Implementierung eines webbasierten BPMN-Editors mit interaktiver Bearbeitung und JSON-zu-BPMN-Konvertierung

DIPLOMARBEIT

VisuPro

Höhere Abteilung für Informatik

01.07.2025 – 26.03.2026

PROJEKTMITGLIEDER:

Mathias Grubbauer
Justin Luckeneder
Alexander Vorreither

PROJEKTBETREUER:

Prof. Dipl.-Ing. Christian Reisinger

IN ZUSAMMENARBEIT MIT BMD SYSTEMHAUS GESMBH:

Andreas Fuchs
Hannes Großauer

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet. Das generative KI-Tool ChatGPT wurde von uns zur Überprüfung der Rechtschreibung und Grammatik der gesamten Arbeit genutzt. Die Korrekturen sowie Formulierungen wurden vom Diplomarbeitsteam selbstständig umgesetzt.

Mathias Grubbauer

Justin Luckeneder

Alexander Vorreither

Gendererklärung

Im Rahmen dieser Diplomarbeit wird aus Gründen der besseren Lesbarkeit auf die gleichzeitige Verwendung der weiblichen sowie der männlichen Form verzichtet. Alle Personenbezeichnungen beziehen sich gleichermaßen auf alle Geschlechter. Diese Vereinfachungen dienen ausschließlich der Lesbarkeit und sollen in keiner Weise Diskriminierung oder Benachteiligung bestimmter Geschlechter implizieren.

Mathias Grubbauer

Justin Luckeneder

Alexander Vorreither

Danksagung

Wir bedanken uns bei sämtlichen Personen der HTL-Perg und BMD, die diese Diplomarbeit ermöglicht haben.

Unser besonderer Dank gilt unserem Diplomarbeitsbetreuer, Herrn Professor OStR. Dipl.-Ing. Christian Reisinger, für die fachliche Unterstützung, hilfreichen Anregungen und die konstruktive Begleitung während der gesamten Arbeit.

Ebenso danken wir unseren Betreuern Andreas Fuchs und Hannes Großauer von der BMD Systemhaus GesmbH, welche uns jederzeit tatkräftig während unseres Praktikums unterstützt haben.

Darüber hinaus möchten wir allen Kolleginnen und Kollegen der BMD danken, die uns durch ihre Hilfsbereitschaft, ihre Ratschläge und die Zusammenarbeit geholfen haben, den praktischen Teil unserer Arbeit erfolgreich zu realisieren.

Kurzbeschreibung

Die BMD Systemhaus GesmbH entwickelt betriebswirtschaftliche Softwarelösungen für Unternehmen verschiedenster Branchen. Innerhalb dieser Systeme werden zahlreiche Qualitätsmanagementprozesse digital dokumentiert. Zur Zeit liegen die vorhandenen Abläufe jedoch hauptsächlich in tabellarischer Form, beziehungsweise als Auflistung vor, was für Anwender oft unübersichtlich und schwer verständlich ist. Beispielsweise sind Zusammenhänge zwischen den einzelnen Prozessschritten nicht sofort ersichtlich. Das Ziel von „VisuPro“ ist eine verständliche und übersichtliche Visualisierung dieser Prozesse bereitzustellen, was nicht nur die Dokumentation von Abläufen verbessert, sondern auch Schulungen neuer Mitarbeiter erleichtert.

Aus diesem Grunde wurde im Rahmen dieser Diplomarbeit eine webbasierte Anwendung entwickelt, welche die bestehenden Prozessdaten automatisch in ein grafisches Prozessmodell überführt. Als Grundlage dient dabei die Modellierungssprache Business Process Model and Notation (BPMN), ein international verbreiteter Standard zur Beschreibung von Geschäftsprozessen. Durch die Verwendung dieser Notation können Abläufe strukturiert dargestellt werden und sind dadurch für technische sowie fachliche Anwender leichter nachvollziehbar. Die entwickelte Anwendung besteht aus mehreren Komponenten.

Über eine REST-Schnittstelle ruft das System Prozessdaten aus dem bestehenden Datenhaltungssystem ab und bereitet sie strukturiert auf. Ein Konvertierungsalgorithmus transformiert diese Daten in ein BPMN-konformes XML-Format, das sich anschließend in einem Web-Editor anzeigen lässt. Änderungen im Diagramm werden anschließend wieder in das ursprüngliche, strukturierte Datenformat zurückgeführt.

Das Ergebnis der Diplomarbeit ist ein funktionsfähiger Prototyp eines webbasierten BPMN-Editors, der die Darstellung und das Verständnis von Geschäftsprozessen verbessert. Die bestehenden Prozessdaten werden durch einen Algorithmus in die BPMN-Form gebracht und eine REST-Schnittstelle ermöglicht die reibungslose Kommunikation zwischen der App und dem Backend der BMD. Die Webanwendung demonstriert die Integration einer grafischen Prozessdarstellung in eine bestehende Unternehmenssoftware und wie man bestehende Teile einer etablierten Softwarelösung weiter verbessern kann.

Abstract

This diploma thesis was carried out in cooperation with BMD Systemhaus GesmbH, a company that develops business software solutions for various businesses. The goal of the project was to develop a web-based application that converts existing quality management process data into graphical process models.

Business processes in enterprise software are often stored and managed in structured formats such as tables or lists. While these formats are suitable for documentation and data storage, they can make workflows, especially complex ones, difficult to understand. Relationships between individual process steps, for example, are not always immediately visible, which complicates training, analysis and documentation of business processes.

The developed system fetches process data from the BMD software environment via a REST interface and prepares it for further processing. A conversion algorithm transforms the structured data into BPMN-compliant XML, which can then be visualized and edited within a browser-based editor. The graphical rendering and interaction with the diagrams are implemented using the open-source library `bpmn-js`. Changes made to the diagrams can subsequently be converted back into the original data format, including model-specific information such as element position and size.

The resulting prototype demonstrates how existing process data can be converted into BPMN-diagrams and integrated into an enterprise software environment. By providing a graphical representation of workflows, the application improves the transparency and understandability of business processes.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel werden die Entstehung und der Hintergrund dieser Diplomarbeit erläutert. Die Ziele des Projekts und das Projektumfeld werden ebenfalls dargestellt.

1.1 Ausgangslage

In vielen Unternehmen werden Geschäftsprozesse digital dokumentiert. Die Darstellung dieser Abläufe erfolgt jedoch häufig in tabellarischer Form oder als einfache Listen. Eine solche Form der Aufbereitung ist für Anwender oft nur schwer nachvollziehbar und erschwert insbesondere die Analyse, Kommunikation sowie die Weiterentwicklung von Prozessen.

Auch beim Auftraggeber dieses Projekts, der BMD SYSTEMHAUS GesmbH mit Sitz in Steyr, werden betriebliche Qualitätsmanagement-Prozesse in der vorhandenen Software digital erfasst, jedoch überwiegend in einer listenbasierten Struktur dargestellt. Dadurch fehlt eine übersichtliche und anschauliche Darstellung der einzelnen Prozessschritte und ihrer Zusammenhänge. Gerade für Schulungen, die Einarbeitung neuer Mitarbeiter oder externe Prüfungen besitzt eine verständliche grafische Aufbereitung der Abläufe große Bedeutung. Die organisatorischen Rahmenbedingungen des Projekts werden im Abschnitt 3.1 näher beschrieben.

Eine visuelle Modellierung von Geschäftsprozessen erleichtert es, auch komplexe Abläufe verständlich darzustellen. Standards wie BPMN (Business Process Model and Notation) ermöglichen dabei eine einheitliche und weit verbreitete Form der Prozessdarstellung. Grafische Modelle unterstützen die Analyse, Dokumentation und Bewertung von Abläufen und helfen dabei, Verbesserungspotenziale leichter zu erkennen. Eine Einführung in die verwendeten Grundlagen und Konzepte findet sich im Kapitel 2.

Aus dieser Ausgangssituation entstand die Idee, bestehende Prozessinformationen in einer grafischen Form aufzubereiten und damit eine übersichtlichere Darstellung betrieblicher Abläufe zu ermöglichen. Dadurch rücken Zusammenhänge zwischen einzelnen Prozessschritten stärker in den Vordergrund und Prozesse werden leichter zugänglich.

Das im Rahmen dieser Diplomarbeit entwickelte System greift diese Problemstellung auf und schafft die Grundlage für eine verständlichere und praxisnähere Darstellung von Geschäftsprozessen. Die Ergebnisse der entwickelten Lösung werden in Kapitel 5 präsentiert.

1.2 Zielsetzung

Ziel dieser Diplomarbeit ist die Entwicklung eines Prototyps zur grafischen Darstellung und Bearbeitung von Geschäftsprozessen. Es handelt sich dabei um eine Neuentwicklung, die auf bereits vorhandenen Prozessdaten des Auftraggebers aufbaut. Diese Prozessdaten werden in übersichtlicher und verständlicher Form aufbereitet.

Aus fachlicher Sicht verfolgt das Projekt das Ziel, betriebliche Abläufe besser nachvollziehbar, leichter analysierbar und einfacher kommunizierbar zu machen. Darüber hinaus verbessert die Anwendung die Dokumentation von Prozessen und unterstützt deren Nutzung in Schulungen, bei der Einarbeitung neuer Mitarbeiter sowie im Bereich der Qualitätssicherung. Damit greift die Arbeit die in der Projektdefinition festgelegten Geschäftsziele auf, insbesondere die Verbesserung der Übersichtlichkeit und Verständlichkeit von QM-Prozessen sowie die Unterstützung der internen Kommunikation.

Auf Projektebene umfasst die Arbeit die Aufbereitung ausgewählter Prozessdaten des Auftraggebers und deren Überführung in grafische Prozessmodelle. Darauf aufbauend entsteht ein Prototyp, in dem diese Modelle nicht nur dargestellt, sondern auch bearbeitet und weiterentwickelt werden können. Zusätzlich untersucht das Projekt exemplarisch, ob Änderungen an grafischen Prozessmodellen wieder in eine strukturierte Datenbasis zurückgeführt werden können. Dieser Aspekt wird als Proof of Concept umgesetzt.

Langfristig trägt die entwickelte Lösung dazu bei, Geschäftsprozesse im Unternehmen verständlicher und leichter nutzbar zu machen. Sie unterstützt damit eine bessere Dokumentation, eine klarere interne Abstimmung und qualitätssichernde Maßnahmen im Unternehmensalltag.

1.3 Überblick über den Inhalt der Arbeit

Im Rahmen dieser Diplomarbeit wurde eine Anwendung zur Visualisierung und Bearbeitung von Geschäftsprozessen auf Basis des BPMN-Standards entwickelt. Im Mittelpunkt steht dabei die grafische Aufbereitung bestehender Prozessinformationen sowie die Möglichkeit, diese Modelle gezielt anzupassen und weiterzuentwickeln.

Kapitel 2 behandelt die fachlichen und technischen Grundlagen des Projekts. Dort werden die verwendeten Konzepte, Standards und Technologien erläutert, die für das Verständnis der Arbeit erforderlich sind.

Kapitel 3 beschreibt die organisatorischen Rahmenbedingungen des Projekts. Dazu zählen insbesondere die Projektplanung, die Zusammenarbeit im Team sowie die Abstimmung mit dem Auftraggeber.

Kapitel 4 widmet sich der Umsetzung der entwickelten Lösung. In diesem Kapitel werden das Mock-up und der Prototyp sowie die zentralen Bestandteile der Anwendung näher beschrieben.

Kapitel 5 stellt die erzielten Ergebnisse vor und enthält eine Analyse der entwickelten Lösung im Hinblick auf die gesetzten Ziele.

1.4 Projektumfeld

Das im Rahmen dieser Diplomarbeit entwickelte System entstand in Zusammenarbeit mit der BMD SYSTEMHAUS GesmbH. Es orientiert sich an den bestehenden Anforderungen des Unternehmens und wurde für den Einsatz im Umfeld der vorhandenen Softwarelösung konzipiert. Damit steht das Projekt in engem Bezug zu den betrieblichen Abläufen und zur bestehenden Prozessdokumentation des Auftraggebers.

Die entwickelte Anwendung wird im Kontext der BMD-Software NTCS (siehe Glossar, Eintrag: NTCS) bereitgestellt und in deren Arbeitsumgebung eingebunden. Dadurch entsteht eine Lösung, die nicht losgelöst vom Unternehmensalltag betrachtet wird, sondern in einem bereits vorhandenen organisatorischen und fachlichen Rahmen eingesetzt werden kann.

Im Mittelpunkt steht dabei die grafische Darstellung und Bearbeitung von Geschäftsprozessen. Bestehende Prozessinformationen können in einer verständlichen Form aufbereitet und für unterschiedliche Anwendungsbereiche nutzbar gemacht werden. Dazu zählen insbesondere die Dokumentation betrieblicher Abläufe, die interne Abstimmung, Schulungszwecke sowie die Unterstützung qualitätssichernder Maßnahmen.

Das Projektumfeld ist somit durch die enge Zusammenarbeit mit dem Auftraggeber, die Orientierung an realen Unternehmensprozessen sowie die Einbindung in eine bestehende Softwarelandschaft geprägt. Die konkrete technische Umsetzung der entwickelten Lösung wird im Kapitel 4 näher beschrieben.

2 Grundlagen und Methoden

In diesem Kapitel werden die grundlegenden Begriffe, Methoden und Technologien vorgestellt, die für das Projekt relevant sind. Dazu zählen zentrale Fachbegriffe, sowie die im Projekt verwendeten Technologien, Entwicklungswerkzeuge, Bibliotheken, Softwarekomponenten und Schnittstellen.

2.1 Grundlegende Fachbegriffe und Konzepte

2.1.1 Grafisch orientierte Modellierungsmethoden im Geschäftsprozessmanagement

Im Geschäftsprozessmanagement (GPM) werden betriebliche Abläufe in Unternehmen systematisch beschrieben, dokumentiert, optimiert und überwacht. Ziel des Geschäftsprozessmanagements ist es, vorhandene Informationen über betriebliche Abläufe zu nutzen, um Prozesse effizienter, transparenter und anpassungsfähiger zu gestalten. Dies erfolgt durch die Analyse, Modellierung, Implementierung, Ausführung, Überwachung sowie die kontinuierliche Verbesserung von Geschäftsprozessen.

Geschäftsprozesse bestehen aus einer Abfolge von Ereignissen, Aktivitäten und Entscheidungen, die in einem logischen Zusammenhang stehen und daher nicht isoliert betrachtet werden können. Eine zentrale Rolle spielt dabei die grafische Prozessmodellierung. Durch standardisierte Notationen können komplexe Abläufe verständlich dargestellt werden, was das Zusammenspiel verschiedener Abteilungen und Interessengruppen verbessert. Zu den bekanntesten Modellierungssprachen in diesem Bereich zählen die Ereignisgesteuerte Prozesskette (EPK) sowie die Business Process Model and Notation (BPMN).[1]

[1] Vgl. Kocian, *Geschäftprozessmodellierung mit BPMN 2.0. Business Process Model and Notation im Methodenvergleich*, S. 4–5; Rump, *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten*, S. 12.

2.1.1.1 Business Process Model and Notation (BPMN)

Ursprung

Die BPMN ist eine standardisierte grafische Modellierungssprache für Geschäftsprozesse, welche 2004 von der Business Process Management Initiative (BPMI) entwickelt und veröffentlicht wurde, um eine standardisierte, verständliche Methode zur Modellierung von Geschäftsprozessen bereitzustellen. Nach der Übernahme durch die Object Management Group (OMG) wurde BPMN weiterentwickelt und 2011 als Version 2.0 veröffentlicht, aktuell gültig ist die Version 2.0.2.[2] BPMN ist inzwischen international anerkannt und gilt als der De-facto-Standard für die Geschäftsprozessmodellierung.[3]

Grundprinzip

Die BPMN dient der grafischen Modellierung von Geschäftsprozessen, wobei sie zwei Hauptziele verfolgt:

1. Kommunikation: BPMN schafft eine einheitliche Notation, die sowohl von Fachanwendern, als auch von Entwicklern verstanden werden kann. Dies erleichtert die Zusammenarbeit zwischen Prozessanalysten, Entwicklern und Management.
2. Dokumentation und Ausführbarkeit: Neben der reinen Visualisierung erlaubt BPMN eine präzise Spezifikation von Abläufen, die in Workflow- oder Prozessmanagementsystemen teilweise direkt ausführbar sind.[4]

Relevanz für diese Arbeit

Im Rahmen dieser Diplomarbeit werden vorhandene Prozessdaten aus einem bestehenden System automatisiert in BPMN-Diagramme überführt und anschließend grafisch dargestellt. Die

[2] Vgl. Object Management Group, *Business Process Model & Notation (BPMN)*; Wikipedia contributors, *Business Process Model and Notation*.

[3] Vgl. Object Management Group, *Business Process Model & Notation (BPMN)*; Wikipedia contributors, *Business Process Model and Notation*; Object Management Group, *Business Process Model and Notation (BPMN) Version 2.0.2*.

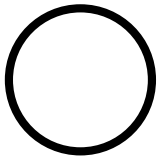
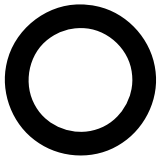
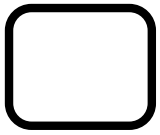
[4] Vgl. Göpfert und Lindenbach, *Geschäftsprozessmodellierung mit BPMN 2.0: Business Process Model and Notation*, S.3; Partsch, *Transformation zwischen den ausgewählten Modelliersprachen EPK und BPMN*, S.25; Kocian, *Geschäftprozessmodellierung mit BPMN 2.0. Business Process Model and Notation im Methodenvergleich*, S.6-7.

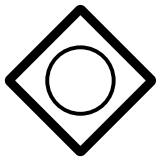
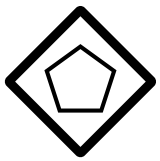
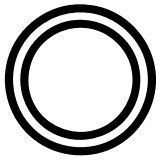
BPMN eignet sich dafür besonders gut, da sie sowohl eine klare visuelle Struktur als auch ein standardisiertes Austauschformat besitzt.

Die erzeugten BPMN-Modelle können dadurch im entwickelten Editor visualisiert, bearbeitet und als strukturierte Prozessbeschreibung weiterverwendet werden.

Notationselemente der BPMN

In der entwickelten Anwendung werden nicht alle verfügbaren BPMN-Notationselemente verwendet. In der folgenden Tabelle sind daher nur jene BPMN-Notationselemente aufgeführt, die im Rahmen dieser Arbeit verwendet werden. Eine vollständige Übersicht der BPMN-Notationselemente findet sich in der Online-Referenz Camunda, *BPMN-Notationselemente*.

Symbol	**Erklärung**
	**Start-Ereignis (Start Event)**: Markiert den Beginn eines Prozessablaufs. Ein Start-Ereignis löst den Prozess aus und ist der Punkt, an dem eine Instanz des Prozesses entsteht. Es wird durch einen einfachen Kreis dargestellt.
	**End-Ereignis (End Event)**: Zeigt das Ende eines Prozessablaufs an. Wenn ein End-Ereignis erreicht wird, wird der Prozess beendet und es werden keine weiteren Aktivitäten mehr ausgeführt.
	**Aufgabe (Task)**: Stellt eine einzelne Aktivität dar, die im Prozess ausgeführt wird. Tasks repräsentieren Arbeitsschritte oder Tätigkeiten, die von Menschen oder Systemen ausgeführt werden.
	**Gateway**: Gateways steuern den Fluss des Prozessablaufs, indem sie Verzweigungen, Zusammenführungen oder Entscheidungen darstellen. Sie selbst führen keine Arbeit aus, sondern definieren die Pfadlogik.

Symbol	Erklärung
	**Paralleles Gateway (Parallel Gateway)**: Ein Gateway, das den Prozessfluss gleichzeitig auf mehrere Pfade aufteilt oder mehrere Pfade wieder zusammenführt. Alle Wege werden parallel und unabhängig ausgeführt.
	**Inklusives Gateway (Inclusive Gateway)**: Erlaubt die Auswahl mehrerer Pfade, bei denen eine oder mehrere Bedingungen erfüllt werden können. Anders als beim exklusiven Gateway können mehrere Ausgänge gleichzeitig aktiviert werden.
	**Exklusives Gateway (Exclusive Gateway)**: Trifft eine Entscheidung, bei der genau ein Ausgangspfad gewählt wird. Basierend auf Bedingungen oder Daten wird ein einziger Pfad durchlaufen.
	**Ereignis-basiertes Gateway (Event-Based Gateway)**: Trifft Entscheidungen basierend auf externen Ereignissen, die eintreten können. Anstatt Bedingungen auszuwerten, wartet das Gateway auf eines von mehreren möglichen Ereignissen.
	**Komplexes Gateway (Complex Gateway)**: Ein Gateway für komplexe Verzweigungsregeln, die nicht durch einfache Bedingungen oder Standardlogiken abgedeckt werden. Es erlaubt benutzerdefinierte Regeln für Pfadauswahl und Synchronisation.
	**Zwischenereignis (Intermediate Event)**: Ein Ereignis, das während der Ausführung eines Prozesses auftritt und den weiteren Ablauf beeinflussen oder unterbrechen kann, beispielsweise durch Timer-, Nachrichten- oder Fehlersignale. In der entwickelten Anwendung wird dieses Symbol abweichend von der allgemeinen BPMN-Semantik zur Kennzeichnung eines Meilensteins beziehungsweise zum Markieren des Erreichens eines Meilensteins verwendet.

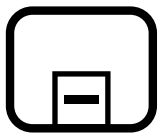
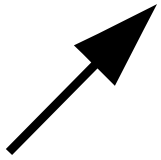
Symbol	Erklärung
	**Teilprozess Expanded Subprocess)**: Ein Subprozess fasst eine Gruppe von Aktivitäten zu einem modularen Baustein zusammen. Ein „expanded“ Subprozess zeigt seinen inneren Ablauf im Diagramm und kann komplexe Abläufe kapseln. Ein solcher Subprozess kann auch als „Collapsed Subprocess“ dargestellt werden, wodurch der interne Ablauf verborgen bleibt. Dadurch wird eventuell die Lesbarkeit erhöht, da das Diagramm nicht so lang erscheint, und man diese komplexen Subprozesse nur dann aufklappen und lesen kann, wenn man sie braucht.
	**Sequenzfluss (Sequence Flow)**: Verbindet BPMN-Elemente und zeigt die Reihenfolge und Richtung des Prozessablaufs an. Sequenzflüsse bestimmen, wie ein Token von einem Aktivitäts- oder Ereigniselement zum nächsten fließt.

Tabelle 1: Verwendete BPMN-Notationselemente[5]

2.1.1.2 Ereignisgesteuerte Prozesskette (EPK)

Ursprung

Die EPK wurde in den 1990er Jahren von August-Wilhelm Scheer im Rahmen des ARIS-Konzepts an der Universität Saarbrücken entwickelt. Ziel war es, eine Methode zur grafischen Beschreibung von Geschäftsprozessen bereitzustellen, bei der Ereignisse und Funktionen in einer logischen Abfolge dargestellt werden. Die EPK wurde insbesondere im deutschsprachigen Raum weit verbreitet und wird häufig zur fachlichen Beschreibung von Geschäftsprozessen eingesetzt.[6]

[5]Symbole: Camunda Services GmbH (BPMN SVG Icons).
Erklärungen: Vgl. (Camunda, *BPMN-Notationselemente*).

[6]Vgl. Partsch, *Transformation zwischen den ausgewählten Modelliersprachen EPK und BPMN*, S.18-21; Wikipedia, *Ereignisgesteuerte Prozesskette*.

Einordnung

Im Gegensatz zur BPMN ist die EPK primär auf die fachliche Beschreibung von Prozessen ausgerichtet und weniger stark auf eine technische Ausführung ausgelegt. Beide Modellierungsmethoden verfolgen jedoch dasselbe Ziel: Geschäftsprozesse strukturiert und nachvollziehbar darzustellen. Im Rahmen dieser Arbeit hat sich jedoch gezeigt, dass BPMN für die betrachteten Anwendungsfälle besser geeignet ist (siehe Abschnitt 5.4). Aus diesem Grund dient die EPK in dieser Arbeit nur als ergänzende Darstellung einer weiteren verbreiteten Prozessmodellierungssprache, während die Umsetzung der entwickelten Anwendung auf der BPMN basiert.[7]

Einsatz und Nutzen

1. Visualisierung und Analyse von Geschäftsprozessen: EPKs erlauben eine verständliche Darstellung von Prozessen, wobei Ereignisse (Zustände, die Funktionen auslösen) und Funktionen (Tätigkeiten innerhalb des Prozesses) abwechselnd angeordnet werden. Dadurch können Verantwortliche Engpässe, redundante Aufgaben oder Optimierungspotenziale identifizieren.
2. Unterstützung der Prozessmodellierung und Standardisierung: Durch die standardisierte Symbolik von EPKs wird sichergestellt, dass Prozesse konsistent über Abteilungen hinweg dokumentiert werden. Dies erleichtert natürlich die Kommunikation zwischen Fachabteilungen, IT-Abteilungen und externen Partnern.
3. Basis für Informationssysteme: EPKs dienen als konzeptionelle Grundlage für die Implementierung von Prozessen in Enterprise Resource Planning (ERP)- oder Workflow-Systemen. Sie sind in der Regel nicht direkt ausführbar, liefern aber die notwendige Prozesslogik für IT-Systeme.[8]

[7] Vgl. Thomas und Fellmann, *Semantische Ereignisgesteuerte Prozessketten*, S.1-3; Nüttgens und Rump, „Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK)“; Wikipedia, *Ereignisgesteuerte Prozesskette*.

[8] Vgl. Kocian, *Geschäftprozessmodellierung mit BPMN 2.0. Business Process Model and Notation im Methodenvergleich*, S.20; Rump, *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten*, S.55-60; Thomas und Fellmann, *Semantische Ereignisgesteuerte Prozessketten*, S.2-3.

Notationselemente der EPK

In der untenstehenden Tabelle sind die relevantesten Elemente der EPK-Notationselemente dargestellt. Eine vollständige Übersicht der EPK-Notationselemente ist in der Internetquelle GBTEC, *EPK-Notationselemente* zu finden.

Symbol	Erklärung
	**Aktivität**: Eine Aktivität, auch Funktion genannt, stellt eine aktive Tätigkeit innerhalb eines Geschäftsprozesses dar. Sie beschreibt einen Arbeitsschritt, der von einer Organisationseinheit ausgeführt wird und einen bestimmten Zustand verändert. Funktionen sind zentrale Elemente der EPK und werden typischerweise durch Ereignisse ausgelöst und abgeschlossen.
	**Ereignis**: Ein Ereignis beschreibt einen Zustand oder eine Bedingung im Prozessablauf, die entweder den Beginn oder das Ende einer Funktion kennzeichnet. Ereignisse selbst führen keine Aktivitäten aus, sondern dienen der Beschreibung von Zustandsänderungen innerhalb des Prozesses. Eine EPK beginnt und endet in der Regel mit einem Ereignis.
	**Operator UND**: Der UND-Operator (AND) ermöglicht die parallele Aufteilung oder Synchronisation von Prozesspfaden. Bei einer Verzweigung werden alle ausgehenden Pfade gleichzeitig aktiviert. Bei einer Zusammenführung wartet der Prozess, bis alle eingehenden Pfade abgeschlossen sind. Der UND-Operator dient der Modellierung paralleler Abläufe.
	**Operator ODER**: Der ODER-Operator (OR) erlaubt eine bedingte Verzweigung, bei der ein oder mehrere ausgehende Pfade gleichzeitig aktiviert werden können. Die Auswahl der Pfade erfolgt abhängig von definierten Bedingungen. Bei der Zusammenführung wird der Prozess fortgesetzt, sobald alle zuvor aktivierten Pfade abgeschlossen sind.

Symbol	Erklärung
	**Operator XOR**: Der XOR-Operator (exklusives Oder) steuert eine alternative Verzweigung oder Zusammenführung des Prozessflusses. Bei einer Verzweigung wird genau einer der möglichen Pfade aktiviert. Bei einer Zusammenführung wird der Prozess fortgesetzt, sobald genau ein eingehender Pfad erreicht wurde. XOR-Operatoren werden zur Modellierung von Entscheidungen eingesetzt.
	**Organisationseinheit**: Eine Organisationseinheit beschreibt die fachliche oder organisatorische Zuordnung einer Funktion innerhalb eines Geschäftsprozesses. Sie repräsentiert beispielsweise eine Abteilung, Rolle oder Person, die für die Durchführung einer Funktion verantwortlich ist. Organisationseinheiten dienen der Transparenz von Zuständigkeiten, beeinflussen jedoch nicht den Kontrollfluss des Prozesses.
	**Prozessschnittstelle**: Eine Prozessschnittstelle dient in der EPK dazu, die Modellkomplexität zu senken. Sie fungiert als Verknüpfungselement, um aus einem übergeordneten Prozess (z. B. Auftragsbearbeitung) auf ausgelagerte Unterprozesse (z. B. Rechnungslegung) zu verweisen. Deswegen besteht eine EPK nicht nur aus Start- und Endereignissen, sondern ein Prozessmodell kann anstelle eines abschließenden Endereignisses auch direkt mit einem Prozesswegweiser enden, um in eine andere Prozesskette überzuleiten.
	**Datenfluss**: Der Datenfluss stellt den Austausch von Informationen oder Daten zwischen Funktionen und weiteren Elementen dar. Er zeigt, welche Informationen von einer Funktion erzeugt, genutzt oder weitergegeben werden.

Tabelle 2: EPK-Notationselemente[9]

2.1.2 Softwaretests

Die nachfolgenden drei Testarten bilden unterschiedliche Ebenen im Softwareentwicklungsprozess, wobei jede Art eint, dass sie einzelne Projektergebnisse auf ihre Korrektheit prüfen. Sie wurden auch zu diesem Zwecke im Projekt verwendet.

2.1.2.1 Unit-Tests

Unittests dienen der Überprüfung einzelner Funktionen oder kleiner Softwarekomponenten. Dabei wird eine isolierte Codeeinheit, beispielsweise eine Methode oder Klasse, losgelöst von anderen Systemteilen getestet. Ziel ist es, sicherzustellen, dass die jeweilige Funktionalität korrekt implementiert ist und sich wie erwartet verhält. Durch die isolierte Betrachtung können dann Fehler frühzeitig erkannt und behoben werden, bevor sie sich auf andere Teile des Systems auswirken. Unit-Tests werden häufig automatisiert ausgeführt und stellen eine wichtige Grundlage für die Qualitätssicherung im Softwareentwicklungsprozess dar.[10]

2.1.2.2 Integrationstests

Im Gegensatz zu Unittests überprüfen Integrationstests das Zusammenspiel mehrerer Softwarekomponenten innerhalb eines Systems. Sobald einzelne Module durch Unittests validiert wurden, werden diese dann miteinander kombiniert und gemeinsam getestet. Hier liegt der Fokus darauf, sicherzustellen, dass die Schnittstellen zwischen den Komponenten korrekt funktionieren und Daten ordnungsgemäß ausgetauscht werden. Auf diese Weise können Fehler identifiziert werden, die erst beim Zusammenspiel mehrerer Module auftreten könnten.[11]

2.1.2.3 Systemtests

Systemtests überprüfen die Software als vollständig integriertes Gesamtsystem. Dabei wird überprüft, ob alle Komponenten gemeinsam korrekt funktionieren und die definierten Anforde-

[9]Symbole: ARIS Express.
Je nach Modellierungswerkzeug können die einzelnen EPK-Komponenten unterschiedlich aussehen.
Erklärungen: (Vgl. GBTEC, *EPK-Notationselemente*; HarzOptics, *Einige Grundregeln für die Modellierung ereignisgesteuerter Prozessketten*)

[10]Vgl. Parasoft, *Was ist Unit-Testing? Eine vollständige Anleitung.*

[11]Vgl. Parasoft, *Softwareintegrationstests.*

rungen erfüllt werden. Im Gegensatz zu Unit- und Integrationstests steht hier nicht mehr die einzelne Funktion, bzw. Schnittstelle, im Vordergrund, sondern das Verhalten der Anwendung aus Sicht des gesamten Systems. Systemtests simulieren häufig reale Nutzungsszenarien und dienen dazu, die Stabilität und allgemeine Funktionalität der Anwendung im praktischen Einsatz zu überprüfen. Systemtests können sowohl automatisiert, als auch manuell erfolgen. Im Projekt wurden diese manuell durchgeführt, indem die vollständig implementierte Anwendung überprüft wurde, damit man das Zusammenspiel zwischen den Komponenten und die gelieferten Ergebnisse kontrollieren kann.[12]

[12] Vgl. Parasoft, *Software System Testing*.

2.2 Verwendete Technologien

Dieser Abschnitt beschreibt die zentralen Technologien, die bei der Entwicklung der Anwendung eingesetzt wurden und die technische Grundlage des Systems bilden. Die verwendeten Technologien wurden nicht vom Auftraggeber vorgeschrieben und oblagen dem Projektteam. Eine Ausnahme bildet hierbei Javascript, wobei aufgrund der im Abschnitt 2.2.1 erwähnten Eigenschaften eine Erweiterung um Typescript erfolgte.

2.2.1 TypeScript

Abbildung 1: TypeScript-Logo[13]

TypeScript ist eine von Microsoft entwickelte Programmiersprache, die auf JavaScript basiert und diese um eine statische Typisierung erweitert. Durch die Verwendung von Typ-Definitionen kann man dann mögliche Programmierfehler bereits während der Entwicklung erkennen, bevor der Code ausgeführt wird. Der TypeScript-Code wird anschließend in JavaScript kompiliert, sodass man ihn in Webbrowsern ausführen kann. So bleibt TypeScript mit JavaScript-Laufzeitumgebungen kompatibel, aber man hat ebenfalls zusätzliche Tools zur Verfügung.

Im Rahmen dieses Projekts wird TypeScript eingesetzt, um die Wartbarkeit und Lesbarkeit des Quellcodes durch statische Typisierung zu verbessern und Fehler frühzeitig zu erkennen.[14]

2.2.2 Vite

Abbildung 2: Vite-Logo[15]

Vite ist ein Frontend-Build-Tool sowie ein Entwicklungsserver für Webanwendungen. Es wurde von Evan You entwickelt und dient als Entwicklungswerkzeug für moderne Webanwendungen mit Fokus auf kurze Startzeiten und schnelle Aktualisierung während der Entwicklung.

[14] Vgl. Microsoft, *What is TypeScript?*; React, *Using TypeScript in React*.

Vite stellt einen lokalen Entwicklungsserver bereit, der Quellcode direkt über native ES-Module im Browser ausliefert. Dadurch können Anwendungen ohne aufwendiges Vorab-Bundling gestartet werden. Außerdem sind Änderungen am Code mithilfe eines Hot-Module-Replacements unmittelbar in der Weboberfläche sichtbar, ohne dass man die gesamte Anwendung neu laden muss. Wenn man Produktionsversionen erstellen will, bündelt Vite den Anwendungscode und erzeugt optimierte statische Dateien.

Im Projekt kommt Vite zum Einsatz, um die Entwicklungsumgebung für das React-Frontend bereitzustellen und um den TypeScript-Code für die Ausführung im Browser zu verarbeiten.[16]

2.2.3 SQLite

SQLite ist ein relationales Datenbankmanagementsystem. Es wurde von Dr. Richard Hipp im Jahr 2000 entwickelt. Es ist eine abgeschlossene, serverlose Plattform für eingebettete Systeme, bei der keine separate Server-Infrastruktur erforderlich ist. Die Architektur von SQLite wird in 4 Teile unterteilt.

[16] Vgl. Vite, *The Build Tool for the Web*; Vite, *Vite Next Generation Frontend Tooling*.

- **Core:** In diesem Teil verarbeitet die SQLite-Engine den SQL-Text und wandelt diesen, unter Verwendung des SQL-Compiler, in Bytecode um und führt diesen in der virtuellen Maschine aus.
- **SQL-Compiler:** Dieser Teil ist für die Umwandlung des SQL-Text in Bytecode zuständig. Zuerst wird dabei der SQL-Text tokenisiert, das bedeutet, dass dieser in Stücke unterteilt wird. Jedes dieser Tokens wird dann an den Parser weitergeleitet. Dieser kreiert einen AST (abstrakter Syntaxbaum) anhand der Eingabe. Der Code Generator analysiert den AST (abstrakter Syntaxbaum) und generiert anhand davon einen Bytecode, der die Logik der SQL-Anweisung durchführen kann.
- **Backend:** Übernimmt die Speicherung und Verwaltung der Datenbank in einer Datei. Zuerst bildet es für jede Tabelle einen B-Baum. Dann konvertieren der Pager und das OS-Interface diese in eine `.sqlite` Datei.
- **Zusatzfunktionen:** Beinhaltet Zusatzfunktion die SQLite implementiert.

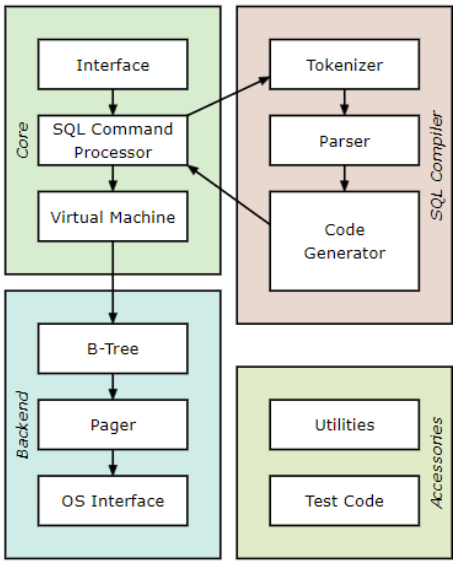


Abbildung 3: SQLite-Architektur

[17,18]

In unserem Projekt verwenden wir SQLite für die Speicherung der Daten im Backend (siehe: 3.5.3). Sie ist besonders geeignet, da SQLite die Tabellen mit B-Bäume in eine lokale Datei schreibt und somit keine separate Server-Infrastruktur braucht. Darüber hinaus ist keine zusätzliche Konfiguration erforderlich, um die Datenbank lauffähig zu machen.

2.2.4 REST-API

Eine REST-API (Representational State Transfer Application Programming Interface) ist eine Schnittstelle, über die verschiedene Softwarekomponenten über das HTTP-Protokoll miteinander

[18]Vgl. SQLite Development Team, *Architecture of SQLite*
[19]Vgl. SQLite Development Team, *How SQLite Works*

kommunizieren können. Dabei werden Daten zwischen einem Client und einem Server über standardisierte Webanfragen ausgetauscht.

Bei einer REST-basierten Architektur stellt der Server sogenannte Ressourcen bereit, auf die ein Client über eine eindeutige Adresse, den sogenannten URI (Uniform Resource Identifier), zugreifen kann. Der Client sendet eine HTTP-Anfrage an den Server, welcher diese verarbeitet und eine entsprechende Antwort zurückgibt. Die übertragenen Daten können dabei in unterschiedlichen Formaten vorliegen, beispielsweise als JSON, XML oder einfacher Text.[20]

Ein wesentliches Merkmal von REST ist die zustandslose Kommunikation zwischen Client und Server. Das bedeutet, dass jede Anfrage alle notwendigen Informationen enthalten muss und der Server keinen Zustand zwischen mehreren Anfragen speichert. Dadurch können Systeme einfacher skaliert werden und bleiben unabhängig voneinander.

Weitere Eigenschaften einer REST-Schnittstelle sind eine einheitliche Schnittstelle zwischen Systemkomponenten sowie eine mehrschichtige Architektur, bei der mehrere Dienste miteinander kombiniert werden können. Dadurch lassen sich Anwendungen modular aufbauen und leichter erweitern.

Im Rahmen dieses Projekts wurde eine eigene REST-API implementiert, um die Kommunikation zwischen Frontend und Backend des Prototyps zu ermöglichen. Diese Schnittstelle dient ausschließlich der internen Kommunikation innerhalb der entwickelten Anwendung und ist nicht Teil der bestehenden Infrastruktur des Auftraggebers.

Das Frontend sendet dabei HTTP-Anfragen an das Backend (siehe Kapitel 4.6), um BPMN-Diagramme zu speichern, abzurufen oder zu aktualisieren. Das Backend verarbeitet diese Anfragen und speichert die entsprechenden Daten in einer lokal verwendeten Datenbank.

Da im Rahmen des Projekts kein direkter Zugriff auf die produktiven Datenbanken des Auftraggebers (BMD) möglich war, wurde diese REST-Schnittstelle als technische Lösung implementiert, um die Speicherung und Verarbeitung der Diagrammdaten im Prototyp zu ermöglichen. Dadurch konnte die Funktionalität der Anwendung unabhängig von der bestehenden Systemlandschaft entwickelt und getestet werden.

Die grundlegende Architektur der Anwendung folgt dabei dem Prinzip:

$$\text{Frontend} \rightarrow \text{REST-API} \rightarrow \text{Backend} \rightarrow \text{Datenbank}$$

[20] Vgl. Red Hat, *What is a REST API?*

Der Benutzer arbeitet im Frontend der Anwendung. Aktionen wie das Erstellen oder Bearbeiten eines BPMN-Diagramms lösen HTTP-Anfragen an das Backend aus. Dieses verarbeitet die Anfrage und speichert die entsprechenden Daten in der Datenbank oder liefert gespeicherte Daten zurück an das Frontend.

Durch die Verwendung einer REST-API wird eine klare Trennung zwischen Benutzeroberfläche und Datenverarbeitung erreicht. Dadurch können Frontend und Backend unabhängig voneinander entwickelt und erweitert werden.

2.2.5 Hibernate

Hibernate ist ein Framework zur objektrelationalen Abbildung (Object-Relational Mapping, ORM) in Java. Es ermöglicht die Abbildung von Java-Objekten auf relationale Datenbanktabellen. Dadurch wird die direkte Arbeit mit SQL-Abfragen reduziert, da Datenbankoperationen über Java-Objekte durchgeführt werden können. Entwickler können somit mit Klassen und Objekten arbeiten, während Hibernate die Kommunikation mit der Datenbank übernimmt.[a]

Abbildung 4: Logo des Hibernate Frameworks

Das Framework stellt verschiedene zentrale Komponenten bereit, darunter die Klassen `SessionFactory`, `Session` und `StatelessSession`. Zusätzlich implementiert Hibernate die *Jakarta Persistence API (JPA)* und stellt damit auch die standardisierten Schnittstellen `EntityManagerFactory` und `EntityManager` zur Verfügung, welche für den Zugriff auf die Datenbank verwendet werden.

Ein zentraler Bestandteil von Hibernate ist das sogenannte *Mapping* zwischen Java-Klassen und Datenbanktabellen. Dabei werden Klassen als sogenannte *Entities* definiert, deren Attribute automatisch den Spalten einer Tabelle zugeordnet werden.

[a] Vgl. Hibernate Team, *Hibernate ORM Documentation*.

Dieses Mapping erfolgt üblicherweise über Annotationen direkt im Quellcode. Dadurch können typische Datenbankoperationen wie *Create*, *Read*, *Update* und *Delete* (CRUD) direkt über

Java-Objekte ausgeführt werden, ohne dass dafür manuell SQL-Abfragen geschrieben werden müssen.

Im Projekt wird Hibernate in Kombination mit JPA verwendet, um den Zugriff auf die Datenbank zu vereinfachen. Die Konfiguration erfolgt über eine Datei `persistence.xml`, in der unter anderem der verwendete JDBC-Treiber sowie die Verbindungsparameter zur Datenbank definiert sind.

Ein Beispiel für eine solche Konfiguration ist in Listing 1 dargestellt.

Listing 1: Beispiel einer Datenbankkonfiguration in der persistence.xml

```
1  <property name="jakarta.persistence.jdbc.driver" value="org.h2.Driver" />
2  <property name="jakarta.persistence.jdbc.url"
3            value="jdbc:h2:file:./data/bpmn;AUTO_SERVER=TRUE;DB_CLOSE_DELAY=-1" />
4  <property name="jakarta.persistence.jdbc.user" value="sa" />
5  <property name="jakarta.persistence.jdbc.password" value="" />
```

Durch diese Konfiguration kann Hibernate eine Verbindung zur Datenbank herstellen und die definierten Entities automatisch verwalten. Der Einsatz von Hibernate erleichtert insbesondere die Entwicklung datenbankbasierter Anwendungen, da sich Entwickler stärker auf die Geschäftslogik konzentrieren können, während die Verwaltung der Datenbankzugriffe vom Framework übernommen wird.

2.2.6 Open Liberty

Open Liberty ist ein leichtgewichtiger, Open-Source-basierter Application Server für die Entwicklung und den Betrieb von Java-Anwendungen. Er eignet sich besonders für die Bereitstellung von Microservices und REST-basierten Webanwendungen. Der Server stellt eine Laufzeitumgebung zur Verfügung, in der Java-Anwendungen ausgeführt und über HTTP-Schnittstellen bereitgestellt werden können.[a]
Ein wesentliches Merkmal von Open Liberty ist die modulare Architektur. Funktionen werden über sogenannte *Features* aktiviert, wodurch nur die tatsächlich benötigten Komponenten geladen werden.
Ein weiterer Vorteil ist die sogenannte *Zero-Migration Architecture*. Diese Architektur ermöglicht es, neue Versionen der Laufzeitumgebung zu verwenden, ohne dass bestehende Konfigurationsdateien oder Anwendungen angepasst werden müssen.
Im Rahmen dieses Projekts wird Open Liberty als Laufzeitumgebung für das Backend eingesetzt. Der Server hostet die implementierten REST-Schnittstellen und stellt diese über HTTP zur Verfügung.

Abbildung 5: Logo des Open Liberty Application Servers

[a] Vgl. Open Liberty, *Open Liberty Documentation*.

Das Frontend kann über diese Schnittstellen BPMN-Daten speichern, laden und bearbeiten. Aufgrund seiner geringen Komplexität und der guten Unterstützung moderner Java-Webtechnologien eignet sich Open Liberty besonders gut für den prototypischen Einsatz im Rahmen dieses Projekts.

