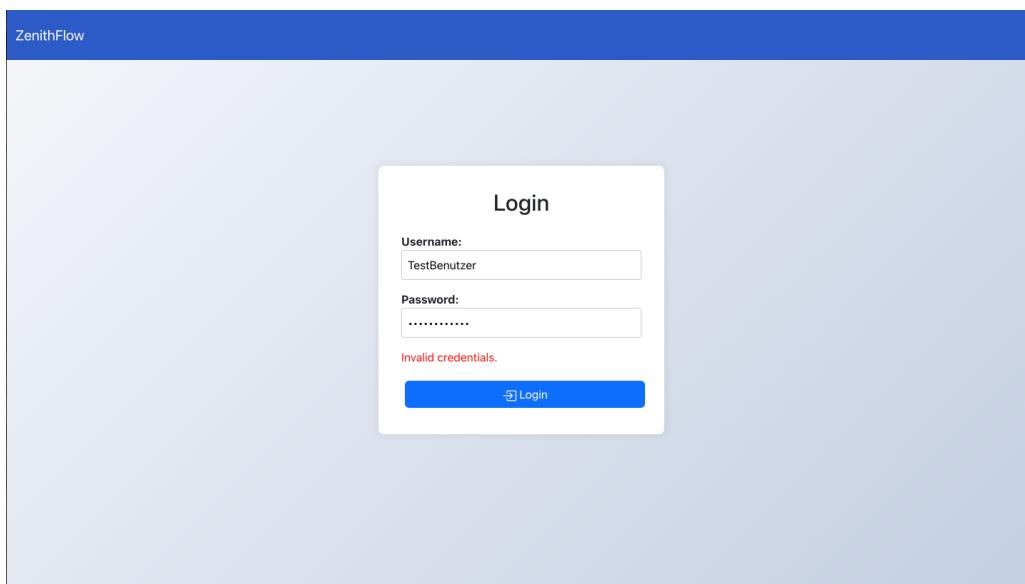


Abbildung A2: Fehlernachricht beim Login – falsche Anmeldedaten

```
1 // App.js
2 import React from 'react';
3 import { BrowserRouter as Router, Routes, Route } from 'react-
4 router-dom';
5 import LoginPage from './loginPage';
6 import StartPage from './startpage';
7 import ManageProcess from './manageProcess';
8 import ExecuteProcess from './executeProcess';
9 import UserManagement from './userManagement';
10 import ProjectManagement from './projectManagement';
11 import PrivateRoute from './privateRoute';
12 import Notifications from './notificationPage';
13 import ChangePassword from './changePassword';
14 import ArchivedInstancesPage from './archivedInstances';
15
16 function App() {
17   return (
18     <Router>
19       <Routes>
20         <Route path="/" element={<LoginPage />} />
21         <Route path="/start" element={
22           <PrivateRoute>
23             <StartPage />
24           </PrivateRoute>
25         }
26         <Route path="/manage-users" element={
27           <PrivateRoute>
28             <UserManagement />
29           </PrivateRoute>
30         }
31         <Route path="/manage-process" element={
32           <PrivateRoute>
33             <ManageProcess />
34           </PrivateRoute>
35         }
36         <Route path="/execute-process" element={
37           <PrivateRoute>
38             <ExecuteProcess />
39           </PrivateRoute>
40         }
41       </Routes>
42     </Router>
43   )
}
```

```
44      <Route path="/notifications" element={  
45          <PrivateRoute>  
46              <Notifications />  
47          </PrivateRoute>  
48      }  
49  />  
50  <Route path="/manage-projects" element={  
51      <PrivateRoute>  
52          <ProjectManagement />  
53      </PrivateRoute>  
54  }  
55  />  
56  <Route path="/change-password" element={  
57      <PrivateRoute>  
58          <ChangePassword />  
59      </PrivateRoute>  
60  }  
61  />  
62  <Route path="/archived-instances" element={  
63      <PrivateRoute>  
64          <ArchivedInstancesPage />  
65      </PrivateRoute>  
66  }  
67  />  
68  </Routes>  
69  </Router>  
70  );  
71 }  
72  
73 export default App;
```