2.2.7 JPA

Die *Java Persistence API* (JPA) ist eine Spezifikation für den Zugriff auf relationale Datenbanken in Java-Anwendungen. Sie definiert eine standardisierte Schnittstelle zur Speicherung und Verwaltung von Datenbankobjekten in Form von sogenannten Entities. Durch die Verwendung

von JPA können Entwickler mit Java-Objekten arbeiten, während die Abbildung dieser Objekte auf Datenbanktabellen automatisch erfolgt.[21]

Ein zentrales Element von JPA ist der *EntityManager*. Dieser stellt eine Schnittstelle zur Interaktion mit der Datenbank dar und ermöglicht Operationen wie das Speichern, Aktualisieren oder Löschen von Entity-Objekten. Der EntityManager wird üblicherweise von einer *EntityManagerFactory* erzeugt, welche mehrere EntityManager-Instanzen bereitstellen kann, die auf dieselbe Datenbank zugreifen.

JPA verwendet außerdem den sogenannten *Persistence Context*. Dabei handelt es sich um eine Menge von verwalteten Entity-Instanzen, deren Zustand automatisch mit der Datenbank synchronisiert wird. Innerhalb dieses Kontexts existiert für jede Entity-Identität genau eine Objektinstanz. Der Persistence Context kann entweder auf eine einzelne Transaktion beschränkt sein oder über mehrere Operationen hinweg bestehen.

Eine *Persistence Unit* definiert die Menge der Entity-Klassen, die von einem EntityManager verwaltet werden können. Die Konfiguration dieser Persistence Unit erfolgt üblicherweise in der Datei `persistence.xml`. In dieser Datei werden unter anderem die Datenbankverbindung sowie weitere Konfigurationsparameter festgelegt.

Im Rahmen dieses Projekts wird JPA zur Verwaltung der Datenbankzugriffe im Backend eingesetzt. Die BPMN-Daten werden dabei in Form von Entity-Objekten gespeichert und über Repository-Klassen verarbeitet. Die Verbindung zur Datenbank wird über die Konfigurationsdatei `persistence.xml` hergestellt, in der die notwendigen Verbindungsparameter definiert sind.

2.2.8 Headless Browser

Unter dem Begriff Headless Browser versteht man einen Browser ohne grafischer Oberfläche. In diesem Projekt wird diese Technologie für die Simulation einer Browserumgebung genutzt. Diese wird benötigt, weil man Screenshots von einer dynamisch generierten Oberfläche zur Laufzeit aufnehmen muss, während man keine Browserumgebung zur Verfügung hat.

[21] Vgl. Red Hat, *Java Persistence API (JPA) Documentation*.

2.2.9 Git

Git ist ein verteiltes Versionskontrollsystem, das zur Nachverfolgung von Änderungen im Quellcode verwendet wird. Es wurde ursprünglich von Linus Torvalds ins Leben gerufen, um die Zusammenarbeit mehrerer Entwickler an einem Projekt zu erleichtern und gleichzeitig die vollständige Historie aller Änderungen zu speichern.

Abbildung 6: Git-Logo[22]

In Git wird der Zustand eines Projekts zu bestimmten Zeitpunkten in sogenannten Commits gespeichert. Diese bilden eine Historie, anhand derer Änderungen nachvollzogen, verglichen oder bei Bedarf auch rückgängig gemacht werden können.[23]

GitHub ist eine webbasierte Plattform zur Verwaltung von Git-Repositories. Sie ermöglicht es Entwicklern, Projekte zentral zu speichern und gemeinsam daran zu arbeiten. Darüber hinaus stellt GitHub Funktionen zur Zusammenarbeit bereit, zum Beispiel Pull Requests, Issue Tracking oder Code-Reviews.[25]

Abbildung 7: GitHub-Logo[24]

Im Projekt wird Git zur Versionsverwaltung des gesamten Quellcodes verwendet. Dadurch können Änderungen am Code nachvollzogen und verfolgt werden und mehrere Entwickler können parallel an unterschiedlichen Teilen der Anwendung arbeiten, ohne dass sie sich gegenseitig überschreiben. Wir verwenden GitHub dabei als zentrale Plattform zur Verwaltung des gemeinsamen Repositorys.

2.2.10 npm

npm ist ein Paketmanager für die JavaScript-Laufzeitumgebung Node.js. Er dient dazu, externe Bibliotheken und Module zu installieren, zu verwalten und in eigene Projekte zu integrieren. Zusätzliche stellt npm

Abbildung 8: npm-Logo[26]

[23] Vgl. Chacon und Straub, *Pro Git Book*.
[25] Vgl. ebd.

ein Online-Repository, die sogenannte npm-Registry, bereit, in dem man Pakete veröffentlichen und für andere Pakete verfügbar machen kann.

Der Paketmanager wird normalerweise über das Terminal verwendet und ermöglicht es, Abhängigkeiten eines Projekts automatisch zu installieren und zu aktualisieren. Die benötigten Pakete werden dabei typischerweise in einer `package.json` definiert, welche die Abhängigkeiten und Metadaten eines Projekts beschreibt.[27]

Im Projekt wird npm verwendet, um externe JavaScript-Bibliotheken zu verwalten und zu installieren, die für die Entwicklung des Frontends benötigt werden (siehe 2.4).

[27] npm, Inc., *About npm*.

2.3 Verwendete Entwicklungssysteme

Für die Entwicklung der Anwendung wurden verschiedene Entwicklungsumgebungen verwendet. Diese unterstützen während des Entwickelns durch Debugging-Werkzeuge, Code-Editoren, sowie Erweiterungen zur Verbesserung der Produktivität. Die Wahl der Entwicklungssysteme fiel auf das Projektteam und wurde vom Auftraggeber nicht vorgegeben.

2.3.1 Visual Studio Code

Visual Studio Code (VS Code) ist eine Open-Source-Programmierumgebung, die von Microsoft entwickelt worden ist. Sie bietet im Gegensatz zu traditionellen Entwicklungsumgebungen keine integrierten sprachspezifischen Debugging-Tools und ein wesentliches Merkmal von VS Code ist die Erweiterbarkeit über den Visual Studio Marketplace. Dadurch können Erweiterungen installiert werden, die zusätzliche Funktionen wie Unterstützung für bestimmte Programmiersprachen oder Entwicklungsframeworks hinzufügen.[29]

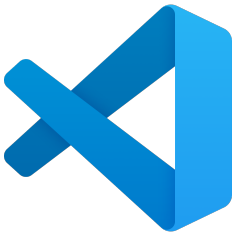


Abbildung 9: VS Code-Logo[28]

Im Projekt wird Visual Studio Code hauptsächlich für die Entwicklung des Frontends verwendet.

2.3.2 IntelliJ Community

IntelliJ IDEA Community Edition ist eine von JetBrains entwickelte integrierte Entwicklungsumgebung für Java-basierte Anwendungen. Die IDE bietet Funktionen wie Syntaxhervorhebung, integrierte Debugging-Werkzeuge, Werkzeuge zur Projektverwaltung, Versionskontrolle und Build-Systeme wie Maven oder Gradle.[31]

Abbildung 10: Intellij-Logo[30]

[29] Vgl. Microsoft, *Der Open-Source-KI-Code-Editor*.
[31] Vgl. ebd.

Hier wurde Intellij zur Entwicklung des Java-Backends verwendet.

2.3.3 MySQL Workbench

MySQL Workbench ist ein grafisches Werkzeug zur Entwicklung und Verwaltung von MySQL-Datenbanken. Es stellt Funktionen zur Datenbankmodellierung, SQL-Entwicklung sowie zur Administration von Datenbanksystemen bereit. Die Anwendung ermöglicht es, Datenbankschemas visuell zu entwerfen und diese anschließend automatisch in SQL-Strukturen zu überführen. Dadurch kann die Struktur einer Datenbank bereits in einer frühen Phase der Entwicklung geplant und überprüft werden.

Abbildung 11: MySQL Workbench-Logo[32]

Neben der Modellierung hat MySQL Workbench auch Werkzeuge zum Schreiben und Testen von SQL-Abfragen sowie Funktionen zur Verwaltung von Serververbindungen und Datenbankbenutzern. Außerdem kann man bestehende Datenbanken analysiert und als visuelle Modelle dargestellen.[33]

Im Projekt wurde MySQL Workbench zur Modellierung eines Entity Relationship Diagram für die Datenbankstruktur verwendet (siehe Abschnitt 3.5.3.2).

[33]Vgl. Oracle, *Enhanced Data Migration*.

2.4 Verwendete Bibliotheken

Um den Entwicklungs-Prozess zu vereinfachen, werden gezielt Bibliotheken eingesetzt. Anstatt das Rad neu zu erfinden, kann man bestehende, ausführlich getestete und gepflegte Funktionen übernehmen, um die Entwicklungszeit zu reduzieren und die Zuverlässigkeit zu erhöhen. In diesem Abschnitt werden ausschließlich jene Bibliotheken beschrieben, die für die Kernfunktionalität der Anwendung wesentlich sind und eine vollständige Auflistung aller Abhängigkeiten wurde bewusst nicht aufgenommen.

2.4.1 React

React ist eine Open-Source-JavaScript-Bibliothek zur Entwicklung von Benutzeroberflächen für Webanwendungen. React basiert auf einer komponentenorientierten Architektur, das heißt, man zerlegt komplexe Benutzeroberflächen in kleinere, voneinander unabhängige Komponenten, was die Wiederverwendbarkeit und Wartbarkeit erhöht.

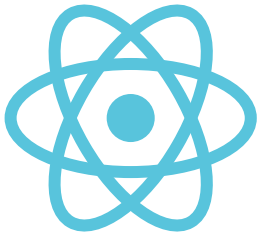


Abbildung 12: React-Logo[34]

Wir benutzen React zur Implementierung des Frontends. Zur Navigation zwischen verschiedenen Ansichten der Anwendung wird zusätzlich die Bibliothek React Router eingesetzt. Diese ermöglicht ein clientseitiges Routing innerhalb der Anwendung, sodass unterschiedliche Komponenten abhängig von der URL dargestellt werden können, ohne dass die gesamte Seite neu geladen werden muss.[35]

2.4.2 bpmn-js

Für die Umsetzung der grafischen Anforderungen des Projekts wird die Bibliothek `bpmn-js` eingesetzt. Dabei handelt es sich um ein in JavaScript entwickeltes Open-Source-Toolkit, das speziell für die Darstellung und Interaktion mit BPMN 2.0-Diagrammen in Webanwendungen

[35] Vgl. React, *The library for web and native user interfaces*; React, *Was ist React.js? Ein Blick auf die beliebte JavaScript-Bibliothek*; React, *Getting started with react router*.

konzipiert wurde. Die Bibliothek wird vom Projekt bpmn.io (siehe 2.5.3) entwickelt und hat sich als weit verbreitete Lösung etabliert, wenn es um die Integration von BPMN-Funktionalitäten in Webanwendungen geht.

`bpmn-js` ermöglicht sowohl die Darstellung bestehender BPMN-Diagramme, als auch deren interaktive Bearbeitung direkt im Browser. Dabei werden BPMN-Elemente auf einer SVG-basierten Zeichenfläche dargestellt und können durch den Benutzer verschoben, gelöscht oder hinzugefügt werden. Die Bibliothek baut unter anderem auf den Komponenten `diagram-js` und `bpmn-moddle` auf. Während `diagram-js` für die grafische Darstellung und Benutzerinteraktionen verantwortlich ist, übernimmt `bpmn-moddle` die Verarbeitung der zugrundeliegenden BPMN-Datenstrukturen. (siehe 2.4.3)

Durch diese modulare Architektur kann `bpmn-js` sowohl als reiner Viewer zur Anzeige von Diagrammen, als auch als Modeler zur Erstellung und Bearbeitung von BPMN-Prozessen dienen. Diese Bibliothek bildet die Grundlage für die Visualisierung und Bearbeitung der BPMN-Diagramme (siehe 4.2) und die Nutzung dieser wurde vom Auftraggeber vorgegeben.[36]

2.4.3 bpmn-moddle

Die Bibliothek `bpmn-moddle` stellt die Datenmodell-Ebene der BPMN-Verarbeitung bereit. Sie wird innerhalb des `bpmn-js`-Ökosystems verwendet, um BPMN-Diagramme aus XML-Dateien einzulesen, zu verarbeiten und wieder zu serialisieren. Dabei implementiert `bpmn-moddle` das BPMN 2.0-Metamodell der OMG und stellt sicher, dass die erzeugten oder bearbeiteten Diagramme der offiziellen BPMN-Spezifikation entsprechen. Dadurch können BPMN-Diagramme standardkonform gespeichert und mit anderen Prozessmanagementsystemen ausgetauscht werden.

Im Projekt wird `bpmn-moddle` genutzt, um BPMN-Datenstrukturen zu interpretieren und zu manipulieren, während die grafische Darstellung durch `bpmn-js` erfolgt. (siehe 2.4.2)[37]

[36] Vgl. bpmnio, *bpmn-js - BPMN 2.0 for the web*; bpmnio, *Understanding bpmn-js Internals*.
[37] Vgl. bpmnio, *bpmn-moddle*.

2.4.4 fast-xml-parser

Die Bibliothek `fast-xml-parser` dient der Verarbeitung von XML-Daten innerhalb von JavaScript- und TypeScript-Anwendungen. Sie ermöglicht das effiziente Parsen von XML-Daten in eine Datenstruktur, und umgekehrt.

Im Projekt wird `fast-xml-parser` verwendet, um XML-basierte Daten aus externen Schichten einzulesen und in eine Form zu überführen, die innerhalb der Anwendung weiterverarbeitet werden kann.[38]

2.4.5 vitest

`Vitest` ist ein modernes Testframework für JavaScript- und TypeScript-Anwendungen. Es wurde für Entwicklungsumgebungen entwickelt, die auf `Vite` basieren und ermöglicht die schnelle Ausführung automatisierter Tests innerhalb des Projekts. Das Framework unterstützt unter anderem Unit- und Integrationstests, sowie Funktionen wie Mocking oder Snapshot-Tests.

Abbildung 13: Vitest-Logo[39]

Im Rahmen des Projekts wird `Vitest` eingesetzt, um zentrale Funktionen der Anwendung automatisiert zu überprüfen und mögliche Fehler frühzeitig im Entwicklungsprozess zu erkennen.[40]

2.4.6 axios

`Axios` ist eine JavaScript-Bibliothek zur Durchführung von HTTP-Anfragen zwischen Client und Server. Es handelt sich um einen Promise-basierten HTTP-Client und die Bibliothek ermöglicht das Senden und Empfangen von Daten über gängige HTTP-Methoden und stellt dabei eine einfache Programmierschnittstelle bereit, mit der man asynchrone Anfragen ausführen und deren Antworten verarbeiten kann.

Abbildung 14: Axios-Logo[41]

[38] Vgl. Natural Intelligence, *fast-xml-parser*.
[40] Vgl. Vitest, *Get Started*; Vitest, *Next generation testing framework powered by Vite.*

Im Projekt wird `Axios` verwendet, um die Kommunikation zwischen dem React-Frontend und der BMD-REST-Schnittstelle umzusetzen. Darüber werden Authentifizierungsprozesse durchgeführt, sowie projektspezifische Daten vom Backend abgerufen, die anschließend zur Verarbeitung und Visualisierung innerhalb der Anwendung verwendet werden.[42]

2.4.7 puppeteer

`Puppeteer` ist eine JavaScript-Bibliothek, die es erlaubt, über das DevTools-Protokoll den Chrome Browser zu steuern. Sie unterstützt dabei die Verwendung von Headless Browsern (siehe Kapitel 2.2.8) und erlaubt zusätzliche Funktionen wie die Erstellung von Screenshots. Diese Funktion ist für dieses Projekt wichtig, um aus dynamisch generiertem HTML einen Screenshot zu exportieren, damit man das Bild als `Base64-String` bereitstellen kann.

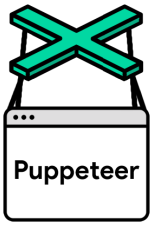


Abbildung 15: Puppeteer-Logo

[42] Vgl. Axios, *Promise based HTTP client for the browser and node.js.*

2.5 Sonstige verwendete Software

in diesem Abschnitt werden zusätzliche Softwarewerkzeuge beschrieben, die während der Entwicklung und Dokumentation des Projekts verwendet wurden.

2.5.1 Figma

Figma ist ein webbasiertes Designwerkzeug, das zur Erstellung und Bearbeitung von Entwürfen wie Webseiten oder Anwendungen verwendet wird. Die Software ermöglicht es, Designs zu kreieren, zu teilen und gemeinsam zu bearbeiten, sodass mehrere Personen gleichzeitig am selben Projekt arbeiten können.[44]

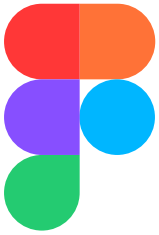


Abbildung 16: Figma-Logo[43]

Im Projekt wurde Figma für die Erstellung des Mockups verwendet, welches die Benutzeroberfläche darstellt (siehe 4.1).

2.5.2 Overleaf

Overleaf ist ein LaTeX-Editor, der speziell für die Erstellung wissenschaftlicher Dokumente mit LaTeX entwickelt wurde. Es handelt sich dabei um einen Online-Editor, der es ermöglicht, LaTeX-Dokumente direkt im Browser zu erstellen und zu bearbeiten, ohne zusätzliche Software installieren zu müssen. Darüber hinaus unterstützt Overleaf die gleichzeitige Bearbeitung von Dokumenten in Echtzeit sowie diverse Funktionen zum Teilen und Verwalten von Projekten.[46]

Abbildung 17: Overleaf-Logo[45]

LaTeX ist ein Dokumenterstellungssystem, das häufig für wissenschaftliche Arbeiten verwendet wird. Im Gegensatz zu klassischen Texteditoren wird ein Dokument dabei über Quelltext

[44] Vgl. Figma, *What is Figma?*
[46] Vgl. ebd.

beschrieben, in dem Struktur- und Formatierungsbefehle definiert werden. Durch eine sogenannte TEX-Engine wird der Quelltext verarbeitet und es entsteht ein formatiertes PDF.[47]

Diese Arbeit wurde mithilfe von LATEX in der Online-Umgebung Overleaf verfasst. Dabei wurde die Arbeit in mehrere `.tex`-Dateien aufgeteilt, die jeweils einzelnen Kapiteln oder Abschnitten entsprechen und in einer zentralen Hauptdatei eingebunden werden. Diese Struktur erleichtert die Organisation des Dokuments und verbessert die Übersichtlichkeit im Vergleich zu einem einzelnen großen Dokument.

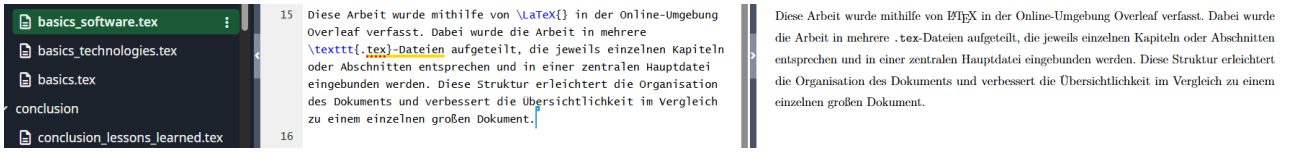


Abbildung 18: Overleaf-Arbeitsumgebung mit Projektstruktur, LATEX-Quelltext und kompiliertem PDF-Dokument

2.5.3 BPMN.io

BPMN.io ist ein Open-Source-Projekt, das webbasierte Werkzeuge zur Modellierung und Darstellung von Geschäftsprozessen bereitstellt. Die Plattform stellt verschiedene Toolkits zur Verfügung, mit denen sich Modellierungsfunktionen direkt in Webanwendungen integrieren lassen.

BPMN.iO

Abbildung 19: BPMN.io-Logo[48]

Zu diesen Werkzeugen gehören unter anderem Bibliotheken zur Erstellung, Darstellung und Bearbeitung von BPMN-Diagrammen im Browser. Dadurch können Prozessmodelle grafisch dargestellt und innerhalb einer Anwendung interaktiv bearbeitet werden.[49]

Die Entwicklung des Projekts erfolgt öffentlich über GitHub, wo mehrere Werkzeuge und Bibliotheken für die Arbeit mit BPMN bereitgestellt und weiterentwickelt werden.[50]

[47] Vgl. Overleaf, *Learn LaTeX in 30 minutes*.
[49] Vgl. ebd.
[50] Vgl. bpmn-io, *bpmn-io Organization on GitHub*.

Im Projekt bildet BPMN.io die Grundlage für die eingesetzten BPMN-Werkzeuge, insbesondere für die Bibliothek `bpmn-js`, die zur Visualisierung und Bearbeitung der BPMN-Diagramme innerhalb der Anwendung verwendet wird (siehe 2.4.2).

2.5.4 Postman

Postman ist ein Werkzeug zur Entwicklung, zum Testen und zur Dokumentation von Programmierschnittstellen (APIs). Es ermöglicht das Senden von HTTP-Requests an Webservices sowie die Analyse der dazugehörigen Antworten. Dabei können insbesondere REST-Schnittstellen mit Methoden wie `GET`, `POST` oder `PUT` einfach überprüft werden. Auch die Übertragung von Daten im JSON-Format sowie das Testen von Headern, Parametern und Antwortcodes wird unterstützt.[52]

Abbildung 20: Postman-Logo[51]

Ein zentrales Konzept von Postman sind sogenannte *Collections*. Dabei handelt es sich um Sammlungen von gespeicherten Requests, die strukturiert organisiert und mehrfach verwendet werden können. Dadurch eignet sich Postman sowohl für manuelle Tests als auch für wiederkehrende Prüfungen mehrerer Endpunkte.[53]

Zusätzlich bietet Postman die Möglichkeit, Pre-Request- und Post-Response-Skripte zu verwenden. Damit können beispielsweise Testfälle formuliert, dynamische Parameter gesetzt oder Antworten automatisiert überprüft werden.[54]

Im Rahmen dieses Projekts wurde Postman verwendet, um die implementierten REST-Schnittstellen des Backends unabhängig vom Frontend zu testen. Dadurch konnten einzelne Endpunkte gezielt aufgerufen und die vom Server zurückgegebenen Daten direkt kontrolliert werden. Dies war insbesondere während der Implementierung und Fehlersuche hilfreich, da Eingabedaten schrittweise angepasst und die Auswirkungen unmittelbar überprüft werden konnten.

Abbildung 21 zeigt beispielhaft einen in Postman vorbereiteten `POST`-Request auf den Endpoint `/rest/`. Im linken Bereich ist die zugehörige Collection mit mehreren Testanfragen sichtbar.

[52] Vgl. Postman, Inc., *Postman Learning Center*.
[53] Vgl. Postman, Inc., *Using Postman Collections*.
[54] Vgl. Postman, Inc., *Writing Tests and Scripts in Postman*.

Im Hauptbereich werden die gewählte HTTP-Methode, die Zieladresse des lokalen Servers sowie der im JSON-Format definierte Request-Body dargestellt. Auf diese Weise konnten neue `BpmnNode`-Objekte direkt an das Backend übermittelt und dessen Verarbeitung überprüft werden.

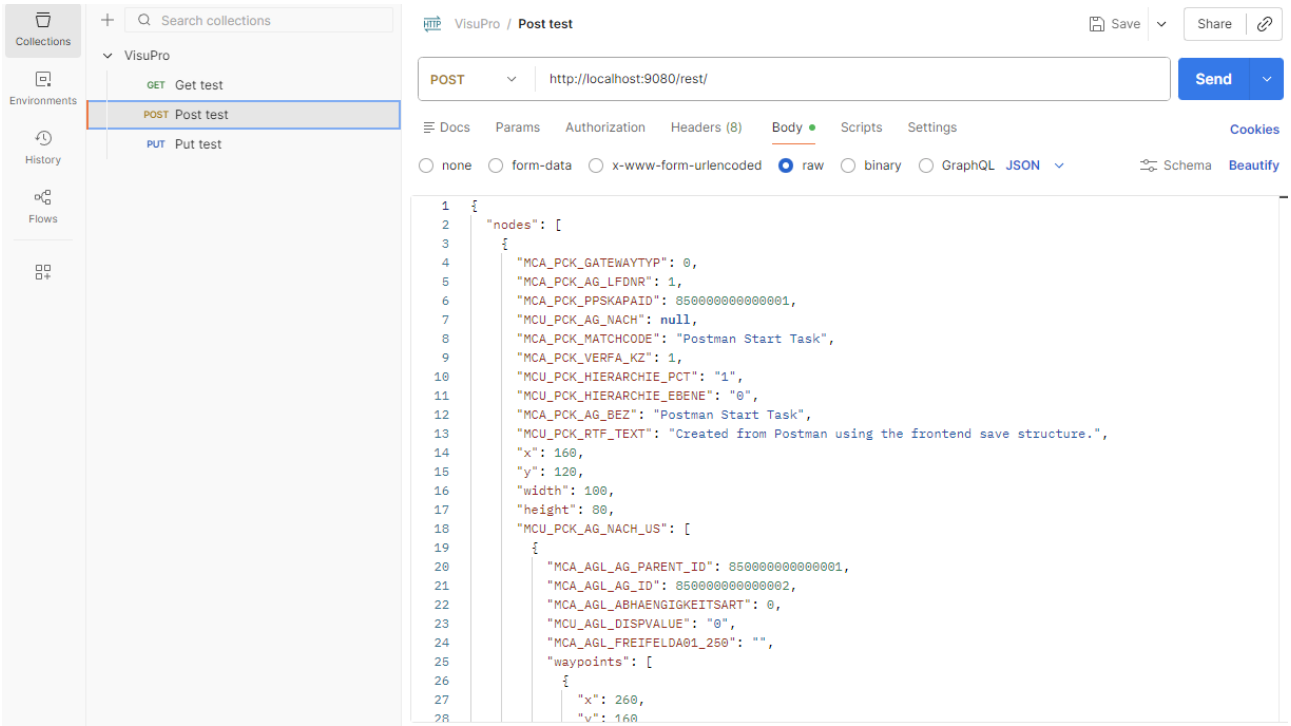


Abbildung 21: Test eines POST-Requests auf den REST-Endpoint mit Postman

Der gezeigte Request enthält exemplarische Felder eines BPMN-Knotens, darunter Bezeichnungen sowie Positions- und Größenangaben. Durch das manuelle Absenden solcher Anfragen konnten die Schnittstellen des Backends schrittweise validiert werden, bevor deren Nutzung über das Frontend erfolgte. Postman diente damit als praktisches Hilfsmittel zur Verifikation der korrekten Kommunikation zwischen Client und Server.

3 Planung und Realisierung

Dieses Kapitel beschreibt die Planung und organisatorische Struktur des Projekts sowie die konzeptionellen Grundlagen der Umsetzung. Dabei werden zunächst die Projektorganisation, Kommunikationswege, wie auch die Meilensteine dargestellt. Anschließend werden die Risiken, die zentralen Anforderungen und Use Cases beschrieben. Abschließend wird der technische Entwurf des Systems inklusive der Architektur, dem Datenmodell sowie der geplanten Maßnahmen zur Qualitätssicherung vorgestellt.

3.1 Projektorganisation

In diesem Abschnitt wird beschrieben, wie das Projekt organisatorisch aufgebaut war. Dazu gehören der Auftraggeber, die Ausgangssituation, das Projektteam sowie die Kommunikations- und Abstimmungswege während der Umsetzung.

3.1.1 Auftraggeber

Der Auftraggeber des Projekts ist die BMD SYSTEMHAUS GesmbH mit Sitz in Steyr. Das Unternehmen entwickelt betriebswirtschaftliche Softwarelösungen für unterschiedliche Branchen.

3.1.2 Ausgangssituation

Die bestehenden Prozessabläufe werden derzeit in Form von Listen gespeichert. Diese Art der Darstellung ist für Anwender oft unübersichtlich und erschwert vor allem die Analyse, das Verständnis und die Weiterentwicklung von Geschäftsprozessen.

Besonders für Schulungen und zur Einarbeitung neuer Mitarbeiter fehlt eine übersichtliche grafische Darstellung der Abläufe. Daraus ergab sich die Anforderung, die vorhandenen Prozess-

daten automatisch in BPMN-Diagramme zu überführen und diese zusätzlich in einem Editor bearbeitbar zu machen.

Bereits im ersten Meeting mit dem Auftraggeber wurde außerdem festgelegt, dass die BPMN-Oberfläche den gesamten verfügbaren Bildschirmbereich einnimmt. Dadurch wird eine möglichst einfache Integration in das bestehende NTCS-System ermöglicht.

3.1.3 Team

Das Projekt wurde im Rahmen der Diplomarbeit von einem dreiköpfigen Team umgesetzt. Die Aufgaben wurden entsprechend der jeweiligen Schwerpunkte im Team aufgeteilt, damit die einzelnen Bereiche effizient bearbeitet werden konnten.

- **Mathias Grubbauer** übernahm die Projektleitung. Zu seinen Aufgaben gehörten vor allem die Projektplanung, die Datenaufbereitung, die Datenbankmodellierung sowie Teile der Backend-Implementierung.
- **Alexander Vorreither** war für die Integration der BPMN-Diagrammfunktionalität und für zentrale Teile der Anwendungslogik zuständig. Dazu gehörten insbesondere die Einbindung der BPMN-Modellierungsbibliothek sowie Funktionen zur Darstellung und Bearbeitung der Diagramme.
- **Justin Luckeneder** war hauptsächlich für die Entwicklung des Frontends verantwortlich. Seine Aufgaben umfassten die Gestaltung und technische Umsetzung der Benutzeroberfläche sowie die Interaktion zwischen Benutzer und System.

Zur übersichtlichen Darstellung der Aufgabenverteilung wurde zusätzlich eine IMV-Matrix erstellt. Diese zeigt, welche Teammitglieder bei den einzelnen Arbeitspaketen informiert, mitwirkend oder verantwortlich waren.

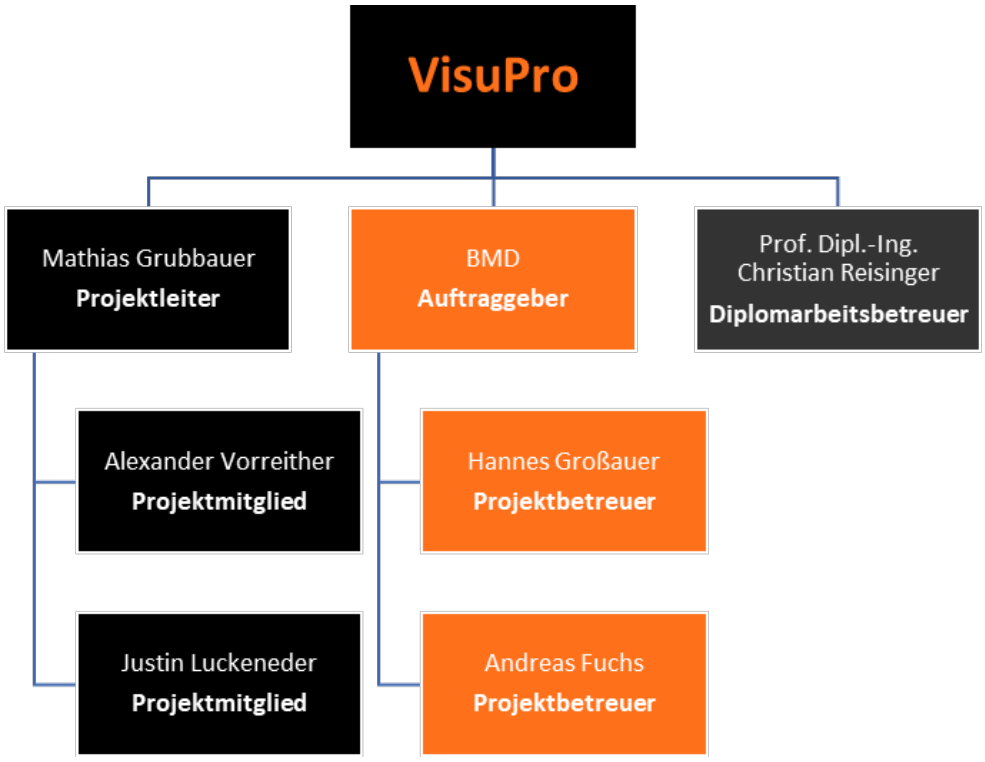


Abbildung 22: Organigramm der Projektaufteilung

Aufgabe / Arbeitspaket		Mathias Grubbauer	Justin Luckeneder	Alexander Vorreither	Kommentar
DA - VisuPro					
1	Projektplanung	V	M	M	
2	Anforderungsanalyse	V	M	M	
3	Use-Case-Erstellung	V	M	M	
4	Design u. Mockup	I	V	I	
5	Systemarchitektur	M	I	V	
6	Backend-Architektur	V	M	I	
7	Datenbankmodellierung	V	I	I	
8	Implementierung Backend	M	V	I	
9	Editor-Implementierung	I	M	V	
10	BPMN-Algorithmus	M	M	V	
11	Dynamischer Bildexport	V	M	I	
12	EPK-BPMN-Vergleich	V	M	M	
13	Schnittstellen	V	M	I	
14	Testplanung	V	M	M	
15	Tests	V	M	M	
16	Dokumentation	V	M	M	
17	Abnahme	V	M	M	
		PL	Dev	Dev	

I: Informiert, **M:** Mitarbeit, **V:** Verantwortlich

Abbildung 23: Informations-, Mitwirkungs- und Verantwortungsmatrix des Projektteams

3.1.4 Kommunikationswege

Zu Beginn des Projekts fand ein Kick-off-Meeting mit dem Auftraggeber statt. Dabei wurden die Projektziele, zentrale Anforderungen und der grundsätzliche Projektablauf besprochen.

Während der Entwicklungsphase erfolgte die Abstimmung mit dem Auftraggeber in regelmäßigen Meetings. In diesen Besprechungen wurde der aktuelle Projektstand vorgestellt, offene Fragen wurden geklärt und weiteres Feedback direkt in die Umsetzung aufgenommen.

Zur Organisation der praktischen Projektarbeit wurde zusätzlich ein Kanban-Board verwendet. Dieses diente dazu, Arbeitspakete zu erfassen, Teammitgliedern zuzuordnen und ihren Bearbeitungsstand nachvollziehbar darzustellen. Auf diese Weise konnten Aufgaben strukturiert verwaltet und der Projektfortschritt laufend überblickt werden.

Eine Übersicht über die verwendete Aufgabenverwaltung zeigt Abbildung 24.

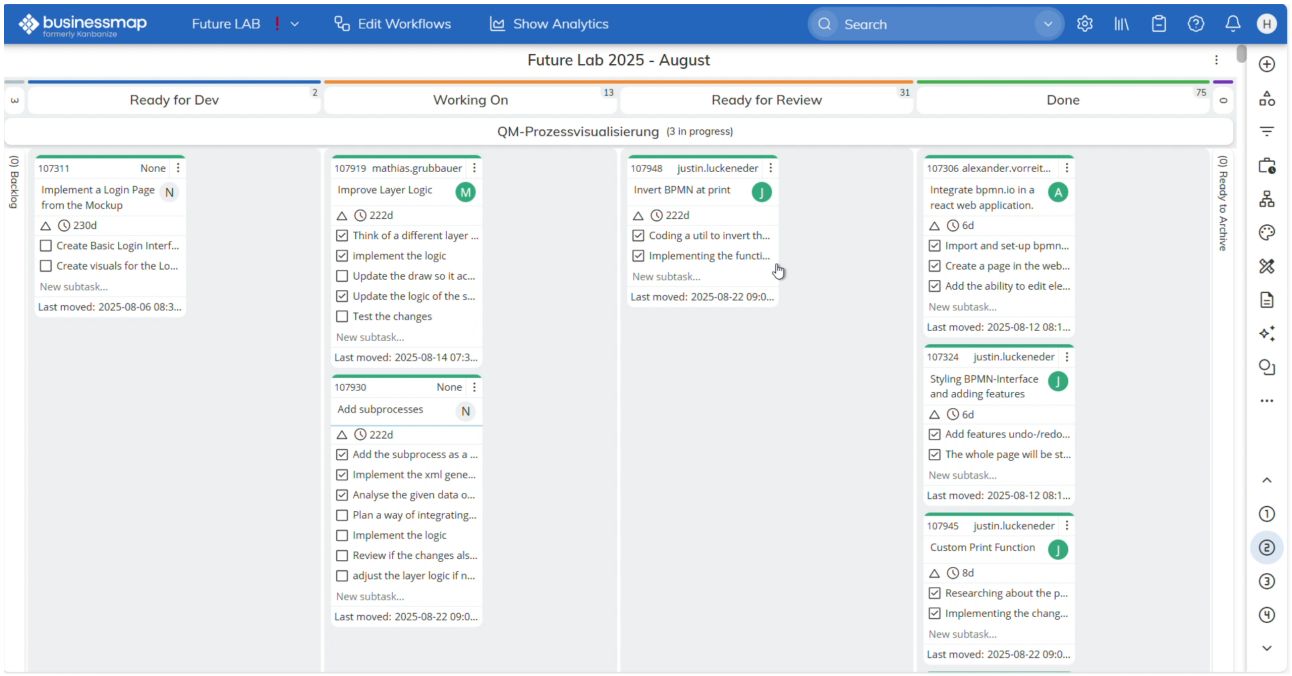


Abbildung 24: Kanban-Board zur Verwaltung von Arbeitspaketen und Bearbeitungsständen während des Projekts

Die wichtigsten Termine sind in Tabelle 3 dargestellt.

Termin	Beschreibung
01.07.2025	Kick-off-Meeting
06.08.2025	Abstimmungsmeeting 1 mit Auftraggeber
13.08.2025	Abstimmungsmeeting 2 mit Auftraggeber
19.08.2025	Statusbericht 1 und 2
20.08.2025	Abstimmungsmeeting 3 mit Auftraggeber
26.08.2025	Statusbericht 3
27.08.2025	Abstimmungsmeeting 4 mit Auftraggeber
29.08.2025	Abschlusspräsentation bei BMD

Tabelle 3: Projekttermine und Meetings

3.2 Meilensteine

Zur Strukturierung und Kontrolle des Projektverlaufs wurden im Rahmen von VisuPro mehrere Meilensteine definiert. Diese markieren zentrale Zwischenergebnisse und dienten als Orientierung für die zeitliche und inhaltliche Planung des Projekts.

Die Meilensteine orientieren sich an den wesentlichen Projektphasen. Dazu zählen insbesondere die Projektinitialisierung, die Konzeption der Benutzeroberfläche, die Umsetzung der Backend-Schnittstellen sowie die Integration und Bearbeitung von BPMN-Diagrammen in der Webanwendung.

3.2.1 Zeitplanung

Tabelle 4 zeigt die wichtigsten geplanten Meilensteine des Projekts in komprimierter Form.

Datum	Meilenstein
01.07.2025	Kick-off-Meeting und offizieller Projektstart
04.08.2025	Beginn der technischen Umsetzungsphase im Rahmen des Praktikums
04.08.2025	Datenbankschema und grundlegendes UI-Design fertiggestellt
18.08.2025	Anzeige von BPMN-Diagrammen im Viewer implementiert
28.08.2025	Backend-Schnittstellen fertiggestellt
28.08.2025	Erstellung und Bearbeitung von BPMN-Diagrammen umgesetzt
28.08.2025	Integration von Frontend und Backend abgeschlossen
29.08.2025	Abschlusspräsentation beim Auftraggeber

Tabelle 4: Zentrale Meilensteine des Projekts

Ergänzend zur tabellarischen Übersicht wurde der Projektablauf in Form von Zeit- und Planungsdiagrammen ausgearbeitet. Diese verdeutlichen die zeitliche Reihenfolge der Arbeitspakete sowie deren Abhängigkeiten. Da eine vollständige Detaildarstellung im Fließtext nur eingeschränkt lesbar ist, werden im Folgenden die drei Hauptphasen des Projekts zusammenfassend beschrieben.

Die **Projektinitialisierungsphase** umfasste organisatorische und planerische Grundlagenarbeiten. Dazu zählten die Projektdefinition, der Projektstrukturplan, der CM-Plan, die Aufwandsschätzung, der Zeitplan, das Risikomanagement sowie die Erstellung von Dokumentvorlagen. Diese Phase endete mit dem Kick-off und bildete die Grundlage für die anschließende Umsetzung.

Die **Backend-Entwicklung** konzentrierte sich zunächst auf die Datenverwaltung. In diesem Zusammenhang wurden das Datenmodell erstellt, das Datenbankschema ausgearbeitet sowie die Datenbank eingerichtet. Darauf aufbauend erfolgte die Implementierung der REST-Schnittstellen, über die Daten ausgelesen, neu angelegt und verändert werden konnten. Ergänzend wurden Optimierungen, Fehlerbehebungen sowie begleitende Dokumentationsarbeiten durchgeführt.

Die **Frontend-Entwicklung** begann mit der Konzeption der Benutzeroberfläche in Form eines Mockups. Anschließend wurde die Designoberfläche technisch umgesetzt. Aufbauend auf dieser Grundstruktur wurden die BPMN-Funktionalitäten integriert, darunter das Laden, Anzeigen, Bearbeiten und Erstellen von Diagrammen. Den Abschluss bildeten qualitätssichernde Funktionen wie Validierung und Undo-Mechanismen sowie die Anbindung an das Backend.

3.2.2 Abhängigkeiten

Zwischen den einzelnen Projektphasen bestanden sowohl organisatorische als auch technische Abhängigkeiten. Die Projektinitialisierung musste zunächst abgeschlossen werden, da erst dadurch eine verbindliche Grundlage für die Umsetzung geschaffen wurde.

Im Backend stellte insbesondere die Erstellung des Datenmodells eine Voraussetzung für die weitere technische Entwicklung dar. Erst nach Definition der Datenstruktur konnten die Datenbank eingerichtet und die REST-Schnittstellen umgesetzt werden.

Im Frontend konnte die Gestaltung der Benutzeroberfläche zunächst unabhängig vom Backend erfolgen. Die Implementierung zentraler Diagrammfunktionen sowie die dauerhafte Verarbeitung

von Prozessdaten setzten jedoch eine funktionierende Backend-Anbindung voraus. Dadurch ergab sich gegen Ende des Projekts eine Zusammenführung beider Entwicklungsstränge.

Insgesamt zeigt die Planung, dass das Projekt in logisch aufeinander aufbauende Phasen gegliedert war. Die Visualisierung der Meilensteine und Abhängigkeiten unterstützte dabei die Kontrolle des Fortschritts sowie die Abstimmung innerhalb des Teams.

3.3 Risiken und Maßnahmen

Im Rahmen der Projektplanung wurden mögliche Risiken identifiziert, die den Projektverlauf negativ beeinflussen können. Ziel des Risikomanagements ist es, potenzielle Probleme frühzeitig zu erkennen, deren Auswirkungen einzuschätzen und geeignete Gegenmaßnahmen festzulegen. Dabei wurden technische, organisatorische, inhaltliche, zeitliche und qualitative Risiken betrachtet.

3.3.1 Risikoidentifikation und Bewertung

Für das Projekt VisuPro wurden mehrere Risiken identifiziert, die während der Projektumsetzung auftreten können. Die Bewertung erfolgt anhand der Eintrittswahrscheinlichkeit sowie der möglichen Auswirkungen auf den Projektverlauf.

Risiko	**Wahrscheinlichkeit**	**Auswirkung**
Unzureichende Erfahrung mit neuen Technologien	mittel	mittel bis hoch
Eingeschränkter Zugriff auf notwendige Systeme oder Hardware	niedrig bis mittel	mittel
Unklare Anforderungen oder verspätete Abstimmung mit dem Auftraggeber	mittel	hoch
Unvollständige oder widersprüchliche Prozessdokumentation	mittel	mittel
Zeitliche Verzögerungen durch unvorhergesehene Probleme	mittel	hoch
Mangelnde Benutzerfreundlichkeit der Anwendung	niedrig bis mittel	mittel

Tabelle 5: Bewertung der wesentlichen Projektrisiken

Die Bewertung zeigt, dass insbesondere unklare Anforderungen, technische Herausforderungen sowie mögliche zeitliche Verzögerungen den größten Einfluss auf das Projekt haben können. Aus diesem Grund wurden für jedes identifizierte Risiko geeignete Maßnahmen definiert.

3.3.2 Risiko-Maßnahmen-Map

Die in Tabelle 6 dargestellte Risiko-Maßnahmen-Map zeigt die wesentlichen Projektrisiken sowie die jeweils festgelegten Gegenmaßnahmen.

Risiko	**Maßnahme**
Unzureichende Erfahrung mit neuen Technologien	Frühzeitige Einarbeitung in die eingesetzten Technologien, technische Recherchen vor Beginn der Implementierung sowie prototypische Umsetzung kritischer Funktionen.
Eingeschränkter Zugriff auf notwendige Systeme oder Hardware	Frühzeitige Abstimmung mit dem Auftraggeber bezüglich benötigter Zugänge, wiederholtes Nachfassen bei fehlenden Ressourcen sowie Ausweichen auf alternative Test- oder Entwicklungsumgebungen.
Unklare Anforderungen oder verspätete Abstimmung mit dem Auftraggeber	Regelmäßige Abstimmungsmeetings, laufende Klärung offener Fragen und frühzeitige Rücksprache bei Unklarheiten, um Fehlentwicklungen zu vermeiden.
Unvollständige oder widersprüchliche Prozessdokumentation	Gemeinsame Durchsicht der vorhandenen Prozessbeschreibungen mit dem Auftraggeber sowie fachliche Präzisierung unklarer oder widersprüchlicher Inhalte.
Zeitliche Verzögerungen durch unvorhergesehene Probleme	Berücksichtigung von Pufferzeiten im Projektplan, laufende Fortschrittskontrolle anhand der Meilensteine und Priorisierung zentraler Funktionen.
Mangelnde Benutzerfreundlichkeit der Anwendung	Erstellung und Abstimmung von Mockups, frühes Einholen von Feedback sowie schrittweise Verbesserung der Benutzeroberfläche auf Basis der Rückmeldungen.