Code A1: Das vollständige Routing im Process-Manager

## Start-Seite

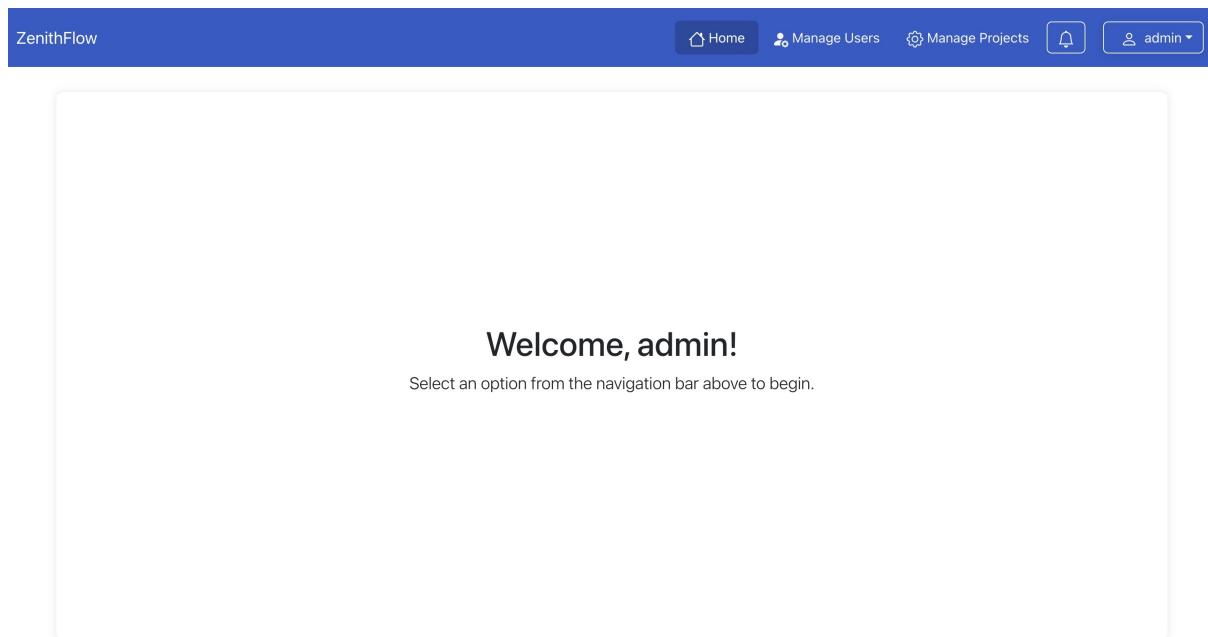
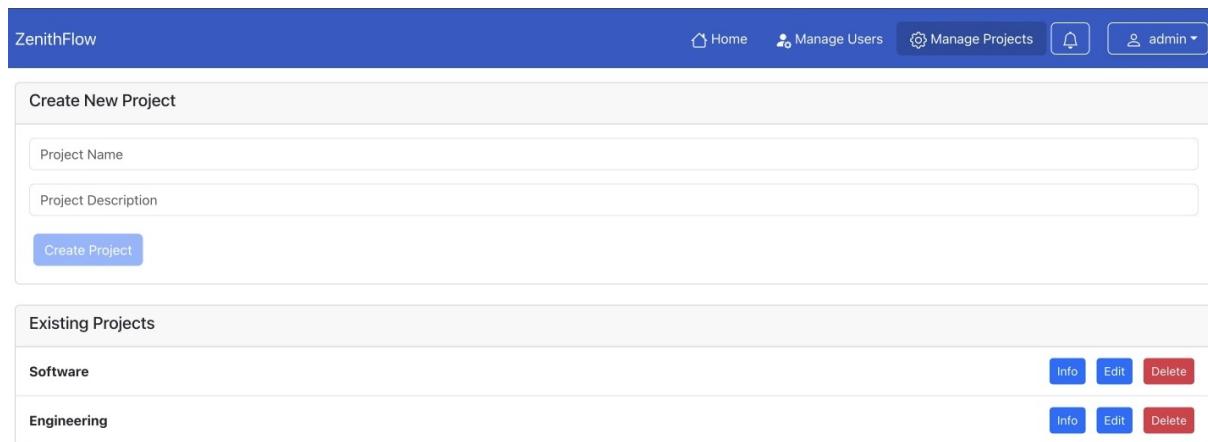


Abbildung A3: Die Admin Start-Seite

## Projekt-Management



The screenshot shows the ZenithFlow Project Management interface. At the top, there is a blue header bar with the text "ZenithFlow" on the left and navigation links for "Home", "Manage Users", "Manage Projects", a notification bell, and a user account for "admin". Below the header, there are two main sections: "Create New Project" and "Existing Projects". The "Create New Project" section contains fields for "Project Name" and "Project Description", and a "Create Project" button. The "Existing Projects" section lists categories "Software" and "Engineering", each with "Info", "Edit", and "Delete" buttons.