Tabelle 6: Risiko-Maßnahmen-Map

3.3.3 Zusammenfassung der Maßnahmen

Die definierten Gegenmaßnahmen dienen dazu, die Eintrittswahrscheinlichkeit identifizierter Risiken zu verringern beziehungsweise deren Auswirkungen auf den Projektverlauf möglichst gering zu halten. Besonders wichtig waren dabei die laufende Abstimmung mit dem Auftraggeber, die frühzeitige technische Vorbereitung sowie die regelmäßige Überprüfung des Projektfortschritts anhand der festgelegten Meilensteine. Auf diese Weise konnten potenzielle Probleme früh erkannt und rechtzeitig berücksichtigt werden.

3.4 Use Cases und Anforderungen

Dieser Abschnitt beschreibt die wesentlichen Anwendungsfälle des Systems VisuPro sowie die daraus abgeleiteten funktionalen und nicht-funktionalen Anforderungen. Use Cases stellen typische Interaktionen zwischen Benutzer und System strukturiert dar und machen die erwarteten Funktionen der Anwendung nachvollziehbar.

VisuPro unterstützt die Visualisierung, Erstellung, Bearbeitung, Speicherung und den Export von Geschäftsprozessen in Form von BPMN-Diagrammen. Benutzer laden bestehende Prozessmodelle, zeigen diese an oder legen neue Diagramme an und bearbeiten sie weiter.

3.4.1 Use Cases

Tabelle 7 fasst die wichtigsten Anwendungsfälle des Systems zusammen.

Use Case	**Beschreibung**
Diagramm laden	Der Benutzer lädt ein bereits gespeichertes BPMN-Diagramm aus der Datenbank, um es im System weiterzuverwenden.
Diagramm anzeigen	Das System stellt ein geladenes BPMN-Diagramm grafisch im Editor dar.
Neues Diagramm erstellen	Der Benutzer legt ein neues BPMN-Diagramm an und fügt erste Prozesselemente hinzu.
Diagramm bearbeiten	Der Benutzer verändert oder verschiebt bestehende BPMN-Elemente und ergänzt bei Bedarf neue Elemente.
Diagramm speichern	Das System speichert ein neu erstelltes oder bearbeitetes Diagramm über die REST-Schnittstelle dauerhaft im Backend.
Diagramm exportieren	Der Benutzer exportiert ein BPMN-Diagramm in ein anderes Ausgabeformat, beispielsweise als Bild oder Dokument.

Tabelle 7: Wesentliche Use Cases des Systems VisuPro

Abbildung 25 und Abbildung 26 zeigen zwei zentrale Nutzungsszenarien des Systems in Form von Use-Case-Diagrammen. Einerseits veranschaulichen sie den Umgang mit bereits vorhandenen Diagrammen, andererseits den Ablauf beim Erstellen und Speichern neuer beziehungsweise bearbeiteter Diagramme.

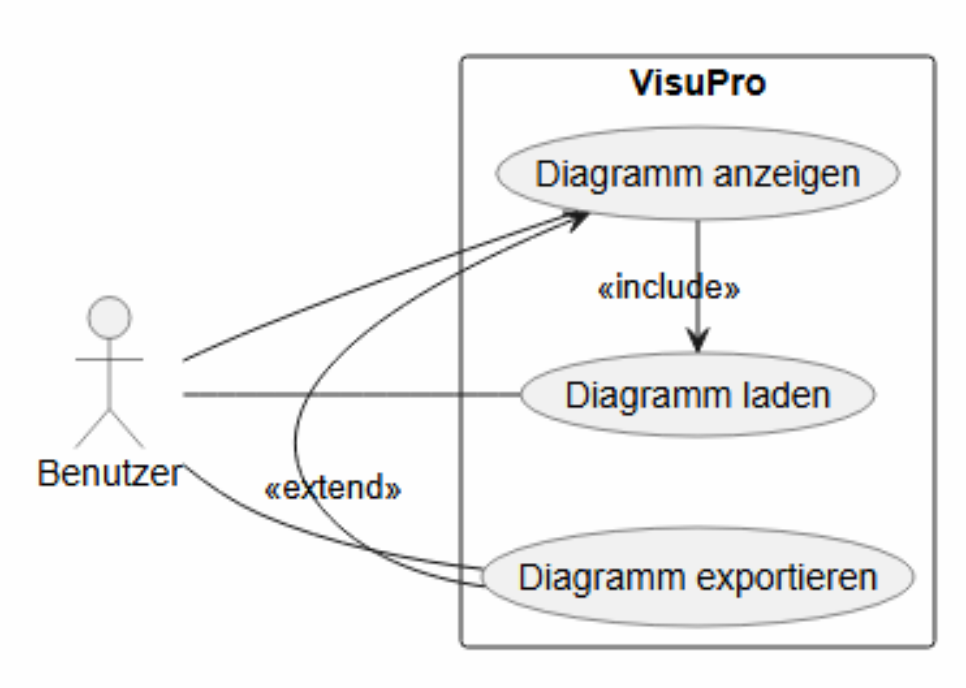


Abbildung 25: Use-Case-Diagramm zum Laden, Anzeigen und Exportieren eines Diagramms

Abbildung 25 verdeutlicht den Ablauf beim Arbeiten mit bereits gespeicherten Diagrammen. Der Benutzer lädt ein bestehendes BPMN-Diagramm, betrachtet dieses im Editor und exportiert es bei Bedarf in ein anderes Format. Dieser Anwendungsfall spielt insbesondere für die Analyse, Dokumentation und Weiterverwendung vorhandener Prozesse eine wichtige Rolle.

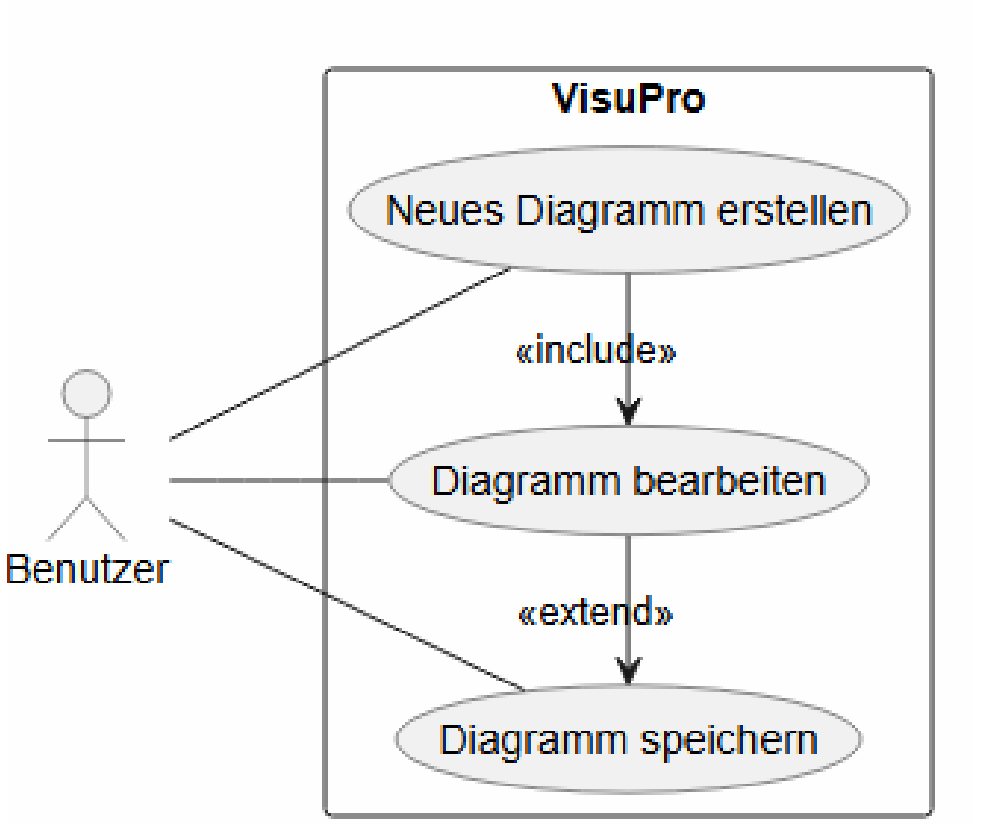


Abbildung 26: Use-Case-Diagramm zum Erstellen, Bearbeiten und Speichern eines Diagramms

Abbildung 26 beschreibt den zweiten wesentlichen Nutzungskontext des Systems. Der Benutzer erstellt ein neues Diagramm, bearbeitet dieses im Editor und speichert es anschließend dauerhaft. Damit unterstützt VisuPro den vollständigen Ablauf von der Modellierung bis zur Ablage eines Prozessmodells.

Die dargestellten Use Cases bilden die Grundlage für die Umsetzung der zentralen Funktionen von VisuPro. Gleichzeitig erleichtern sie die strukturierte Herleitung der Anforderungen an das System.

3.4.2 Anforderungen

Aus den beschriebenen Anwendungsfällen ergeben sich funktionale und nicht-funktionale Anforderungen, die den Aufbau und die Qualität des Systems festlegen.

Funktionale Anforderungen

Funktionale Anforderungen beschreiben die Leistungen, mit denen das System die definierten Use Cases unterstützt.

- Das System lädt gespeicherte BPMN-Diagramme aus der Datenbank.
- Das System stellt BPMN-Diagramme im Editor grafisch dar.
- Benutzer erstellen neue BPMN-Diagramme.
- Benutzer bearbeiten und erweitern bestehende Diagramme.
- Das System speichert neu erstellte oder veränderte Diagramme dauerhaft.
- Das System überträgt BPMN-Daten über eine REST-Schnittstelle zwischen Frontend und Backend.
- Das System exportiert Diagramme in andere Formate.

Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben qualitative Eigenschaften des Systems, die unabhängig von einzelnen Funktionen dessen Nutzbarkeit und technische Qualität prägen.

- Eine übersichtliche und intuitive Benutzeroberfläche unterstützt eine einfache Bedienung.
- Die Anwendung läuft in gängigen Webbrowsern.
- Die Kommunikation zwischen Frontend und Backend erfolgt über standardisierte REST-Schnittstellen.
- Ein modularer Aufbau erleichtert Wartung und Erweiterung des Systems.
- Die Speicherung der BPMN-Daten gewährleistet Zuverlässigkeit und Konsistenz.
- Die Darstellung und Bearbeitung von Diagrammen erfolgt mit angemessener Reaktionsgeschwindigkeit.

Die strukturierte Darstellung der Use Cases und Anforderungen schafft eine nachvollziehbare Grundlage für die Implementierung des Systems. Zugleich erleichtert diese Gliederung die spätere Zuordnung der umgesetzten Funktionen zu den fachlichen und technischen Zielsetzungen des Projekts.

3.5 Systementwurf

In diesem Kapitel wird beschrieben, wie die geplante Architektur des Programms aufgebaut ist.

3.5.1 Systemarchitektur

Das Projekt ist, um eine übersichtlichere Struktur zu erlangen, in verschiedene Komponenten unterteilt. Dieses Kapitel widmet sich der Darstellung dieser. Abbildung 27 zeigt eine grafische Übersicht über all diese Komponenten.

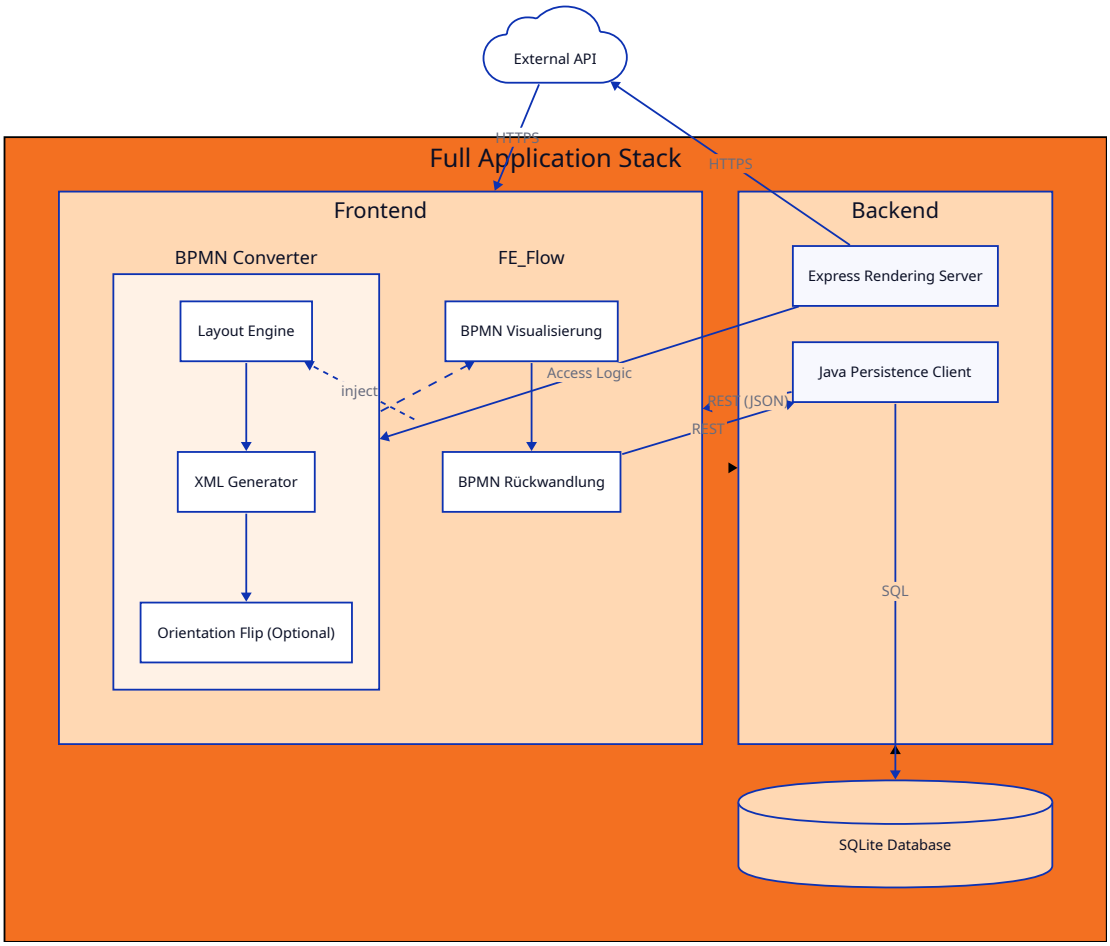


Abbildung 27: Darstellung der Architektur, Module, Schnittstellen und des Datenflusses

3.5.1.1 Frontendarchitektur

In diesem Abschnitt werden die verschiedenen Komponenten des Frontends genauer beschrieben. Das Frontend kann Daten von der externen BMD-API sowohl als auch von unserem entwickelten Java-REST-Programm abfragen. Diese Daten liegen im JSON-Format vor. Die Struktur der Daten ist von der BMD vorgegeben und von uns nicht beeinflussbar.

BPMN-Converter

Diese Komponente ist dafür zuständig, die JSON-Daten in ein, laut dem offiziellen Schema der OMG, valides BPMN-XML umzuwandeln. Der Converter erhält die JSON-Daten vom Frontend und übergibt sie an die Layout-Engine. Diese verarbeitet die Daten und generiert basierend darauf ein Grid mit den einzelnen Knoten, das anschließend wieder an den Converter zurückgegeben wird. Der Converter ruft mit diesen Daten den XML-Generator auf, welcher das Grid benutzt, um damit das BPMN-XML zu erstellen. Es besteht auch die Möglichkeit, mittels des Orientation Flips die Fließrichtung des generierten BPMN-Diagramms zu ändern.

- ***Layout-Engine***

  Die Layout-Engine benutzt einen Algorithmus (siehe Abschnitt 4.3.2) um automatisch ein hierarchisches Layout aufzubauen. Diese beruht auf den Prinzipien der wissenschaftlichen Arbeit Ko, „A computer scientist's introductory guide to business process management (BPM)“.

- ***XML-Generator***

  Der XML-Generator verwertet das von der Layout-Engine erzeugte Grid und wandelt die einzelnen Knoten und Sequenzflüsse mittels der Bibliothek bpmn-moddle (siehe Abschnitt 2.4.3) in valides BPMN-XML um (siehe Abschnitt 4.3.3).

- ***Orientation Flip***

  Diese Komponente benutzt im Gegensatz zu den anderen das fertige BPMN-XML. Somit besteht die Möglichkeit, diese auch außerhalb des BPMN-Converters aufzurufen.

BPMN Visualisierung

Die BPMN Visualisierungskomponente visualisiert über die Bibliothek `bpmn-js` (vgl. Abschnitt 2.4.2) das BPMN-Diagramm. Zusätzlich schafft sie die Möglichkeit, am Diagramm Änderungen vorzunehmen und diese zu speichern (siehe Abschnitt 4.2 und 5.1).

BPMN Rückwandlung

Diese Komponente ermöglicht es, Änderungen, die in der Visualisierungskomponente vorgenommen wurden, wieder in den persistenten Speicher zurückzuführen. Dafür wandelt sie das BPMN-XML wieder in JSON-Daten um und stellt mit diesen eine Speicheranfrage an das Backend.

3.5.1.2 Backendarchitektur

Die Backendarchitektur besteht aus Komponenten die Daten für andere Module zur Verfügung stellen.

- ***Java REST Backend***

  Das Java REST Backend dient als Prototyp für die persistente Speicherung von Änderungen der einzelnen Prozesse und für den Test des Algorithmus mit Daten, die durch die eigenen Positionsparameter gesetzt haben. (siehe Abschnitt 4.6.1)

- ***Express-JS Bild Export***

  Die NTCS bietet im Rahmen des Qualitätsmanagements die Funktion, einen Report über ein Projekt zu erzeugen. Um diesen anschaulicher zu gestalten, gibt es die Anforderung, darin auch die Visualisierung in Diagrammform zu inkludieren. Genau für diesen Zweck existiert die Express-JS-Bild-Export Komponente. Um auf Befehl von eine dynamisch gewählten BPMN-Diagramm ein Bild zu schaffen, implementiert diese einen Headless-Browser(siehe Kapitel: 2.2.8).

- ***Verwendete Schnittstellen***

Im folgenden Abschnitt werden kurz alle externen Schnittstellen beschrieben, die innerhalb der Anwendung verwendet werden.

Datenschnittstelle für BMD-interne Daten

Im Rahmen der Diplomarbeit wurde uns von der BMD eine Schnittstelle zur Verfügung gestellt, die es uns ermöglicht, die von der BMD definierten Qualitätsmanagementprozesse in unser Programm zu laden.

Die Basis für alle Anfragen erfolgt über die folgende URL: Grund-URL: `https://bmdwebsvc.bmd.at/b`

Pfade

Für den Abruf der relevanten QM-Daten stehen folgende Pfade zur Verfügung.

/wfm/projectPlan Diese Schnittstelle ermöglicht es, Qualitätsmanagementprozesse vom BMD-Backend zu laden. Im Aufruf können zusätzliche Query-Parameter definiert werden. Die Parameter und wie diese die Query beeinflussen, sind in der Tabelle 8 dargestellt.

Attribut	**Erklärung**
MCA_PRO_PROJEKTNR	Wenn angegeben, wird nur der Qualitätsprozess mit dieser Nummer zurückgegeben
Ist leer	Wenn kein Wert angegeben ist, wird eine Liste aller Projekte zurückgegeben. In diesem Fall werden nicht alle Attribute von den Daten geladen.
RES_FIELDS	Definiert, welche Felder bei der Rückgabe mitgegeben werden.
Ist leer	Alle Felder werden mitgegeben

Tabelle 8: Parameter des Endpunkts `/wfm/projectPlan`

Mann kann die Parameter der Tabelle 8 folgendermaßen in der Query angeben:

```
/wfm/projectPlan?<parameter>=<value>&<parameter2>=<value2>
```

/getkey Lädt den öffentlichen Schlüssel der BMD-API. Dieser Schlüssel wird später für die Authentifizierung verwendet.

/getSession Erzeugt eine Session auf dem BMD-internen Server, über die man anschließend einen Token anfragen kann.

/getToken Dieser Endpunkt wickelt die Authentifizierung ab.

Dabei ist es erforderlich, im HTTP-Body die in der Tabelle 9 definierten Parameter anzugeben.

Attribut	**Erklärung**
apiKey	Kombination aus dem Feld `data` der Antwort von `/getKey` und einem privaten Schlüssel, getrennt durch den String `|`
userName	Persönlicher Benutzername bzw. BMD-Personenkürzel.
password	Ein SHA3-512-Hashwert aus dem zusammengesetzten String: `<sessionId von der /getSession>|<des bmd server passwords>`
sessionId	Die ID aus der Antwort der `/getSession`

Tabelle 9: Argumente

Die Parameter sind im HTTP-Body in folgender Form zu übergeben:

```
<parameter>=<value>&<parameter2>=<value2>...
```

3.5.2 Datenfluss

Die Kommunikation zwischen dem Frontend und dem Backend erfolgt mit http Requests. Für den Abruf der Daten von der BMD steht eine REST-API zur Verfügung. Diese Daten werden

abgerufen und dem BPMN-Converter übergeben.

Der Converter reicht die Daten an die Funktionen seiner Subkomponenten weiter, um ein valides XML zu kreieren.

Die BPMN Visualisierungs Komponente bekommt die XML und zeigt sie in der Benutzeroberfläche an. Falls der Benutzer diese Änderung speichert, reicht diese das XML zur Rückwandlungs Komponente weiter.

Diese wandelt das XML wieder in die JSON-Struktur der BMD zurück und reicht sie an unser Backend weiter, welches diese Daten wieder persistiert.

3.5.3 Datenhaltung und Datenbankmodell

3.5.3.1 Einleitung

Als Datenbank wird SQLite Version 3.50.2 (siehe Kapitel 2.2.3) verwendet. Diese wird nur temporäre für die Entwicklung benutzt, da kein direkter Zugriff auf die BMD-interne Datenbank zur Verfügung gestellt wurde.

3.5.3.2 Datenmodell

Hintergrund

Das Datenmodell (siehe Abbildung 28) basiert auf JSON-Daten, die vom Kunden über eine Schnittstelle zur Verfügung gestellt wurden. Da für diese Daten keine Dokumentation vorhanden ist, erfolgte eine umfangreiche Datenanalyse, um die logische Struktur zu identifizieren (für die genaue Datenbankdokumentation siehe C).

ERD

Abbildung 28 stellt das Datenmodell visuell als ein Entity-Relationship-Diagramm dar. Es wurde dabei das Werkzeug MySQL Workbench (siehe Kapitel 2.3.3) zur Modellierung verwendet.

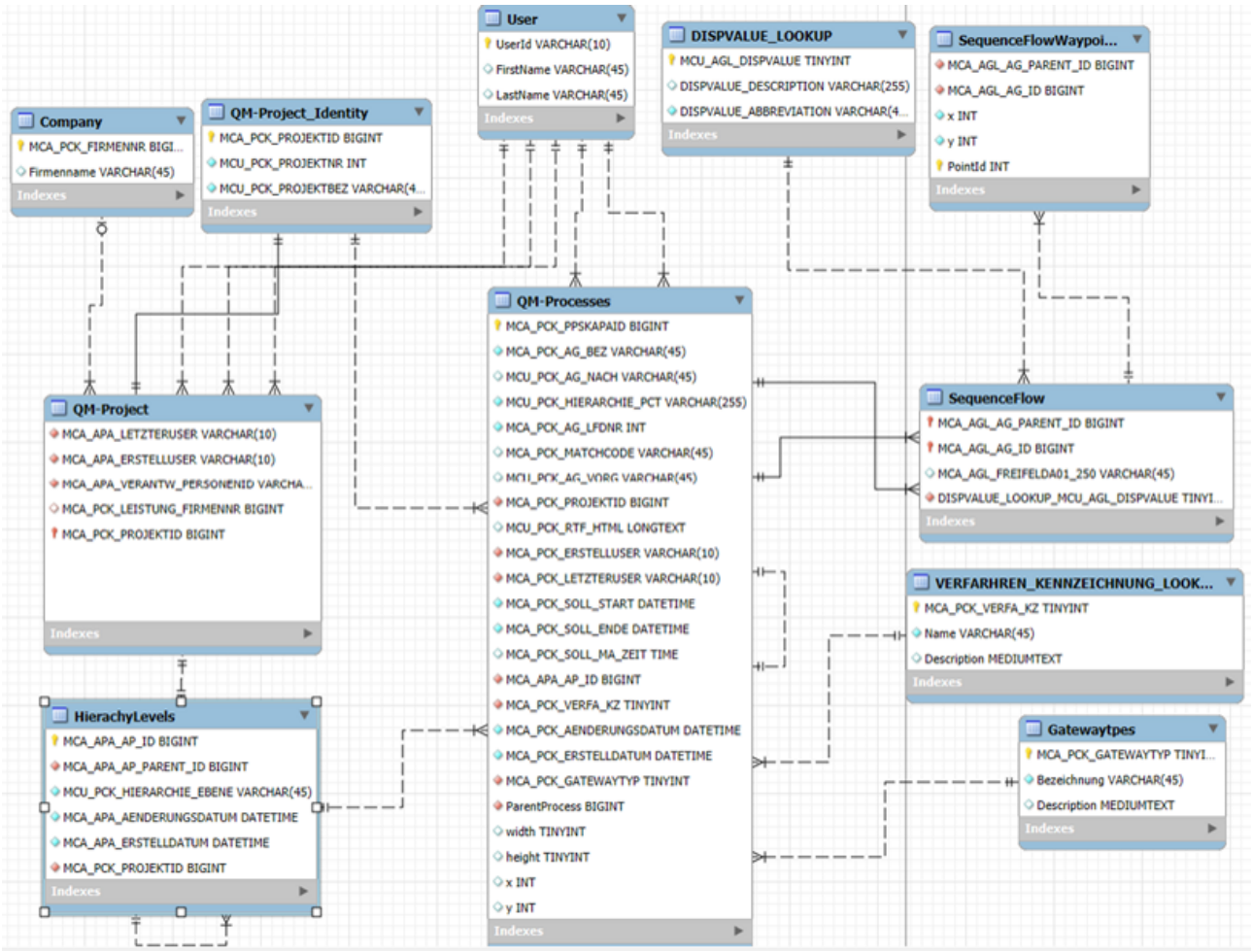


Abbildung 28: Datenmodell

Beschreibung der Entitätsmengen

In der folgenden Liste werden die einzelnen Entitätsmengen des Datenmodells beschrieben. Die komplette Datendokumentation ist im Anhang (siehe Anhang C) vorhanden.

Company

Die Entitätsmenge Company repräsentiert einen Kunden oder Geschäftspartner der BMD SYSTEMHAUS GesmbH.

User

Die Entitätsmenge User repräsentiert einen Nutzer des Produkts der BMD. Dieser Nutzer kann im System Änderungen durchführen.

QM-Project

Die Entitätsmenge QM-Project repräsentiert ein Qualitätsmanagement-Projekt. Pro Projekt kann es bis zu einem BPMN Diagramm geben.

QM-Processes

Die Entitätsmenge QM-Processes stellt eine Arbeitseinheit innerhalb eines Projektablaufs dar. Sie modelliert einen einzelnen, konkreten Schritt in einem BPMN-Geschäftsprozess.

SequenceFlow

Die Entitätsmenge SequenceFlow beschreibt die Verbindung zwischen zwei Prozessen. Sie beinhaltet die Richtung des logischen Kontrollflusses und die Art der Verbindung.

SequenceFlowWaypoint

Die Entitätsmenge SequenceFlowWaypoint beinhaltet die X und Y Koordinaten. Diese verwendet ein Sequenzfluss, um die Punkte, durch die eine graphische Darstellung der Verbindung fließt, zu definieren.

DISPVALUE_LOOKUP

Diese Entitätsmenge ist eine Referenzrelation, in welcher alle Arten von Sequenzflüssen gespeichert werden. Es gibt folgende Arten:

- Ende zu Ende
- Anfang zu Ende
- Ende zu Anfang
- Anfang zu Anfang

HierarchyLevels

Ein Qualitätsmanagement-Prozess ist Teil genau einer Hierarchieebene. Der Prozess bestehet aus einer Wurzelebene und für jeden Grad der Verschachtelung von Subprozessen existiert eine weitere.

Qualitätsmanagement-Prozesse derselben Ebene müssen nicht zum selben Subprozess gehören.

QM-Project_Identity

Diese Entitätsmenge dient der Einhaltung der Boyce-Codd-Normalform.

VERFAHREN_KENNZEICHNUNG_LOOKUP

Diese Entitätsmenge ist eine Referenzrelation. Sie dient zur Speicherung aller Arten eines BPMN-Prozesses.

Es existieren folgende Arten(für eine genauere Beschreibung der einzelnen Arten siehe 2.1.1.1):

- Task
- Event
- Gateway
- Subprocess

Gatewaytypes

In der BPMN gibt es verschiedene Gateway-Logiken. Diese Entitätsmenge dient als Referenzrelation, zum Definieren aller im Programm implementierten Gateway-Typen.

Bedeutung im Zusammenhang mit dem Programm

Für die BPMN-Modellierung werden die Daten wie folgt verwendet. Jede Entität in der Entitätsmenge QM-Project_Identity (siehe: 3.5.3.2) repräsentiert für den Algorithmus ein Diagramm.

Mittels des Attributs MCA_PCK_PROJEKTID werden alle BPMN Elemente des Diagramms von der Entitätsmenge QM-Processes (siehe Kapitel: 3.5.3.2) geladen. Die Art des Elements wird durch die Entitäten VERFAHREN_KENNZEICHNUNG_LOOKUP (siehe Kapitel: 3.5.3.2) und Gatewaytypes (siehe Kapitel: 3.5.3.2) gekennzeichnet. Basierend auf den Positionsattributen generiert der Algorithmus die BPMN-Elemente an deren korrekten Positionen. Falls diese Attribute noch nicht belegt sind, wird die Position basierend auf der Hierarchie später automatisch generiert.

Im nächsten Schritt kann der Algorithmus anhand der Verbindungen zwischen den Elementen die Hierarchie extrapolieren. Diese ist in der Entitätsmenge SequenceFlow (siehe: 3.5.3.2) ersichtlich.

Die visuelle Darstellung eines Sequenzflusses kann durch Wegpunkte, die in der Entitätsmenge SequenceFlowWaypoints (siehe: 3.5.3.2) definiert sind, gesteuert werden.

3.5.4 Maßnahmen zur Qualitätssicherung

Das Gabler Wirtschaftslexikon definiert Qualitätssicherung folgendermaßen: „Die Qualitätssicherung umfasst als Bestandteil des Qualitätsmanagements alle organisatorischen und technischen Maßnahmen, die vorbereitend, begleitend und prüfend der Schaffung und Erhaltung einer definierten Qualität eines Produkts oder einer Dienstleistung dienen.“[55]

Die Qualität der entwickelten Anwendung wird im Rahmen des Projekts anhand der in der Projektdefinition festgelegten Erfolgskriterien definiert. Dazu zählen insbesondere die korrekte Darstellung und Bearbeitbarkeit von BPMN-Diagrammen, sowie die Ausführbarkeit der Anwendung und die positive Bewertung der betreuenden Fachkräfte. (siehe Anhang D.1)

Theoretische Einordnung

Grundsätzlich lassen sich Maßnahmen der Qualitätssicherung in konstruktive und analytische Maßnahmen einteilen. Konstruktive Maßnahmen zielen darauf ab, bereits während der Entwicklung ein qualitätsförderndes Umfeld zu schaffen und Fehler möglichst zu vermeiden. Dazu zählen zum Beispiel die Wahl geeigneter Technologien und Werkzeuge, eine strukturierte Projektorganisation, eine klare Systemarchitektur sowie der Einsatz von Versionsverwaltung und Dokumentation.

Analytische Maßnahmen dienen hingegen der Überprüfung der Qualität bereits erstellter Projektergebnisse. Diese können statisch erfolgen, beispielsweise durch die Analyse von Programmcode oder Konzepten, oder dynamisch durch das Ausführen der Software und die Beobachtung ihres Verhaltens. Letztere Form stellt das eigentliche Testen dar und ermöglicht die Überprüfung, ob die entwickelte Anwendung die definierten Anforderungen korrekt erfüllt.[56]

Konstruktive Maßnahmen im Projekt

Im Rahmen dieses Projekts wurden verschiedene Maßnahmen zur Qualitätssicherung eingeplant. Eine wichtige konstruktive Maßnahme stellt beispielsweise die Wahl geeigneter Technologien dar. Anstatt JavaScript wurde hier TypeScript gewählt, da durch die statische Typisierung mögliche Programmfehler schon im Vorhinein erkannt werden können. Auch wird die Wartbarkeit und

[55] Vogt, *Definition: Was ist "Qualitätssicherung"?*

[56] Vgl. Schwab und Schwab-Matkovits, *Systemplanung und Projektentwicklung: Projektmanagement und Software-Entwicklung*, S. 315-327.

Lesbarkeit des Codes verbessert. Darüber hinaus werden etablierte Open-Source-Bibliotheken wie bpmn.io verwendet, um BPMN-Diagramme zu modellieren und zu bearbeiten. (siehe Kapitel 2.4 und 2.5 für mehr Informationen zu gewählten Technologien und Bibliotheken)

Analytische Maßnahmen im Projekt

Zu den angewendeten analytischen Maßnahmen zählen diverse Unittests, Integrationstests und Systemtests. Diese Testarten decken unterschiedliche Ebenen ab, von einzelnen Funktionen bis hin zur Überprüfung des Gesamtsystems. (für konkrete Tests siehe Kapitel 4.7, für mehr Informationen zu den einzelnen Testarten siehe 2.1.2)

4 Implementierung

In diesem Kapitel wird die technische Umsetzung des Systems beschrieben. Zunächst werden das entwickelte Mockup und der interaktive Editor vorgestellt. Dann werden die implementierten Algorithmen zur Generierung der XML- und JSON-Strukturen sowie der Imageexport erläutert. Abschließend wird die Backend-Architektur dargestellt.

4.1 Mockup

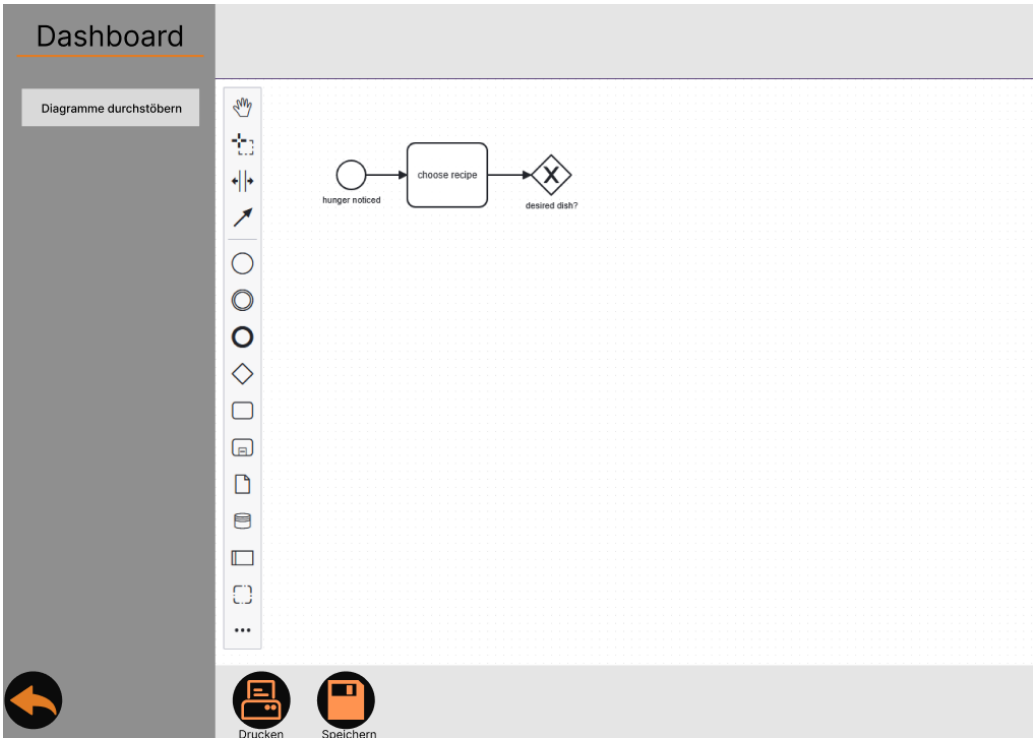


Abbildung 29: Mockup der Benutzeroberfläche, erstellt mit Figma

Wie in Abbildung 29 dargestellt, zeigt das Mockup eine erste konzeptionelle Version der Anwendung. Es wurde mit dem Designwerkzeug Figma (siehe 2.5.1) erstellt und diente als Grundlage für die spätere Implementierung der Benutzeroberfläche im Frontend.

Das Mockup ermöglicht es, zentrale Funktionen und die grundlegende Struktur der Anwendung bereits vor der eigentlichen Implementierung zu visualisieren. Dadurch konnten Layout, Be-

dienkonzept sowie die Anordnung der wichtigsten Benutzeroberflächenkomponenten frühzeitig geplant und mit den Projektbeteiligten abgestimmt werden.

Das Mockup umfasst die grafische Konzeption zentraler Standardfunktionen, darunter:

- das Laden eines bereits bestehenden BPMN-Diagramms,
- die Erstellung und Bearbeitung von BPMN-Diagrammen im Editor,
- die Umwandlung des erstellten BPMN-Diagramms in ein Bildformat (initial PDF, später PNG),
- sowie das Speichern des erstellten BPMN-Diagramms.

Ein zentraler Bestandteil der Benutzeroberfläche ist der BPMN-Diagramm-Editor, der auf der Open-Source-Bibliothek `bpmn.io` basiert. Eine detaillierte Beschreibung dieser Bibliothek und ihrer Funktionsweise ist in Abschnitt 2.5.3 zu finden.

Die Werkzeugleiste orientiert sich daher in ihrem grundlegenden Aufbau an der von `bpmn.io` bereitgestellten Standardoberfläche. Im weiteren Projektverlauf wurde diese Oberfläche jedoch an die spezifischen Anforderungen angepasst, beispielsweise durch die Anpassung der angezeigten Werkzeuge und deren Anordnung innerhalb der Benutzeroberfläche.

Das Mockup diente außerdem als wichtige Grundlage für die spätere Implementierung der Weboberfläche mit modernen Webtechnologien. Die im Mockup definierten UI-Komponenten wurden anschließend schrittweise in HTML, CSS und TypeScript umgesetzt und mit der Diagrammfunktionalität des BPMN-Editors verbunden.

4.1.1 Werkzeugleiste des BPMN-Editors

Die Werkzeugleiste (siehe 30) befindet sich auf der linken Seite des Editors und stellt die zentralen Funktionen zur Erstellung und Bearbeitung von BPMN-Diagrammen bereit.

Im oberen Bereich befinden sich Werkzeuge zur Navigation und Interaktion. Dazu zählen das Verschieben des Diagrammausschnitts, die Auswahl einzelner oder mehrerer Elemente sowie das Erstellen von Verbindungen zwischen Elementen.

Darunter sind die BPMN-Elemente angeordnet, die in das Diagramm eingefügt werden können. Dazu gehören insbesondere Ereignisse, Aktivitäten, Gateways sowie weitere Artefakte. Die fachliche Bedeutung dieser Elemente wird in Abschnitt 2.1.1.1 beschrieben.

Am unteren Ende der Werkzeugleiste können zusätzliche Elemente und erweiterte Funktionen ausgewählt werden.

Abbildung 30: Werkzeugleiste des BPMN-Editors

4.2 BPMN-Editor und interaktive Bearbeitung

Dieser Abschnitt beschreibt die technische Umsetzung des BPMN-Editors und die interaktiven Bearbeitungsfunktionen. Zuerst wird erläutert, wie der BPMN-Modeler in die Anwendung integriert wurde und welche wichtigen Funktionen und Erweiterungen im Rahmen des Projekts implementiert wurden. Dazu zählen die Steuerung des Editors, Anpassungen der Benutzeroberfläche sowie die Anpassung und Erweiterung des Eigenschaften- und Metadatenpanels.

4.2.1 Integration des BPMN-Editors

Der BPMN-Editor ist über die Route `/visualisieren/:mode?` erreichbar. Praktisch werden dabei folgende Aufrufe verwendet:

- `/visualisieren/new` für ein neues Diagramm
- `/visualisieren?MCA_PRO_PROJEKTNR=<Projektnummer>` zum Laden eines Prozesses aus der NTCS.

Der BPMN-Editor ist als React-Seite `Visualisieren` implementiert.

Ablauf, wenn eine Projektnummer angegeben ist:

1. Laden der zugehörigen Prozessdaten aus der NTCS
2. Überführen der Daten in ein gültiges BPMN-Modell durch den im Abschnitt 4.3 beschriebenen Konvertierungsalgorithmus
3. Importieren und Visualisieren des erzeugten BPMN-XMLs im BPMN-Editor

Die Visualisierung und Interaktion selbst erfolgt mit `bpmn-js` (siehe Abschnitt 2.4.2). Dabei übernimmt `diagram-js` die Zeichenfläche und Interaktionsmechanik (z. B. Auswahl, Verschieben, Verbindungen), während `bpmn-moddle` die BPMN-Datenstruktur und XML-Serialisierung bereitstellt (siehe Abschnitt 2.4.3). `bpmn-moddle` wird bereits im toXML-Algorithmus aufgerufen, um aus den geladenen Prozessdaten ein BPMN-konformes Modell aufzubauen und anschließend als BPMN-XML zu serialisieren, was im Editor angezeigt werden kann.

Die technische Kapselung der Editorinitialisierung erfolgt über die Hook `useBpmnModeler`. Diese Hook verfolgt drei Ziele:

1. zentrale Initialisierung und Freigabe der Modeler-Instanz
2. registrierbare Erweiterbarkeit über Module
3. entkoppelte Bereitstellung der Editorfunktionen an die React-UI

Beim Mounten der Seite erstellt die Hook die Modeler-Instanz und registriert zusätzliche Module: eine angepasste Palette, das Eigenschaftenpanel, ein Metadatenmodul sowie den Color-Picker (siehe Listing 2). Beim Unmount werden Event-Listener entfernt und die Instanz freigegeben. Damit wird ein sauberer Lebenszyklus sichergestellt und Speicher-/Event-Leaks werden vermieden.

Listing 2: Registrierung von Erweiterungsmodulen in der Modeler-Initialisierung

```
const m = new Modeler({
  container: containerRef.current!,
  propertiesPanel: { parent: propertiesPanelRef.current || "#properties" },
  additionalModules: [
    { __init__: ["paletteProvider"], paletteProvider: ["type", CustomPaletteProvider] },
    ColorPickerModule,
    BpmnPropertiesPanelModule,
    ProjectMetaModule
  ]
});
```

4.2.2 Interaktive Kernfunktionen

Die Hook stellt zentrale Interaktionsfunktionen bereit, die in der Toolbar gebunden werden: Undo/Redo, Zoom In/Out und Zoom-Reset.

Die Undo/Redo-Funktionalität basiert auf dem `commandStack`. Jede modellrelevante Änderung wird als Command protokolliert. Über `canUndo()` und `canRedo()` wird der Zustand des Stacks ausgelesen und dann in den UI-Status (aktiv/deaktiviert) übergeben. Die eigentlichen Aktionen verweisen auf `undo()` bzw. `redo()` des Services.

Listing 3: CommandStack-Integration für Undo/Redo inklusive UI-Zustand

```
const commandStack = m.get("commandStack");
setCanUndo(commandStack.canUndo());
setCanRedo(commandStack.canRedo());
const onStackChanged = () => {
  setCanUndo(commandStack.canUndo());
  setCanRedo(commandStack.canRedo());
};

eventBus.on("commandStack.changed", onStackChanged);
```

Die Zoom-Funktionen basieren auf dem canvas-Service von `bpmn-js`. Über die Hook werden dafür eigene Funktionen, wie `zoomIn()`, `zoomOut()` und `zoomReset()` bereitgestellt, die dann von der Toolbar aufgerufen werden können. Zusätzlich wird der mögliche Zoombereich begrenzt, um extreme Skalierungen zu verhindern.

4.2.3 UI-Erweiterungen und Eigenschaften-/Metadatenpanel

Neben der Modellierungsfläche wurden mehrere UI-Erweiterungen umgesetzt: eine feste Toolbar (Speichern, Undo/Redo, Zoom), ein Lade-Overlay und ein rechtes Seitenpanel. Das Seitenpanel ist ein- und ausklappbar und unterstützt eine manuelle Breitenanpassung per Drag, dadurch kann es sich der Benutzer so einstellen, wie er möchte. Inhaltlich kombiniert dieses Panel zwei Ebenen, einmal die Standard-Eigenschaften des `bpmn-js-properties-panel` und zweitens die projektspezifischen Metadaten über das `ProjectMetaModule`.

Das `ProjectMetaModule` ergänzt zusätzliche Felder und bindet Prozessdaten kontextbezogen an das aktuell ausgewählte BPMN-Element. Wenn kein BPMN-Element ausgewählt ist, werden die von der NTCS-gepflegten Metadaten zum Gesamtprozess im Panel angezeigt. Beispielsweise Prozessname, `ID` und weitere Metadaten. Die Feldlogik unterscheidet explizit zwischen read-only und editierbaren Werten. Read-only-Werte sind beispielsweise Metadaten, die aus der NTCS entnommen wurden und gar nicht editierbar sein sollten, wie die verschiedenen IDs eines Elements oder das Erstelldatum (des in der Buchhaltungssoftware gepflegten Prozesselements, nicht dem in der Webanwendung generierten). Felder wie die Beschreibung oder Bezeichnung hingegen sind frei editierbar.

4.2.4 Visuelle Anpassungen des Editors

Neben funktionalen Erweiterungen wurde auch die Darstellung des Standard-`bpmn-js`-Editors gezielt angepasst. Hierfür werden eigene Stylesheets (`styles.css`, `properties-panel.css`) eingesetzt, die über spezifische Selektoren und priorisierte Regeln das Erscheinungsbild zentraler UI-Bereiche verändern (u.a. Properties-Panel, Toolbar, Panel-Gruppen).

Listing 4: Überschreiben von Standard-Styling zu den BMD-Stammfarben

```
...
.djs-palette {
  background-color: rgba(234, 114, 3, 1) !important;
  margin-top: 60px;
  margin-left: 2.5px;
  width: fit-content;
  border-color: #000;
}

.djs-palette .entry {
  color: var(--text-muted);
  padding: 6px;
  margin: 2px 0;
  transition: all 0.2s ease;
}
...
```

Zusätzlich wurde die Standardpalette nicht nur visuell angepasst, sondern auch funktional reduziert. Der in Listing 5 dargestellte `CustomPaletteProvider` erweitert den Standard-`PaletteProvider` von `bpmn-js`, um die verfügbare Palette an Modellierungselementen einzuschränken, da nicht alle BPMN-Elemente in der NTCS gepflegt und verwendet werden. Nach der Registrierung des Providers im Konstruktor über `palette.registerProvider(this)` wird die Standardpalette zunächst über `super.getPaletteEntries()` übernommen. Anschließend werden nicht benötigte Einträge gezielt entfernt, indem entsprechende Aktionen aus dem `actions`-Objekt gelöscht werden (z.B. `create.data-store`, `create.data-object`, `create.participant-expanded` und `create.group`). Deswegen stehen diese Elemente im Editor nicht mehr zur Auswahl und die Palette wird auf den im Projekt benötigten Modellierungsumfang reduziert.

Listing 5: Eigener Palette-Provider zur funktionalen Reduktion der Standardpalette (Auszug)

```
export default class CustomPaletteProvider extends PaletteProvider {
  constructor(
    palette: any,
    create: any,
    elementFactory: any,
    spaceTool: any,
    lassoTool: any,
    handTool: any,
    globalConnect: any,
    translate: any
  ) {
    super(palette, create, elementFactory, spaceTool, lassoTool, handTool, globalConnect,
        translate);
    palette.registerProvider(this);
  }

  getPaletteEntries(): any {
    const actions = super.getPaletteEntries();
    delete actions["create.data-store"];
    delete actions["create.data-object"];
    delete actions["create.participant-expanded"];
    delete actions["create.group"];
    return actions;
  }
}
```

4.3 Algorithmus zur Generierung eines BPMN-Diagramms

Um das BPMN-Diagramm darzustellen, müssen die vom BMD-Backend erhaltenen Daten zunächst in XML umgewandelt werden. Auch der Aufbau des BPMN-Diagramms spielt hierbei eine wichtige Rolle. Das generierte BPMN-Diagramm muss nicht nur dem offiziellen XML-Schema (siehe offizielle XSD-Datei Object Management Group, *BPMN 2.02 XML Schema Definition (XSD)*) folgen, sondern zusätzlich so angenehm und überschaubar wie möglich gestaltet sein. Um dies zu erreichen, muss man die einzelnen Elemente und die Sequenzflüsse in einer einheitlichen Struktur darstellen. Die Eingabedaten geben dabei zwar die Struktur vor, jedoch gibt es keine Garantie, dass die Elemente in der richtigen Reihenfolge angeordnet sind.

Probleme

Bei der Generierung der Hierarchie treten aufgrund der Struktur der Daten einige Schwierigkeiten auf.

- Die einzelnen Elemente speichern ihre eigenen absoluten und relativen Positionen im Diagramm nicht. Der Algorithmus benötigt diese jedoch, da die Platzierung der folgenden Elemente auch einen Einfluss auf die Position eines vorher platzierten nehmen kann.
- BPMN unterstützt zyklische Abläufe, diese können bei weiteren Prozessen zu doppelt verarbeiteten Knoten führen.
- BPMN unterstützt Subprozesse. Die Unterelemente sind wiederum zu strukturieren.

4.3.1 Eingabedaten

Die Eingabedaten beschreiben Qualitätsmanagementprozesse, die innerhalb der BMD für die Qualitätssicherung verwendet werden. Im Rahmen der Projektentwicklung stellte uns die BMD zwei verwendete Qualitätsmanagementprozesse zur Verfügung. Diese beinhalten alle von der BMD-Software unterstützten Elemente. Diese Daten stehen im JSON-Format, unter einem fest definierten Schema, in einer BMD-internen Schnittstelle für die Abfrage zur Verfügung.

Dafür steht unserem Projektteam jedoch keine Datendokumentation zur Verfügung. Deshalb ist im Rahmen des Projekts eine ausführliche Datenanalyse von unserem Team durchgeführt worden. Die wichtigsten Felder sind im Rahmen des Abschnitts 3.5.3.2 einzusehen. Die genaue Dokumentation der Analyse ist in der Datendokumentation (siehe Anhang C) enthalten. Das Listing 6 zeigt die wichtigsten JSON-Daten anhand eines Beispiels. Diese ermöglichen die Generierung einer Hierarchie.

Listing 6: Eingabedaten

```
{
  "MCA_PCK_PPSKAPAID": 596460689002536,            // Die eindeutige ID eines BPMN-Elements

  "MCU_PCK_HIERARCHIE_PCT": "3.4.1",               // Position des Elements innerhalb der Hierarchie
                                                   // (Angabe der Subebene)
                                                   // wird benutzt um zu identifizieren in welchen
                                                   // Subprozess ein Element gehört

  "MCA_PCK_AG_BEZ": "Daten aufbereiten/Ã¼bernehmen: Einkauf", // Bezeichnung des Elements
  "MCA_PCK_VERFA_KZ": 1,                           // Bestimmt den Typ des BPMN-Elements

  "MCU_PCK_AG_NACH_US": [                          // Liste der Nachfolgeelemente
                                                   // Wichtig um die Abfolge der Elemente zu bestimmen
    {
      "MCA_AGL_AG_PARENT_ID": 596460689002536,     // ID des Elternelements
                                                   // (hier identisch mit der ID dieses Elements)

      "MCA_AGL_AG_ID": 596460689003536             // ID des nachfolgenden Elements

    }
  ]
}
```

4.3.2 Verarbeitungsschritte

Die Implementierung des Algorithmus kann man in mehrere Teile, die in der Abbildung 31 ersichtlich sind, aufteilen.

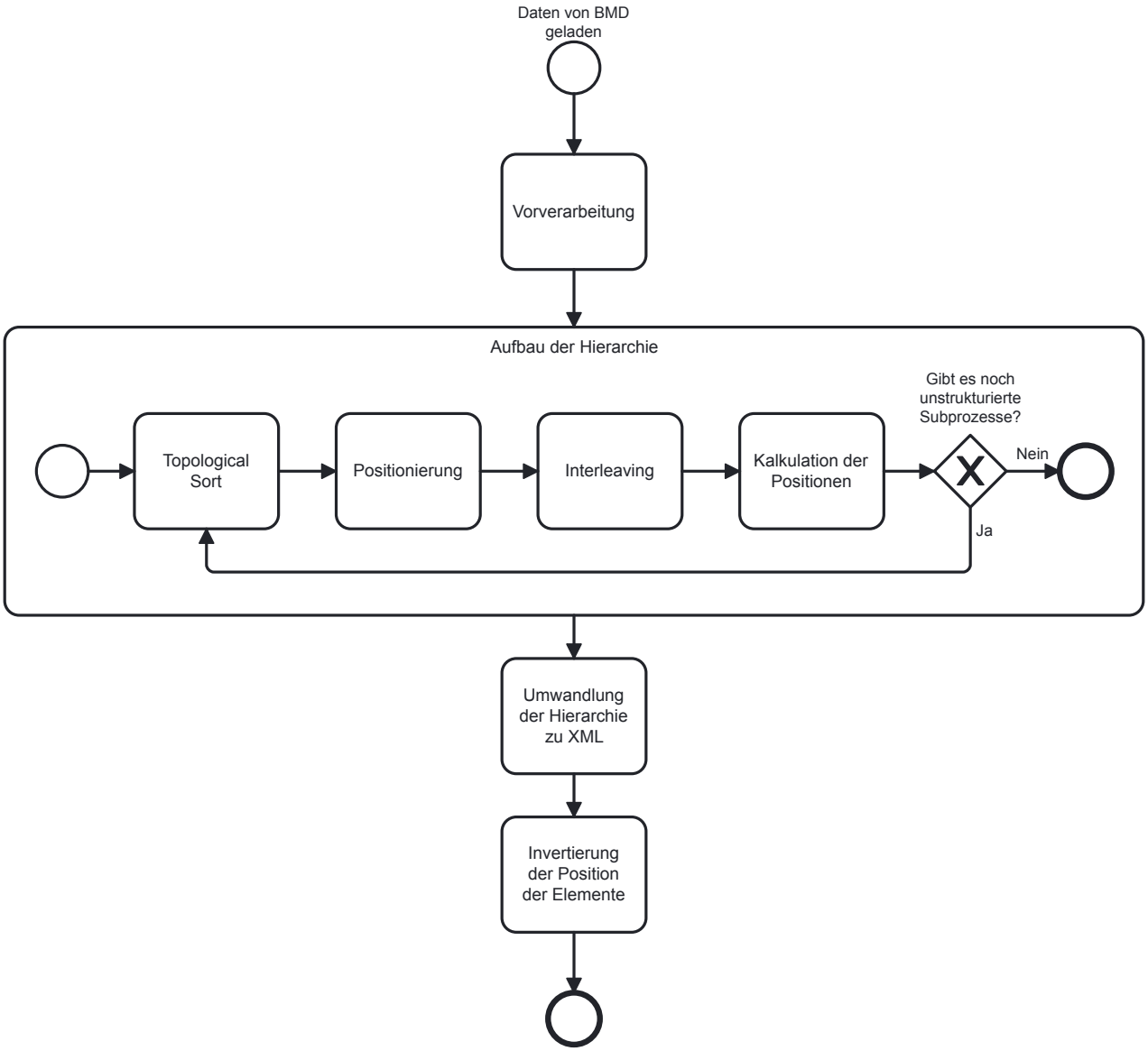


Abbildung 31: Ablauf des Konvertierungsprozesses

Im ersten Schritt werden die Daten vorverarbeitet. Danach behandelt der Algorithmus die Umgehung der definierten Probleme (siehe Abschnitt: 4.3). Deshalb erfolgen vor der eigentlichen Generierung des XML-Dokuments einige Vorverarbeitungsschritte, die diesen Prozess ermöglichen:

1. Das Vorsortieren der Daten, ermöglicht dem Algorithmus die Generierung des XML-Textes ohne vorherige Positionen zu berücksichtigen.
2. Positionierung der Daten in einem Grid (siehe Abschnitt: 4.3.2.2), welches die Hierarchie simuliert.
3. Überarbeitung, die Artefakte der vorherigen Prozesse bereinigt.

4. Berechnung der Position.

Da BPMN Subprozesse (siehe Kapitel: 2.1.1.1, Tabelle 1, Teilprozess 1) unterstützt, reichen diese Schritte alleine jedoch nicht aus. Diese verursachen ein weiteres Hindernis, das der Algorithmus überwinden muss. Subprozesse unterstützen komplette Teildiagramme in sich selbst. Somit reicht es nicht aus, den Algorithmus über alle Elemente laufen zu lassen, sondern es muss für jeden Subprozess die Hierarchie unabhängig berechnet werden.

Der Algorithmus erreicht dies, indem er rekursiv jeden Subprozess sucht und die folgenden 4 Schritte 4.3.2.2, 4.3.2.2, 4.3.2.2, 4.3.2.3 für jeden Subprozess unabhängig durchführt.

4.3.2.1 Vorverarbeitung

Um den Prozess effizienter zu gestalten, baut der Algorithmus zuerst einige Datenstrukturen auf. Diese verhindern, dass bei jedem Schritt das komplette Array zu durchsuchen ist.

Dabei entstehen die folgenden Strukturen:

- Eine HashMap, die alle Elemente über deren ID referenziert.
- Eine HashMap, die alle Joins mit deren Splits verbindet.

Zusätzlich korrigiert dieser Schritt Eigenheiten der BMD-Struktur. Zum Beispiel speichert die BMD Start- und Endknoten doppelt. Deshalb fügt der Algorithmus in diesem Schritt auch all diese Knoten zusammen.

4.3.2.2 Aufbau der Hierarchie

Für die visuelle Darstellung des Diagramms muss vorerst die Hierarchie aufgebaut werden. Die Generierung dieser folgt, mit Ausnahme der Generierung von Sequenzflüssen, den Basisideen des wissenschaftlichen Papiers Ko, „A computer scientist's introductory guide to business process management (BPM)“.

Grundidee des Grids

Das Grid hat den Vorteil, dass man neue Zweige bei Gateways einfügen kann, ohne dabei die relative Position des Knotens zu ändern. Ein Beispiel dafür sieht man in der Abbildung 32. Hier wird eine neue Reihe, die grau markiert ist, eingefügt. Das beeinflusst die Reihe oberhalb des Gateways und die Reihe darunter nicht. Das Grid startet mit einer Spalte und einer Reihe. Sobald ein Split auftritt, erfolgt die Einfügung einer neuen Reihe.[57]

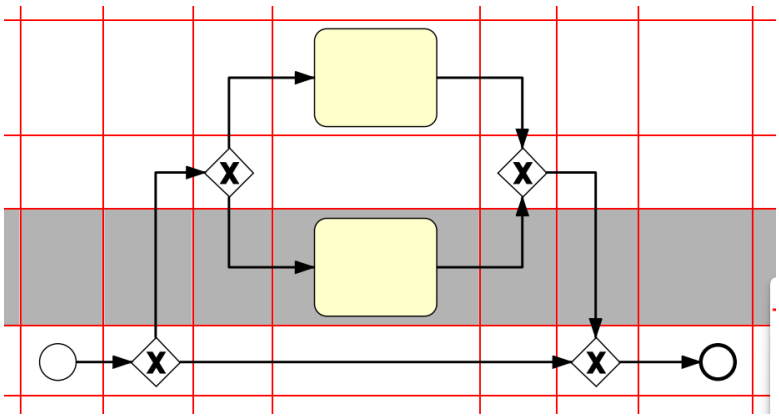


Abbildung 32: Einfügen eines neuen Zweiges

Modified Topological Sort

Wie vorher schon erwähnt, kommen die Eingangsdaten nicht sortiert an. Das führt zu Problemen, falls nicht alle Vorgänger des neuen Knoten ausgelegt sind, da ein neuer Knoten keine Informationen über seine Position im Diagramm speichert. Wenn ein Nachfolger schon modelliert wurde, kann der neue Knoten nicht mehr feststellen, ob er vorher oder nachher eingefügt werden muss.

Der Sortieralgorithmus erfüllt zusätzlich die Aufgabe, Schleifen aufzulösen. Dafür führt dieser folgende Schritte durch:

- Zuerst traversiert dieser das Layout, bis er eine Schleife entdeckt oder alle Knoten besucht wurden. Der Algorithmus erreicht das, indem er alle ausgehenden Sequenzflüsse, also die Nachfolger des Elements, in eine Queue legt.

  Dann, nachdem alle Operationen für dieses Element durchgeführt sind, nimmt er sich das oberste Element von der Queue und führt den Schritt für dieses durch.

- Falls die Queue leer ist, aber nicht alle Knoten besucht wurden, liegt eine Schleife vor.

[57] Vgl. Ko, „A computer scientist's introductory guide to business process management (BPM)“.

- In diesem Fall sucht der Algorithmus unter den bestehenden Knoten nach dem Knoten, der einen eingehenden Sequenzfluss hat. Dieser Knoten wurde schon besucht, was bedeutet, dass dieser fix zu diesem Teil des Diagramms gehört. Da er jedoch immer noch in der Queue ist, bedeutet das, dass er mehr als einmal aufgerufen wurde und somit Teil der Schleife ist.

  Falls dieser nicht vorhanden ist, benutzt man das erste Element.

- Im nächsten Schritt ändert der Algorithmus alle eingehenden Sequenzflüsse des Elements virtuell zu ausgehenden.

  Dadurch wird die Schleife gebrochen, und das Layout wird azyklisch. Weiters führt es dazu, dass man einen Vorgänger-Knoten später als ein neuer Knoten modellieren kann.

- Um das für die ganze Schleife zu erreichen, führt der Algorithmus eine virtuelle Drehung aller Sequenzflüsse bis zum Anfang der Schleife durch.

Für diesen Schritt wird eine Modifizierung von Kahn's Algorithmus verwendet.

Positionierung eines Elements

Nachdem die Daten mit dem Modified Topological Sort sortiert wurden, können die Elemente anhand der Reihenfolge im Grid platziert werden. Dafür gibt es drei Optionen:

- Falls das Element nur einen Vorgänger und Nachfolger hat, wird er einfach rechts neben dem vorherigen Knoten im Grid platziert.

- Falls das Element mehr als einen Nachfolger hat, wird es Split bezeichnet. Dabei werden Reihen oberhalb und unterhalb der Split-Reihe eingefügt, sodass diese stets die mittlere Reihe bleibt.
  Die folgenden Elemente wissen jedoch nicht, wer ihre Nachbarn sind und somit können sie nicht bestimmen, welchen Platz sie unter den erweiterten Reihen einnehmen müssen. Deshalb wird eine neue Map erstellt, die zur ID des Nachfolgers die Reihe speichert, da das Split-Element die Reihen reservieren muss. Somit muss ein Knoten nur überprüfen ob eine Reservierung vorliegt um zu wissen in welcher Reihe sie zu platzieren sind.

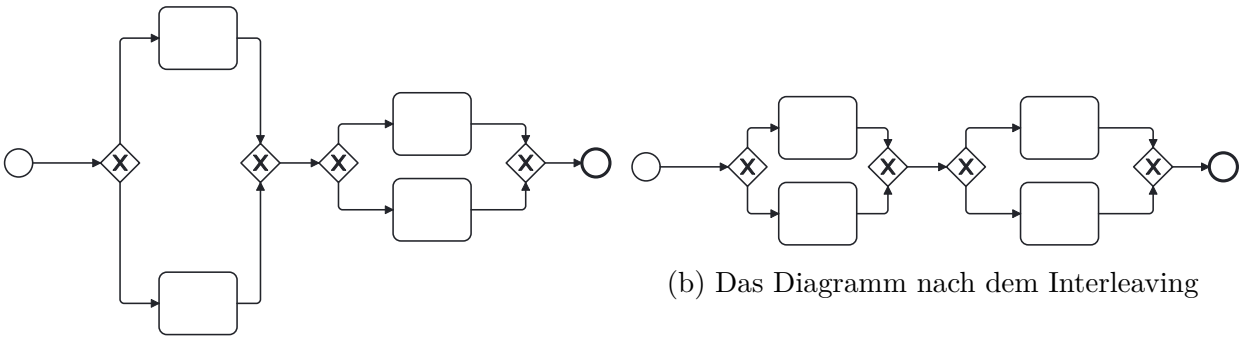
- Falls das Element mehr als einen Vorgänger besitzt, handelt es sich um ein sogenanntes Join. Sofern der Split den Vorgänger gefunden hat, wird der Join in dieselbe Reihe des

Splits gesetzt, sonst wird er einfach in die mittlere Zeile gesetzt. Der Join wird in die Spalte direkt rechts von den Vorgängerknoten platziert.

Interleaving

Da beim vorherigen Verarbeitungsschritt die Chance besteht, dass bei einem Split neue Reihen im Grid eingefügt werden, gibt es Stellen, bei denen das Diagramm die Knoten zu weit voneinander entfernt platziert. Ein Beispiel hierfür ist die Abbildung 33a.[58]

Nach der automatischen Generierung des Diagramms wird deshalb der Interleavingprozess begonnen. Dieser führt alle Reihen des Grids, wo keine Überlappung der Knoten vorhanden ist, zusammen. Aus dem Diagramm der Abbildung 33a entsteht nach dem Interleaving das Diagramm der Abbildung 33b



(a) Das Diagramm vor dem Interleaving

(b) Das Diagramm nach dem Interleaving

Abbildung 33: Interleaving

Implementierung

Methodendefinition der Konversion Im folgenden Absatz wird eine Funktion definiert, die bei allen Reihen des Grids überprüft, ob man sie zusammenführen kann. Diese Funktion stellt die zentrale Funktion des Interleaving-Prozesses dar.

Methode: `interleaveRows()`

Parameter: `grid: Grid`: In diesem Parameter ist das zu optimierende Grid, welches das BPMN-Diagramm beinhaltet, dargestellt

Returns: `grid: Grid`: Das interleaved Grid.

[58] Vgl. Ko, „A computer scientist's introductory guide to business process management (BPM)“.

Definition: Nachdem alle BPMN-Elemente platziert wurden, wird diese Methode aufgerufen. Sie führt zwei Reihen zusammen, falls bei diesen zwei Gridreihen keine Überlappung vorhanden ist. Dabei schreibt diese die Werte in eine gemeinsame Reihe. Dieser Prozess wiederholt sich, bis keine weiteren mehr übrig sind.

Listing 7: Interleaving

```
1  /**
2   * Attempts to interleave rows in the grid to reduce overall height.
3   * @param {Grid} grid The layout grid to optimize.
4   * @returns {Grid} The optimized grid.
5   */
6  function interleaveRows(grid: Grid): Grid {
7    let i = 0;
8    while (i < grid.length - 1) {
9        const row1 = grid[i];
10       const row2 = grid[i + 1];
11
12       if (canInterleave(row1, row2)===true) {
13         mergeRows(grid, i, i+1);
14         if(i !== 0) {
15           rippleUpwards(grid, i-1);
16         }
17         i--;
18       }
19       i++;
20   }
21
22   return grid;
23 }
```

Der Algorithmus überprüft in jeder Reihe mit der Methode `canInterleave()`, ob man zwei Reihen kollisionsfrei zusammenführen kann.

Falls dies eintritt, erfolgt in der Zeile 8, über die Funktion `mergeRows()`, die Zusammenführung der Reihen. Diese Änderung kann bewirken, dass die Reihe über der aktuellen keine Kollisionen mit der aktuellen mehr beinhalten. Dies wird in der Zeile 15 mit der Methode `rippleUpwards()` überprüft. Falls das der Fall ist, wird dasselbe für die nächsthöhere Stufe durchgeführt.

Diese Vorgehensweise vermeidet durch den Ripple-Effekt eine weitere Schleife die den Interleavingprozess wiederholt. Dadurch ist diese Implementation effizienter als die im wissenschaftlichen Papier (siehe: Vgl. Ko, „A computer scientist's introductory guide to business process management (BPM)“, S. 6) vorgeschlagene.

4.3.2.3 Kalkulation der Pixelposition der Elemente

Es werden die Positionen der einzelnen Knoten relativ zu deren Subprozess oder der obersten Ebene, falls keiner oberhalb existiert, berechnet. Für diese Rechnung muss man die maximale Höhe der einzelnen Reihen und die maximale Breite jeder Spalte berechnen. Im letzten Schritt erfolgt die Positionierung der Elemente in der Mitte der Spalte und der Reihe.

Für die Subprozesse werden die Höhe aller Reihen als Höhe und die Länge aller Spalten als Breite benutzt.

4.3.2.4 Konvertieren des Grids in BPMN-XML

Die Positionen all dieser Elemente werden anschließend verwendet, um mit der Bibliothek moddle (siehe Kapitel 2.4.3) die einzelnen Elemente zu kreieren (siehe Listing 8).

Listing 8: Kreation eines Elements

```
currentElement = moddle.create("bpmn:InclusiveGateway", {
                  id: elementId,
                  name: elementName,
                  MCU_PCK_RTF_TEXT: rtfText,
                });
```

Die `moddle.create` Methode nimmt im ersten Feld die Art des Elements und im zweiten ein Objekt, das alle nötigen Attribute beinhaltet.

Die Form eines Elements wird unabhängig davon erstellt. Diese Methode benutzt wiederum die Methode `moddle.create`, der das entsprechende Element zugewiesen wird, um die Form zu kreieren.

Listing 9: Kreation der Form

```
const shape = moddle.create("bpmndi:BPMNShape", {
          id: "Shape_" + currentElement.id,
          bpmnElement: currentElement,
          bounds: moddle.create("dc:Bounds", absolutePos),
          isExpanded: node.MCA_PCK_VERFA_KZ === BpmnNodeType.Subprozess,
        });
        plane.get("planeElement").push(shape);
        bpmnShapes.set(currentElement.id, shape);
```

Zuletzt wird auch für jede Verbindung ein Sequenzfluss zugewiesen. Falls das Zielelement in einer anderen Reihe liegt, wird ein extra Datenpunkt als eine rechtwinklige Ecke generiert. Die Position wird wie folgt bestimmt:

- X-Koordinate: Die X-Koordinate des Herkunftselements + die halbe Breite und
- Y-Koordinate: Die Y-Koordinate des Zielelements - der halben Höhe

Bestehende Daten Falls die Daten bereits bestehende Positionen enthalten, verwendet der Algorithmus diese. Für die Sequenzflüsse werden ebenfalls die bestehenden Datenpunkte verwendet, sofern diese verfügbar sind.

4.3.2.5 Invertierung der Diagrammausrichtung

Nach der Generierung des BPMN-Diagramms kann dessen Ausrichtung optional angepasst werden. Standardmäßig wird das Diagramm horizontal aufgebaut. Für bestimmte Anwendungsfälle ist jedoch eine vertikale Darstellung zweckmäßiger. Zu diesem Zweck wurde ein zusätzlicher Verarbeitungsschritt implementiert, der die grafische Darstellung des Diagramms transformiert, ohne die fachliche Struktur des BPMN-Modells zu verändern.

Die Grundidee des Algorithmus entstand zunächst in Form einer Skizze, in der die horizontale Anordnung der Elemente gedanklich in eine vertikale Darstellung überführt wurde. Darauf aufbauend erfolgte die Umsetzung durch eine Transformation der im BPMN-DI gespeicherten Koordinaten.

Die Invertierung erfolgt in den folgenden Schritten:

1. Zunächst werden alle grafischen Elemente des BPMN-Diagramms durchlaufen. Dabei wird unterschieden, ob es sich um eine Form, eine Kante oder eine Beschriftung handelt.
2. Bei gewöhnlichen Formen, etwa Aufgaben, Ereignissen oder Gateways, wird die neue Position aus dem Mittelpunkt des Elements berechnet. Dadurch bleibt die Lage des Elements auch nach der Transformation konsistent.
3. Bei Subprozessen ist auch eine Vertauschung der Breite und Höhe durchzuführen, da diese Elemente eine Containerfunktion besitzen und ihre Ausdehnung an die neue Ausrichtung angepasst werden muss.
4. Bei Kanten, insbesondere Sequenzflüssen, werden die Koordinaten aller Waypoints vertauscht. Anschließend werden der erste und der letzte Punkt der Kante nachkorrigiert, damit die Verbindung weiterhin visuell korrekt an den verbundenen Elementen anliegt.
5. Falls ein Element eine Beschriftung besitzt, werden auch deren Begrenzungskoordinaten transformiert und an die neue Ausrichtung angepasst.
6. Nach der Verarbeitung aller Elemente liegt das zuvor horizontal aufgebaute Diagramm in vertikaler Ausrichtung vor.

Das Grundprinzip der Transformation besteht darin, die X- und Y-Koordinaten grafischer Elemente zu vertauschen. Für einen einzelnen Punkt (x, y) ergibt sich daraus der transformierte Punkt (y, x). Im implementierten Algorithmus genügt dieser einfache Tausch jedoch nicht in

allen Fällen, da Formen, Kanten und Beschriftungen zusätzliche Eigenschaften wie Breite, Höhe oder Anschlusspositionen besitzen.

Bei gewöhnlichen Formen mit den Begrenzungskoordinaten (x, y) sowie Breite w und Höhe h wird zunächst der Mittelpunkt berechnet:

$$x_m = x + \frac{w}{2}, \qquad y_m = y + \frac{h}{2}$$

Anschließend wird die neue Position so bestimmt, dass das Element auch nach der Invertierung konsistent ausgerichtet bleibt. Entsprechend der Implementierung ergibt sich:

$$x' = y_m - \frac{h}{2}, \qquad y' = x_m - \frac{w}{2}$$

Breite und Höhe bleiben bei diesen Formen unverändert.

Bei Subprozessen werden zusätzlich Breite und Höhe vertauscht. Die neue Position ergibt sich dabei direkt aus dem Tausch der Koordinaten:

$$x' = y, \qquad y' = x, \qquad w' = h, \qquad h' = w$$

Für Kanten werden sämtliche Waypoints einzeln transformiert:

$$(x_i, y_i) \mapsto (y_i, x_i)$$

Da ein reiner Koordinatentausch nicht in jedem Fall zu einer sauberen Verbindung an den Rändern der Elemente führt, werden der erste und der letzte Punkt der Kante anschließend geringfügig korrigiert. Die Korrektur berücksichtigt die Abmessungen der verbundenen Elemente, damit die Kanten weiterhin visuell korrekt andocken.

Beschriftungen werden analog zu den übrigen grafischen Elementen transformiert. Dabei werden ebenfalls Position und Ausdehnung angepasst, damit die Labels nach der Invertierung an einer sinnvollen Stelle im Diagramm verbleiben.

Listing 10: Transformation der Diagrammausrichtung

```
export function swapXYInDiagramBounds(
  definitions: BpmnModdle.Definitions
): BpmnModdle.Definitions {
  if (!definitions?.diagrams?.length) return definitions;

  for (const diagram of definitions.diagrams ?? []) {
    const plane = diagram?.plane;
    const elements = plane?.planeElement ?? [];
    if (!elements.length) continue;

    for (const el of elements) {
      if (isShape(el)) {
        rotateShapeBounds(el);
      } else if (isEdge(el)) {
        rotateEdgeWaypoints(el, elements);
      }
      rotateLabelOnElement(el as AnyShape | AnyEdge);
    }
  }
  return definitions;
}
```

Listing 10 zeigt die zentrale Funktion dieses Verarbeitungsschritts. Sie durchläuft die grafischen Elemente des Diagramms und ruft abhängig vom Elementtyp die jeweilige Transformationslogik auf.

4.3.3 Ausgabe

Ein komplettes BPMN-XML mit Form und allen Sequenzflüssen ausmodelliert. Ein Beispiel mit zwei Elementen und Sequenzflüssen sieht man im Beispiel 11

Listing 11: Auszug aus dem XML-Ergebnis

```
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
    xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
    xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
    xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
    xmlns:bioc="http://bpmn.io/schema/bpmn/biocolor/1.0" id="Definitions_1qsmrfl"
    targetNamespace="http://bpmn.io/schema/bpmn" exporter="bpmn-js (https://demo.bpmn.io)"
    exporterVersion="18.6.1">
  <bpmn:process id="Process_18kr0ed" isExecutable="true">
    <bpmn:task id="Activity_0mf8ddy" name="Funktionstest">
      <bpmn:incoming>Flow_03ahg78</bpmn:incoming>
      <bpmn:outgoing>Flow_0oky6hk</bpmn:outgoing>
    </bpmn:task>
    <bpmn:sequenceFlow id="Flow_03ahg78" sourceRef="Gateway\_0vzoq6f"
        targetRef="Activity_0mf8ddy" />
    <bpmn:sequenceFlow id="Flow_1u8ksem" sourceRef="Gateway\_0vzoq6f"
        targetRef="Gateway_0czhk2y" />
    <bpmn:task id="Activity_0x1uwi9" name="Techniken beschließen">
      <bpmn:incoming>Flow_1aeweba</bpmn:incoming>
      <bpmn:outgoing>Flow_0hy04uy</bpmn:outgoing>
    </bpmn:task>
    ...
    </bpmn:process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="Process_18kr0ed">
      <bpmndi:BPMNShape id="Activity_0mf8ddy_di" bpmnElement="Activity_0mf8ddy">
        <dc:Bounds x="550" y="347" width="100" height="80" />
      </bpmndi:BPMNShape>
```

```
20          </bpmndi:BPMNEdge>
21        </bpmndi:BPMNPlane>
22        ...
23      </bpmndi:BPMNDiagram>
24    </bpmn:definitions>
```

4.4 Algorithmus zur BPMN-Rückkonvertierung

4.4.1 Einsatzszenarien und Motivation

Im Editor können Benutzer BPMN-Diagramme bearbeiten. Sie ändern Elementbezeichnungen, fügen neue Knoten (nodes) hinzu, entfernen Elemente oder verschieben Verbindungen. Um diese Änderungen dauerhaft zu speichern, müssen sie in das BMD-JSON-Format zurückkonvertiert werden, das die NTCS-Datenhaltung versteht. Die wichtigsten Felder sind im Kapitel 3.5.3.2 beschrieben. Die genaue Dokumentation der Analyse ist in der Datendokumentation (siehe Anhang C) enthalten.

Der Gesamtablauf der Rückkonvertierung umfasst vier Systemschritte:

1. Benutzer bearbeitet Diagramm im Editor
2. Editor exportiert BPMN-XML vom Modeler
3. `convertToJSON` wandelt dieses XML zurück ins BMD-Format
4. Das Backend speichert anschließend das resultierende JSON.

Anmerkung zur Architektur: Die Backend-Persistierung (siehe 4.6) ist eine `Proof-of-Concept`-Implementierung für diesen Editor. In der Produktivversion würde die BMD die Daten selbst speichern, da sie über ein eigenes Backend-System und eigene Persistierungs-Mechanismen verfügt. Die `convertToJSON`-Funktion ist hingegen eine standardisierte Datentransformationskomponente, unabhängig vom Backend und vollständig übertragbar.

Das eigentliche Problem besteht darin, dass BPMN und BMD-Format sind strukturell unterschiedlich. BPMN arbeitet mit expliziten, gerichteten Kanten (Sequence Flows). Die BMD speichert Verbindungen implizit in jedem Knoten (als Nachfolger und Vorgänger). Zusätzlich muss die hierarchische Struktur der Subprozesse erhalten bleiben und neue Knoten (die im Editor hinzugefügt wurden) müssen gültige BMD-Typinformationen bekommen, auch wenn sie keine historischen Daten aus der Datenhaltung der BMD, also der NTCS, haben.

4.4.2 Eingabe und Ausgabe

4.4.2.1 Eingabe

Die Eingabe ist BPMN-XML, das aus einer dieser Quellen kommt:

- Direkt von `convertToXML` (siehe Abschnitt 4.3)
- Vom `bpmn-js`-Modeler nach Benutzeränderungen: modeler.saveXML()

Optionaler Eingabeparameter: `responseData`: Array aus BMD-Knoten aus der NTCS-API. Enthält Original-Metadaten (Matchcode, RTF-Text, Hierarchie-Info), die fehlen könnten, wenn der Benutzer völlig neue Elemente hinzugefügt hat.

Beispiel BPMN-XML-Fragment (vereinfacht):

Listing 12: BPMN XML Eingabe (Auszug)

```xml
<bpmn:definitions ...>
  <bpmn:process id="Process_1">
    <bpmn:task id="E_123" name="Aufgabe 1"/>
    <bpmn:exclusiveGateway id="E_456" name="Verzweigung"/>
    <bpmn:sequenceFlow id="E_flow_1" sourceRef="E_123" targetRef="E_456"/>
  </bpmn:process>
  <bpmndi:BPMNDiagram>
    <bpmndi:BPMNPlane>
      <bpmndi:BPMNShape bpmnElement="E_123">
        <dc:Bounds x="100" y="50" width="100" height="80"/>
      </bpmndi:BPMNShape>
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>
</bpmn:definitions>
```

4.4.2.2 Ausgabe

Die Ausgabe ist eine BMD-JSON-Struktur:

Listing 13: BMD JSON Output-Schema

```
interface BmdBpmnJson {
    nodes: BmdBpmnNode[];   // Hierarchische Knotenstruktur (flach mit children)
}

interface BmdBpmnNode {
  MCA_PCK_GATEWAYTYP: number;
  MCA_PCK_AG_LFDNR: number;
  MCA_PCK_PPSKAPAID: number | null;
  MCA_APA_AP_PARENT_ID?: number;
  MCU_PCK_RTF_TEXT?: string;

  MCU_PCK_AG_VORG_US?: Array<{
    ...
  }>;

  MCU_PCK_AG_NACH_US?: Array<{
    ...
```

```
    }>;

    MCU_PCK_AG_NACH: string | null;
    MCA_PCK_MATCHCODE: string;
    MCA_PCK_VERFA_KZ: number | null;
    MCU_PCK_HIERARCHIE_PCT: string | null;
    MCU_PCK_HIERARCHIE_EBENE: string;
    MCA_PCK_AG_BEZ: string;

    x?: number;
    y?: number;
    width?: number;
    height?: number;
    children?: BmdBpmnNode[];
}
```

Zentrale Besonderheit: Verbindungen zwischen Knoten in einem Prozessdiagramm und die dazugehörigen Waypoints, also die Punkte, durch die die Linie von einem Element zum anderen verläuft, sind keine eigenen Objekte, sondern in den Nachfolger- und Vorgänger-Arrays eingebettet. Zusätzlich speichert jeder Knoten seine Position (x, y) und Größe (width, height) aus dem Diagramm. Position und Größe sind optional, da sie nur aus dem bpmn-js Modeler stammen, nicht aus convertToXML. Für mehr Informationen zu den restlichen Feldern, die aus der NTCS kommen, siehe das Kapitel 3.5.3.2. Die genaue Dokumentation der Analyse ist in der Datendokumentation (siehe Anhang C) enthalten.

4.4.3 Verarbeitungsschritte im Überblick

Der Algorithmus zur Rückkonvertierung besteht aus fünf Schritten, die wir im Folgenden durchlaufen (siehe auch Abbildung 34 für einen genaueren Überblick):

1. XML-Parsing und Extraktion der Diagramm-Metadaten: Extrahiert Positionsdaten (x, y, Breite, Höhe) und Waypoints aus der BPMN-Diagram-Ebene.
2. Element-Sammlung und Typ-Annotation: Sammelt alle Flow-Knoten (Tasks, Events, Gateways, Subprocesses) und annotiert sie mit ihrem BPMN-Typ.
3. Hierarchische Knotenverarbeitung: Die BMD-Knotenstruktur wird rekursiv aufgebaut, mit Hierarchie-Nummerierung und Subprozess-Verschachtelung.
4. Verbindungs-Ableitung aus Sequence-Flows: Aus den BPMN-Sequence-Flows werden Nachfolger- und Vorgänger-Arrays errechnet und in jeden Knoten eingebettet.
5. Metadaten-Integration: BMD-spezifische Felder werden aus dem BPMN-XML und optionalen responseData zusammengeführt.

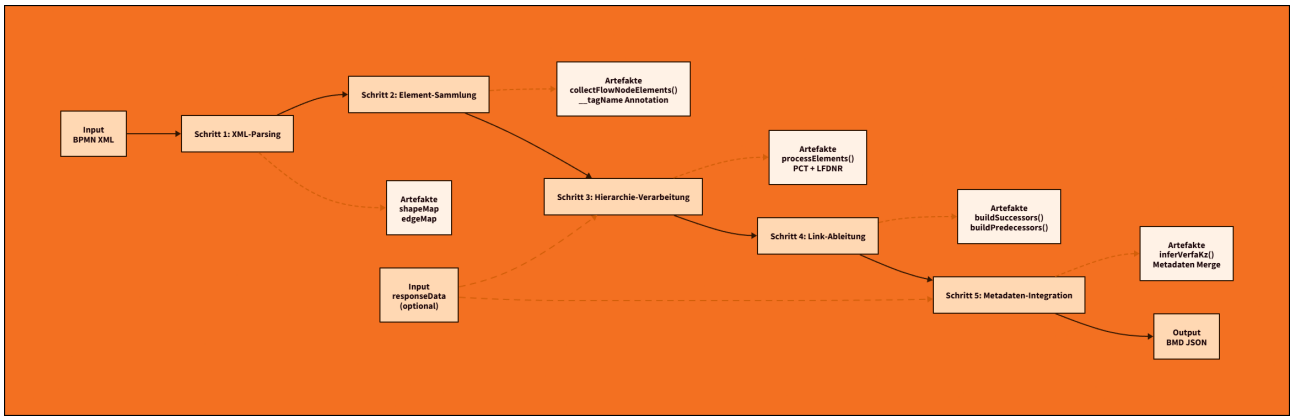


Abbildung 34: Datenfluss durch die fünf Verarbeitungsschritte von convertToJSON()

4.4.4 Detaillierte Verarbeitungsschritte

4.4.4.1 Schritt 1: XML-Parsing und Extraktion der Diagramm-Metadaten

Kontext:

`bpmn-js` speichert Layout-Informationen (Positionen, Größen, Kurvenrouten) nicht im semantischen Modell, sondern in einer separaten Diagram-Ebene. Diese Daten sind wichtig, da das persistierte Diagramm ohne sie keine visuellen Informationen hätte und daher Änderungen wie das Verschieben von Knoten, wenn das Diagramm später neu geladen wird, nicht mehr vorhanden sind und in der `convertToXML`-Funktion neu berechnet werden müssten.

Die Diagram-Ebene enthält zwei Artefakte:

- BPMNShape: Knoten-Positionen und -Größen
- BPMNEdge: Waypoints der Sequence-Flow-Kurven

Diese werden in zwei Maps für schnelle Lookups extrahiert.

Implementierung:

Die `buildShapeMap()`-Funktion iteriert über alle `BPMNShape`-Elemente und speichert die Bounds unter der Element-ID ab:

Listing 14: Extraktion der Shape-Daten (vereinfacht)

```
1 for (const shape of shapes) {
```

```
    const id = shape.bpmnElement.replace(/^E_/, "");
    const bounds = shape["dc:Bounds"];

    shapeMap[id] = {
      x: parseFloat(bounds.x),
      y: parseFloat(bounds.y),
      width: parseFloat(bounds.width),
      height: parseFloat(bounds.height)
    };
  }
```

`buildEdgeMap()` speichert die Waypoints eines jeden Sequence-Flows mit seiner Flow-ID:

Listing 15: Edge-Extraktion

```
function buildEdgeMap(edgeShapes: any[]): Record<string, {x: number; y: number}[]> {
  const map: Record<string, {x: number; y: number}[]> = {};
  for (const edge of edgeShapes) {
    const id = edge.bpmnElement.replace(/^E_/, "");
    map[id] = normalize(edge["di:waypoint"]).map((wp: any) => ({
      x: parseFloat(wp.x),
      y: parseFloat(wp.y),
    }));
  }
  return map;
}
```

Das Ergebnis sind zwei Maps: `shapeMap` (Element-ID; x, y, width, height und `edgeMap` (Flow-ID; Array von Waypoints). Diese beiden Datenstrukturen enthalten ausschließlich visuelle Informationen des Diagramms und werden in den folgenden Schritten verwendet, um die erzeugten BMD-Knoten mit Layoutinformationen zu ergänzen.

4.4.4.2 Schritt 2: Element-Sammlung und Typannotation

Kontext:

Im BPMN-XML sind Flow-Knoten nach Typ getrennt gespeichert (z.B. `bpmn:task`, `bpmn:exclusiveGateway`). Für die weitere Verarbeitung brauchen wir eine gemeinsame Liste aller Flow-Knoten, wobei der ursprüngliche BPMN-Typ erhalten bleiben muss.

Deswegen wird jeder Knoten mit seinem Tag-Namen (`__tagName`) annotiert, sodass er nach der Konkatenation noch identifizierbar ist.

Implementierung:

Die `withTag()`-Funktion markiert Array-Elemente mit ihrem Typ:

Listing 16: Element-Annotation

```
function withTag(elements: any[], tag: string): any[] {
  return elements.map(el => ({ ...el, __tagName: tag }));
}
```

Die `collectFlowNodeElements()`-Funktion sammelt alle relevanten BMD-Flow-Knoten eines Containers und führt die nach Typ getrennten XML-Elemente zu einer gemeinsamen Liste zusammen. Dabei wird für jedes Element der ursprüngliche BPMN-Typ als zusätzliches Feld (`__tagName`) gespeichert. Dadurch bleibt der Elementtyp auch nach der Zusammenführung eindeutig identifizierbar und kann in den Folgeschritten verwendet werden.

Das Ergebnis ist eine flache Liste aller BPMN-Flow-Knoten. Diese Liste bildet die Grundlage für die hierarchische Verarbeitung im nächsten Schritt.

4.4.4.3 Schritt 3: Hierarchische Knotenverarbeitung

Kontext:

Dieser Schritt ist der Kern des Algorithmus. Hier wird die zuvor erzeugte flache Elementliste in eine hierarchische BMD-Knotenstruktur umgewandelt. Dabei werden mehrere strukturelle Eigenschaften berechnet:

- Laufende Nummer pro Hierarchieebene (LFDNR)
- Hierarchiepfad (PCT) ,z. B. '1.2.3'
- Parent-Beziehungen
- Verschachtelung von Subprozessen

Implementierung:

Die Hauptfunktion `processElements` verarbeitet alle Elemente eines Containers und erzeugt daraus BMD-Knoten. Der Zähler `lfdnrCounter` wird bei jedem Rekursionsaufruf neu initialisiert, sodass Geschwisterknoten innerhalb einer Hierarchieebene fortlaufend nummeriert werden.

Listing 17: Hierarchische Knotenverarbeitung (Auszug)

```
for (const el of elements) {
  const lfdnr = lfdnrCounter++;
  const pct = parentPct ? `${parentPct}.${lfdnr}` : String(lfdnr);

```

```
    const node = {
      MCA_PCK_AG_LFDNR: lfdnr,
      MCU_PCK_HIERARCHIE_PCT: pct,
      MCU_PCK_HIERARCHIE_EBENE: String(depth)
    };

    if (el.__tagName === "bpmn:subProcess") {
      node.children = processElements(
        collectFlowNodeElements(el),
        normalize(el["bpmn:sequenceFlow"]),
        depth + 1,
        pct
      );
    }

    nodes.push(node);
  }
```

Das Ergebnis dieses Schritts ist eine hierarchische BMD-Knotenstruktur, die die Prozesshierarchie sowie die grundlegenden Metadaten der Elemente enthält. Im nächsten Schritt werden aufbauend die expliziten Verbindungen zwischen den Knoten berechnet.