Abbildung A4: Die vollständige Projektverwaltungs-Seite

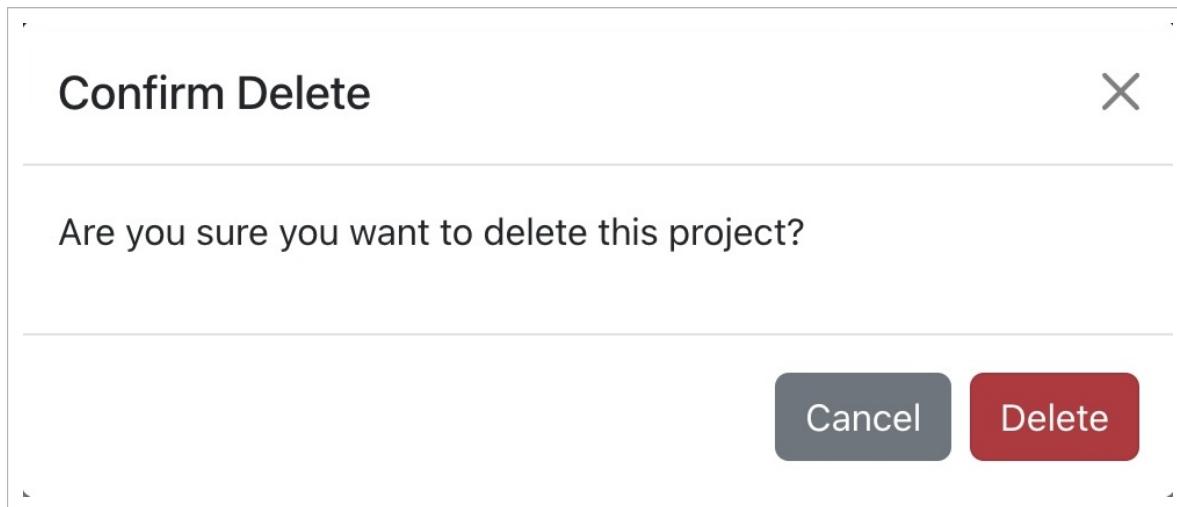
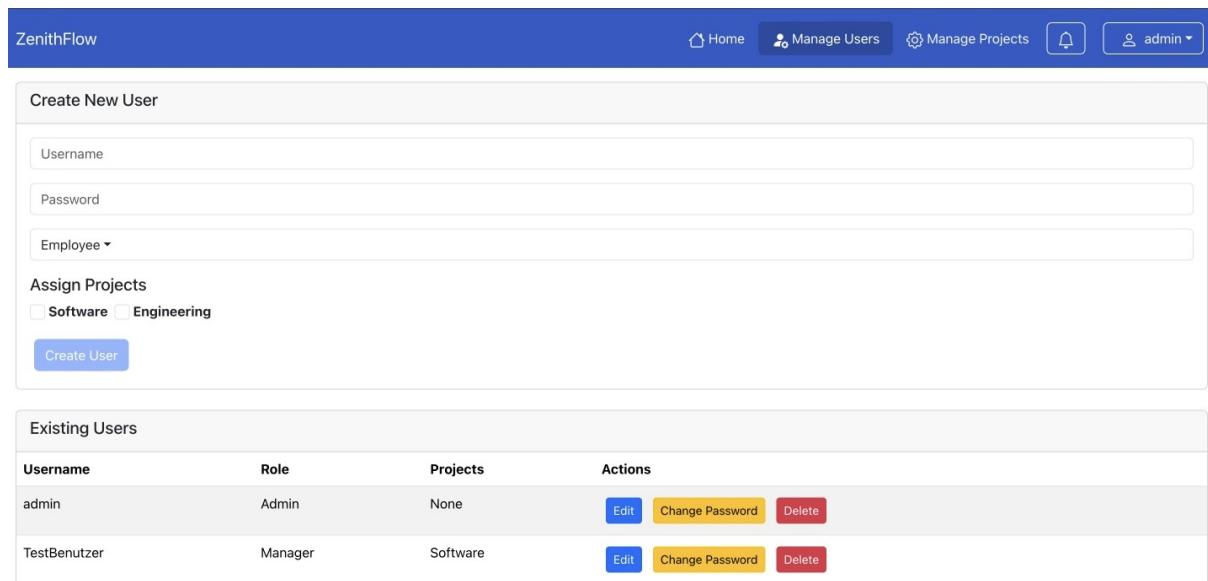