4.4.4.4 Schritt 4: Verbindungs-Ableitung aus Sequence Flows

Kontext:

BPMN und BMD stellen Verbindungen zwischen Elementen unterschiedlich dar:

- BPMN verwendet explizite `<bpmn:sequenceFlow>`-Elemente mit `sourceRef` (Herkunftsknoten) und `targetRef` (Ziel-Knoten)
- In der BMD werden Verbindungen direkt in den Knoten gespeichert, als `NACH_US` (Nachfolger) und `VORG_US` (Vorgänger).

Deshalb müssen die Verbindungen aus den BPMN-Sequence-Flows neu berechnet werden.

Implementierung:

Die Nachfolger eines Elements werden aus den BPMN-`sequenceFlow`-Elementen abgeleitet. Dazu werden alle Flows gefiltert, bei denen das aktuelle Element als Quelle (`sourceRef`) eingetragen ist. Aus diesen Flows werden anschließend die entsprechenden BMD-Verbindungsobjekte erzeugt.

Listing 18: Nachfolger-Ableitung (Auszug)

```
const successors = sequenceFlows
  .filter(sf => sf.sourceRef === el.id)
  .map(sf => ({
    MCA_AGL_AG_PARENT_ID: sourceId,
    MCA_AGL_AG_ID: targetId,
    MCA_AGL_ABHAENGIGKEITSART: dependencyType,
```

```
7       MCA_AGL_FREIFELDA01_250: sf.name ?? "",
8       waypoints: edgeMap[sf.id] ?? []
9   }));
```

Die Funktion `buildPredecessors()` arbeitet analog, filtert aber nach `targetRef` statt `sourceRef`, um eingehende Verbindungen eines Knotens zu bestimmen.

Abhängigkeitstypen:

Das Feld `MCA_AGL_ABHAENGIGKEITSART` signalisiert ob eine Verbindung über eine Subprocess-Grenze geht:

- 0: Quelle und Ziel sind auf gleicher Ebene
- 3: Mindestens ein Element liegt in einem Subprozess

Nach diesem Schritt enthält jeder Knoten sowohl seine Vorgänger (`MCU_PCK_AG_VORG_US`) als auch seine Nachfolger (`MCU_PCK_AG_NACH_US`). Damit ist die vollständige Prozessstruktur inklusive aller Verbindungen rekonstruiert.

4.4.4.5 Schritt 5: Metadaten-Integration

Kontext:

Viele BMD-spezifische Felder sind im BPMN-XML nicht vorhanden. Diese Informationen müssen aus verschiedenen Quellen zusammengeführt werden.

- BPMN-XML (aktuelle Benutzereingaben)
- optionale `responseData` aus der NTCS
- Standardwerte für neu erzeugte Elemente

Typableitungen:

Der BMD-Knotentyp (`MCA_PCK_VERFA_KZ`) wird aus dem BPMN-Elementtyp abgeleitet. Tasks werden als Typ `1` gespeichert, Events und Gateways als Typ `3` und Subprozesse als Typ `4`. Unbekannte oder nicht unterstützte Elemente erhalten keinen zugeordneten Typwert.

Dabei gilt die Priorität: Daten aus dem BPMN-XML haben Vorrang, da sie die aktuellsten Änderungen des Benutzers enthalten.

4.4.5 Besonderheiten und Spezialfälle

Typableitungen bei neuen Elementen

Wenn ein Benutzer ein neues Element im Editor hinzufügt, hat es keine Einträge in responseData. Die `inferVerfaKz()`-Funktion leitet den Typ aus dem BPMN-Tag ab (siehe Schritt 5), wodurch neue Elemente gültige BMD-Typinformation erhalten.

Verbindungs-Ableitung aus Sequence Flows

Während der Bearbeitung können alte BMD-Verbindungen inkonsistent mit der Diagramm-Struktur sein. Statt alten Verbindungen zu vertrauen, werden sie deshalb in Schritt 4 vollständig neu aus den BPMN-Sequence-Flows berechnet. Dies garantiert, dass die gespeicherten Verbindungen exakt der sichtbaren Struktur entsprechen.

Abhängigkeitstypen und Subprocess-Scoping

Wenn eine Verbindung über eine Subprocess-Grenze geht (z.B. von Element A außerhalb zu Element B darin), wird `MCA_AGL_ABHAENGIGKEITSART = 3` gesetzt. Dies signalisiert, dass eine spezielle Behandlung nötig ist.

Hierarchische Nummerierung in Subprozessen

Die PCT (Hierarchie-Pfad) wird nicht aus responseData entnommen, sondern neu kalkuliert, damit es nicht zu verwaisten oder doppelten Pfaden kommt, wenn Subprozesse umgeordnet werden.

Fehlende Positionsdaten

Wenn ein Diagramm frisch in `bpmn-js` erstellt wird (noch nicht gelayoutet), sind shapeMap und edgeMap leer. Die Ausgabe ist trotzdem gültiges BMD-JSON, die x / y / width / height-Felder sind in diesem Fall nicht vorhanden.

4.4.6 Zusammenfassung

Die gesamte Pipeline transformiert BPMN-XML schrittweise in eine BMD-kompatible JSON-Struktur mit hierarchischen Knoten. Diese Struktur wird an das Backend geschickt und persistiert. In der Produktivversion würde die BMD dieses JSON direkt über das eigene Backend speichern.

4.5 Generierung von PNG aus BPMN-Diagramm

In diesem Abschnitt wird der Code vorgestellt, der für die Generierung eines PNG über eine Schnittstelle zuständig ist.

Kontext

In der NTCS gibt es die Funktion, einen Report für jeden Qualitätssicherungsprozess zu erstellen. Die Komplexität dieses Dokuments kann jedoch sehr schnell steigen, was oft zu Problemen von der Verständlichkeit führt. Deshalb wird der Report um eine Funktion erweitert, die es ermöglicht, das BPMN-Diagramm direkt anzuzeigen. Um das zu erreichen, lädt ein Makro in der NTCS ein PNG des BPMN-Diagramms über eine Schnittstelle in das Dokument.

Das Problem dabei ist, dass man für jede Anfrage einen anderen Prozess lädt. Somit muss der Algorithmus zuerst das BPMN-Diagramm aufbauen und anschließend davon ein PNG erzeugen. Es kann jedoch ohne Zeichenfläche keine PNG Generierung stattfinden, deshalb funktioniert ein normaler Export nicht.

Die Lösung

Es wird nun eine neue Schnittstelle über einen JavaScript Express-Server zur Verfügung gestellt. Auf diesem Server läuft mittels Puppeteer (siehe Abschnitt 2.4.7) ein Headless Browser, der die Oberfläche eines normalen Browsers simuliert und somit ein PNG generiert. Dieses PNG wird anschließend an das Makro weitergegeben, das es anschließend im Report anzeigt.

4.5.1 Code

Der folgende Abschnitt beschreibt den Code zur Konvertierung des BPMN-Diagramms in ein PNG-Bild.

Start des Headless Browsers

Repräsentiert den Code für den Start des Headless-Browsers

Methode: `getBrowser()`

Returns: `Promise<puppeteer.Browser>`: Ein Promise auf den Headless-Browser von Puppeteer (siehe Abschnitt 2.4.7)

Definition: Da es langsam und ineffizient ist, für jede Anfrage einen komplett neuen Headless Browser mit Puppeteer aufzubauen, wird dieser beim ersten Aufruf dieser Methode in einer In-Memory-Variable zwischengespeichert. Zukünftige Anfragen verwenden anschließend diese Instanz des Headless Browsers.

Listing 19: Aufbau des geteilten Headless Browsers

```
1  // This variable will hold the shared browser instance.
2  let browserInstance: puppeteer.Browser | null = null;
3  /**
4   * Initializes and returns a shared Puppeteer browser instance.
5   * If an instance already exists, it returns the existing one.
6   * @returns {Promise<Browser>} A promise that resolves to a Puppeteer Browser instance.
7   */
8  export async function getBrowser(): Promise<puppeteer.Browser> {
9    if (!browserInstance) {
10     console.log("Launching new browser instance...");
11     browserInstance = await puppeteer.launch({
12       // Recommended args for running in a server/docker environment
13       args: ["--no-sandbox", "--disable-setuid-sandbox"],
14     });
15   }
16   return browserInstance;
17 }
```

Falls noch keine Instanz der `browserInstance` existiert, wird in den Zeilen 11-14 der Headless-Browser mit der Methode `puppeteer.launch()` von puppeteer gestartet. Dieser Methode beinhaltet verschiedene Startoptionen die dessen Funktionalität leicht verändern. Für dieses Projekt ist die wichtigste Option `args`. Mit dieser Option können verschiedene Terminal-Kommandos an den Start-Prozess weitergegeben werden. Beispielsweise erfolgt mit der `-disable-setuid-sandbox`-Option die Deaktivierung aller Sandbox-Schichten. Das hat zur Folge dass auch auf Linux und somit auch auf simple Docker-Images der Headless Browser funktioniert, da diese keine Sandbox unterstützen.

Initialisierung des Headless Browsers

Im folgenden Abschnitt erfolgt die Erklärung, wie die Bibliothek `bpmn-js` in den Headless Browser eingebettet wird.

Listing 20: Headless Browser Initialisierung

```
1       // Load the bpmn-js viewer library into the page.
2       // require.resolve finds the path to the installed module file.
3       const bpmnJsScript = await fs.readFile(
4         require.resolve("bpmn-js/dist/bpmn-viewer.development.js"),
5         "utf-8"
6       );
7       // Set the HTML content of the page, injecting the viewer script.
8       const htmlContent = `<!DOCTYPE html><html><head><title>BPMN
            Renderer</title></head><body><div id="container" style="display:
            inline-block;"></div><script>${bpmnJsScript}</script></body></html>`;
```

Das Importieren und Visualisieren des BPMN-Diagramms erfolgt über `bpmn-js` (siehe Abschnitt 2.4.2). Diese Bibliothek kann jedoch nicht einfach mit `import` geladen werden, da auf der von Puppeteer generierten Browserinstanz das Node-Environment nicht laufen kann. Deshalb ist der Quellcode der Bibliothek direkt in die HTML-Webseite, mittels eines `<script>`-Tags, einzubetten.

Der Quellcode wird folgendermaßen geladen: Zuerst erfolgt der Aufruf der Funktion `require.resolve()`. Im Gegensatz zu `require()` lädt diese nicht den kompilierten Code direkt in den Laufzeitkontext, sondern löst den absoluten Dateipfad des Node-Modules auf und gibt diesen zurück.

Im nächsten Schritt wird die Datei als Text eingelesen und in den initialen HTML Code eingebettet.

Methodendefinition der Konversion

Methode: `bpmnXMLtoBase64()`

Parameter: `bpmnXML: string` — Dieser Parameter dient zur Übergabe des anzuzeigende Diagramm in XML-Form an die Methode

Returns: `Promise<string>`: — Ein Promise auf einen String der das Base64-enkodierte Bild des BPMN-Diagramms beinhaltet.

Definition: Konvertiert ein BPMN-Diagramm in ein Base64 enkodiertes PNG-Bild. Im ersten Schritt baut die geteilte Instanz des Headless-Browser eine neue Seite auf. Diese zeigt mittels `bpmn-js` (siehe Abschnitt: 2.4.2) das im Parameter `bpmnXML` enthaltene BPMN-XML an. Von diesem Diagramm generiert der Algorithmus dann den Screenshot.

Initialisierung des Headless Browsers

Im folgenden Codeabschnitt ist ein Ausschnitt der `bpmnXMLtoBase64()`-Methode ersichtlich, in der eine neue Seite des Headless Browsers aufgerufen und das BPMN-Skript importiert wird.

Listing 21: Headless Browser Initialisierung

```
1       // load
2       const browser = await getBrowser();
3       page = await browser.newPage();
4       await page.setViewport({ width: 1200, height: 800});
5
6       await page.setContent(htmlContent);
7
8       await page.evaluate(async (xml: string) => {
9         // BpmnJS is available on the window object thanks to the script injection.
10        const viewer = new (window as any).BpmnJS({ container: "#container" });
11        try {
12          await viewer.importXML(xml);
13          // 'fit-viewport' zooms the canvas to fit the diagram.
14          viewer.get("canvas").zoom("fit-viewport");
15        } catch (err: any) {
16          // This error will be thrown in the browser context.
17          throw new Error(`Failed to import BPMN XML: ${err.message}`);
18        }
19      }, bpmnXML);
```

Die Codezeilen 2 bis 3 wird laden den Browser (siehe Abschnitt 4.5.1) und erstellen für jede Anfrage eine neue Seite. Die Funktion `page.evaluate(method, input)` führt einen Befehl innerhalb der Seite aus. Darin wird mithilfe des globalen Objekts `window` ein neuer BPMN-Viewer im HTML-Element mit der ID `container` erzeugt und das valide BPMN-XML mittels des BPMN-Viewers als BPMN-Diagramm dargestellt.

Bildkonvertierung

Im folgenden Abschnitt ist ein Ausschnitt der `bpmnXMLtoBase64()`-Methode ersichtlich. Diese konvertiert, mittels der im Abschnitt 4.5.1 initialisierten Browserseite, das BPMN-Diagramm in einer Base64-kodierte `string` Variable.

Listing 22: Konvertierung des Diagramms in ein Bild

```
1       const imageBuffer = await diagramContainer.screenshot({
2         encoding: "binary",
3         omitBackground: true,
4       });
5
6       // Convert the binary image buffer to a Base64 data URI.
7       const base64String = Buffer.from(imageBuffer).toString("base64");
8       return base64String;
```

Für eine optimale Darstellung des generierten PNGs erfolgen vor der Erreichung dieses Codeabschnitt in der Methode `bpmnXMLtoBase64()` einige Anpassungen am Styling des Browsers.

In den Zeilen 1-4 nimmt Puppeteer einen Screenshot auf, der dann in der Zeile 7 zu einem Base64-kodierten String umgewandelt und zurückgegeben wird. Nachdem alles abgeschlossen ist, wird das Browserfenster wieder geschlossen.

4.6 Java REST-Endpunkte

Das Backend des Systems ist als REST-basierter Webservice implementiert und stellt eine Schnittstelle bereit, über die das Frontend des BPMN-Editors mit der Datenbank kommunizieren kann. Über diese können BPMN-Elemente gespeichert, aktualisiert sowie aus der Datenbank geladen werden. Die grundlegenden Konzepte einer REST-Architektur wurden bereits in Abschnitt 2.2.4 erläutert.

Die Implementierung der REST-Schnittstelle erfolgt mithilfe der Jakarta REST API (JAX-RS). Die Anwendung wird auf einem Open-Liberty-Server ausgeführt, der als Laufzeitumgebung für das Backend dient (vgl. Abschnitt 2.2.6). Die REST-Endpunkte sind unter dem Basispfad `/rest` erreichbar. Das Frontend kommuniziert mit diesen Endpunkten über HTTP-Anfragen, wobei die übertragenen Daten im JSON-Format serialisiert werden.

Für die Persistenz der Daten wird die Java Persistence API (JPA) eingesetzt (vgl. Abschnitt 2.2.7). JPA ermöglicht den Zugriff auf die Datenbank über objektorientierte Entitäten, wodurch Datenbankoperationen direkt auf Java-Objekten ausgeführt werden können. Als Implementierung der ORM-Funktionalität wird das Framework Hibernate verwendet (vgl. Abschnitt 2.2.5). Die BPMN-Daten werden in einer SQLite-Datenbank gespeichert (siehe Abschnitt 2.2.3).

Der Datenbankzugriff erfolgt über ein Repository, das als Zwischenschicht zwischen den REST-Endpunkten und der Persistenzschicht fungiert. Dieses Repository kapselt die Datenbankoperationen und sorgt dafür, dass die REST-Ressourcen unabhängig von der konkreten Implementierung der Datenbankzugriffe bleiben.

Das zentrale Datenmodell des Backends bildet die Klasse `BpmnNode`. Diese Klasse repräsentiert ein BPMN-Element und enthält sowohl fachliche Attribute des Geschäftsprozesses als auch Layoutinformationen, beispielsweise die Position und Größe eines Elements im Diagramm (`x`, `y`, `width`, `height`).

4.6.1 Übersicht der REST-Schnittstelle

Die wichtigsten Endpunkte der REST-Schnittstelle sind in der Tabelle 10 dargestellt.

Methode	Endpunkt	Beschreibung
GET	`/rest/`	Liefert alle gespeicherten `BpmnNode`-Objekte
POST	`/rest/`	Speichert einen neuen `BpmnNode` in der Datenbank
PUT	`/rest/`	Aktualisiert bestehende oder erstellt neue `BpmnNode`-Objekte

Tabelle 10: Übersicht der REST-Endpunkte

4.6.2 Beschreibung der einzelnen Endpunkte

GET /rest/

Dieser Endpunkt stellt eine Liste aller gespeicherten BPMN-Knoten aus der Datenbank bereit.

HTTP-Methode: GET

Rückgabewert: `List<BpmnNode>`

Beschreibung: Die Schnittstelle dient dazu, alle vorhandenen BPMN-Elemente aus der Datenbank zu laden. Die Daten werden als JSON-Array an das Frontend zurückgegeben und anschließend im BPMN-Editor visualisiert.

POST /rest/

Dieser Endpunkt dient zum Speichern eines neuen BPMN-Knotens.

HTTP-Methode: POST

Eingabewert: `BpmnNode` im JSON-Format.

Beschreibung: Das übergebene JSON-Objekt wird im Backend in ein `BpmnNode`-Objekt umgewandelt und anschließend über das Repository in der Datenbank gespeichert.

Ein möglicher Input kann wie folgt aussehen:

Listing 23: Beispiel eines POST-Requests für einen BPMN-Knoten

```
{
    "MCA_PCK_PPSKAPAID": 1001,
    "MCA_APA_AP_BEZ": "Start Task",
    "MCA_APA_AP_TYP": 1,
    "MCA_PCK_AG_BEZ": "Demo Step",
    "MCU_PCK_AP_BEZ": "Demo BPMN Node",
    "x": 120,
    "y": 80,
    "width": 160,
    "height": 90
}
```

PUT /rest/

Dieser Endpunkt dient zum Aktualisieren bestehender BPMN-Knoten.

HTTP-Methode: PUT

Eingabewert: `BpmnNode` im JSON-Format

Beschreibung: Der Endpunkt verarbeitet eine Liste von BPMN-Knoten und führt für jedes Element eine `Upsert`-Operation aus. Eine Upsert-Operation kombiniert die Operationen `Update` und `Insert`. Existiert ein Datensatz mit der entsprechenden ID bereits in der Datenbank, wird dieser aktualisiert. Falls kein entsprechender Datensatz vorhanden ist, wird ein neuer Datensatz erstellt. Dadurch können mehrere BPMN-Elemente effizient in einem einzigen Request aktualisiert oder neu gespeichert werden.

4.7 Testing

Zur Qualitätssicherung wurden nachträglich diverse automatisierte Tests mit dem Framework `Vitest` (siehe Abschnitt 2.4.5) geschrieben. Dieses Kapitel befasst sich mit drei konkreten Tests, nämlich zwei Unittests und einem Integrationstest.

4.7.1 Unit-Tests

Für eine Erklärung von Unittests siehe Abschnitt 2.1.2.1.

Unit-Test 1: Linearer Ablauf - convertToBpmnXml

Der erste Unit-Test `convertToBpmnXml` überprüft die Funktion `convertToXML` (näher beschrieben im Abschnitt 4.3). Diese wandelt strukturierte BMD-Projektdaten in valides BPMN-XML um, das anschließend vom `bpmn-js`-Modeler gerendert werden kann. Ein Fehler in dieser Funktion würde dazu führen, dass entweder kein Diagramm erzeugt wird oder Knoten und Verbindungen falsch dargestellt werden. Der Unit-Test dient deshalb der Prüfung, ob die Funktion für einen minimalen linearen Ablauf die korrekten BPMN-Elementtypen und Sequenzfluss-IDs erzeugt. Als Eingabe dienen drei Objekte: ein Startknoten (ID 100, Typ Milestone), eine Aufgabe (ID 101, Typ Task) und ein Endknoten (102, Typ Milestone). Diese sind über das Feld `MCU_PCK_AG_NACH_US` verknüpft:

Listing 24: Unit-Test: Eingabe für convertToBpmnXml (Auszug)

```
const nodes: BmdBpmnNode[] = [
{
    MCA_PCK_PPSKAPAID: 100,
    MCA_PCK_VERFA_KZ: BpmnNodeType.Milestone,
    MCA_PCK_GATEWAYTYP: BpmnGatewayTypes.Milestone,
    MCA_PCK_AG_LFDNR: 1,
    MCU_PCK_HIERARCHIE_PCT: "1",
    MCU_PCK_HIERARCHIE_EBENE: "0",
    MCA_PCK_AG_BEZ: "Start",
    MCU_PCK_AG_NACH_US: [link(100, 101)],
},
...
];

    const result = await convertToBpmnXml(nodes, false);
```

Nach dem Aufruf wird der zurückgegebene XML-String auf die Existenz der korrekten BPMN-Elemente überprüft:

Listing 25: Unit-Test: Existenzfeststellung der Elemente

```
expect(result.xml).toContain("<bpmn:startEvent");
expect(result.xml).toContain("<bpmn:task");
expect(result.xml).toContain("<bpmn:endEvent");
expect(result.xml).toContain("Flow_E_100_E_101");
expect(result.xml).toContain("Flow_E_101_E_102");
```

Der Test stellt damit sicher, dass `convertToBpmnXml` den Knotentyp `Milestone` am Anfang eines Ablaufs korrekt als `StartEvent` und am Ende als `endEvent` erkennt - eine Logik, die anhand der Position des Knotens im Ablauf entschieden wird. Zusätzlich wird geprüft, dass Sequenzflüsse nach dem Schema `Flow_E_{source}_E_{target}` korrekt benannt werden.

Unit-Test 2: Verbindungen - bpmnXmlToJson

Der zweite Unit-Test `bpmnXmlToJson` liest gespeichertes BPMN-XML zurück ein und rekonstruiert daraus die strukturierte JSON-Darstellung, die für die Persistenz im BMD-System benötigt wird. Dabei müssen insbesondere die Vorgänger-Nachfolger-Beziehungen zwischen den Knoten korrekt aus den `sequenceFlow`-Elementen des XMLs extrahiert werden.. Fehler hier würden zu Verlusten beim Speichern von Verbindungen oder dem falschen Zuordnen dieser führen. Der Unit-Test prüft deshalb gezielt, ob diese Verbindungen nach dem Parsen korrekt in den Feldern `MCU_PCK_AG_VORG_US` und `MCU_PCK_AG_NACH_US` vorliegen. Da `bpmnXmlToJson` als Eingabe BPMN-XML erwartet, wird das Testdokument als XML-String direkt im Test definiert:

Listing 26: Unit-Test: Statisches Eingabe-XML

```
<bpmn:process id="Process_1" isExecutable="false">
    <bpmn:startEvent id="E_100" name="Start"/>
    <bpmn:task id="E_101" name="Task A"/>
    <bpmn:endEvent id="E_102" name="Ende"/>
    <bpmn:sequenceFlow id="Flow_100_101" sourceRef="E_100" targetRef="E_101"/>
    <bpmn:sequenceFlow id="Flow_101_102" sourceRef="E_101" targetRef="E_102"/>
</bpmn:process>
```

Nach dem Aufruf von `bpmnXmlToJson` werden die Verbindungen des mittleren Knotens (Task A, ID 101) überprüft:

Listing 27: Unit-Test: Überprüfen der Verbindungen

```
// Task A should have Start (100) as predecessor
expect(task!.MCU_PCK_AG_VORG_US).toHaveLength(1);
expect(task!.MCU_PCK_AG_VORG_US![0].MCA_AGL_AG_PARENT_ID).toBe(100);
expect(task!.MCU_PCK_AG_VORG_US![0].MCA_AGL_AG_ID).toBe(101);

// Task A should have Ende (102) as successor
expect(task!.MCU_PCK_AG_NACH_US).toHaveLength(1);
expect(task!.MCU_PCK_AG_NACH_US![0].MCA_AGL_AG_PARENT_ID).toBe(101);
expect(task!.MCU_PCK_AG_NACH_US![0].MCA_AGL_AG_ID).toBe(102);
```

Damit wird die korrekte Funktionsweise der internen Hilfsfunktionen `buildSuccessors` und `buildPredecessors` (siehe Abschnitt 4.4) überprüft, die für das Auslesen der Sequenzflüsse aus dem geparsten XML verantwortlich sind.

4.7.2 Integrationstest

Für eine Erklärung von Integrationstests siehe Abschnitt 2.1.2.2.

Integrationstest: Rundtrip - convertToBpmnXml -> bpmnXmlToJson

Diese beiden Unit-Test prüfen zwar jede Funktion an sich, können jedoch nicht ausschließen, dass Fehler an der Schnittstelle zwischen beiden Modulen auftreten. Ein konkretes Beispiel: `bpmnXmlToJson` entfernt intern das Präfix `E_` von Element-IDs (`convertToJSON`, `el.id.replace(/^E_/, )`). Erzeugt `convertToBpmnXml` dieses Präfix nicht konsistent, würden die IDs nach dem Parsen nicht mehr mit den ursprünglichen BMD-IDs übereinstimmen und Verbindungen würden verloren gehen. Der Integrationstest prüft genau dieses Zusammenspiel, indem er beide Funktionen nacheinander aufruft. Als Eingabe dienen wieder drei BMD-Knoten (IDs 1000, 1001, 1002) mit definierten Vorgänger-Nachfolger–Beziehungen. Das Ergebnis von `convertToBpmnXml` wird direkt als Eingabe an `bpmnXmlToJson` übergeben:

Listing 28: Integrationstest: Rundtrip-Aufruf

```
1       const { xml } = await convertToBpmnXml(nodes, false);
2       const json = await bpmnXmlToJson(xml, nodes as any[]);
```

Dann folgt die Validierung, ob die semantischen Verbindungen den gesamten Prozess unbeschadet überstehen:

Listing 29: Integrationstest: Kontrolle der Verbindungen

```
1  expect(json.nodes.length).toBeGreaterThanOrEqual(3);
2  expect(
3          json.nodes.some((n) => n.MCA_PCK_PPSKAPAID === 1000 &&
               n.MCU_PCK_AG_NACH_US?.some((s) => s.MCA_AGL_AG_ID === 1001))
4       ).toBe(true);
5  expect(
6          json.nodes.some((n) => n.MCA_PCK_PPSKAPAID === 1001 &&
               n.MCU_PCK_AG_NACH_US?.some((s) => s.MCA_AGL_AG_ID === 1002))
7       ).toBe(true);
```

Der Test besteht wenn alle drei Knoten im Ergebnis vorhanden und die Nachfolgerbeziehungen 1000 -> 1001 und 1001 -> 1002 korrekt erhalten geblieben sind.

5 Ergebnisse

Dieses Kapitel stellt die Ergebnisse der entwickelten Anwendung dar. Es folgt die Betrachtung des implementierten Editors und des entwickelten Algorithmus. Zum Schluss wird ein Vergleich zwischen den Modellierungsmethoden EPK und BPMN, inklusive der Beispielmodellierung realer Geschäftsprozesse, vorgenommen.

5.1 Benutzeroberfläche und Interaktion im BPMN-Editor

Dieses Kapitel beschreibt das Ergebnis der implementierten Editorfunktionalität aus Sicht der Benutzeroberfläche. Während Abschnitt 4.2 die technische Umsetzung des BPMN-Editors erläutert, wird hier dargestellt, wie der Editor in der Anwendung erscheint und welche Interaktionsmöglichkeiten dem Benutzer zur Verfügung stehen.

Die entwickelte Bearbeitungsoberfläche ermöglicht die Visualisierung, Erstellung und Bearbeitung von BPMN-Diagrammen innerhalb der Webanwendung. Neben der eigentlichen Modellierungsfläche stehen verschiedene Werkzeuge zur Verfügung, die eine intuitive Interaktion mit dem Prozessdiagramm ermöglichen. Im Zuge dieser Arbeit entstand zudem ein Benutzerhandbuch, welches die Navigation in der Oberfläche und mögliche Aktionen dokumentiert (siehe Anhang E). Somit ist die Anwendung für neue sowie erfahrene Benutzer leichter verständlich und man kann sich schneller in die Software einarbeiten.

5.1.1 Aufbau der Benutzeroberfläche

Die Benutzeroberfläche des BPMN-Editors ist in mehrere funktionale Bereiche gegliedert. Die zentrale Komponente bildet die Modellierungsfläche (Canvas), auf der das BPMN-Diagramm

dargestellt und bearbeitet werden kann. Ergänzend dazu stehen mehrere Bedienelemente zur Verfügung, die den Modellierungsprozess unterstützen.

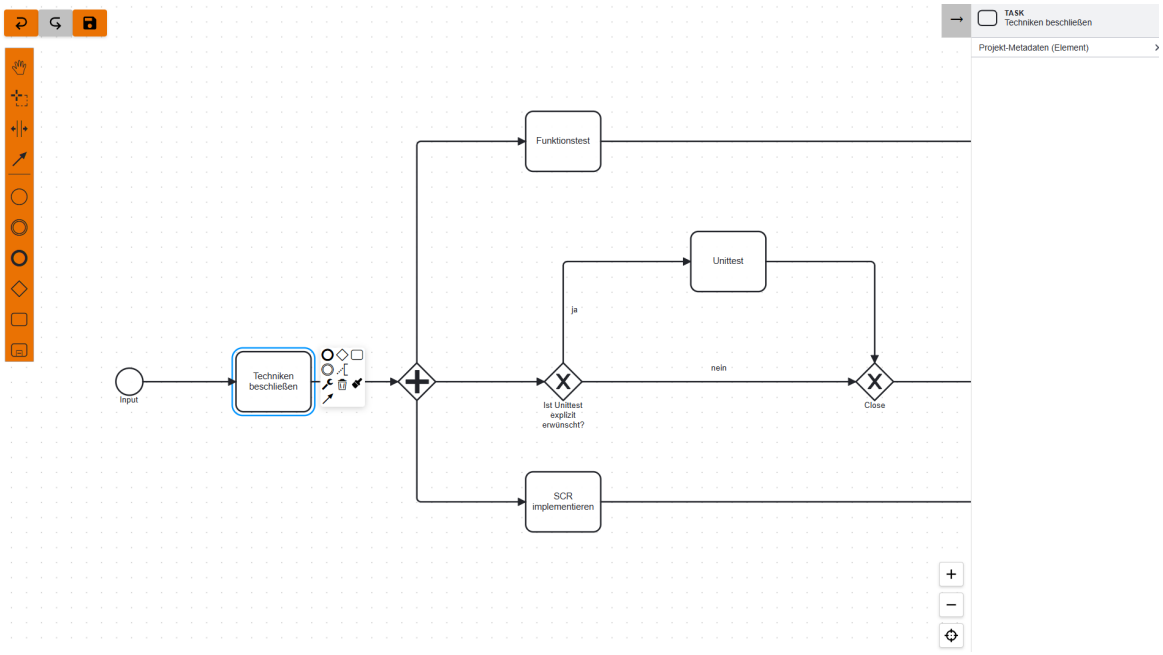


Abbildung 35: Benutzeroberfläche des BPMN-Editors

Wie in Abbildung 35 dargestellt, besteht die Oberfläche aus den folgenden Hauptbereichen:

- **Modellierungsfläche (Canvas)** Die zentrale Zeichenfläche ist für die Darstellung und Bearbeitung des BPMN-Diagramms. Hier kann man Elemente positionieren, verschieben und miteinander verbinden.
- **Palette der Modellierungselemente** Auf der linken Seite befindet sich eine Werkzeugpalette mit den verfügbaren BPMN-Elementen. Diese können ausgewählt und per Drag-and-Drop auf der Zeichenfläche platziert werden.
- **Toolbars** Im oberen Bereich befindet sich eine Werkzeugleiste mit zentralen Editorfunktionen, wie beispielsweise Speichern und Undo/Redo. Im unteren Bereich findet man die Zoom-Steuerung.
- **Eigenschaften- und Metadatenpanel** Auf der rechten Seite befindet sich ein Seitenpanel, in dem Eigenschaften und projektspezifische Metadaten des aktuell ausgewählten BPMN-Elements angezeigt und teilweise bearbeitet werden können.

5.1.2 Erstellung und Bearbeitung von BPMN-Diagrammen

Die Modellierung eines Prozesses erfolgt direkt auf der Zeichenfläche des Editors. BPMN-Elemente können über die Palette (siehe Abbildung 36) ausgewählt und per Drag-and-Drop im Diagramm platziert werden. Nach dem Platzieren können Elemente verschoben, verbunden oder gelöscht werden.
Verbindungen zwischen einzelnen Prozessschritten werden automatisch als BPMN-Sequenzflüsse dargestellt. Dadurch kann der Ablauf eines Geschäftsprozesses visuell modelliert werden. Zusätzlich unterstützt der Editor grundlegende Interaktionsmechanismen wie Auswahlrahmen, Verschieben von Elementen und das automatische Anpassen von Verbindungen.

Abbildung 36: Werkzeugpalette

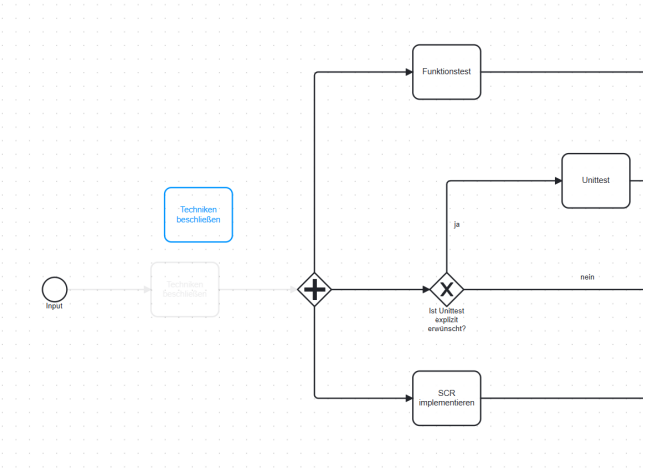


Abbildung 37: Bearbeitung eines BPMN-Diagramms durch Verschieben von Elementen

5.1.3 Bearbeitung von Eigenschaften und Metadaten

Für jedes BPMN-Element können im rechten Seitenpanel (siehe Abbildung 38) zusätzliche Informationen angezeigt werden. Dazu gehören sowohl BPMN-Eigenschaften als auch projektspezifische Metadaten, welche aus der NTCS mit dem Prozess selbst geholt werden.
Wenn ein Element im Diagramm ausgewählt wird, werden im Eigenschaftenpanel die zugehörigen Informationen angezeigt. Dazu zählen beispielsweise Bezeichnungen, Beschreibungen oder systemseitige Identifikatoren. Einige dieser Felder können direkt im Editor bearbeitet werden, während andere ausschließlich zur Anzeige von aus externen Systemen übernommenen Daten dienen.
Ist kein BPMN-Element ausgewählt, werden im Panel stattdessen allgemeine Metadaten des gesamten Prozesses dargestellt.

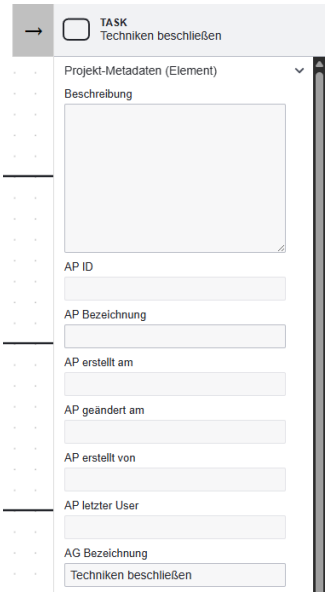


Abbildung 38: Eigenschaften- und Metadatenpanel

5.1.4 Navigation und Editorfunktionen

Neben der eigentlichen Modellierung stellt der Editor verschiedene Funktionen zur Navigation und Bearbeitung bereit.
Zu den wichtigsten Funktionen gehören:

- **Undo/Redo** zur Rücknahme oder Wiederherstellung von Modellierungsänderungen
- **Zoom-Funktionen** zur Anpassung der Darstellung des Diagramms
- **Verschieben der Zeichenfläche** zur Navigation innerhalb größerer Diagramme

Abbildung 39: Toolbar

Die Toolbar stellt zentrale Steuerungsfunktionen des Editors bereit, wie beispielsweise das Speichern von Änderungen sowie das Rückgängig machen oder Wiederherstellen von Bearbeitungsschritten.

Zusätzlich stehen Funktionen zur Anpassung der Zoom-Stufe zur Verfügung, wodurch auch größere Prozessdiagramme komfortabel betrachtet werden können.

Abbildung 40: Zoom-Steuerung

5.1.5 Export von Diagrammen

Nach der Bearbeitung kann das erstellte BPMN-Diagramm aus der Anwendung exportiert werden. Der Editor unterstützt verschiedene Exportformate, darunter BPMN-XML sowie bildbasierte Formate (siehe Listing 22).

Der Export ermöglicht es, das Diagramm außerhalb der Anwendung weiterzuverwenden, beispielsweise zur Dokumentation von Geschäftsprozessen oder zur Integration in andere Systeme (siehe Abschnitt 4.5).

5.2 Algorithmus zur BPMN-Generierung

Dieser Abschnitt zeigt die möglichen Ausgaben, die der BPMN-Umwandlungsalgorithmus erzeugen kann. Im Vergleich zu Abschnitt 4.3 werden hier Beispiele der erzeugten Diagramme dargestellt.

5.2.1 Ausgabe / generierte BPMN-Diagramme

Die generierten Diagramme sind in zwei Darstellungsformen gegliedert:

- vertikale Darstellung
- horizontale Darstellung

Die vertikale Darstellung des Diagramms ist in der Abbildung 41 ersichtlich. Diese wurde von der BMD als zusätzliche Option gewünscht, da vertikale Diagramme in Berichten übersichtlicher dargestellt werden können.

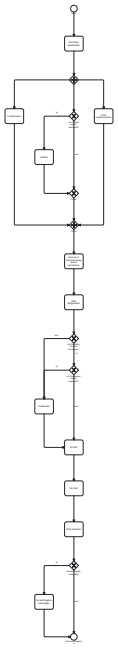


Abbildung 41: Vertikale BPMN-Darstellung

Im Gegensatz dazu zeigt die Abbildung 42 die horizontale Darstellung. Hierbei ist in der Weboberfläche zwar mehr vom Diagramm sichtbar, diese Darstellungsart ist jedoch für den Bericht weniger geeignet.

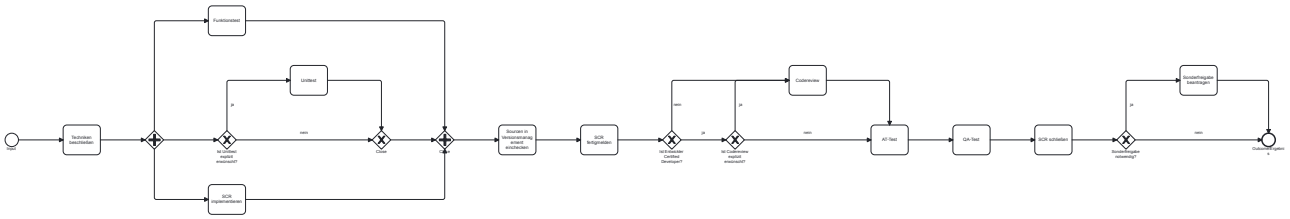


Abbildung 42: Horizontale BPMN-Darstellung

Die generierten Diagramme entsprechen der erwarteten BPMN-Struktur und zeigen, dass der entwickelte Algorithmus die Eingabedaten korrekt in ein visuell strukturiertes BPMN-Diagramm überführen kann. Sowohl die horizontale als auch die vertikale Darstellung ermöglichen eine übersichtliche Visualisierung des Geschäftsprozesses für unterschiedliche Use Cases.

5.3 Endpunkt für die Erzeugung des PNG-Bild

Dieses Kapitel zeigt den Report, der durch die PNG-Export Schnittstelle, ein Bild eines Diagramms beinhaltet.

5.3.1 Schnittstelle

Es steht für alle Makros der folgende Pfad zur Verfügung, um jederzeit ein PNG von einem Diagramm zu erzeugen.

- Pfad: /api/export/:projectNr

Die `:projectNr` repräsentiert dabei die Id des zu ladenden Diagramms.

5.3.2 Report

Die Abbildung 43 zeigt einen Report, der ein BPMN-Diagramm beinhaltet. Um die Erzeugung des Dokuments und das Einfügen des Bildes ist die interne Logik der NTCS zuständig.

Abbildung 43: Bild eines Reports in der NTCS

5.4 Vergleich BPMN und EPK

In diesem Kapitel werden ausgewählte Geschäftsprozesse jeweils in EPK- und BPMN-Notation modelliert und vergleichend analysiert. Das Ziel ist es, strukturelle Unterschiede, Unterschiede im Formalisierungsgrad sowie praktische Einsatzmöglichkeiten beider Modellierungssprachen herauszuarbeiten. Für den Vergleich werden zwei verschiedene Prozesse betrachtet, wobei jeder Prozess zum Vergleich als EPK Diagramm als auch als BPMN Diagramm modelliert wird.

5.4.1 Prozess 1: SCR durchführen

Der erste Prozess ist SSCR durchführen". Dieser Prozess wurde uns von der BMD während des Entwicklungsprozesses zur Verfügung gestellt und dient der systematischen Prüfung von Quellcode zur Sicherstellung von Qualität, Wartbarkeit und Einhaltung von Entwicklungsrichtlinien. Der Prozess ist bereits in der Software der BMD vorhanden und dient der vereinfachten Darstellung eines Source-Code-Review-Prozesses.

5.4.1.1 EPK-Modell

Abbildung 44 zeigt den Prozess „SCR durchführen“ in EPK-Notation.

Abbildung 44: Prozess „SCR durchführen“ in EPK-Notation

Das Erste, was bei der Modellierung des EPK-Diagramms auffällt, ist, dass auf jede Aktivität ein Ereignis folgt, welches den Ausgang der Aktivität beschreibt. Dieser Ablauf ist ein Grundprinzip des EPK-Modells.[59]

[59] Vgl. GBTEC, *EPK-Notationselemente*, S. 69.

5.4.1.2 BPMN-Modell

Abbildung 45 zeigt denselben Prozess in der BPMN-Notation.

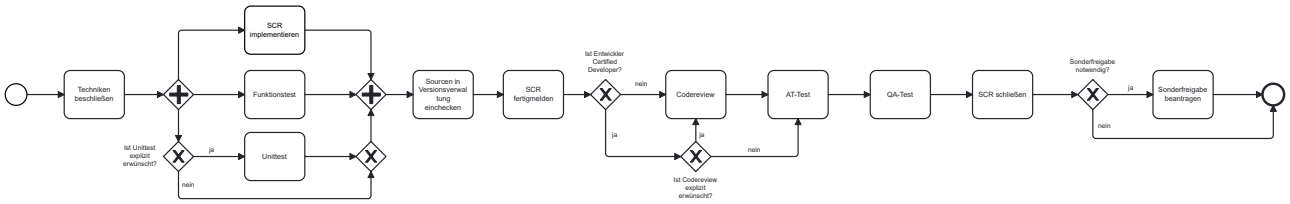


Abbildung 45: Prozess „SCR durchführen“ in BPMN-Notation

Im Unterschied zur EPK werden in der BPMN die einzelnen Aufgaben nicht von Prozessen gesteuert.

5.4.1.3 Vergleich und Bewertung des Prozesses „SCR durchführen“

Kriterien

Der Vergleich und die Bewertung des ”SCR durchführen” Prozesses berücksichtigt folgende Kriterien:

- Verschiedene Funktionalitäten der Notationen
- Lesbarkeit des Prozesses

Vergleich

Für den obigen Prozess zeigt sich, dass zwar beide Modellierungssprachen geeignet sind um Qualitätsmanagementprozesse darzustellen, sie jedoch unterschiedliche Vorteile und Nachteile mit sich bringen.

Durch das Einbringen von Ereignissen nach jeder Aktivität wird ein EPK-Diagramm schnell unübersichtlich, weil jede Aktivität zusätzlich ein weiteres Ereignis mit sich bringt. Dadurch wirkt der Prozess komplexer, als er tatsächlich ist. Außerdem wächst das EPK-Diagramm doppelt so schnell.

Die Ereignisse bringen jedoch auch Vorteile mit sich: Sie machen die Ausgänge für den Leser leicht ersichtlich. Zusätzlich ermöglicht die EPK Notation mit Organisationseinheiten die Zuteilung spezifischer Rollen und Personen zu einzelne Aufgaben.

Für die Gateways/Operatoren wenden beide Methoden ähnliche Prinzipien an. Jedoch ist die Art, wie die Entscheidungen dargestellt werden, unterschiedlich. Die BPMN-Notation beinhaltet die Frage, was eine Verzweigung besagt, direkt im Gateway und die möglichen Ausgänge ist direkt als Beschriftung in jedem Pfad abgebildet.

Im Gegensatz dazu sind Verzweigungen in der EPK-Notation implizit, das heißt, dass keine spezifische Frage im Operator angegeben wird. Die möglichen Ausgänge werden wie folgt behandelt. In jedem Pfad muss zu Beginn ein Ereignis stehen. Dieses Ereignis beinhaltet die Definition, wann dieser Pfad aufgerufen wird.

Bewertung

Die BMD priorisiert eine einfache Darstellung , die jeder leicht verstehen kann, über eine Umfangreichere. Während der Ablauf eines Prozesses in BPMN ohne zusätzliche Komplexität dargestellt werden kann, fügt EPK mit Ereignissen weitere Elemente hinzu, die die Lesbarkeit eines Diagramms negativ beeinflussen.

Bei der Zuteilung von Rollen und Personen zu Aufgaben, übertrifft EPK BPMN. Organisationseinheiten sind leicht zu benutzen und übersichtlich. Jedoch besitzt auch BPMN die Fähigkeit Rollen mit Swimmlanes zu verteilen. Deshalb ist dieser Aspekt nicht so relevant wie andere.

Bei dem Punkt Verzweigung erweist sich die BPMN-Notation, wenn sie fachgerecht benutzt wird, als deutlich übersichtlicher. Da die Entscheidungslogik meist mit einer expliziten Frage direkt im Gateway belegt wird und die einzelnen Ausgänge direkt am Pfad ersichtlich sind, wird der Prozess für den Leser leichter verständlich.

5.4.2 Prozess 2: ERP-Software-Einführung

Dieser Prozess dient der Qualitätssicherung und als Leitfaden für die Einführung des ERP Systems bei einem Kunden.

Der Prozess wurde unserem Projektteam während des Entwicklungsprozesses zur Verfügung gestellt und dient, genau wie der Prozess SSCR durchführen"(siehe 5.4.1), zur Kontrolle der Korrektheit des Quellcodes. Er erweitert die bestehende systematische Prüfung des Quellcodes um die korrekte Verarbeitung von Subprozessen und verschachtelten Subprozessen und dient somit als Referenz um sicherzustellen, dass auch hierarchisch komplexe BPMN-Diagramme vollständig und fehlerfrei dargestellt werden können.

5.4.2.1 EPK-Modell

Die Abbildung 46 zeigt, wie man in EPK den ERP-Software-Einführungsprozess darstellt. Subprozesse werden hierbei als Prozessschnittstellen modelliert. Jede Prozessschnittstelle verlinkt ein anderes EPK Diagramm, das den Inhalt des Subprozesses beinhaltet. Dabei beinhaltet die Abbildung 47 den Inhalt des Subprozesses "Datenaufbereitung und -übernahme". Dieser enthält wiederum einen geschachtelten Subprozess, der in der Abbildung 48 dargestellt wird.

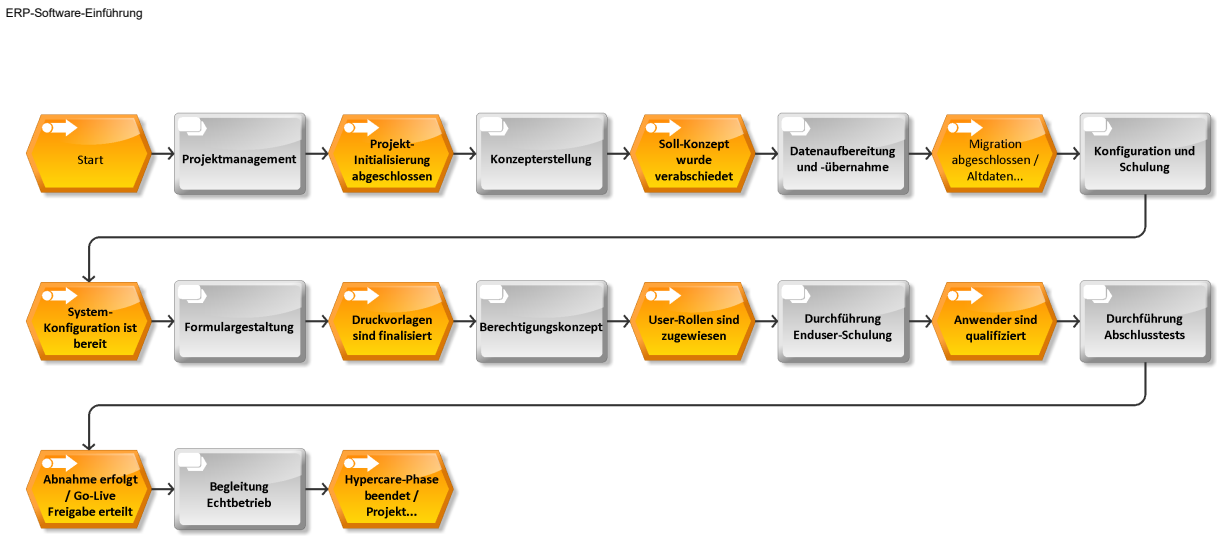


Abbildung 46: Prozess „ERP Software Einführung“ in EPK-Notation

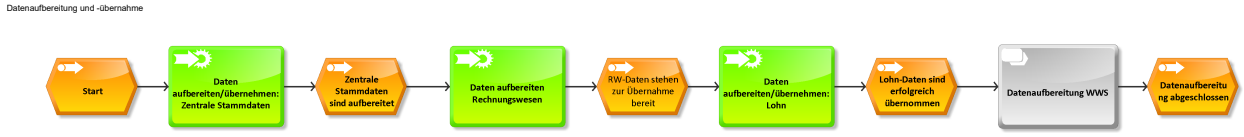


Abbildung 47: Subprozess „Datenaufbereitung und- übernahme“ in EPK-Notation

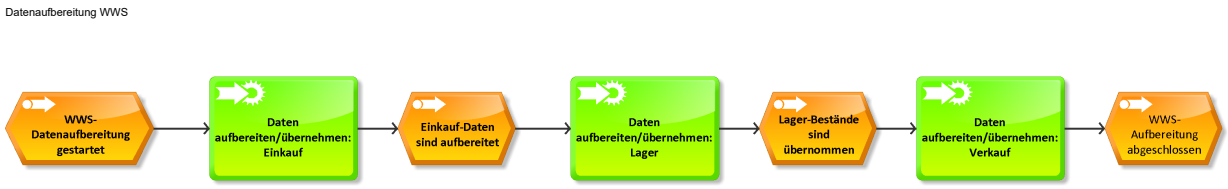


Abbildung 48: Subprozess „WWS durchführen“ in EPK-Notation

5.4.2.2 BPMN-Modell

Die Abbildung 49 zeigt die Basisstruktur des ERP-Software-Einführungsprozesses. Dabei sind alle Subprozesse, außer "Datenaufbereitung und -übernahme", zur besseren Übersicht zusammengeklappt. Der verschachtelte, zusammengeklappte Subprozess "Datenaufbereitung WWS"wird in der Abbildung 49 im zweiten Teil dargestellt.

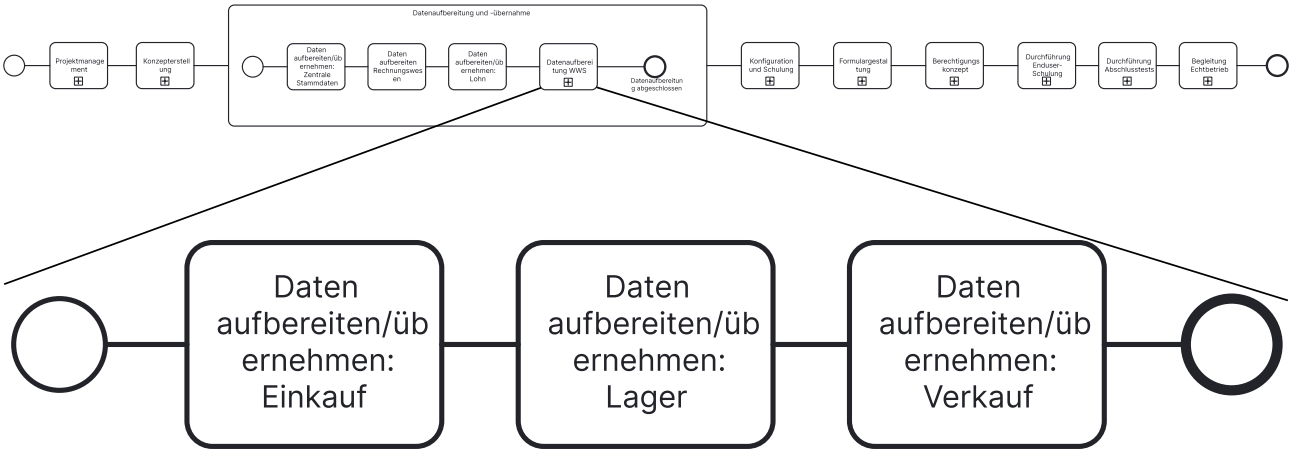


Abbildung 49: Prozess „ERP Software Einführung“ in BPMN-Notation

5.4.2.3 Vergleich und Bewertung

Kriterien

Der Vergleich und die Bewertung des ”ERP Software Einführung” Prozesses berücksichtigt folgende Kriterien:

- Verschiedene Funktionalitäten der Notationen
- Lesbarkeit von Subprozessen

Vergleich

In der EPK-Notation sind Subprozesse durch Prozessschnittstellen dargestellt. Eine Prozessschnittstelle ist eine Referenz auf einen anderen EPK Prozess, der bei diesem Schritt ausgeführt wird. Bei jedem Aufruf eines Teilprozesses muss dieser neu geladen werden. Es besteht auch nicht die Möglichkeit, den vollständigen Subprozess vorübergehend im Hauptprozess darzustellen. Dadurch muss man konstant zwischen den Ansichten wechseln. Diese verschachtelte Ansicht kann bei stark verschachtelten Subprozessen schnell zu Unübersichtlichkeit führen.

Im Gegensatz zu der EPK-Notation verwendet die BPMN-Notation Subprozesse. Dabei wird zwischen zusammengeklappten und aufgeklappten Subprozessen unterschieden. Beim aufgeklappten Subprozess wird dieser auch direkt im Diagramm angezeigt. Der zusammengeklappte Subprozess wird hingegen als ein Element angezeigt, das eine Referenz auf den Teilprozess beinhaltet. Man kann zwischen den zwei Ansichten in jeder BPMN Bearbeitungsoberfläche mit nur zwei bis drei Handgriffen wechseln. Dadurch kann sich der Leser die optimale Ansicht selbst zusammenstellen.

Bewertung

Wenn man die zwei Notationen hinsichtlich der Darstellung von Subprozessen vergleicht, findet man die BPMN-Notation als die bessere Option. Allein durch die gewonnene Flexibilität von frei konfigurierbaren Ansichten kann der Leser die Prozesse besser verstehen. Dazu kommt der zusätzliche Konfigurations- und Wartungsaufwand von EPK durch die Verlinkung von Prozessschnittstellen zu deren einzelnen Subprozessen. BPMN bietet auch Event Handling an, falls bei der Exekution des Prozesses ein Event aufgerufen wird.[60] Dabei wird zwischen Ereignissen im Hauptprozess und Ereignissen innerhalb von Subprozessen unterschieden..[61] Das ermöglicht dem Leser des BPMN-Diagramms, Fehler leichter zu lokalisieren.

Im Gegensatz dazu kann in der EPK-Notation das Konzept eines Fehler- und Eventhandlings, wie bei der Event-Handling von BPMN, laut der Syntax-Definition[62] mathematisch nicht existieren. Da es eine strikte Definition gibt, dass es nur eine eingehende und eine ausgehende Kante gibt, ist es nicht möglich, ein drittes Element, das für das Event-Handling notwendig wäre,

[60] Vgl. Object Management Group, *Business Process Model and Notation (BPMN) Version 2.0.2*, S.175.
[61] Vgl. ebd., S.175.
[62] Nüttgens und Rump, „Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK)“, Vgl.

hinzuzufügen. Deshalb muss in EPKs die Fehlerbehandlung direkt in das Diagramm eingebaut werden.

5.4.3 Gesamtbewertung über alle betrachteten Prozesse

Kriterien

Wenn man die Notationen im Zusammenhang mit dem kompletten Projekt betrachtet kristallisieren sich drei verschiedene Kriterien heraus.

- Einfache Lesbarkeit
- Funktionalität
- Automatisierbarkeit

Von diesen ist die Automatisierbarkeit die wichtigste, da man ohne dieser kein Diagramm generieren kann.

Lesbarkeit

Im Rahmen der Lesbarkeit wurde in Kapitel 5.4.1.3 schon definiert, dass EPK mit Ereignissen den Prozess verkomplizieren. Im Gegensatz dazu stellt BPMN den Prozess übersichtlich und leicht nachvollziehbar dar. Auch bei Subprozesse scheint BPMN mit seiner Versatilität. Dabei unterstützt es wie in Abschnitt 5.4.2.3 erwähnt die gleiche Ansicht wie EPK sowohl als auch weitere.

Funktionalität

Beim Thema Funktionalität bieten beide Notationen Vorteile und Nachteile. Diese sind jedoch nicht von großer Bedeutung für die BMD, da beide Vorgehensweisen Event-Handling und Rollenverteilung bis zu einem gewissen Grad unterstützen.

Aspekt der automatischen Generierung

Es gibt eine klare Definition[63] eines BPMN-Diagramms. Mit BPMN 2.0 gibt es einen Standard für die Speicherung eines BPMN-Diagramms. Der Standard wird von der OMG definiert, die ein XML-Schema[64] zur Verfügung stellen. Da XML leicht manipulierbar ist, ermöglicht diese Struktur eine effiziente und automatische Generierung eines BPMN-Diagramm mittels Programmierlogik. Deshalb existieren auch mehrere Studien über die Erstellung einer vollautomatischen BPMN-Engine;[65]

Da EPK eine proprietäre Entwicklung von ARIS ist, gibt es dafür keine einheitliche Spezifizierung. Daher sind auch die EPK-Engines nicht einheitlich und können nicht auf jede EPK Datei angewendet werden. Deshalb ist EPK für die automatische Generierung eines Diagramms über Programmierlogik eher nicht geeignet.

Schlussfolgerung

Auch wenn EPK, speziell für kleine Prozesse und Prozesse, bei denen die Ressourcen genau verfolgt werden, eine gute Eignung hat, ist sie für unseren Anwendungsfall ungeeignet. Da unser Anwendungsfall primär von der automatischen Generierung eines Diagramms handelt, ist der Automatisierungsaspekt entscheidend, und genau in diesem Aspekt weist BPMN deutliche Vorteile gegenüber EPK auf. Darüber hinaus hat BPMN auch einen leichten Vorteil gegenüber EPK in den Kriterien von der BMD definierten Kriterien bezüglich der Leserlichkeit. Somit gilt BPMN als klarer Gewinner für die Zwecke des Projekts.

[63] Object Management Group, *Business Process Model and Notation (BPMN) Version 2.0.2*.
[64] Object Management Group, *BPMN 2.02 XML Schema Definition (XSD)*.
[65] Vgl. Dijkman, Menjivar und Ouyang, „Semantics and Analysis of Business Process Models in BPMN“.

6 Resümee

Dieses Kapitel fasst die wichtigsten Ergebnisse der Arbeit zusammen. Darüber hinaus wird ein Ausblick auf mögliche Weiterentwicklungen der entwickelten Lösung gegeben. Abschließend werden zentrale Erkenntnisse und Erfahrungen aus der Projektarbeit („Lessons Learned“) dargestellt.

6.1 Fazit

Die Anwendung *VisuPro* wurde im Rahmen dieser Diplomarbeit von unserem Team in Zusammenarbeit mit der BMD entwickelt.

Im Voraus möchten wir uns bei unseren Auftraggebern für die gute Zusammenarbeit während und außerhalb des Praktikums bedanken. Die Diplomarbeit konnte nur durch diese regelmäßige Kommunikation verwirklicht werden.

Im Rahmen des Projektes wurden alle in der Projektdefinition (siehe Anhang: D.1) definierten Ziele zeitgemäß erreicht. Eine spezielle Herausforderung stellte dabei der Algorithmus zur Generierung der Hierarchie dar. Um alle Meilensteine trotzdem rechtzeitig fertigzustellen wurde mehr Aufwand benötigt als geplant, dieser ist in Form von Überstunden geleistet worden. Mit der Version 1.0, dessen Fertigstellungsdatum das letzte Ideengebermeeting war, ist vom Ideengeber die Bestätigung für eine ausreichende Qualität gegeben worden.

Zusätzlich wurden in Absprache mit dem Auftraggeber noch die folgenden Features implementiert:

- Freie Invertierung eines Diagramms
- Der Export eines Bildes des Diagramms über eine Schnittstelle

Die finale Version der Applikation ist am 19. März 2026 der BMD übergeben worden (siehe Anhang: F) und entspricht dem durch umfangreiche Tests sichergestellten Qualitätsstandard der BMD.

6.2 Ausblick

Es wurden alle in der Projektdefinition festgelegten Ziele erreicht. Der Ausblick handelt von der Integration der Anwendung in das aktive System und von weiteren Funktionen, welche das Programm verbessern könnten. Diese sind vom Ideengeber jedoch nicht gefordert.

Aktuell werden die Daten, wie von der BMD gefordert, nur in der lokalen Datenbank gespeichert. Zukünftig ist eine Integration der Datenpersistierung in das System der BMD vorgesehen. Dafür wurde bereits ein Proof-Of-Concept erstellt. Einige BPMN-spezifische-Funktionen, wie beispielsweise die Verfolgung des zuletzt bearbeitenden Nutzers, konnten jedoch nicht umgesetzt werden, da auf die entsprechenden Systemfunktionen kein Zugriff bereitgestellt wurde. Diese Funktionen waren von der BMD auch nicht gefordert.

In Bezug auf den BPMN-Generierungsalgorithmus besteht auch noch die Möglichkeit, bestehende Positionen in den Algorithmus für die Generierung der Hierarchie mit einfließen zu lassen. Dadurch könnte die Generierung der Diagramme weiter verbessert und an bestehende Layoutinformationen angepasst werden. Die Grundlage für diese Funktionalität besteht bereits, es ist jedoch unklar ob dies überhaupt eine Verbesserung darstellt.

Die grundlegende technische Integration der Webanwendung in die NTCS ist bereits implementiert und bildet die Basis für eine zukünftige Einbindung des Editors in die bestehende Systemumgebung.

6.3 Lessons Learned

Im Rahmen dieses Projekts wurde die Wichtigkeit von Datendokumentation und Datenanalyse deutlich. Eine ausführliche Dokumentation erleichtert zahlreiche Entwicklungsschritte im Programmierprozess. Die Software kann im Vorhinein besser analysiert werden, da man zu jedem Attribut eine Beschreibung zur Verfügung hat. Zusätzlich unterstützt eine gute Datenanalyse die Entwicklung komplexer Softwarefunktionen und Algorithmen. In unserem Projekt wurde die Datenanalyse in einem der ersten Schritte durchgeführt, was die Entwicklung sehr erleichterte.

Ein weiterer positiver Aspekt des Projekts, den wir beibehalten werden, waren die regelmäßigen Meetings mit dem Auftraggeber. Oft wurden in diesen Meetings Probleme erwähnt, die aus Sicht des Auftraggebers besonders relevant sind. Zusätzlich wurden Verbesserungsvorschläge gemeinsam im Team diskutiert, und dadurch ergaben sich Möglichkeiten, einen besseren Einblick in die Zielsetzung und die Motivation des Projekts zu erlangen.

Im Nachhinein wird deutlich, dass in der Planungsphase nicht alle Aspekte in den Meetings mit dem Auftraggeber eindeutig definiert wurden. Aus diesem Grund ist in diesem Bereich ein Verbesserungspotential vorhanden. Angesichts des Erfolgs der im Rahmen des Projekts abgehaltenen Ideengebermeetings, ist in zukünftigen Projekten geplant, bereits in der Planungsphase eines mit allen Beteiligten abzuhalten.

Rückblickend ist die Kommunikation einer der wichtigsten Aspekte in der Projektentwicklung. Die Kommunikation mit dem Auftraggeber und innerhalb des Teams hilft dabei, Informationen klar und präzise zu vermitteln. Dazu gehört auch die Formulierung von Anforderungen und Rückmeldungen, da unklare oder missverständliche Formulierungen zu Fehlinterpretationen führen können. In diesem Bereich besteht weiterhin Verbesserungspotenzial hinsichtlich klarer und präziser Formulierungen.

Glossar

Abstrakter Syntaxbaum

Eine Baum Datenstruktur das den syntaktischen Aufbau eine Programmes repräsentiert. Jeder Knoten stellt ein Konstrukt der Sprache dar. Die Blattknoten, sogenannte terminals, repräsentieren dabei die einzelnen Token, das sind die kleinsten individuellen Einheiten einer Programmiersprache, wie Operatoren oder Schlüsselwerte. Die anderen Knoter, also die non-terminals, repräsentieren im Gegensatz dazu Programmierkonstrukte wie Schleifen und Bedingungen. Unnütze Informationen wie die Formatierung werden ausgelassen.[66]

B-Baum

Der B-Baum ist eine Baum-Datenstruktur, die Binäre Suchbäume erweitert. Dabei werden im linken Subknoten kleinere Daten und im rechten die größeren gespeichert. In jedem Knoten sind m Elemente gespeichert, somit besitzt dieser auch m Zweige.[67]

BPMI

Business Process Management Initiative: Ehemalige Organisation zur Entwicklung und Förderung von Standards im Geschäftsprozessmanagement, insbesondere zur Entwicklung der BPMN-Notation, die später von der OMG übernommen wurde.

BPMN

Business Process Model and Notation: Standardisierte grafische Notation zur Modellierung, Analyse und Dokumentation von Geschäftsprozessen. BPMN stellt einen einheitlichen Symbolsatz bereit, um Prozesse zwischen Fach- und IT-Bereichen darzustellen.

Entity Relationship Diagram

Peter Pin-Shan Chen veröffentlichte im März 1976 das Wissenschaftliche Papier, „The entity-relationship model—toward a unified view of data“, in dem er die Idee des ERD erstmals präsentiert. Chen baut auf den 3 schon bestehenden Modellen auf und entwickelt daraus sein eigenes Modell. Es bildet die Basis für eine einheitliche Datenansicht.

EPK

Ereignisgesteuerte Prozesskette: Modellierungsmethode zur Darstellung von Geschäftsprozessen, bei der Ereignisse und Funktionen abwechselnd angeordnet werden. EPKs werden häufig zur konzeptionellen Beschreibung betrieblicher Abläufe eingesetzt.

[66] Alon, Levy und Yahav, *code2seq: Generating Sequences from Structured Representations of Code*.
[67] Cornell Universität, *B-Trees*.

ERP

Enterprise Resource Planning: Integrierte Softwarelösung zur Planung, Steuerung und Überwachung von Geschäftsprozessen in Unternehmen, die verschiedene Funktionsbereiche wie Einkauf, Produktion, Lager, Vertrieb und Finanzen miteinander verbindet. Ziel von ERP-Systemen ist es, Ressourcen effizient einzusetzen und unternehmensweite Transparenz zu schaffen.

GPM

Geschäftsprozessmanagement: Ganzheitlicher Managementansatz zur Analyse, Modellierung, Implementierung, Ausführung, Überwachung und kontinuierlichen Verbesserung von Geschäftsprozessen. Ziel des GPM ist es, betriebliche Abläufe effizient, transparent und anpassungsfähig zu gestalten.

Integrationstest

Integrationstests prüfen das Zusammenspiel verschiedener Komponenten des Systems. Ziel ist es sicherzustellen, dass die Schnittstellen zwischen verschiedenen Systemteilen korrekt funktionieren.

NTCS

Die Business-Standardsoftware der BMD. Sie bietet integrierte Module für die Finanzbuchhaltung, Lohnabrechnung, Warenwirtschaft, Bilanzierung, Controlling, Projektmanagement, und noch weitere Module.

OMG

Object Management Group: Internationale Standardisierungsorganisation, die offene Standards für Software- und Systemarchitekturen entwickelt, darunter die BPMN-Spezifikation.

Referenzrelation

Referenzrelation: Eine Entitätsmenge die dazu dient alle möglichen Werte, für eine andere Entitätsmenge, darstellt, damit ein Wert nicht redundant gespeichert wird

Systemtest

Systemtests betrachten die Anwendung als Gesamtsystem und überprüfen ob die implementierten Funktionen gemeinsam korrekt arbeiten. Es wird überprüft ob die Anwendung die definierten Anforderungen erfüllt und im praktischen Einsatz stabil funktioniert.

Unittest

Unittests dienen der Überprüfung einzelner Funktionen oder kleinerer Module. Hier testet man Komponenten isoliert, um zu prüfen um die Funktionalitäten korrekt implementiert wurden.

Literaturverzeichnis

[1] Claudia Kocian. *Geschäftprozessmodellierung mit BPMN 2.0. Business Process Model and Notation im Methodenvergleich*. HNU Working Paper Nr. 16. letzter Zugriff am 14.01.2026. Hochschule für angewandte Wissenschaften Neu-Ulm, 2011. URL: https://d-nb.info/106696100X/34.

[2] Frank J. Rump. *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten*. letzter Zugriff am 14.01.2026. B.G. Teubner Stuttgart, 1999. URL: https://books.google.at/books?hl=de&lr=&id=d5DNBgAAQBAJ&oi=fnd&pg=PA6&dq=epk+ereignisgesteuerte+prozesskette&ots=ll6skSV6ls&sig=GcaU1qzf1mNiQzfy_agS7-R9-NA&redir_esc=y#v=onepage&q=epk%20ereignisgesteuerte%20prozesskette&f=false.

[3] Object Management Group. *Business Process Model & Notation (BPMN)*. letzter Zugriff am 14.01.2026. 2026. URL: https://www.omg.org/bpmn/.

[4] Wikipedia contributors. *Business Process Model and Notation*. Stand vom 14.01.2026. 2026. URL: https://de.wikipedia.org/wiki/Business_Process_Model_and_Notation.

[5] Object Management Group. *Business Process Model and Notation (BPMN) Version 2.0.2*. letzter Zugriff am 14.01.2026. 2014. URL: https://www.omg.org/spec/BPMN/2.0.2/PDF.

[6] Jochen Göpfert und Heidl Lindenbach. *Geschäftsprozessmodellierung mit BPMN 2.0: Business Process Model and Notation*. letzter Zugriff am 14.01.2026. Oldenbourg Verlag, 2011. URL: https://books.google.de/books?id=XFhQ0joVjqAC&printsec=copyright&hl=de#v=onepage&q&f=false.

[7] Christopher Partsch. *Transformation zwischen den ausgewählten Modelliersprachen EPK und BPMN*. Bachelor-Arbeit. letzter Zugriff am 14.01.2026. 2018. URL: http://bauhaus.cs.uni-magdeburg.de:8080/miscms.nsf/FEA8C8150500AA14C1257449004F79A9/1D5137BD4330EFFAC12583DD0070A0C4/$FILE/Bachelorarbeit%20Christopher%20Partsch.pdf.

[8] Camunda. *BPMN-Notationselemente*. Stand vom 14.01.2026. 2026. URL: https://camunda.com/bpmn/reference/.

[9] Wikipedia. *Ereignisgesteuerte Prozesskette*. Stand vom 14.01.2026. 2026. URL: https://de.wikipedia.org/wiki/Ereignisgesteuerte_Prozesskette.

[10] Oliver Thomas und Michael Fellmann. *Semantische Ereignisgesteuerte Prozessketten*. Working Paper. letzter Zugriff am 14.01.2026. Institut für Wirtschaftsinformatik (IWi) im DFKI Universität des Saarlandes, 2006. URL: https://dl.gi.de/server/api/core/bitstreams/ee724004-cbfc-4bdf-9361-110a165dd144/content.

[11] Markus Nüttgens und Frank J. Rump. „Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK)“. In: *Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen*. Hrsg. von Jörg Desel und Mathias Weske. Bd. P-21. Lecture Notes in Informatics (LNI). Bonn: Gesellschaft für Informatik / Bonner Köllen Verlag, 2002, S. 64–77.

[12] GBTEC. *EPK-Notationselemente*. Stand vom 27.02.2026. 2026. URL: https://www.gbtec.com/de/wiki/prozessmanagement/epk-tool/.

[13] HarzOptics. *Einige Grundregeln für die Modellierung ereignisgesteuerter Prozessketten*. Stand vom 17.02.2026. 2013. URL: https://harzoptics.wordpress.com/2013/04/26/einige-grundregeln-fur-die-modellierung-ereignisgesteuerter-prozessketten/.

[14] Parasoft. *Was ist Unit-Testing? Eine vollständige Anleitung*. Letzter Zugriff: 03.03.2026. URL: https://de.parasoft.com/learning-center/unit-testing-guide/.

[15] Parasoft. *Softwareintegrationstests*. Letzter Zugriff: 03.03.2026. URL: https://de.parasoft.com/learning-center/iso-26262/integration-testing/.

[16] Parasoft. *Software System Testing*. Letzter Zugriff: 03.03.2026. URL: https://www.parasoft.com/learning-center/iso-26262/system-testing/.

[17] React. *Using TypeScript in React*. Letzter Zugriff: 03.03.2026. URL: https://react.dev/learn/typescript.

[18] Microsoft. *What is TypeScript?* Letzter Zugriff: 03.03.2026. URL: https://www.typescriptlang.org/.

[19] Vite. *The Build Tool for the Web*. Letzter Zugriff: 03.03.2026. URL: https://vite.dev/.

[20] Vite. *Vite Next Generation Frontend Tooling*. Letzter Zugriff: 03.03.2026, version 8.3.0. URL: https://github.com/vitejs/vite.

[21] SQLite Development Team. *Architecture of SQLite*. 2025. URL: https://www.sqlite.org/arch.html (besucht am 11. 03. 2026).

[22] SQLite Development Team. *How SQLite Works*. 2025. URL: https://www.sqlite.org/howitworks.html (besucht am 11. 03. 2026).

[23] Red Hat. *What is a REST API?* 2025. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (besucht am 15. 03. 2026).

[24] Hibernate Team. *Hibernate ORM Documentation*. Zugriff am: 15.03.2026. 2024. URL: https://docs.hibernate.org/orm/.

[25] Open Liberty. *Open Liberty Documentation*. Zugriff am: 15.03.2026. 2024. URL: https://openliberty.io/docs/.

[26] Red Hat. *Java Persistence API (JPA) Documentation*. Zugriff am: 15.03.2026. 2023. URL: https://docs.redhat.com/en/documentation/red_hat_jboss_enterprise_application_platform/7.0/html/development_guide/java_persistence_api_jpa.

[27] Scott Chacon und Ben Straub. *Pro Git Book*. 2024. URL: https://git-scm.com/book/en/v2 (besucht am 12. 03. 2026).

[28] GitHub. *About GitHub and Git*. Letzter Zugriff: 12.03.2026. URL: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git.

[29] npm, Inc. *About npm*. 2026. URL: https://docs.npmjs.com/about-npm (besucht am 12. 03. 2026).

[30] Microsoft. *Der Open-Source-KI-Code-Editor*. URL: https://code.visualstudio.com/ (besucht am 12. 03. 2026).

[31] Jetbrains. *Die führende IDE für professionelle Softwareentwicklung*. 2026. URL: https://www.jetbrains.com/de-de/idea/ (besucht am 12. 03. 2026).

[32] Oracle. *Enhanced Data Migration*. 2026. URL: https://www.mysql.com/products/workbench/ (besucht am 12. 03. 2026).

[33] React. *The library for web and native user interfaces*. Letzter Zugriff: 03.03.2026. URL: https://react.dev/.

[34] React. *Was ist React.js? Ein Blick auf die beliebte JavaScript-Bibliothek*. Letzter Zugriff: 03.03.2026. URL: https://kinsta.com/de/blog/was-ist-react-js/.

[35] React. *Getting started with react router*. Letzter Zugriff: 03.03.2026. URL: https://reactrouter.com/home.

[36] bpmnio. *bpmn-js - BPMN 2.0 for the web*. Letzter Zugriff: 03.03.2026, version: 18.13.0. URL: https://github.com/bpmn-io/bpmn-js.

[37] bpmnio. *Understanding bpmn-js Internals*. Letzter Zugriff: 03.03.2026. URL: https://bpmn.io/toolkit/bpmn-js/walkthrough/#bpmn-js-internals.

[38] bpmnio. *bpmn-moddle*. Letzter Zugriff: 04.03.2026, version: 10.0.0. URL: https://github.com/bpmn-io/bpmn-moddle?tab=readme-ov-file.

[39] Natural Intelligence. *fast-xml-parser*. Letzter Zugriff: 03.03.2026, version 5.4.2. URL: https://github.com/NaturalIntelligence/fast-xml-parser/tree/master.

[40] Vitest. *Get Started*. Letzter Zugriff: 03.03.2026. URL: https://v3.vitest.dev/guide/features.html.

[41] Vitest. *Next generation testing framework powered by Vite*. Letzter Zugriff: 03.03.2026. URL: https://github.com/vitest-dev/vitest.

[42] Axios. *Promise based HTTP client for the browser and node.js*. Letzter Zugriff: 03.03.2026. URL: https://axios-http.com/docs/intro.

[43] Figma. *What is Figma?* 2026. URL: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma (besucht am 12. 03. 2026).

[44] Overleaf. *About Overleaf*. Letzter Zugriff: 12.03.2026. 2026. URL: https://www.overleaf.com/about.

[45] Overleaf. *Learn LaTeX in 30 minutes*. Letzter Zugriff: 12.03.2026. 2026. URL: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes.

[46] bpmn.io. *bpmn.io – Web-based tooling for BPMN, DMN and Forms*. Letzter Zugriff: 12.03.2026. URL: https://bpmn.io/.

[47] bpmn-io. *bpmn-io Organization on GitHub*. Letzter Zugriff: 12.03.2026. URL: https://github.com/bpmn-io.

[48] Postman, Inc. *Postman Learning Center*. 2025. URL: https://learning.postman.com/docs/introduction/overview/ (besucht am 15. 03. 2026).

[49] Postman, Inc. *Using Postman Collections*. 2025. URL: https://learning.postman.com/docs/collections/overview/ (besucht am 15. 03. 2026).

[50] Postman, Inc. *Writing Tests and Scripts in Postman*. 2025. URL: https://learning.postman.com/docs/tests-and-scripts/write-scripts/intro-to-scripts/ (besucht am 15. 03. 2026).

[51] Ryan K. L. Ko. „A computer scientist's introductory guide to business process management (BPM)“. In: *XRDS* 15.4 (Juni 2009). ISSN: 1528-4972. DOI: 10.1145/1558897.1558901. URL: https://doi.org/10.1145/1558897.1558901.

[52] Kai-Ingo Vogt. *Definition: Was ist "Qualitätssicherung"?* Stand vom 03.03.2026. 2018. URL: https://wirtschaftslexikon.gabler.de/definition/qualitaetssicherung-44396/version-267707.

[53] Felix Schwab und Ingrid Schwab-Matkovits. *Systemplanung und Projektentwicklung: Projektmanagement und Software-Entwicklung*. Schulbuch für HTL III. Wien: Hölzel Verlag, 2022.

[54] Object Management Group. *BPMN 2.02 XML Schema Definition (XSD)*. 2014. URL: https://www.omg.org/spec/BPMN/20100501/Semantic.xsd (besucht am 04. 03. 2026).

[55] Remco Dijkman, Marlon Dumas Menjivar und Chun Ouyang. „Semantics and Analysis of Business Process Models in BPMN“. In: *Information and Software Technology* 50.12 (2008), S. 1281–1294. DOI: 10.1016/j.infsof.2008.02.006. URL: https://eprints.qut.edu.au/224464/.

[56] Uri Alon, Omer Levy und Eran Yahav. *code2seq: Generating Sequences from Structured Representations of Code*. Aug. 2018. DOI: 10.48550/arXiv.1808.01400.

[57] Cornell Universität. *B-Trees*. 2025. URL: https://www.cs.cornell.edu/courses/cs3110/2012sp/recitations/rec25-B-trees/rec25.html (besucht am 11. 03. 2026).

Abbildungsverzeichnis

[68] React, *Using TypeScript in React*.
[69] Vite, *The Build Tool for the Web*.
[70] Chacon und Straub, *Pro Git Book*.
[71] GitHub, *About GitHub and Git*.
[72] npm, Inc., *About npm*.
[73] Microsoft, *Der Open-Source-KI-Code-Editor*.
[74] Jetbrains, *Die führende IDE für professionelle Softwareentwicklung*.
[75] Oracle, *Enhanced Data Migration*.
[76] React, *The library for web and native user interfaces*.
[77] Vitest, *Get Started*.
[78] Axios, *Promise based HTTP client for the browser and node.js*.
[79] Figma, *What is Figma?*
[80] Overleaf, *About Overleaf*.
[81] bpmn.io, *bpmn.io – Web-based tooling for BPMN, DMN and Forms*.
[82] Postman, Inc., *Postman Learning Center*.

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang

A Aufgabenverteilung

Nachfolgend wird aufgezählt, welches Teammitglied welche Kapitel der Diplomarbeit verfasst hat.

A.1 Alexander Vorreither

- Kurzfassung
- Abstract
- Grundlagen und Methoden (2.1)
  - Verwendete Technologien (2.2)
    - Typescript (2.2.1)
    - Vite (2.2.2)
    - Git (2.2.9)
    - npm (2.2.10)
  - Verwendete Entwicklungssysteme (2.3)
  - Verwendete Bibliotheken (2.4)
    - React (2.4.1)
    - bpmn-js (2.4.2)
    - bpmn-moddle (2.4.3)
    - fast-xml-parser (2.4.4)
    - vitest (2.4.5)
    - Axios (2.4.6)
- Implementierung (4)
  - BPMN-Editor und interaktive Bearbeitung (4.2)
  - Algorithmus zur BPMN-Rückkonvertierung (4.4)
- Ergebnisse (5)
  - Benutzeroberfläche und Interaktion im BPMN-Editor (5.1)

A.2 Justin Luckeneder

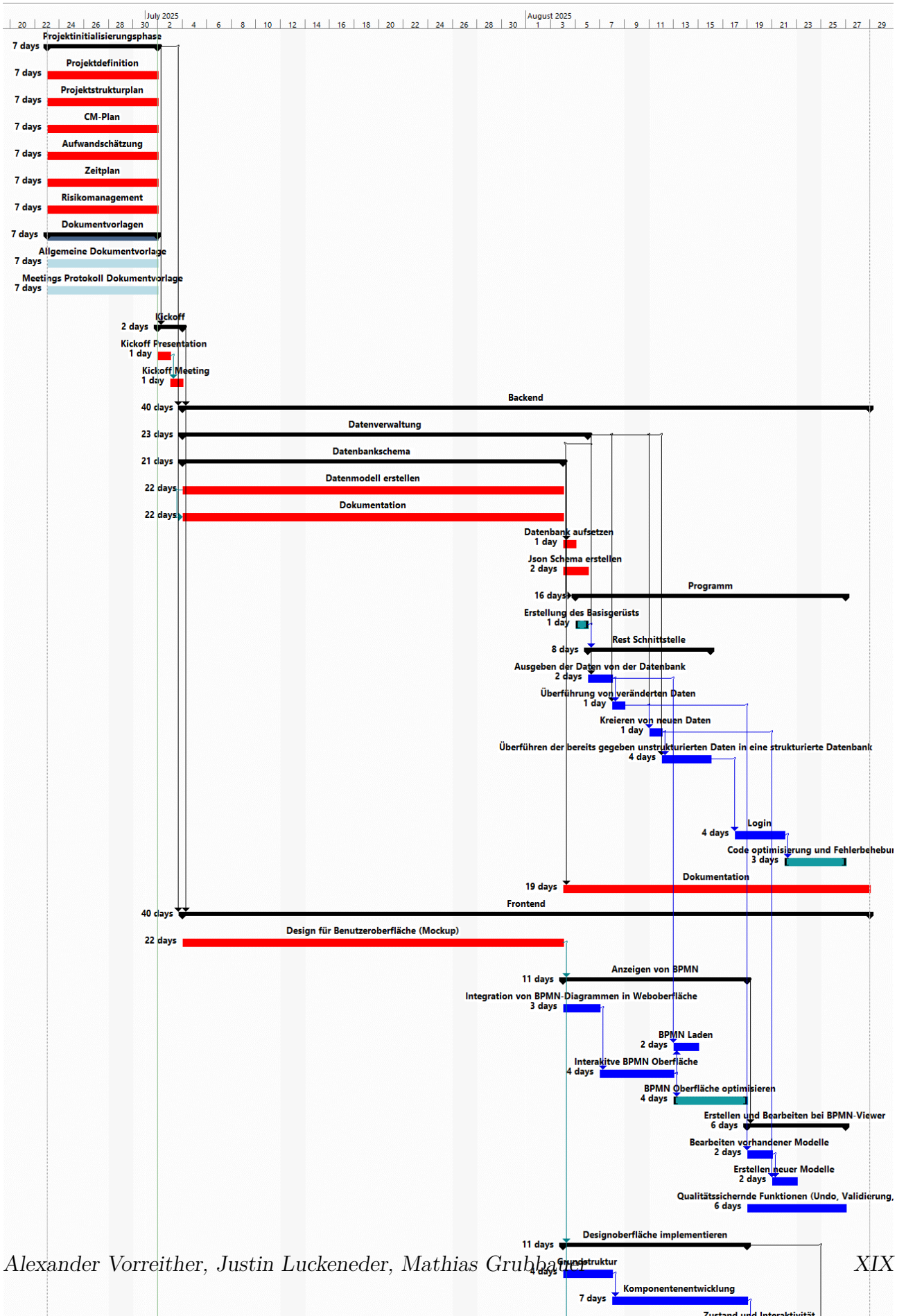
- Einleitung (1)
  - Ausgangslage (1.1)
  - Zielsetzung (1.2)
  - Überblick über den Inhalt der Arbeit (1.3)
  - Projektumfeld (1.4)
- Grundlagen und Methoden (2)
  - Sonstige verwendete Software (2.5)
    - Figma (2.5.1)
    - Postman (2.5.4)
- Planung und Realisierung (3)
  - Projektorganisation (3.1)
  - Meilensteine (3.2)
  - Risiken und Maßnahmen (3.3)
  - Use Cases und Anforderungen (3.4)
- Implementierung (4)
  - Mockup (4.1)
  - Algorithmus zur BPMN-Generierung (4.3)
    - Schritte (4.3.2)
      - Invertieren (4.3.2.5)
  - Java REST-Endpunkte (4.6)

A.3 Mathias Grubbauer

- Grundlagen und Methoden (2)
  - Verwendete Bibliotheken (2.4)
    - Puppeteer (2.4.7)
  - Verwendete Technologien (2.2)
    - SQLite (2.2.3)
    - Headless Browser (2.2.8)
  - Verwendete Schnittstellen (3.5.1.2)
- Planung und Realisierung (2.2)

  - Systementwurf (3.5)
    - Systemarchitektur (3.5.1)
    - Datenhaltung und Datenbankmodell (3.5.3.2)
- Implementierung (4)
  - Algorithmus zur BPMN-Generierung (4.3)
    - Input (4.3.1)
    - Schritte (4.3.2)
      - **nicht** Invertieren (4.3.2.5)
    - Output (4.3.3)
  - Generierung von PNG aus BPMN-Diagramm
- Ergebnisse
  - Algorithmus zur BPMN-Generierung
  - Vergleich BPMN und EPK
- Resümee
  - Ausblick (6.2)
  - Lessons Learned (6.3)
  - Fazit (6.1)

B Zeitplan


July 2025
August 2025
Projektinitialisierungsphase
7 days
Projektdefinition
7 days
Projektstrukturplan
7 days
CM-Plan
7 days
Aufwandschätzung
7 days
Zeitplan
7 days
Risikomanagement
7 days
Dokumentvorlagen
7 days
Allgemeine Dokumentvorlage
7 days
Meetings Protokoll Dokumentvorlage
7 days
Kickoff
2 days
Kickoff Presentation
1 day
Kickoff Meeting
1 day
Backend
40 days
Datenverwaltung
23 days
Datenbankschema
21 days
Datenmodell erstellen
22 days
Dokumentation
22 days
Datenbank aufsetzen
1 day
Json Schema erstellen
2 days
Programm
16 days
Erstellung des Basisgerüsts
1 day
Rest Schnittstelle
8 days
Ausgeben der Daten von der Datenbank
2 days
Überführung von veränderten Daten
1 day
Kreieren von neuen Daten
1 day
Überführen der bereits gegeben unstrukturierten Daten in eine strukturierte Datenbank
4 days
Login
4 days
Code optimisierung und Fehlerbehebu
3 days
Dokumentation
19 days
Frontend
40 days
Design für Benutzeroberfläche (Mockup)
22 days
Anzeigen von BPMN
11 days
Integration von BPMN-Diagrammen in Weboberfläche
3 days
BPMN Laden
2 days
Interakitve BPMN Oberfläche
4 days
BPMN Oberfläche optimisieren
4 days
Erstellen und Bearbeiten bei BPMN-Viewer
6 days
Bearbeiten vorhandener Modelle
2 days
Erstellen neuer Modelle
2 days
Qualitätssichernde Funktionen (Undo, Validierung,
6 days
Designoberfläche implementieren
11 days
Grundstruktur
4 days
Komponentenentwicklung
7 days

C Datendokumentation

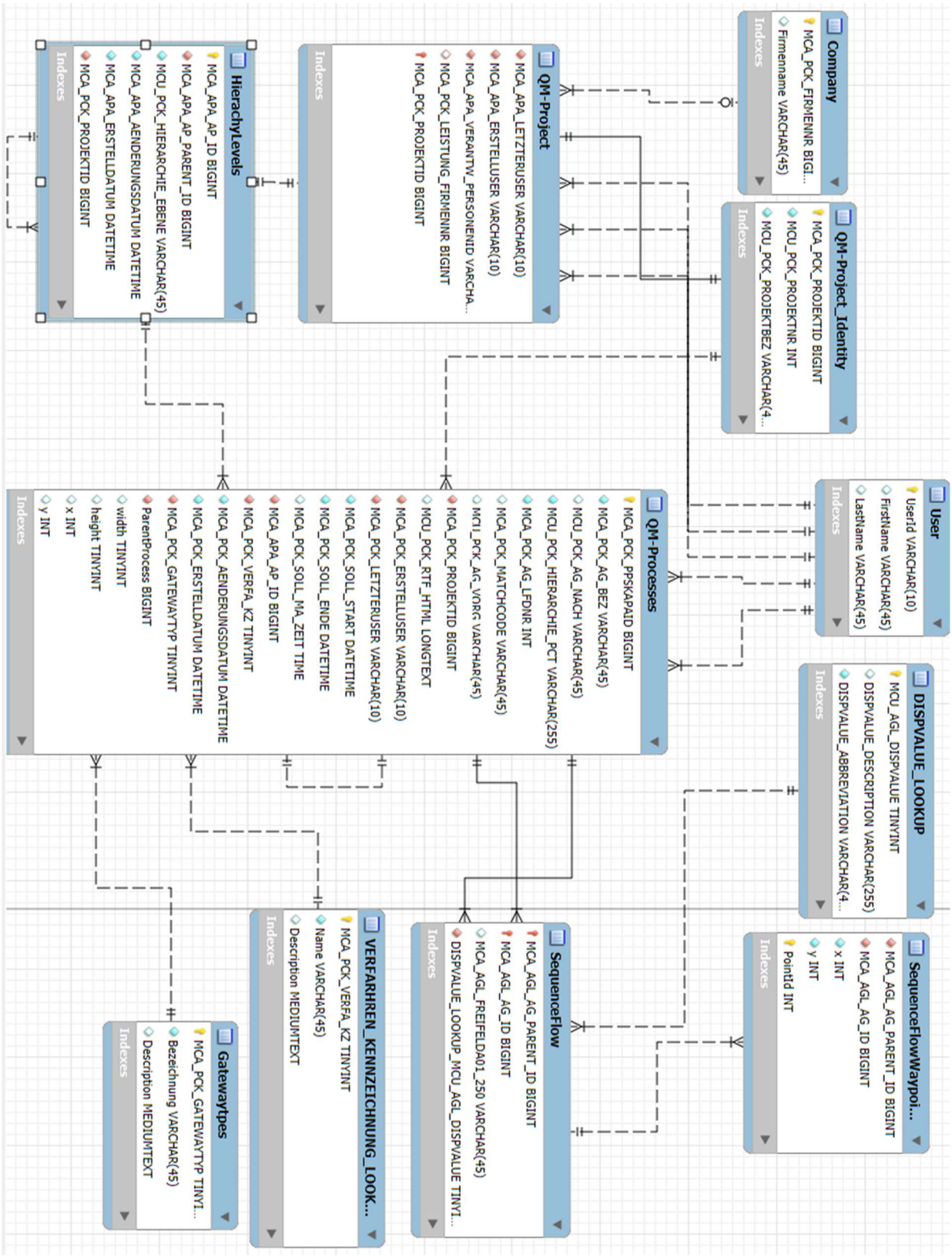
VisuPro

Documentation

Mathias Grubbauer

1. Inhaltsverzeichnis

2. ERD


Company
MCA_PCK_FIRMENNR BIGI...
Firmenname VARCHAR(45)
Indexes
QM-Project_Identity
MCA_PCK_PROJEKTID BIGINT
MCU_PCK_PROJEKTNR INT
MCU_PCK_PROJEKTBEZ VARCHAR(4...
Indexes
User
UserId VARCHAR(10)
FirstName VARCHAR(45)
LastName VARCHAR(45)
Indexes
QM-Project
MCA_APA_LETZTERUSER VARCHAR(10)
MCA_APA_ERSTELLUSER VARCHAR(10)
MCA_APA_VERANTW_PERSONENID VARCHA...
MCA_PCK_LEISTUNG_FIRMENNR BIGINT
MCA_PCK_PROJEKTID BIGINT
Indexes
HierachyLevels
MCA_APA_AP_ID BIGINT
MCA_APA_AP_PARENT_ID BIGINT
MCU_PCK_HIERARCHIE_EBENE VARCHAR(45)
MCA_APA_AENDERUNGSDATUM DATETIME
MCA_APA_ERSTELLDATUM DATETIME
MCA_PCK_PROJEKTID BIGINT
Indexes
QM-Processes
MCA_PCK_PPSKAPAID BIGINT
MCA_PCK_AG_BEZ VARCHAR(45)
MCU_PCK_AG_NACH VARCHAR(45)
MCU_PCK_HIERARCHIE_PCT VARCHAR(255)
MCA_PCK_AG_LFDNR INT
MCA_PCK_MATCHCODE VARCHAR(45)
MCU_PCK_AG_VORG VARCHAR(45)
MCA_PCK_PROJEKTID BIGINT
MCU_PCK_RTF_HTML LONGTEXT
MCA_PCK_ERSTELLUSER VARCHAR(10)
MCA_PCK_LETZTERUSER VARCHAR(10)
MCA_PCK_SOLL_START DATETIME
MCA_PCK_SOLL_ENDE DATETIME
MCA_PCK_SOLL_MA_ZEIT TIME
MCA_APA_AP_ID BIGINT
MCA_PCK_VERFA_KZ TINYINT
MCA_PCK_AENDERUNGSDATUM DATETIME
MCA_PCK_ERSTELLDATUM DATETIME
MCA_PCK_GATEWAYTYP TINYINT
ParentProcess BIGINT
width TINYINT
height TINYINT
x INT
y INT
Indexes
DISPVALUE_LOOKUP
MCU_AGL_DISPVALUE TINYINT
DISPVALUE_DESCRIPTION VARCHAR(255)
DISPVALUE_ABBREVIATION VARCHAR(4...
Indexes
SequenceFlow
MCA_AGL_AG_PARENT_ID BIGINT
MCA_AGL_AG_ID BIGINT
MCA_AGL_FREIFELDA01_250 VARCHAR(45)
DISPVALUE_LOOKUP_MCU_AGL_DISPVALUE TINYI...
Indexes
SequenceFlowWaypoi...
MCA_AGL_AG_PARENT_ID BIGINT
MCA_AGL_AG_ID BIGINT
x INT
y INT
PointId INT
Indexes
VERFAHREN_KENNZEICHNUNG_LOOK...
MCA_PCK_VERFA_KZ TINYINT
Name VARCHAR(45)
Description MEDIUMTEXT
Indexes
Gatewaytypes
MCA_PCK_GATEWAYTYP TINYI...
Bezeichnung VARCHAR(45)
Description MEDIUMTEXT
Indexes

3. Entitätsbeschreibung

3.1. Company

Beschreibung: Die Entität Firma repräsentiert eine externe juristische Person, typischerweise einen Kunden oder Geschäftspartner der BMD.
Entitätstyp: Fundamental
Primary Key: MCA_PCK_FIRMENNR

Attribute:

Attribut	Beschreibung	Bemerkung
Firmennummer	Eine eindeutige Kennung, die von BMD zur internen Verwaltung des Kunden vergeben wird.	Dient als primärer Identifikator.
Firmenname	Eine eindeutige Kennung, die von BMD zur internen Verwaltung des Kunden vergeben wird.	Muss eindeutig sein, um Duplikate zu vermeiden.