Abbildung A5: Sicherheitsabfrage vor dem Löschen eines erstellten Projekts

## Benutzer-Management



The screenshot shows the ZenithFlow User Management interface. At the top, there is a navigation bar with links for Home, Manage Users, Manage Projects, a notification bell, and a user account for 'admin'. Below the navigation bar, there are two main sections: 'Create New User' and 'Existing Users'.

**Create New User:** This section contains input fields for 'Username' and 'Password', and a dropdown menu for 'Employee'. Below these, there is a 'Assign Projects' section with checkboxes for 'Software' and 'Engineering', and a 'Create User' button.

**Existing Users:** This section displays a table of users with columns for 'Username', 'Role', 'Projects', and 'Actions'. The table contains two rows:

Username	Role	Projects	Actions
admin	Admin	None	<a href="#">Edit</a> <a href="#">Change Password</a> <a href="#">Delete</a>
TestBenutzer	Manager	Software	<a href="#">Edit</a> <a href="#">Change Password</a> <a href="#">Delete</a>

Abbildung A6: Die vollständige Benutzerverwaltungs-Seite

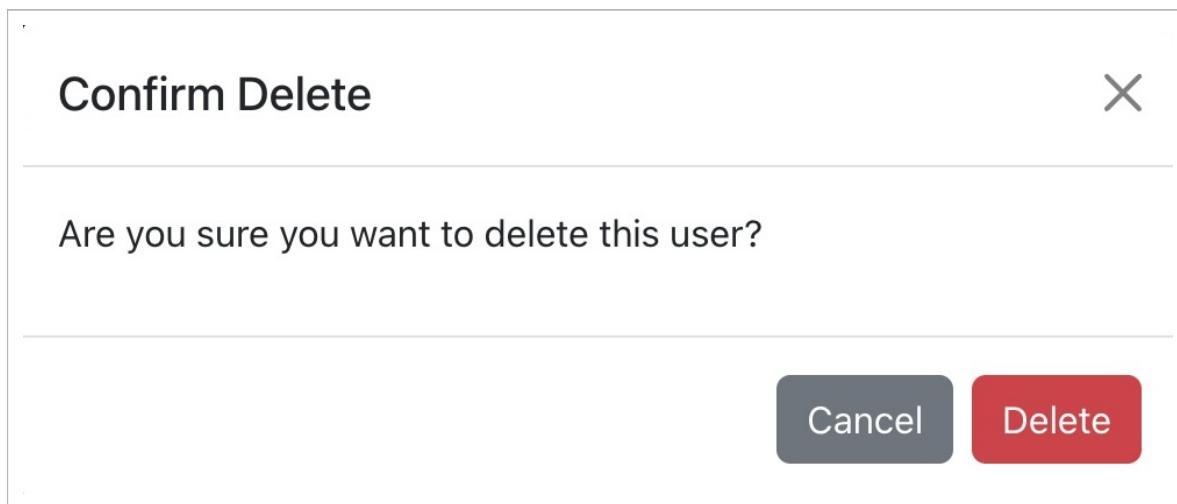


Abbildung A7: Sicherheitsabfrage vor dem Löschen eines erstellten Benutzers

## Passwort ändern

### Change Password

**Current Password**  
...

**New Password**  
.....

**Confirm New Password**  
.....

Current password is incorrect.

**Set new Password**

**← Back**

Abbildung A8: Fehlernachricht aufgrund eines falschen aktuellen Passwort

# Change Password

**Current Password**

**New Password**

**Confirm New Password**

New passwords do not match.

**Set new Password**

**← Back**

Abbildung A9: Fehlernachricht aufgrund nicht übereinstimmende neue Passwörter

# Change Password

**Current Password**

**New Password**

**Confirm New Password**

Password updated successfully.