Existenzregeln

Create: Wenn ein neuer Kunde einen Auftrag bei der BMD stellt

Delete: Wenn ein Kunde die Löschung seines Kontos beantragt (DSGVO)

Ein Firmendatensatz darf nur gelöscht werden, wenn keine aktiven Geschäftsbeziehungen (z. B. laufende Projekte oder offene Verträge) mehr bestehen.

Integritätsregeln

- Der Firmenname sollte eindeutig sein, um Duplikate zu vermeiden und eine klare Identifikation zu gewährleisten.

Anzahl der Ausprägungen

- Initial: 30000
- Wachstumsrate: Etwa 1000 – 1800 neue Kunden im Jahr
- Im Durchschnitt wird man irgendwo zwischen 30000 und 50000 Kunden haben

3.2. User

Beschreibung: Die Entität Benutzer repräsentiert eine natürliche Person, typischerweise einen Mitarbeiter der BMD, der im System Aktionen durchführt und Verantwortlichkeiten zugewiesen bekommt.

Einschränkungen:

Nicht jedes QM-Projekt muss einer Firma zugeordnet sein (z.B. bei internen Projekten).

Entitätstyp: Fundamental

Primary Key: Personalnummer (UserId)

Attribut	Beschreibung	Bemerkung
Personalnummer	Eine eindeutige Kennung, die von der Personalabteilung jedem Mitarbeiter zugewiesen wird.	Dient als primärer Identifikator und Login-Name.
Vorname	Der offizielle Vorname der Person.	
Nachname	Der offizielle Nachname der Person.	

Existenzregeln:

Create: Ein User wird angelegt wenn er offiziell bei der BMD eingestellt wird

Delete: Sobald der Mitarbeiter die BMD beantragt, dass seine Datenlöschung antragt.

Integritätsregeln:

Bei der Löschung darf der User keinem Projekt mehr als Verantwortlicher zugewiesen sein

Anzahl der Ausprägungen:

- Initial: ~900
- Wachstumsrate: Etwa 20 neue User im Jahr
- Im Durchschnitt wird man irgendwo zwischen 1000 und 15000 User haben

3.3. QM-Project

Beschreibung: Die Tabelle QM_Projekt dient zur Verwaltung und Dokumentation von Qualitätsmanagement-Projekten innerhalb eines Unternehmens. Sie enthält grundlegende Informationen zu jedem Projekt, seinen Verantwortlichen, dem Status, relevanten Terminen und Qualitätskennzahlen.

Entitätstyp: Fundamental

Primary Key: MCA_PCK_PROJEKTID

Attribut	Beschreibung	Bemerkung
Projekt-ID	Eine interne, systemweit eindeutige Kennung zur technischen Identifikation.	Dient als primärer Identifikator.
Projektnummer	Eine für den Menschen lesbare, eindeutige Nummer zur Identifikation im Geschäftsalltag.	
Projektbezeichnung	Der offizielle Name des Projekts	
Verantwortlicher	Der Benutzer, der die Gesamtverantwortung für das Projekt trägt	

Existenzregeln

Create: Ein QM-Projekt wird dann erstellt, sobald irgendwo ein Bedarf für ein Qualitätsmanagement Abläufe besteht

Delete: Projekte werden nach Abschluss archiviert und nicht physisch gelöscht, um historische Daten zu bewahren. Die meisten sind aber intern und werden erst gelöscht sobald der Ablauf irrelevant wird

Integritätsregeln:

Jedes Projekt muss genau einem verantwortlichen Benutzer zugeordnet sein. Die Projektnummer und Projektbezeichnung müssen eindeutig sein.

Anzahl der Ausprägungen

- Initial: ~2000
- Wachstumsrate: Etwa 5 neue im Jahr
- Im Durchschnitt wird man irgendwo zwischen 1900 und 2100 Projekte haben

3.4. QM-Processes

Beschreibung: Die Entität QM-Prozess stellt die granulare, operative Arbeitseinheit innerhalb eines Projektablaufs dar. Sie modelliert einen einzelnen, konkreten Schritt in einem Geschäftsprozes.

Entitätstyp: Abhängig von QM-Projekt

Primary Key: MCA_PCK_PPSKAPAID

Attribut	Beschreibung
Prozess-ID	Eine systemweit eindeutige Kennung für den Prozessschritt.

Bezeichnung	Die textuelle Beschreibung der Aufgabe
Prozesstyp	Definiert die Art des BPMN-Elements
Vorgänger/Nachfolger	Speichert die Nachfolger zu der das Element sich verbindet

Existenzregeln

Create: Ein QM-Prozess kann nur innerhalb eines existierenden QM-Projekts erstellt werden.

Delete: Wenn das QM-Projekt gelöscht wird.

Wenn das QM-Projekt modifiziert wird und es sich herausstellt das dieser Prozess nicht mehr gebraucht wird

Integritätsregeln:

Jeder QM-Prozess muss genau einem QM-Projekt zugeordnet sein.

Anzahl der Ausprägungen

- Initial: ~1200 pro Projekt = ~240000
- Wachstumsrate: Etwa 1000 neue QM-Prozesse im Jahr
- Im Durchschnitt wird man irgendwo zwischen 230000 und 270000 QM-Prozesse haben

3.5. SequenceFlow

Beschreibung: Die Entität Prozessverbindung ist eine Verknüpfungsentität (Junction Entity), die den logischen Kontrollfluss und die Abhängigkeiten zwischen zwei einzelnen QM-Prozessen modelliert. Sie repräsentiert den "Pfeil" in einem BPMN-Diagramm, der die Reihenfolge, Parallelität oder bedingte Ausführung von Arbeitsschritten festlegt.

Einschränkungen: Eine Verbindung kann nur zwischen Prozessen innerhalb desselben Projekts bestehen.

Entitätstyp: Fundamental abhängig von QM-Processes

Primary Key: Kombination aus Quell-Prozess und Ziel-Prozess (MCA_AGL_AG_PARENT_ID, MCA_AGL_AG_ID)

Attribut	**Beschreibung**
Quell-Prozess	Der QM-Prozess, von dem die Verbindung ausgeht
Ziel-Prozess	Der QM-Prozess, zu dem die Verbindung führt
Verbindungsart	Der Typ der Verbindung

Existenzregeln

Create: Eine Prozessverbindung wird erstellt, wenn zwei QM-Prozess verbunden werden.

Delete: Wenn die Verbindung zwischen den Prozessen gelöscht wird
Wenn einer der Prozesse gelöscht wird

Integritätsregeln:

Ein Prozess kann nicht mit sich selbst verbunden sein. Die Kombination aus Quell- und Ziel-Prozess muss eindeutig sein.

Anzahl der Ausprägungen

- Initial: ~120 pro Projekt = ~240000
- Wachstumsrate: Etwa 1000 neue SequenceFlows im Jahr
- Im Durchschnitt wird man irgendwo zwischen 230000 und 270000 Sequence-Flows haben

3.6. HierachyLevels

Beschreibung: Jeder QM-Prozess ist auf einer Ebene. Wenn ein Process oder Subprocess Unterprozesse hat, ist für diese Unterprozesse die Ebene eins höher als der Originalprozess

Entitätstyp: abhängig von QM-Projekt

Primary Key: MCA_APA_AP_ID

Attribut	Beschreibung	Bemerkung
Ebenen-ID	Eine systemweit eindeutige Kennung für die Hierarchieebene	
Übergeordnete Ebene	Die Ebenen-ID der direkt übergeordneten Ebene.	Bei der obersten Ebene (Wurzel) verweist dieses Attribut auf sich selbst.
Hierarchieebene	Die wievielte Ebene es ist	

Existenzregeln

Create: Sobald in einem Prozess ein Unterelement angelegt wird und es für diese Ebene noch keine Ebene gibt

Beim kreieren des QM-Projekt wird die 1. Ebene angelegt

Delete: Eine Ebene kann nur gelöscht werden, wenn ihr keine untergeordneten Ebenen und keine QM-Prozesse mehr zugeordnet sind.

Integritätsregeln:

Jede Ebene (außer der Wurzel) muss eine gültige, existierende übergeordnete Ebene haben.

Anzahl der Ausprägungen

- Initial: durchschnittlich 1.5 pro Projekt = 3000
- Wachstumsrate: Etwa 180 neue Ebenen im Jahr
- Im Durchschnitt wird man irgendwo zwischen 3000 und 5000 Ebenen haben

3.7. SequenceFlowWaypoints

Beschreibung: Der Zweck der Entität SequenceFlowWaypoints ist es, rein visuelle Darstellungsdaten für eine Prozessverbindung zu speichern. Jeder Eintrag repräsentiert einen X/Y-Koordinatenpunkt, der verwendet wird, um die Linie einer Verbindung im Prozessdiagramm zu zeichnen, insbesondere wenn diese um andere Elemente herumgeführt werden muss.

Entitätstyp: abhängig von SequenceFlow

Primary Key: PointId

Attribut	Beschreibung
ID	Eine eindeutige Kennung für den Koordinatenpunkt
Zugehörige Verbindung	Die Prozessverbindung, zu der dieser Punkt gehört.
Koordinaten	Die Koordinaten wo durch die der Sequenceflow durchgeht

Existenzregeln

Create: Wegpunkte werden erstellt, wenn ein Benutzer eine Verbindungslinie im Editor manuell verschiebt oder knickt.

Bei der ersten Positionsgenerierung werden automatisch bei manchen Verbindungen ein oder mehrere Waypoints generiert

Delete: Alle Wegpunkte einer Verbindung werden automatisch gelöscht, wenn die zugehörige Prozessverbindung gelöscht wird.

Die Punkte können im Editor vom User gelöscht werden

Integritätsregeln:

Jeder Wegpunkt muss zu einer existierenden Prozessverbindung gehören und es dürfen nicht zwei Wegpunkte im selben Projekt überlappen

Anzahl der Ausprägungen

- Initial: ~80 pro Projekt = 160000
- Wachstumsrate: Etwa 500 neue Koordinaten im Jahr
- Im Durchschnitt wird man irgendwo zwischen 170000 und 200000 User haben

3.8. Verfahrenskennzeichen

Beschreibung: Verfahrenskennzeichen bestimmen was für eine Art von BPMN-Element einen QM-Prozess ist.

Entitätstyp: Fundamental

Primary Key: MCA_PCK_VERFA_KZ

Attribut	Beschreibung
ID	Eine eindeutige, Nummer für den Verfahrenstyp
Name	Der Name des Verfahren

Existenzregeln

Create: Die Werte in dieser werden bei der Konfiguration gleich konfiguriert und wird nur geändert, wenn man noch nicht implementierte/neue BPMN Features hinzugefügt.

Delete: Nur bei komplettem Löschen von dem ganzen Modell

Integritätsregeln:

Der Name des Verfahrens muss eindeutig sein.

Anzahl der Ausprägungen

- Initial: 3
- Wachstumsrate: 0
- Es weden immer 3 sein

3.9. Gateway-Typ

Beschreibung: Falls das Verfahren ein Meilenstein ist bestimmt dieses Feld wleche Art von Event/Gateway es ist

Einschränkungen: Dieses Attribut ist nur relevant und darf nur zugewiesen werden, wenn das Verfahrenskennzeichen des QM-Prozesses auf "Meilenstein“ gesetzt ist.

Entitätstyp: Fundamental

Primary Key: MCA_PCK_GATEWAYTYP

Attribut	Beschreibung	Bemerkung

ID	Die Nummer die den Typen bestimmt	
Name	Der Name des Typen	

Existenzregeln

Create: Die Werte in dieser werden bei der Konfiguration gleich konfiguriert und wird nur geändert, wenn man noch nicht implementierte/neue BPMN Features hinzugefügt.

Delete: Nur bei komplettem Löschen von dem ganzen Modell

Anzahl der Ausprägungen

- Initial: 6
- Wachstumsrate: 0
- Es werden immer 6 sein

3.10. DISPVALUE_LOOKUP

Beschreibung: Die Art wie ein SequenceFlow die QM-Prozesse verbindet

Entitätstyp: Fundamental

Primary Key: MCU_AGL_DISPVALUE

Attribut	Beschreibung
ID	Ein eindeutiger, numerischer Schlüssel für die Verbindungsart.
Beschreibung	Der für Menschen lesbare, vollständige Name der Verbindungsart
Abkürzung	Eine kurze, technische Abkürzung für die Verbindungsart

Existenzregeln

Create: Nur bei der Erstellung des Datenmodell

Delete: Nur bei Löschung der Entität

Integritätsregeln:

Anzahl der Ausprägungen

- Initial: 4
- Wachstumsrate: 0
- 4

3.11. QM-Project_Identity

Beschreibung: Die Entität Projektidentität ist eine rein technische, fundamentale Entität, um Datenbanknormalisierung (BCNF) durchführen

Einschränkungen:

Entitätstyp: Fundamental

Primary Key:

Existenzregeln

Create: Ein Projektidentität-Datensatz wird erstellt, sobald ein neues QM-Projekt formal initiiert wird. Dies ist der erste Schritt bei der Erstellung eines neuen Projekts.

Delete: Sobald QM-Project gelöscht wird

Integritätsregeln:

Sowohl die Projektnummer als auch die Projektbezeichnung müssen systemweit absolut eindeutig sein. Dies wird durch UNIQUE-Constraints in der Datenbank sichergestellt und ist der Hauptgrund für die Existenz dieser Entität.

Anzahl der Ausprägungen

1 zu 1 wie QM-Projects

4. Relationsdefinitionen

4.1. User – QM-Project Verantwortlicher

DEFINITION, ZWECK: Diese Beziehung stellt die personelle Gesamtverantwortung für ein Projekt sicher. Jedem Projekt wird ein Hauptverantwortlicher zugewiesen, um eine klare Zuständigkeit für den Erfolg, das Budget und die Einhaltung von Fristen zu gewährleisten.

dazugehörige ENTITÄTEN: User, QM-Project

VON - NACH - BEZIEHUNG: User ist verantwortlich für QM-Project

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Project wird verantwortet von User

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Die Personalnummer des verantwortlichen Users, die im QM-Project hinterlegt wird, muss als aktiver User im System existieren.

4.2. Company – QM-Project

RELATION: QM Prozess wird für eine externe Firma gemacht

DEFINITION, ZWECK: Diese Beziehung ordnet ein QM-Projekt einem externen Kunden zu. Sie dient dazu, kundenbezogene Aufträge von internen Initiativen zu unterscheiden und eine korrekte Abrechnung und Berichterstattung zu ermöglichen.

dazugehörige ENTITÄTEN: Company, QM-Project

VON - NACH - BEZIEHUNG: Company ist der Erwartende des QM-Project

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Project wird durchgeführt für Company

Kardinalität: min: 0 max: 1

INTEGRITÄTSREGELN:

Wenn einem QM-Project eine Firmennummer zugewiesen wird, muss diese Firma im System existieren.

4.3. QM-Project – QM-Processes

RELATION: QM-Project besteht aus QM-Processes

DEFINITION, ZWECK: Dies ist die zentrale strukturelle Beziehung des Systems. Sie definiert, dass ein Projekt der Container für eine Menge von einzelnen Arbeitsschritten (Prozessen) ist. Ein Prozess kann nicht außerhalb eines Projekts existieren.

dazugehörige ENTITÄTEN: QM-Project, QM-Processes

VON - NACH - BEZIEHUNG: QM-Project besteht aus QM-Processes

Kardinalität: min: 1 max: m

UMGEKEHRTE BEZIEHUNG: QM-Processes ist Teil von QM-Project

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Jeder QM-Process muss einer gültigen, existierenden Projekt-ID zugeordnet sein. Wird ein Projekt gelöscht, müssen auch alle zugehörigen Prozesse gelöscht werden (Kaskadierung).

4.4. QM-Processes – QM-Processes

RELATION: QM-Processes ist übergeordnet zu QM-Processes

DEFINITION, ZWECK: Diese hierarchische Beziehung ermöglicht die Modellierung von Sub-Prozessen. Ein komplexer Prozess kann in mehrere detailliertere, untergeordnete Prozesse zerlegt werden, um die Übersichtlichkeit und Modularität des Projektplans zu erhöhen.

dazugehörige ENTITÄTEN: QM-Processes, QM-Processes

VON - NACH - BEZIEHUNG: QM-Processes ist übergeordnet zu QM-Processes

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Processes ist untergeordnet zu QM-Processes

Kardinalität: min: 0 max: 1

INTEGRITÄTSREGELN:

Wenn in QM-Processes ein ParentProcess definiert ist, muss dessen ID (MCA_PCK_PPSKAPAID) in derselben Tabelle existieren. Ein Prozess kann nicht sein eigener übergeordneter Prozess sein

4.5. QM-Processes – QM-Processes | SequenceFlow

RELATION: QM-Processes ist verbunden mit QM-Processes (via SequenceFlow)

DEFINITION, ZWECK: Diese Beziehung modelliert den Kontrollfluss (die "Pfeile") zwischen den Arbeitsschritten eines Projekts. Sie definiert die Ausführungsreihenfolge und die logischen Abhängigkeiten und ist damit die Grundlage für die Workflow-Engine.

Associative Table: SequenceFlow

dazugehörige ENTITÄTEN: QM-Processes, QM-Processes (über die Verknüpfungstabelle SequenceFlow)

VON - NACH - BEZIEHUNG: QM-Processes geht über in QM-Processes

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Processes wird erreicht von QM-Processes

Kardinalität: min: 0 max: m

INTEGRITÄTSREGELN:

Der Quell- (MCA_AGL_AG_PARENT_ID) und Zielprozess (MCA_AGL_AG_ID) einer SequenceFlow-Verbindung müssen in QM-Processes existieren und dürfen nicht identisch sein.

4.6. QM-Project_Identity – QM-Project

RELATION: QM-Project_Identity hat Details in QM-Project

DEFINITION, ZWECK: Diese Beziehung ist das Ergebnis der Datenbanknormalisierung (BCNF). Sie stellt sicher, dass zu jeder eindeutigen Projektidentität (QM-Project_Identity) genau ein Satz von beschreibenden Attributen (QM-Project) gehört. Es handelt sich um eine strikte 1:1-Beziehung, die die Identitätsdaten von den Detaildaten trennt.

dazugehörige ENTITÄTEN: QM-Project_Identity, QM-Project

VON - NACH - BEZIEHUNG: QM-Project_Identity hat Details in QM-Project

Kardinalität: min: 1 max: 1

UMGEKEHRTE BEZIEHUNG: QM-Project beschreibt QM-Project_Identity

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Jeder Datensatz in QM-Project muss sich auf einen existierenden Datensatz in QM-Project_Identity beziehen. Die MCA_PCK_PROJEKTID in QM-Project muss eindeutig sein, um die 1:1-Beziehung sicherzustellen.

4.7. QM-Processes - VERFARHREN_KENNZEICHNUNG_LOOKUP

RELATION: QM-Processes wird kategorisiert durch VERFARHREN_KENNZEICHNUNG_LOOKUP

DEFINITION, ZWECK: Diese Beziehung ordnet jedem QM-Prozess einen fundamentalen BPMN-Typ zu. Sie dient der grundlegenden Klassifizierung und steuert das Verhalten und die verfügbaren Optionen für einen Prozessschritt im System.

dazugehörige ENTITÄTEN: QM-Processes, VERFARHREN_KENNZEICHNUNG_LOOKUP

VON - NACH - BEZIEHUNG: VERFARHREN_KENNZEICHNUNG_LOOKUP klassifiziert QM-Processes

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Processes wird kategorisiert durch VERFARHREN_KENNZEICHNUNG_LOOKUP

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Jeder Datensatz in QM-Processes muss eine gültige MCA_PCK_VERFA_KZ haben, die in der VERFARHREN_KENNZEICHNUNG_LOOKUP-Tabelle existiert.

4.8. QM-Processes - HierachyLevels

RELATION: QM-Processes ist zugeordnet zu HierachyLevels

DEFINITION, ZWECK: Diese Beziehung ordnet einen Prozessschritt einer bestimmten Ebene von Subprozessen zu. Es hilft dabei das man untergeordnete Elemente schneller findet und besser Verwalten kann.

dazugehörige ENTITÄTEN: QM-Processes, HierachyLevels

VON - NACH - BEZIEHUNG: HierachyLevels gruppiert QM-Processes

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Processes ist zugeordnet zu HierachyLevels

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Jeder Datensatz in QM-Processes muss einer gültigen, existierenden MCA_APA_AP_ID aus der HierachyLevels-Tabelle zugeordnet sein.

4.9. SequenceFlow - DISPVALUE_LOOKUP

RELATION: SequenceFlow wird beschrieben durch DISPVALUE_LOOKUP

DEFINITION, ZWECK: Diese Beziehung klassifiziert die Art einer SequenceFlow-Verbindung (z.B. "Standard", "Bedingt"). Sie definiert die logischen Regeln, unter denen der Kontrollfluss von einem Prozess zum nächsten übergeht.

dazugehörige ENTITÄTEN: SequenceFlow, DISPVALUE_LOOKUP

VON - NACH - BEZIEHUNG: DISPVALUE_LOOKUP beschreibt SequenceFlow

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: SequenceFlow wird beschrieben durch DISPVALUE_LOOKUP

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Jeder Datensatz in SequenceFlow muss eine gültige DISPVALUE_LOOKUP_MCU_AGL_DISPVALUE haben, die in der DISPVALUE_LOOKUP-Tabelle existiert.

4.10. User – QM-Project | Erstelluser

RELATION: User erstellt QM-Project

DEFINITION, **ZWECK**: Diese Beziehung dient der Nachvollziehbarkeit und Protokollierung. Sie speichert, welcher Benutzer ein Projekt ursprünglich angelegt hat. Dies ist wichtig für Audits und zur Klärung von Verantwortlichkeiten bei der initialen Projekteinrichtung.

dazugehörige ENTITÄTEN: User, QM-Project

VON - NACH - BEZIEHUNG: User erstellt QM-Project

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Project wird erstellt von User

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Der MCA_APA_ERSTELLUSER in der QM-Project-Tabelle muss auf einen gültigen, existierenden UserId in der User-Tabelle verweisen

4.11. User – QM-Project | Bearbeitet

RELATION: User bearbeitet QM-Project

DEFINITION, ZWECK: Diese Beziehung dient der Nachvollziehbarkeit, indem sie festhält, welcher Benutzer die letzte Änderung an einem Projekt vorgenommen hat. Dies hilft bei der Fehleranalyse und der Verfolgung von Änderungen.

dazugehörige ENTITÄTEN: User, QM-Project

VON - NACH - BEZIEHUNG: User bearbeitet QM-Project

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Project wird bearbeitet von User

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Der MCA_APA_LETZTERUSER in der QM-Project-Tabelle muss auf einen gültigen, existierenden UserId in der User-Tabelle verweisen.

4.12. User – QM-Processes

RELATION: User erstellt QM-Process

DEFINITION, ZWECK: Ähnlich wie bei Projekten, dient diese Beziehung der detaillierten Nachvollziehbarkeit auf Prozessebene. Sie speichert, welcher Benutzer einen bestimmten Prozessschritt ursprünglich angelegt hat.

dazugehörige ENTITÄTEN: User, QM-Processes

VON - NACH - BEZIEHUNG: User erstellt QM-Processes

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Processes wird erstellt von User

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Der MCA_PCK_ERSTELLUSER in QM-Processes muss auf einen gültigen, existierenden UserId in der User-Tabelle verweisen.

4.13. HierachyLevels - HierachyLevels

RELATION: HierachyLevels ist übergeordnet zu HierachyLevels

DEFINITION, ZWECK: Diese rekursive Beziehung ermöglicht den Aufbau einer Baumstruktur. Jede Hierarchieebene kann genau eine übergeordnete Ebene haben, was die Modellierung von unterschiedlichen Ebenen möglich macht.

dazugehörige ENTITÄTEN: HierachyLevels, HierachyLevels

VON - NACH - BEZIEHUNG: HierachyLevels ist übergeordnet zu HierachyLevels

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: HierachyLevels ist untergeordnet zu HierachyLevels

Kardinalität: min: 0 max: 1

INTEGRITÄTSREGELN:

Die MCA_APA_AP_PARENT_ID muss auf eine existierende MCA_APA_AP_ID in derselben Tabelle verweisen. Eine Ebene kann nicht ihre eigene übergeordnete Ebene sein.

4.14. SequenceFlow – SequenceFlowWaypoints

RELATION: SequenceFlow hat Wegpunkte in SequenceFlowWaypoints

DEFINITION, ZWECK: Diese Beziehung verknüpft eine logische Prozessverbindung (SequenceFlow) mit ihren visuellen Darstellungskoordinaten (SequenceFlowWaypoints). Sie ermöglicht das Zeichnen von Verbindungslinien im Prozesseditor, die nicht geradlinig sind.

dazugehörige ENTITÄTEN: SequenceFlow, SequenceFlowWaypoints

VON - NACH - BEZIEHUNG: SequenceFlow hat SequenceFlowWaypoints

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: SequenceFlowWaypoints gehört zu SequenceFlow

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Ein SequenceFlowWaypoints-Datensatz kann nur existieren, wenn die zugehörige SequenceFlow-Verbindung (identifiziert durch MCA_AGL_AG_PARENT_ID und MCA_AGL_AG_ID) existiert. Das Löschen eines SequenceFlow muss zum Löschen aller zugehörigen Wegpunkte führen.

4.15. User - QM-Processes | bearbeitet

RELATION: User bearbeitet QM-Processes

DEFINITION, ZWECK: Diese Beziehung dient der detaillierten Nachvollziehbarkeit auf der Ebene einzelner Prozessschritte. Sie protokolliert, welcher Benutzer die letzte inhaltliche oder strukturelle Änderung an einem QM-Prozess vorgenommen hat. Dies ist entscheidend für Audits, die Fehleranalyse und die Verfolgung von Prozessänderungen über die Zeit.

dazugehörige ENTITÄTEN: User, QM-Processes

VON - NACH - BEZIEHUNG: User bearbeitet QM-Processes

Kardinalität: min: 0 max: m

UMGEKEHRTE BEZIEHUNG: QM-Processes wird bearbeitet von User

Kardinalität: min: 1 max: 1

INTEGRITÄTSREGELN:

Der MCA_PCK_LETZTERUSER in der QM-Processes-Tabelle muss auf einen gültigen, existierenden UserId in der User-Tabelle verweisen.

.

5. Datenkatalog

5.1. User

Spaltenname	Datentyp	Beschreibung	Null abel	Schlüssel/Einschränkung
UserId	VARCHAR (10)	Die Personalid des Users	Nein	Primary Key
FirstName	VARCHAR (45)	Der Vorname der Person	Ja	
LastName	VARCHAR (45)	Der Nachname der Person		

5.2. HierachyLevels

Spaltenname	Datentyp	Beschreibung	Null abel	Schlüssel/Einschränkung
MCA_APA_AP_ID	BIGINT	Die eindeutige Id der Ebene	Nein	PRIMARY KEY
MCA_APA_AP_PARENT_ID	BIGINT	Die Hierachieebene eins über dieser. Der Parent der obersten Hoierachieebene ist er selbst.	Nein	FOREIGN KEY (zu HierachyLevels.MCA_APA_AP_ID)
MCU_PCK_HIERARCHIE_EBENE	VARCHAR (45)	Die Nummer von Ebenen über dieser plus 1 für die aktuelle Ebene	Nein	FOREIGN KEY (zu HierachyLevels.MCA_APA_AP_ID)
MCA_APA_AENDERUNGSDATUM	DATETIME	Das Datum an dem die Ebene das letzte Mal geändert wurde	Nein	DEFAULT CURRENT_TIMESTAMP, ON UPDATE CURRENT_TIMESTAMP (updates the field automatically to the current time when the table is updated)
MCA_APA_ERSTELLDATUM	DATETIME	Das Datum an dem die Ebene erstellt wurde	Nein	DEFAULT CURRENT_TIMESTAMP
MCA_PCK_PROJEKTID	BIGINT	In welchen Projekt die Ebene ist		

5.3. VERFARHREN_KENNZEICHNUNG_LOOKUP

Spaltenname	Datentyp	Beschreibung	Null abel	Schlüssel/Einschränkung

MCA_PCK_VERFA_KZ	TINYINT	Welche Art von BPMN Element der QM-Prozess ist	Nein	PRIMARY KEY
Name	VARCHAR (45)	Die namentliche Artbezeichnung des BPMN-Elements	Nein	UNIQUE
Description	MEDIUMTEXT	Die Beschreibung welche Funktion dieses Verfahren hat	Ja	

5.4. Gatewaytpes

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung
MCA_PCK_GATEWAYTYP	TINYINT	Wenn das BPMN Elemen ein Meilenstein ist, kennzeichnet dieses Attribut von Event/Gateway das BPMN-Element ist	Nein	PRIMARY KEY
Bezeichnung	VARCHAR (45)	Der Name des Gateways/Events	Nein	UNIQUE
Description	MEDIUMTEXT	Die Beschreibung wie dieses Event/Gateway funktioniert	Ja	

5.5. QM-Project_Identity

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung
MCA_PCK_PROJEKTID	BIGINT	Eine eindeutige generierte id	Nein	PRIMARY KEY
MCU_PCK_PROJEKTNR	INT	Projektnummer die dem Projekt zugewiesen wurde	Nein	UNIQUE
MCU_PCK_PROJEKTBEZ	VARCHAR (45)	Der Name oder der Term der das Projekt bezeichnet	Nein	UNIQUE

5.6. QM-Project

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung
MCA_PCK_PROJEKTID	BIGINT	Die eindeutige Projekt-ID	Nein	PRIMARY KEY, FOREIGN KEY (zu QM-

				Project_Identity.MCA_PCK_PROJEKTID)
MCA_APA_LETZTERUSER	VARCHAR (10)	Der User der im Projekt als letzter etwas geändert hat	Nein	FOREIGN KEY (zu User.UserId)
MCA_APA_ERSTELLUSER	VARCHAR (10)	Der User der das Projekt erstellt hat	Nein	FOREIGN KEY (zu User.UserId)
MCA_APA_VERANTW_PERSONENID	VARCHAR (10)	Der User der für das Projekt verantwortlich ist	Nein	FOREIGN KEY (zu User.UserId)
MCA_PCK_LEISTUNG_FIRMENNR	BIGINT	Die Firma (falls es eine gibt) für den der QM-Prozess gemacht wird	Ja	FOREIGN KEY (zu Company.MCA_PCK_FIRMENNR)

5.7. Company

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung
MCA_PCK_FIRMENNR	BIGINT	Eine eindeutige Kennung, die von BMD zur internen Verwaltung des Kunden vergeben wird.	Nein	PRIMARY KEY
Firmenname	VARCHAR (45)	Der Name der Firma	Ja	UNIQUE

5.8. DISPVALUE_LOOKUP

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung
MCU_AGL_DISPVALUE	TINYINT	Dieser Wert bestimmt welche Art von Verbindung es ist	Nein	PRIMARY KEY
DISPVALUE_DESCRIPTION	VARCHAR (255)	Die Beschreibung was diese Verbindung ausmacht	Ja	UNIQUE
DISPVALUE_ABBREVIATION	VARCHAR (45)	Die Abkürzung für die Verbindungsart	Nein	UNIQUE

5.9. SequenceFlow

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung

MCA_AGL_AG_PARENT_ID	BIGINT	Der Prozess von dem die Verbindung ausgeht	Nein	Zusammengesetzter PRIMARY KEY, FOREIGN KEY (zu QM-Processes.MCA_PCK_PPSKAPAID)
MCA_AGL_AG_ID	BIGINT	Der QM-Process an den die Verbindung geht	Nein	Zusammengesetzter PRIMARY KEY, FOREIGN KEY (zu QM-Processes.MCA_PCK_PPSKAPAID)
MCA_AGL_FREIFELDA01_250	VARCHAR (45)	Die Beschreibung der Verbindung	Ja	
DISPVALUE_LOOKUP_MCU_AGL_DISPVALUE	TINYINT	Die Art der Verbindung	Nein	FOREIGN KEY (zu DISPVALUE_LOOKUP.MCU_AGL_DISPVALUE)

5.10. SequenceFlowWaypoints

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung
PointId	INT	Eine automatisch generierte Id	Nein	PRIMARY KEY, AUTO_INCREMENT
MCA_AGL_AG_PARENT_ID	BIGINT	Der Prozess von dem die Verbindung ausgeht	Nein	FOREIGN KEY (zu SequenceFlow)
MCA_AGL_AG_ID	BIGINT	Ziel der Verbindung	Nein	FOREIGN KEY (zu SequenceFlow)
x	INT	X Coordinate wo der Punkt des Sequence Flows platziert wird	Nein	
y	INT	y Coordinate wo der Punkt des Sequence Flows platziert wird	Nein	

5.11. QM-Processes

Spaltenname	Datentyp	Beschreibung	Nullabel	Schlüssel/Einschränkung
MCA_PCK_PPSKAPAID	BIGINT	Die eindeutige id die generiert wird (Länge 15 Nummern)	Nein	PRIMARY KEY
MCA_PCK_AG_BEZ	VARCHAR (45)	Die BPMN-Prozess Beschreibung	Nein	
MCU_PCK_AG_NACH	VARCHAR (45)	Die Matchcodes der nächsten folgenden Prozesse (diese ist ziemlich unzulässig da die Matchcodes und diese Spalte nicht automatisch		

		genereiert werden)		
MCU_PCK_HIERARCHIE_PCT	VARCHAR (255)	Beschreibt die Position und Ebene eines QM-Prozesses	Nein	UNIQUE (mit MCA_PCK_PROJEKTID)
MCA_PCK_AG_LFDNR	INT	Die laufende Nummer eines BPMN-Elementss	Nein	UNIQUE (mit MCA_PCK_PROJEKTID)
MCA_PCK_MATCHCODE	VARCHAR (45)	Der Matchcode von dem QM-Prozess	Ja	
MCU_PCK_AG_VORG	VARCHAR (45)	Matchcode von den Vorgaengern (diese ist ziemlich unzulässig da die Matchcodes und diese Spalte nicht automatisch genereiert werden)	Ja	
MCA_PCK_PROJEKTID	BIGINT	Die Id des QM-Projektes zu dem der Prozess gehört	Nein	FOREIGN KEY (zu QM-Project_Identity.MCA_PCK_PROJEKTID)
MCU_PCK_RTF_HTML	LONGTEXT	Beschreibung des Prozess in HTML	Nein	
MCA_PCK_ERSTELLUSER	VARCHAR (10)	Der User der den QM-Prozess erstellt hat	Nein	FOREIGN KEY (zu User.UserId)
MCA_PCK_LETZTERUSER	VARCHAR (10)	Der User der den QM-Prozess erstellt hat	Nein	FOREIGN KEY (zu User.UserId)
MCA_PCK_SOLL_START	DATETIME	Tag an dem der QM-Prozess gestartet werden soll	Nein	
MCA_PCK_SOLL_ENDE	DATETIME	Tag an dem der QM-Prozess fertig werden soll	Nein	
MCA_PCK_SOLL_MA_ZEIT	TIME	Die Arbeitszeit die für diesen Schritt gebraucht werden soll	Ja	
MCA_APA_AP_ID	BIGINT	Die Id der Ebenen auf dem sich das BPMN-Element befindet	Nein	FOREIGN KEY (zu HierachyLevels.MCA_APA_AP_ID)
MCA_PCK_VERFA_KZ	TINYINT	Die Art des BPMN-Elements	Nein	FOREIGN KEY (zu VERFARHREN_KENNZEICHNUNG_LOOKUP.MCA_PCK_VERFA_KZ)
MCA_PCK_AENDERUNGSDATUM	DATETIME	Zeit und Datum der letzten Änderung des QM-Prozesses	Nein	DEFAULT CURRENT_TIMESTMP ON UPDATE CURRENT_TIMESTMP

MCA_PCK_ERSTELLDATUM	TINYINT	Zeit und Datum der letzten Änderung des QM-Prozesses	Nein	DEFAULT CURRENT_TIMESTAMP
MCA_PCK_GATEWAYTYP	BIGINT	Wenn Meilenstein, markiert dies die Art von Event/Gateway	Ja	FOREIGN KEY (zu QM-Processes.MCA_PCK_PPSKAPAID)
ParentProcess	TINYINT	Speichert den übergeordneten Prozess	Nein	FOREIGN KEY (zu QM-Processes.MCA_PCK_PPSKAPAID)
width	TINYINT	Die weite die das bpmn Element von anfang der x achse hat bis zum gewollten Ende Nullable, da es in der NTCS erstellt werden kann	Nein	
height	TINYINT	Die höhe die das bpmn Element von anfang der y achse hat bis zum gewollten Ende Nullable, da es in der NTCS erstellt werden kann	Ja	
x	INT	x Coordinate wo der Start des BPMN-ELements ist	Ja	
y	INT	y Coordinate wo der Start des BPMN-ELements ist	Ja	

6. CRUD Matrix

Prozess	User	Company	QM-Project_Identity	QM-Project	QM-Processes	SequenceFlow	SequenceFlowWaypoints	HierachyLevels	VERFARHREN_KENNZEICHNUNG	Gatewaytypes	DISPVALUE_LOOKUP
benutzerAnlegen()	C										
firmaVerwalten()		CR UD									
projektAnlegen()	R	R	C	C							
projektdetailsBearbeiten()	R	R	U	U							
projektViusalisieren()	R	R	R	R	R	R	R	R	R	R	R
prozessErstellen()	R		R	R	C			R/ C	R	R	
prozessVerbinden()					R	C	C		R	R	R
prozessBearbeitein()	R		R	R	U						R
prozessLöschen()					D	D	D				
projektLöschen()			D	D	D	D	D				