**Set new Password**

**← Back**

Abbildung A10: Erfolgreiche Änderung des Passworts durch den Benutzer

## Benachrichtigung

## BPMN-Erweiterungen

```
1  export default {
2      __init__: ['customRules'],
3      customRules: ['type', CustomRules]
4  };
5
6  function CustomRules(eventBus, bpmnRules) {
7      // Add a custom rule for deleting elements.
8      bpmnRules.addRule('elements.delete', 1500, (context) => {
9          const { elements } = context;
10         if (!elements) {
11             return true;
12         }
13         // Block deletion if any element's business object id is "
14         // StartEvent_1"
15         return elements.every(element => {
16             if (element.businessObject && element.businessObject.id
17                 === 'StartEvent_1') {
18                 return false;
19             }
20             return true;
21         });
22     }
23     CustomRules.$inject = ['eventBus', 'bpmnRules'];

```

Code A2: Die Datei customRules

```
1 export const customExtension = {
2     name: 'CustomExtension',
3     uri: 'http://example.com/schema/bpmn/role',
4     prefix: 'role',
5     xml: {
6         tagAlias: 'lowerCase'
7     },
8     types: [
9         {
10             name: 'customExtension',
11             extends: ['bpmn:BaseElement'],
12             properties: [
13                 {
14                     name: 'role',
15                     isAttr: true,
16                     type: 'String'
17                 },
18                 {
19                     name: 'description',
20                     isAttr: true,
21                     type: 'String'
22                 }
23             ]
24         }
25     ]
26 };
```

Code A3: Die Datei customExtensions

## Verwalten der Prozesse

Process Information

Complex\_Software\_Test\_Process

Software ▾

Request Save Create New Process

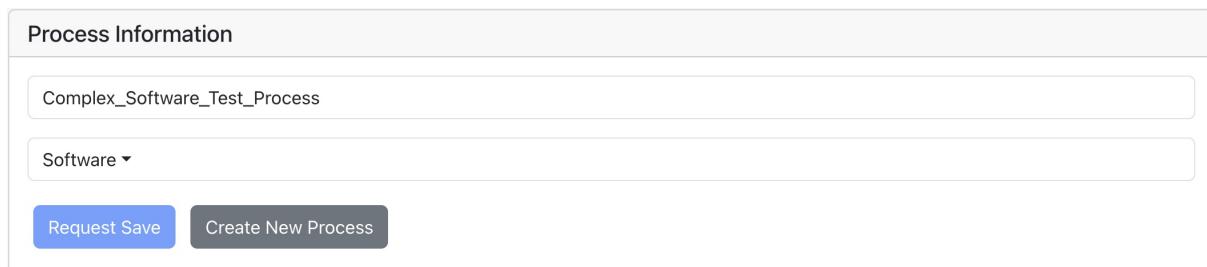


Abbildung A11: Das Prozess-Informationsfenster eines Benutzers

Process Information

Complex\_Software\_Test\_Process\_Change

Software ▾

Save to Database Create New Process



Abbildung A12: Das Prozess-Informationsfenster eines Managers mit Änderungen

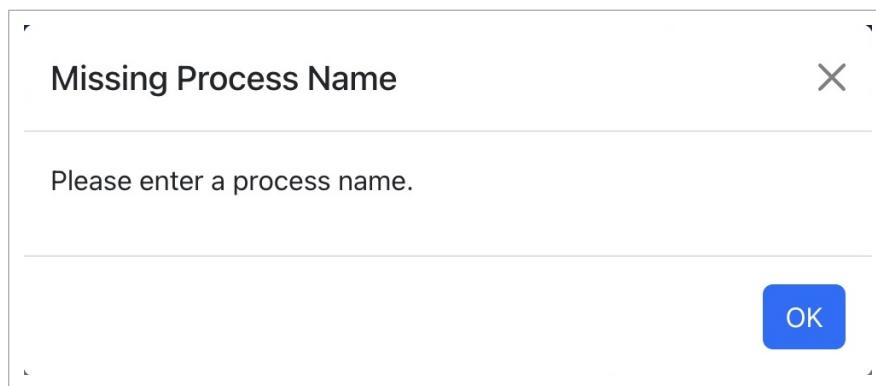


Abbildung A13: Die Fehlernachricht bei fehlendem Prozess-Namen

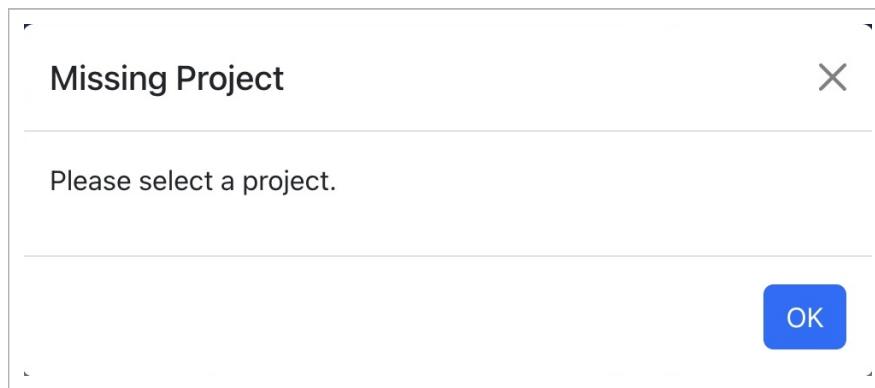


Abbildung A14: Die Fehlernachricht bei fehlendem Projekt

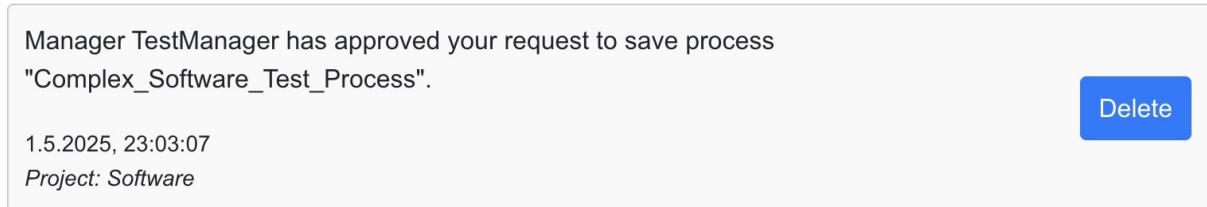


Abbildung A15: Die positive Rückmeldung des Änderungsantrags

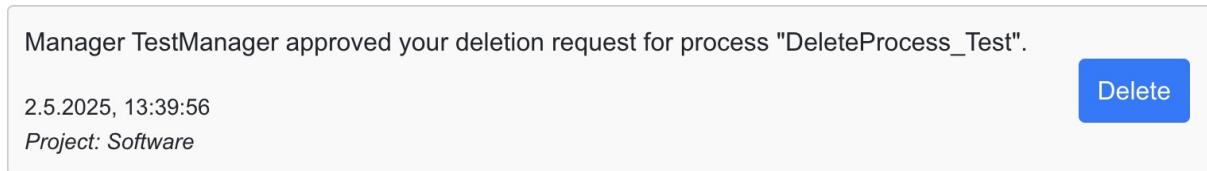


Abbildung A16: Die negative Rückmeldung nach Anfrage zum Löschen

## Ausführen der Prozesse

# Glossar

## **Asynchrone Programmierung**

Asynchrone Programmierung bezeichnet ein Programmierparadigma, bei dem Funktionen nicht-blockierend ausgeführt werden. Anstatt auf das Ergebnis einer Operation (z. B. einer Datenbankabfrage oder Netzwerkkommunikation) zu warten und dabei den Programmfluss anzuhalten, wird die Ausführung fortgesetzt und das Ergebnis später verarbeitet, sobald es verfügbar ist. In JavaScript erfolgt dies typischerweise durch `async/await`. Dieses Konzept erhöht die Effizienz und Reaktionsfähigkeit von Anwendungen, insbesondere im Kontext von I/O-lastigen Operationen wie Datenbankzugriffen oder API-Anfragen.. 36

## **Docker**

Docker ist eine Open-Source-Plattform zur Containerisierung von Anwendungen. Sie ermöglicht es, Software samt aller benötigten Abhängigkeiten und Konfigurationen in standardisierten Containern zu isolieren und auszuführen. Diese Container laufen unabhängig vom zugrunde liegenden Betriebssystem und bieten dadurch eine konsistente und reproduzierbare Ausführungsumgebung über verschiedene Systeme hinweg. Docker wird insbesondere für Entwicklung, Test und Deployment eingesetzt, da es die Portabilität von Anwendungen erhöht, manuelle Konfigurationsaufwände reduziert und die Integration in automatisierte Bereitstellungsprozesse (CI/CD) erleichtert.. 82