7. SQL Create Script

```
-- MySQL Script generated by MySQL Workbench
-- Fri Oct 24 05:00:14 2025
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_
DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBST
ITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `mydb` ;

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`User`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`User` ;

CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `UserId` VARCHAR(10) NOT NULL COMMENT 'Die Personalid des Users',
  `FirstName` VARCHAR(45) NULL COMMENT 'Der Vorname der Person',
  `LastName` VARCHAR(45) NULL COMMENT 'der Nachname der Person',
  PRIMARY KEY (`UserId`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Company`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Company` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Company` (
  `MCA_PCK_FIRMENNR` BIGINT NOT NULL COMMENT 'Die Zugeteilte Firmennnummer',
  `Firmenname` VARCHAR(45) NULL COMMENT 'Der Name der Firma',
  PRIMARY KEY (`MCA_PCK_FIRMENNR`),
  UNIQUE INDEX `Firmenname_UNIQUE` (`Firmenname` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`QM-Project_Identity`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`QM-Project_Identity` ;

CREATE TABLE IF NOT EXISTS `mydb`.`QM-Project_Identity` (
  `MCA_PCK_PROJEKTID` BIGINT NOT NULL COMMENT 'Eine unique id die generiert wird',
  `MCU_PCK_PROJEKTNR` INT NOT NULL COMMENT 'Projektnummer die dem Projekt zugewiesen wurde',
  `MCU_PCK_PROJEKTBEZ` VARCHAR(45) NOT NULL COMMENT 'Der Name oder der Term der das Projekt bezeichnet',
  PRIMARY KEY (`MCA_PCK_PROJEKTID`),
  UNIQUE INDEX `MCU_PCK_PROJEKTNR_UNIQUE` (`MCU_PCK_PROJEKTNR` ASC) VISIBLE,
  UNIQUE INDEX `MCU_PCK_PROJEKTBEZ_UNIQUE` (`MCU_PCK_PROJEKTBEZ` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`QM-Project`
```

```
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`QM-Project` ;


CREATE TABLE IF NOT EXISTS `mydb`.`QM-Project` (
  `MCA_APA_LETZTERUSER` VARCHAR(10) NOT NULL COMMENT 'Der User der im Projekt als letzter etwas geändert hat',
  `MCA_APA_ERSTELLUSER` VARCHAR(10) NOT NULL COMMENT 'Der User der das Projekt erstellt hat',
  `MCA_APA_VERANTW_PERSONENID` VARCHAR(10) NOT NULL COMMENT 'Der User der für das Projekt verantwortlich ist',
  `MCA_PCK_LEISTUNG_FIRMENNR` BIGINT NULL COMMENT 'Die Firma (falls es eine gibt) für den der QM-Prozess gemacht wird',
  `MCA_PCK_PROJEKTID` BIGINT NOT NULL,
  PRIMARY KEY (`MCA_PCK_PROJEKTID`),
  INDEX `fk_QM-Project_User1_idx` (`MCA_APA_LETZTERUSER` ASC) VISIBLE,
  INDEX `fk_QM-Project_User2_idx` (`MCA_APA_ERSTELLUSER` ASC) VISIBLE,
  INDEX `fk_QM-Project_User3_idx` (`MCA_APA_VERANTW_PERSONENID` ASC) VISIBLE,
  INDEX `fk_QM-Project_Company1_idx` (`MCA_PCK_LEISTUNG_FIRMENNR` ASC) VISIBLE,
  INDEX `fk_QM-Project_QM-Project_Identity1_idx` (`MCA_PCK_PROJEKTID` ASC) VISIBLE,
  CONSTRAINT `fk_QM-Project_User1`
    FOREIGN KEY (`MCA_APA_LETZTERUSER`)
    REFERENCES `mydb`.`User` (`UserId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Project_User2`
    FOREIGN KEY (`MCA_APA_ERSTELLUSER`)
    REFERENCES `mydb`.`User` (`UserId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Project_User3`
    FOREIGN KEY (`MCA_APA_VERANTW_PERSONENID`)
    REFERENCES `mydb`.`User` (`UserId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Project_Company1`
```

```
    FOREIGN KEY (`MCA_PCK_LEISTUNG_FIRMENNR`)
    REFERENCES `mydb`.`Company` (`MCA_PCK_FIRMENNR`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Project_QM-Project_Identity1`
    FOREIGN KEY (`MCA_PCK_PROJEKTID`)
    REFERENCES `mydb`.`QM-Project_Identity` (`MCA_PCK_PROJEKTID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`HierachyLevels`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`HierachyLevels` ;


CREATE TABLE IF NOT EXISTS `mydb`.`HierachyLevels` (
  `MCA_APA_AP_ID` BIGINT NOT NULL COMMENT 'Die eindeutige Id der Ebene',
  `MCA_APA_AP_PARENT_ID` BIGINT NOT NULL COMMENT 'Die Hierachieebene eins über dieser\nDer Parent der obersten Hoierachieebene ist er selbst',
  `MCU_PCK_HIERARCHIE_EBENE` VARCHAR(45) NOT NULL COMMENT 'Wie viele Ebenen darüber sind +1',
  `MCA_APA_AENDERUNGSDATUM` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT 'Das Datum an dem die Ebene das letzte Mal geändert wurde',
  `MCA_APA_ERSTELLDATUM` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Das Datum an dem die Ebene erstellt wurde',
  `MCA_PCK_PROJEKTID` BIGINT NOT NULL,
  PRIMARY KEY (`MCA_APA_AP_ID`),
  INDEX `fk_HierachyLevels_HierachyLevels1_idx` (`MCA_APA_AP_PARENT_ID` ASC) VISIBLE,
  INDEX `fk_HierachyLevels_QM-Project1_idx` (`MCA_PCK_PROJEKTID` ASC) VISIBLE,
  CONSTRAINT `fk_HierachyLevels_HierachyLevels1`
    FOREIGN KEY (`MCA_APA_AP_PARENT_ID`)
    REFERENCES `mydb`.`HierachyLevels` (`MCA_APA_AP_ID`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_HierachyLevels_QM-Project1`
    FOREIGN KEY (`MCA_PCK_PROJEKTID`)
    REFERENCES `mydb`.`QM-Project` (`MCA_PCK_PROJEKTID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`VERFARHREN_KENNZEICHNUNG_LOOKUP`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`VERFARHREN_KENNZEICHNUNG_LOOKUP` ;


CREATE TABLE IF NOT EXISTS
`mydb`.`VERFARHREN_KENNZEICHNUNG_LOOKUP` (
  `MCA_PCK_VERFA_KZ` TINYINT NOT NULL COMMENT 'Welche Art von BPMN
Element der QM-Prozess ist\n',
  `Name` VARCHAR(45) NOT NULL COMMENT 'Die namentliche Artbezeichnung
des BPMN-Elements',
  `Description` MEDIUMTEXT NULL COMMENT 'Die Beschreibung welche Funktion
dieses Verfahren hat',
  PRIMARY KEY (`MCA_PCK_VERFA_KZ`),
  UNIQUE INDEX `Name_UNIQUE` (`Name` ASC) VISIBLE)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Gatewaytpes`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Gatewaytpes` ;


CREATE TABLE IF NOT EXISTS `mydb`.`Gatewaytpes` (
  `MCA_PCK_GATEWAYTYP` TINYINT NOT NULL COMMENT 'Wenn das BPMN
Elemen ein Meilenstein ist, kennzeichnet dieses Attribut von Event/Gateway das
BPMN-Element ist',
  `Bezeichnung` VARCHAR(45) NOT NULL,
```

```
  `Description` MEDIUMTEXT NULL COMMENT 'Die Beschreibung wie dieses Event/Gatewazy funktioniert',
  PRIMARY KEY (`MCA_PCK_GATEWAYTYP`),
  UNIQUE INDEX `Bezeichnung_UNIQUE` (`Bezeichnung` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`QM-Processes`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`QM-Processes` ;

CREATE TABLE IF NOT EXISTS `mydb`.`QM-Processes` (
  `MCA_PCK_PPSKAPAID` BIGINT NOT NULL COMMENT 'Die eindeutige id die generiert wird \nLänge 15 Nummern',
  `MCA_PCK_AG_BEZ` VARCHAR(45) NOT NULL COMMENT 'Die BPMN-Prozess Beschreibung',
  `MCU_PCK_AG_NACH` VARCHAR(45) NULL COMMENT 'Die Matchcodes der nächsten folgenden Prozesse\n(diese ist ziemlich unzulässig da die Matchcodes und diese Spalte nicht automatisch genereiert werden)\n',
  `MCU_PCK_HIERARCHIE_PCT` VARCHAR(255) NOT NULL COMMENT 'Dieses Feld beschreibt an welcher position und ebene ein qm prozess ist\nes besteht aus der PCT des Parent Processes wenn es einen gibt dann \".\" gefolgt von der laufenden Nummer des Prozesses',
  `MCA_PCK_AG_LFDNR` INT NOT NULL COMMENT 'Die laufende Nummer eines Bpmn-Elements wird bestimmt von der Reihenfolge in dem die Prozesse im NTCS eingetragen sind.\nDiese Nummer folgt aber nicht dem Sequence Flow',
  `MCA_PCK_MATCHCODE` VARCHAR(45) NULL COMMENT 'Der Matchcode von dem QM-Prozess',
  `MCU_PCK_AG_VORG` VARCHAR(45) NULL COMMENT 'Matchcode von den Vorgaengern\n(diese ist ziemlich unzulässig da die Matchcodes und diese Spalte nicht automatisch genereiert werden)',
  `MCA_PCK_PROJEKTID` BIGINT NOT NULL COMMENT 'Die Id des QM-Projektes zu dem der Prozess gehört',
  `MCU_PCK_RTF_HTML` LONGTEXT NULL COMMENT 'Beschreibung des Prozess in HTML',
  `MCA_PCK_ERSTELLUSER` VARCHAR(10) NOT NULL COMMENT 'Der User der den QM-Prozess erstellt hat',
  `MCA_PCK_LETZTERUSER` VARCHAR(10) NOT NULL COMMENT 'Der User der den QM-Prozess erstellt hat',
```

```
  `MCA_PCK_SOLL_START` DATETIME NOT NULL COMMENT 'Tag an dem der QM-Prozess gestartet weden soll',
  `MCA_PCK_SOLL_ENDE` DATETIME NOT NULL COMMENT 'Tag an dem der QM-Prozess fertig weden soll',
  `MCA_PCK_SOLL_MA_ZEIT` TIME NULL COMMENT 'Die Arbeitszeit die für diesen schritt gebraucht werden soll',
  `MCA_APA_AP_ID` BIGINT NOT NULL COMMENT 'Die Id der Ebenen auf dem sich das BPMN-Element befindet',
  `MCA_PCK_VERFA_KZ` TINYINT NOT NULL COMMENT 'Die Art des BPMN-Elements',
  `MCA_PCK_AENDERUNGSDATUM` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT 'Die Zeit und das Dateum der letzten Änderung des QM-Prozesses',
  `MCA_PCK_ERSTELLDATUM` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Die Zeit und das Dateum der Erstellung des QM-Prozesses',
  `MCA_PCK_GATEWAYTYP` TINYINT NOT NULL COMMENT 'Wenn es ein Meilenstein Verfahren Kennzeichen ist markiert die Wekche Art von Event/Gateway das BPMN-Element ist',
  `ParentProcess` BIGINT NOT NULL COMMENT 'Speichert den übergeordneten Prozess, wenn der QM-Prozess Teil eines Prozesses oder eines Subprozesses  ist.',
  `width` TINYINT NULL COMMENT 'Die weite die das bpmn Element von anfang der x achse hat bis zum gewollten Ende\nNullable, da es in der NTCS erstellt werden kann\n',
  `height` TINYINT NULL COMMENT 'Die höhe die das bpmn Element von anfang der y achse hat bis zum gewollten Ende\nNullable, da es in der NTCS erstellt werden kann\n',
  `x` INT NULL COMMENT 'x Coordinate wo der Start des BPMN-ELements ist',
  `y` INT NULL COMMENT 'y Coordinate wo der Start des BPMN-ELements ist',
  PRIMARY KEY (`MCA_PCK_PPSKAPAID`),
  INDEX `fk_QM-Processes_User1_idx` (`MCA_PCK_ERSTELLUSER` ASC) VISIBLE,
  INDEX `fk_QM-Processes_User2_idx` (`MCA_PCK_LETZTERUSER` ASC) VISIBLE,
  INDEX `fk_QM-Processes_HierachyLevels1_idx` (`MCA_APA_AP_ID` ASC) VISIBLE,
  INDEX `fk_QM-Processes_VERFARHREN_KENNZEICHNUNG_LOOKUP1_idx` (`MCA_PCK_VERFA_KZ` ASC) VISIBLE,
  INDEX `fk_QM-Processes_Gatewaytpes1_idx` (`MCA_PCK_GATEWAYTYP` ASC) VISIBLE,
  INDEX `fk_QM-Processes_QM-Processes1_idx` (`ParentProcess` ASC) VISIBLE,
  UNIQUE INDEX `uq_QM-Processes_LFDNR` (`MCA_PCK_AG_LFDNR` ASC, `MCA_PCK_PROJEKTID` ASC) INVISIBLE,
```

```
  UNIQUE INDEX `uq_QM-Processes_PCT` (`MCU_PCK_HIERARCHIE_PCT` ASC,
`MCA_PCK_PROJEKTID` ASC) VISIBLE,
  INDEX `fk_QM-Processes_QM-Projekt_idx` (`MCA_PCK_PROJEKTID` ASC)
VISIBLE,
  CONSTRAINT `fk_QM-Processes_User1`
    FOREIGN KEY (`MCA_PCK_ERSTELLUSER`)
    REFERENCES `mydb`.`User` (`UserId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Processes_User2`
    FOREIGN KEY (`MCA_PCK_LETZTERUSER`)
    REFERENCES `mydb`.`User` (`UserId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Processes_HierachyLevels1`
    FOREIGN KEY (`MCA_APA_AP_ID`)
    REFERENCES `mydb`.`HierachyLevels` (`MCA_APA_AP_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-
Processes_VERFARHREN_KENNZEICHNUNG_LOOKUP1`
    FOREIGN KEY (`MCA_PCK_VERFA_KZ`)
    REFERENCES `mydb`.`VERFARHREN_KENNZEICHNUNG_LOOKUP`
(`MCA_PCK_VERFA_KZ`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Processes_Gatewaytpes1`
    FOREIGN KEY (`MCA_PCK_GATEWAYTYP`)
    REFERENCES `mydb`.`Gatewaytpes` (`MCA_PCK_GATEWAYTYP`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Processes_QM-Processes1`
    FOREIGN KEY (`ParentProcess`)
    REFERENCES `mydb`.`QM-Processes` (`MCA_PCK_PPSKAPAID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Processes_QM-Projekt`
```

```
  FOREIGN KEY (`MCA_PCK_PROJEKTID`)
  REFERENCES `mydb`.`QM-Project_Identity` (`MCA_PCK_PROJEKTID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`DISPVALUE_LOOKUP`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`DISPVALUE_LOOKUP` ;

CREATE TABLE IF NOT EXISTS `mydb`.`DISPVALUE_LOOKUP` (
  `MCU_AGL_DISPVALUE` TINYINT NOT NULL COMMENT 'Dieser Wert bestimmt welche Art von Verbindung es ist',
  `DISPVALUE_DESCRIPTION` VARCHAR(255) NULL COMMENT 'Die Beschreibung was diese Verbindung ausmacht\n',
  `DISPVALUE_ABBREVIATION` VARCHAR(45) NOT NULL COMMENT 'Die Abkürzung für die Verbindungsart',
  PRIMARY KEY (`MCU_AGL_DISPVALUE`),
  UNIQUE INDEX `DISPVALUE_DESCRIPTION_UNIQUE` (`DISPVALUE_DESCRIPTION` ASC) VISIBLE,
  UNIQUE INDEX `DISPVALUE_ABBREVIATION_UNIQUE` (`DISPVALUE_ABBREVIATION` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`SequenceFlow`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`SequenceFlow` ;

CREATE TABLE IF NOT EXISTS `mydb`.`SequenceFlow` (
  `MCA_AGL_AG_PARENT_ID` BIGINT NOT NULL COMMENT 'Der Prozess von dem die Verbindung ausgeht',
  `MCA_AGL_AG_ID` BIGINT NOT NULL COMMENT 'Der QM-Process an den die Verbindung geht',
```

```
  `MCA_AGL_FREIFELDA01_250` VARCHAR(45) NULL COMMENT 'Die
Beschreibung der Verbindung',
  `DISPVALUE_LOOKUP_MCU_AGL_DISPVALUE` TINYINT NOT NULL COMMENT
'Die Art der Verbindung',
  PRIMARY KEY (`MCA_AGL_AG_PARENT_ID`, `MCA_AGL_AG_ID`),
  INDEX `fk_QM-Processes_has_QM-Processes_QM-Processes1_idx`
(`MCA_AGL_AG_ID` ASC) VISIBLE,
  INDEX `fk_QM-Processes_has_QM-Processes_QM-Processes_idx`
(`MCA_AGL_AG_PARENT_ID` ASC) VISIBLE,
  INDEX `fk_QM-Processes_has_QM-Processes_DISPVALUE_LOOKUP(Self
crea_idx` (`DISPVALUE_LOOKUP_MCU_AGL_DISPVALUE` ASC) VISIBLE,
  CONSTRAINT `fk_QM-Processes_has_QM-Processes_QM-Processes`
    FOREIGN KEY (`MCA_AGL_AG_PARENT_ID`)
    REFERENCES `mydb`.`QM-Processes` (`MCA_PCK_PPSKAPAID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Processes_has_QM-Processes_QM-Processes1`
    FOREIGN KEY (`MCA_AGL_AG_ID`)
    REFERENCES `mydb`.`QM-Processes` (`MCA_PCK_PPSKAPAID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_QM-Processes_has_QM-
Processes_DISPVALUE_LOOKUP(Self create1`
    FOREIGN KEY (`DISPVALUE_LOOKUP_MCU_AGL_DISPVALUE`)
    REFERENCES `mydb`.`DISPVALUE_LOOKUP` (`MCU_AGL_DISPVALUE`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`SequenceFlowWaypoints`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`SequenceFlowWaypoints` ;

CREATE TABLE IF NOT EXISTS `mydb`.`SequenceFlowWaypoints` (
  `MCA_AGL_AG_PARENT_ID` BIGINT NOT NULL COMMENT 'Source von dem der
',
```

```
  `MCA_AGL_AG_ID` BIGINT NOT NULL,
  `x` INT NOT NULL COMMENT 'X Coordinate wo der Punkt des Sequence Flows platziert wird',
  `y` INT NOT NULL COMMENT 'y Coordinate wo der Punkt des Sequence Flows platziert wird',
  `PointId` INT NOT NULL AUTO_INCREMENT COMMENT 'Eine automatisch generierte Id ',
  PRIMARY KEY (`PointId`),
  CONSTRAINT `fk_SequenceFlowWaypoints_SequenceFlow1`
    FOREIGN KEY (`MCA_AGL_AG_PARENT_ID` , `MCA_AGL_AG_ID`)
    REFERENCES `mydb`.`SequenceFlow` (`MCA_AGL_AG_PARENT_ID` , `MCA_AGL_AG_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

8. Seeding

```
-- -----------------------------------------------------------------
-- SQL Seeding-Skript für die Nachschlagetabellen der 'mydb'-Datenbank
-- Füllt die Tabellen mit den initialen, vordefinierten Werten.
-- -----------------------------------------------------------------

USE `mydb`;

-- Um die Datenintegrität sicherzustellen, wird der gesamte Vorgang in einer Transaktion gekapselt.
START TRANSACTION;

-- 1. Befüllen der Tabelle `VERFARHREN_KENNZEICHNUNG_LOOKUP`
```

-- HINWEIS: Das bereitgestellte Enum `BpmnNodeType` scheint die BPMN-Grundtypen "Event" und "Gateway" nicht vollständig abzubilden.

-- "Milestone" ist typischerweise eine Art von "Event/Gateway".

-- Dieses Skript verwendet eine Standard-BPMN-konforme Befüllung, die zu Ihrem Schema passt.

-- Task = 1 und Subprozess = 4 wurden aus dem Enum übernommen.

INSERT INTO `VERFARHREN_KENNZEICHNUNG_LOOKUP` (`MCA_PCK_VERFA_KZ`, `Name`, `Description`) VALUES

(1, 'Task', 'Eine atomare Aufgabe, die von einem Benutzer oder System ausgeführt wird.'),

(2, 'Event', 'Ein Ereignis, das etwas auslöst, z.B. einen Start, ein Ende oder eine Zwischennachricht.'),

(3, 'Gateway', 'Ein Entscheidungspunkt, der den Fluss des Prozesses aufteilt oder zusammenführt.'),

(4, 'Subprozess', 'Ein eigenständiger Prozess, der innerhalb dieses Prozesses gekapselt ist.');

-- 2. Befüllen der Tabelle `Gatewaytpes`

-- Diese Werte stammen direkt aus dem `BpmnGatewayTypes`-Enum.

INSERT INTO `Gatewaytpes` (`MCA_PCK_GATEWAYTYP`, `Bezeichnung`, `Description`) VALUES

(0, 'Milestone', 'Ein spezifisches Ereignis, das einen wichtigen Fortschritt oder Abschluss markiert.'),

(1, 'Exclusive', 'Genau ein ausgehender Pfad wird basierend auf einer Bedingung gewählt (XOR).'),

(2, 'EventBased', 'Der ausgehende Pfad wird basierend auf dem Eintreten eines nachfolgenden Events gewählt.'),

(3, 'Inclusive', 'Ein oder mehrere ausgehende Pfade werden basierend auf Bedingungen aktiviert (OR).'),

(5, 'Parallel', 'Alle ausgehenden Pfade werden gleichzeitig und ohne Bedingung aktiviert (AND).'),

(6, 'Complex', 'Eine komplexe Verzweigungs- oder Zusammenführungslogik, die über die anderen Typen hinausgeht.');

-- 3. Befüllen der Tabelle `DISPVALUE_LOOKUP`

-- Diese Werte stammen direkt aus der Liste der Verbindungstypen.

```
-- Die ID (`MCU_AGL_DISPVALUE`) wurde sequenziell vergeben.

INSERT INTO `DISPVALUE_LOOKUP` (`MCU_AGL_DISPVALUE`, `DISPVALUE_DESCRIPTION`, `DISPVALUE_ABBREVIATION`) VALUES
(1, 'Anfang zu Anfang', 'AA'),
(2, 'Anfang zu Ende', 'AE'),
(3, 'Ende zu Anfang', 'EA'),
(4, 'Ende zu Ende', 'EE'); -- "Ene" wurde zu "Ende" korrigiert


-- Schließt die Transaktion ab und macht die Änderungen permanent.
COMMIT;

-- ------------------------------------------------------------------
-- Seeding-Skript erfolgreich abgeschlossen.
-- ------------------------------------------------------------------
```

D Projektinitialisierung

D.1 Projektdefinition

Hier befindet sich die vollständige Projektdefinition.

VisuPro

Projektdefinition

Alexander Vorreither

Erstelldatum: 28.06.2025

1. Inhaltsverzeichnis

2. Hintergrund

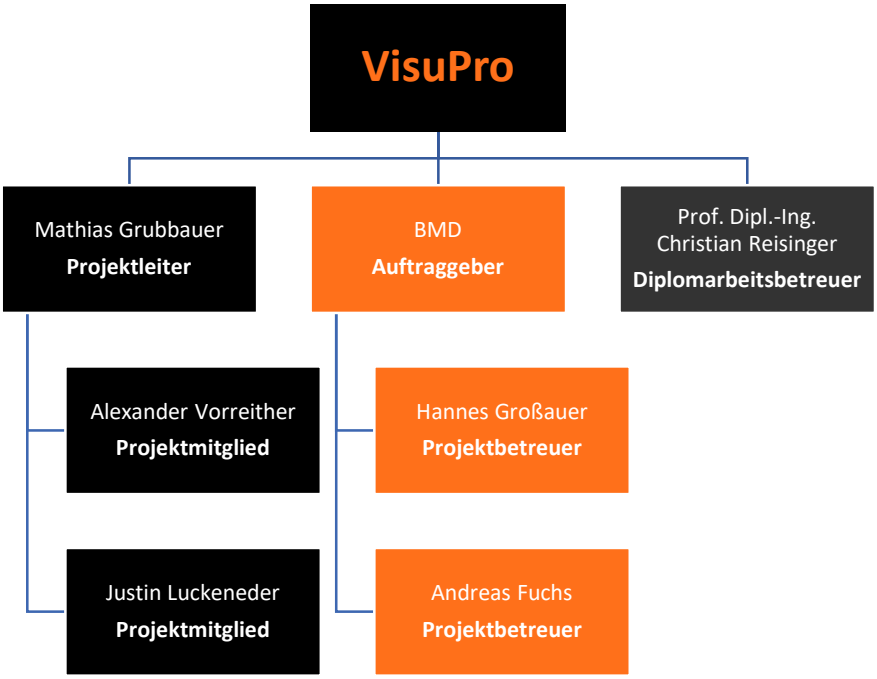
Die Projektidee entstand aus dem Bedarf innerhalb des Software-Unternehmens BMD, betriebliche Qualitätsmanagement-Prozesse (QM-Prozesse) übersichtlicher und anwenderfreundlicher darzustellen. Aktuell werden diese Abläufe in der BMD-Software zwar digital erfasst, jedoch überwiegend in tabellarischer oder Listenform dargestellt. Diese Darstellungsform ist insbesondere für Schulungszwecke, neue Mitarbeitende oder externe Prüfstellen wenig anschaulich und schwer nachvollziehbar.

Im Rahmen eines internen Projekts wurde daher das Ziel formuliert, eine Möglichkeit zur grafischen Visualisierung der QM-Prozesse zu schaffen. Diese Idee basiert auf der Erfahrung aus dem betrieblichen Alltag, in dem immer wieder deutlich wurde, dass eine einfache und verständliche Darstellung der Abläufe fehlt. Gerade bei standardisierten Prozessen – wie etwa bei Einschulungen neuer Mitarbeiter – kommt es auf klare Strukturen und transparente Prozessschritte an.

Um die Umsetzung zu realisieren, entwickeln die Projektteilnehmer im Zuge einer Diplomarbeit eine Anwendung, die es ermöglicht, bestehende QM-Prozesse in Form von BPMN-Diagrammen (Business Process Model and Notation) darzustellen. Die dafür notwendigen Prozessdaten werden aus der BMD-Software exportiert, aufbereitet und in ein standardisiertes Format überführt. Die Visualisierung erfolgt unter Einsatz gängiger Webtechnologien wie JavaScript / TypeScript in Kombination mit HTML / CSS und Modellierungswerkzeugen wie der Open-Source-Bibliothek bpmn.io, die eine interaktive Darstellung und Bearbeitung von BPMN-Prozessen ermöglicht.

Ein wichtiger Aspekt des Projekts ist, dass nicht nur fertige Modelle angezeigt, sondern direkt in der Anwendung bearbeitet und angepasst werden können – z. B. durch das Verschieben oder Ergänzen von Elementen. Dies soll zukünftig auch die Rückspeicherung in die Datenquelle ermöglichen (Proof of Concept).

3. Projektorganisation



4. Ziele

4.1. Geschäftsziele

- Verbesserung der Übersichtlichkeit und Verständlichkeit von QM-Prozessen im Unternehmen.
- Erhöhung der Qualität und Effizienz bei der Dokumentation von Abläufen.
- Unterstützung der internen Kommunikation durch anschauliche Prozessdarstellungen.
- Reduzierung des Aufwands für Dokumentation und Nachvollziehbarkeit betrieblicher Abläufe.

4.2. Ziele des Projekts

- Analyse und Auswahl praxisrelevanter QM-Prozesse zur Visualisierung (z. B. Einschulung, Zertifizierung).
- Extraktion und Aufbereitung der Prozessdaten aus der BMD-Software (via REST oder SQL-Zugriff).
- Entwicklung eines standardisierten BPMN-Modells für ausgewählte Prozesse.
- Implementierung eines Prototyps zur Darstellung und interaktiven Bearbeitung von BPMN-Diagrammen mit bpmn.io.
- Sicherstellung der Rückführbarkeit von Änderungen in die Datenbasis (Proof of Concept).

- Schaffung einer benutzerfreundlichen Oberfläche zur intuitiven Bearbeitung und Anzeige.
- Iterative Entwicklung nach agilen Prinzipien in enger Abstimmung mit den betreuenden Fachkräften.

4.3. Erfolgskriterien des Projekts

- Fehlerfreie Kompilierung
- Zeitgerechte Fertigstellung des Projekts
- Korrekte Anzeige und Bearbeitbarkeit der BPMN-Diagramme in der Anwendung.
- Positive Evaluation durch betreuende Fachkräfte im Hinblick auf Verständlichkeit und Praxistauglichkeit.
- System läuft stabil im prototypischen Einsatz und kann ggf. erweitert oder produktiv integriert werden.

5. Leistungsumfang

5.1. Leistungsumfang hinsichtlich der Geschäftsprozesse

In Scope	Out of Scope
Modellierung von QM-Prozesse (z.B. Einschulung, Zertifizierung)	Vollständige Neudefinition oder Reorganisation bestehender Prozesse
BPMN-basierte Visualisierung dokumentierter Abläufe	Vollständige Unternehmensprozessanalyse

5.2. Leistungsumfang hinsichtlich der Lokationen

In Scope	Out of Scope
Umsetzung und Nutzung am Hauptstandort (Steyr)	Einbindung mehrerer Standorte oder internationaler Einheiten

5.3. Leistungsumfang hinsichtlich der Applikation

In Scope	Out of Scope
Prototyp zur Anzeige und Bearbeitung von BPMN-Diagrammen	Produktivsystem oder vollständige Applikationsarchitektur
Nutzung von bpmn.io zur Diagrammbearbeitung	Eigene BPMN-Engine oder Entwicklung einer BPMN-Alternative

Grundfunktionen wie Prozessanzeige, Elementverschiebung, einfache Bearbeitung	Live-Synchronisation mit Produktivdatenbank
Lokale Speicherung / Proof-of-Concept der Rückführung	

5.4. Leistungsumfang hinsichtlich der Technologie

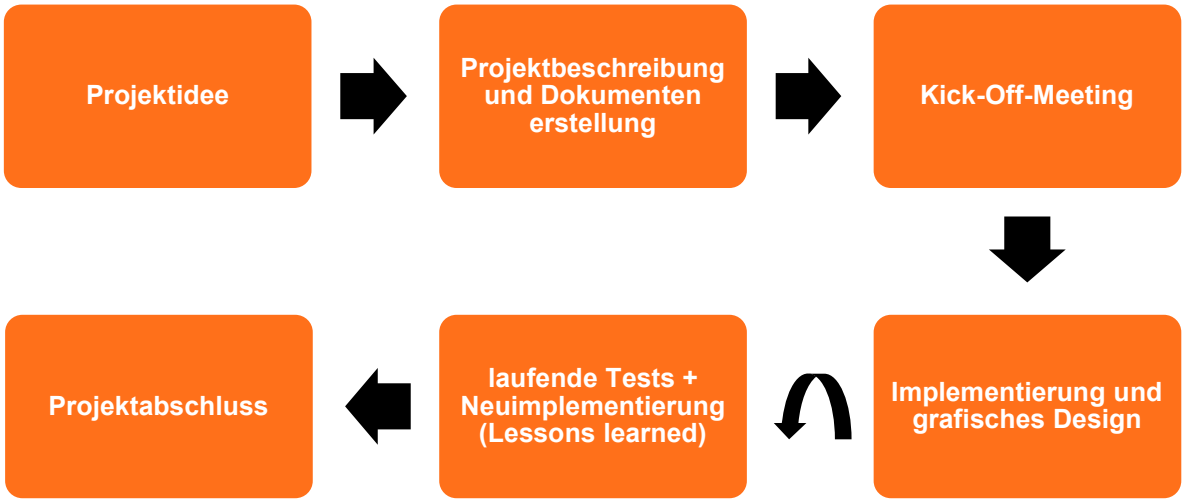
In Scope	Out of Scope
Verwendung von JavaScript/TypeScript, HTML/CSS	Einsatz nativer Mobile- oder Desktop-Apps ohne Web-Technologien
Nutzung der bpmn.io-Bibliothek zur BPMN-Modellierung	Verwendung proprietärer Modellierungswerkzeuge
Zugriff auf Daten via REST-API oder MS SQL Server	

5.5. Leistungsumfang hinsichtlich der Daten

In Scope	Out of Scope
Aufbereitung der Daten für BPMN-Diagramme	Vollautomatisierter Datenabgleich in Echtzeit
Speicherung veränderter Diagramme als strukturierte Daten	Vollständige semantische Prüfung aller Prozessregeln
Export ausgewählter QM-Prozessdaten	Verarbeitung unstrukturierter Daten (z. B. Freitexte, PDFs)

6. Vorgehensweise

6.1. Methodisch-technischer Ansatz



6.2. Projektmanagement Ansatz

6.2.1. Projektorganisation

Name	*Bereiche*
Mathias Grubbauer	Projektleiter, Datenaufbereitung, Datenbank
Alexander Vorreither	Diagramm-Integration & Anwendungslogik (BPMN)
Justin Luckeneder	Frontend & Benutzeroberfläche

6.2.2. Verfahren zur Projektfortschrittskontrolle

- Wöchentliche SCRUM-Meetings
- Überprüfung ob Meilensteine eingehalten werden (Meilensteinliste, Zeitplan)

6.2.3. Projektkommunikation und -berichtswesen

Mündlich:

- Telefongespräche
- Discord-Server

Schriftlich:

- E-Mail
- Discord-Server
- WhatsApp

Persönlich:

- SCRUM-Meetings, unregelmäßige Sitzungen
- Tägliche Meetings während Praktikumszeit

Berichtwesen:

- Verfassen von Statusberichten, auf Nachfrage

6.2.4. Problem- und Fehlermanagement

Probleme werden im Team, bzw. mit den betreuenden Fachkräften des Auftraggebers besprochen und es wird gemeinsam eine Lösung ausgearbeitet.

- Unit Tests

7. Risiken

Risiken können der Risikoinventar_VisuPro – Datei entnommen werden.

8. Annahmen

- Kein längerfristiger Ausfall eines oder mehrerer Teammitglieder
- Die für das Projekt relevanten QM-Prozesse sind in der BMD-Software bereits erfasst oder rekonstruierbar
- Benötigte Daten werden vom Auftraggeber zur Verfügung gestellt
- Verwendete kostenlose Lizenzen werden nicht kostenpflichtig

9. Einschränkungen

- Browserkompatibilität
- Die Anwendung wird vorrangig für Desktopbrowser optimiert; volle Kompatibilität mit mobilen Geräten ist nicht sichergestellt.
- Keine garantierte Unterstützung aller Browser oder Betriebssysteme – Fokus liegt auf aktuellen Chromium-basierten Browsern (z. B. Chrome, Brave), Firefox und Safari.

10. Kritische Erfolgskriterien

- Abschließen innerhalb des Zeitplans
- Funktionsfähige Features
- Übersichtliche Oberfläche

11. Technische Infrastruktur

11.1. Entwicklungsteam

Hardware

- PC / Laptop (Windows, Mac)
- Drucker
- Flashmedium
- Headset (mit Mikrofon)
- Internet-Infrastruktur (Router, WLAN-Access Point)
- Smartphones

Software

- Windows 10/11
- Microsoft 365
- PDF-Reader
- Git
- Browser
- DIE

11.2. Auftraggeber

Hardware

- PC / Laptop mit entsprechendem OS (Windows, Mac) in der jeweils aktuellen Version

Software

- Entsprechendes OS
- Aktueller Browser (Chromium-basiert, Firefox, Safari)

12. Unterschriften zum Projektstart

_____

BMD

_____

Mathias Grubbauer

_____

Prof. Dipl.-Ing, Christian Reisinger

13. Änderungsverzeichnis

Version	Autor	Datum	Inhalt
0.1	Alexander Vorreither	28.06.2025	Erstentwurf

D.2 CM-Plan

VisuPro

CM Plan

Mathias Grubbauer

Erstelldatum: 28.06.2025

1. Inhaltsverzeichnis

2. Einleitung

Dieser CM-Plan gilt für alle Teilprojekte des VisuPro Projekts.

3. Verwaltungseinheiten

3.1. Dokumentation

- Dokumente für die Projekt Initialisierung
- Meetings Protokolle
- Pflichtenheft
- Produkt Dokumentation
- Diplomarbeitsschreiben

3.2. Development

- Backend
- Webapp

4. Werkzeuge

4.1. Werkzeuge für Dokumente

Dokumente:

Dateiendung: .docx

Werkzeug: MS Word

Spreadsheet:

Dateiendung: .xls

Werkzeuge: Excel

Zeitdiagramme;

Dateiendung: .mpp

Werkzeug: Microsoft Projekts

Diagramme:

Dateiendung: .vsdx, .mdj

Werkzeuge: MS Visio, StarUML

Werkzeuge für Versionsverwaltung Git Versionsverwaltung

4.2. Dokumentennamen

{Dokumentname}.{File type}

Meetings Protokolle:

{Datum}_Meetings_Protokoll.docx

4.3. Abkürzungen

Abkürzung	Bezeichnung
PI-Phase	Projektinitialisierungsphase
DokN	Dokumentname
CM	Configuration-Management
RM	Risiko-Management

4.4. Änderungshistorie

Änderungen für Code und Dokumentation werden im internen Github von der BMD mitverfolgt

Änderungen für Dokumente außerhalb der BMD werden in einem Teaminternen Github mitverfolgt

5. Ablageschema

5.1. Internes Git:

- Branch: ProjectManagement
- Struktur:
    - Alles von der PI-Phase
        - Directory: ProjectInitialization
    - Meetings:
        - Directory: Meetings
        - Name: Meeting_{Date}

- Branch: DiplomaThesis
- Struktur:
    - Die individuellen Teile der Diplomarbeit
        - Directory: {Nachname des Schreibers}_DiplomaThesis

- Die vollständige Diplomarbeit
  - VisuPro_FinalDiplomaThesis.docx
  - VisuPro_FinalDiplomaThesis.pdf

5.2. BMD Git

Backend Development:

Alles, vom backend, an dem aktiv gearbeitet wird:

- Branch: BackendDevelopment
  - Project
  - Alles zur Datenbank

Web Development

- Branch: WebDevelopment
  - Javascript Webapp

Dokumentation

- Branch: Documentation
  - Alle Dokumentationen der Programme
  - Es muss im Dokumentnamen bekannt werden zu wo die Dokumentation dazugehört

5.3. Branch Berechtigungen

- Auf den main-Branch wird **nie** gepusht.
- Daten kommen auf den main-Branch nur mit Pull-Requests
- Die Pull-Requests müssen zuerst von einer Person durchgeschaut werden
- Am Ende ist das fertige Projekt im main-Branch unter einem Verzeichnis Project abzulegen, wobei in diesem Verzeichnis drei Subverzeichnisse für die Datenbank, die Webapp und das Backend zu unterteilen ist. Die Dokumentation ist im main-Branch, unter dem Verzeichnis Documentation, abzulegen.

6. Änderungsmanagement und Mängelverfolgung

6.1. Änderungsmanagement

Änderungen können während des Praktikums jederzeit genannt werden.

Zu berücksichtigen ist dabei, dass bei zu vielen oder zu großen Änderungen andere Features vielleicht nicht fertig werden.

6.2. Mängelverfolgung

Mängel können bei Meetings festgestellt werden welche danach bearbeitet werden können.

7. Git

7.1. Git commits

Git commits werden auf Englisch verfasst.

Standard Git commits

- Added { DokN }: Wenn Dokument hinzugefügt wird
- Changed {DokN}: {Change} Wenn Dokument geändert wird.
- Removed {DokN}: Wenn Dokument gelöscht wird.

- **Bei Code:** Nachricht soll beinhalten was geändert worden ist und den Status des geänderten Features.

7.2. Git merges

Inhalt:

- Detaillierte Beschreibung, was vom letzten Stand auf den Gewollten geändert worden ist.
- Detaillierte Beschreibung der Ziele des merges.
- Verfasst in Englisch

8. Allgemeines

8.1. Verwendete Sprachen

Code:	Englisch
Protokolle:	Deutsch

Verwendungsdokumentation:	Deutsch
Code-Dokumentation:	Englisch (und/oder Deutsch)
Diplomarbeit:	Deutsch

9. Änderungsverzeichnis

Autor	Datum	Inhalt
Mathias Grubbauer	29.06.2025	Ersterstellung des CM-Plans

D.3 PSP

VisuPro

PSP

Alexander Vorreither, Mathias Grubbauer, Justin Luckender

Erstelldatum: 29.06.2025

1. Inhaltsverzeichnis

2. Projektstrukturplan

1. Projektmanagement
   - 1.1. Projektinitialisierung
     - 1.1.1. Projektdefinition
     - 1.1.2. CM-Plan
     - 1.1.3. Projektstrukturplan
     - 1.1.4. Aufwandschätzung
     - 1.1.5. Zeitplan
     - 1.1.6. Risikomanagement
     - 1.1.7. Dokumentvorlagen
       - 1.1.7.1. Allgemeine Dokumentvorlage
       - 1.1.7.2. Meetings Protokoll Dokumentvorlage
     - 1.1.8. Kickoff
   - 1.2. Management
     - 1.2.1. Meetings
     - 1.2.2. Statusbericht
2. Backend
   - 2.1. Datenverwaltung
     - 2.1.1. Datenmodell erstellen
     - 2.1.2. Datenbank konfigurieren
     - 2.1.3. Datenbank einrichten
     - 2.1.4. Dokumentation
   - 2.2. Programm
     - 2.2.1. Rest Schnittpunkt
       - 2.2.1.1. Überführen der bereits gegeben unstrukturierten Daten in eine strukturierte Datenbank
       - 2.2.1.2. Ausgeben der Daten von der Datenbank
       - 2.2.1.3. Überführung von veränderten Daten
       - 2.2.1.4. Kreieren von neuen Daten
       - 2.2.1.5. Dokumentation
     - 2.2.2. Login
       - 2.2.2.1. Implementierung mit BMD internen Zertifizierungsverfahren
       - 2.2.2.2. Dokumentation
   - 2.3. Code prüfen
   - 2.4. Tests durchführen
   - 2.5. Bugs fixen
3. Web-Development
   - 3.1. UI/UX-Design
     - 3.1.1. Erstellen von Mockup mit Figma
     - 3.1.2. Abstimmung von Designvorgaben mit Stakeholdern
     - 3.1.3. Gestaltung visueller Komponenten (Buttons, Icons, Menüs)
   - 3.2. Dokumentation
   - 3.3. Layout-Implementierung
     - 3.3.1. Umsetzung des Figma-Designs in HTML/CSS/TypeScript
   - 3.4. Schnittstellenintegration
     - 3.4.1. Anbindung mit Backend über Schnittstellen

3. Änderungsverzeichnis

Autor	Datum	Inhalt
Mathias Grubbauer	29.06.2025	Erstentwurf

D.4 Risiko-Plan

VisuPro

Risikomanagement-Plan

Justin Luckeneder

Erstelldatum: 29.06.2025

1. Inhaltsverzeichnis

2. Risikoidentifikation

2.1. Technische Risiken:

- Unzureichende Kenntnisse
- Notwendige Zugriffe oder Hardware ist zum Teil oder vollständig nicht zugänglich

2.2. Organisatorische Risiken:

- Fehlende oder verspätete Absprache mit Auftraggeber
- Unklare Verantwortlichkeiten zwischen Projekt-Team und Auftraggeber

2.3. Inhaltliche Risiken:

- Prozesse sind nicht eindeutig dokumentiert oder widersprüchlich

2.4. Zeitliche Risiken:

- Zeitplan kann durch unvorhergesehene Schwierigkeiten nicht eingehalten werden

2.5. Qualitative Risiken:

- Nutzerfreundlichkeit nicht ausreichen gegeben

3. Risikobewertung

Risikobezeichnung	Eintrittswahrscheinlichkeit			Auswirkung auf das Projekt		
-	hoch	mittel	gering	hoch	mittel	gering
Zugriffe			x	x		
Fehlende Kenntnisse		x		x		
Absprache			x		x	
Verantwortlichkeiten		x				x
Dokumentation			x			x
Zeitplan			x	x		
Nutzerfreundlichkeit		x		x		

4. Maßnahmenplanung

4.1. Technische Risiken:

- Unzureichende Kenntnisse

Maßnahme: Recherchieren von den genutzten Technologien vor dem Praktikum.

- Notwendige Zugriffe oder Hardware ist zum Teil oder vollständig nicht zugänglich

Maßnahme: Mit Auftraggeber einen Lösungsweg finden; wiederholt erwähnen, um eine schnelle Lösung zu finden.

4.2. Organisatorische Risiken:

- Fehlende oder verspätete Absprache mit Auftraggeber

Maßnahme: Regelmäßige Meetings

- Unklare Verantwortlichkeiten zwischen Projekt-Team und Auftraggeber

Maßnahme: Klare Rollenverteilung zu Beginn, Ansprechpartner bestimmen

4.3. Inhaltliche Risiken:

- Prozesse sind nicht eindeutig dokumentiert oder widersprüchlich

Maßnahme: Meetings mit Ansprechpartnern, Abgleichen mit bestehenden Abläufen

4.4. Zeitliche Risiken:

- Zeitplan kann durch unvorhergesehene Schwierigkeiten nicht eingehalten werden

Maßnahme: Pufferzeiten einplanen, Überstunden

4.5. Qualitative Risiken:

- Nutzerfreundlichkeit nicht ausreichen gegeben

Maßnahme: Usability-Feedback einholen durch Tests, Figma-Mockup erstellen

5. Änderungsverzeichnis

Version	Autor	Datum	Inhalt
0.1	Justin Luckeneder	29.06.2025	Erstentwurf
1.0	Justin Luckeneder	29.06.2025	Finalversion

E Benutzerhandbuch

Benutzerhandbuch – VisuPro

1 Inhaltsverzeichnis

2 Überblick – Was ist VisuPro?

VisuPro ist ein BPMN-Prozessvisualisierungs-Editor, der in das BMD/NTCS-System integriert ist. Die Applikation ermöglicht es Ihnen, Geschäftsprozesse visuell darzustellen, zu bearbeiten und zu exportieren.

2.1 Zweck der Applikation

- Prozesse visualisieren: Laden Sie Prozesse aus der NTCS und sehen Sie diese als interaktive BPMN-Diagramme
- Prozesse bearbeiten: Passen Sie Prozessabläufe, Elemente und Metadaten an
- Prozesse erstellen: Erstellen Sie neue Prozesse von Grund auf
- Exportieren: Speichern Sie Ihre Arbeit lokal oder exportieren Sie sie als PNG

2.2 Systemvorraussetzungen

- Browser: Aktueller Webbrowser (Chrome, Firefox, Safari, Edge) mit JavaScript-Unterstützung, oder im Integration im NTCS-eigenen Webbrowser
- Netzwerkzugang: Verbindung zum BMD-Server und NTCS-System

3 Prozess laden

3.1 URL-Struktur

Um einen Prozess zu laden, verwenden Sie folgende URL:

```
/Visualisieren/?MCA_PRO_PROJEKTNR=<ID>
```

Ersetzen Sie <ID> durch die eindeutige Projektnummer des zu ladenden Prozesses.

3.2 Was passiert beim Laden

1. VisuPro ruft den Prozess aus dem BMD-Server ab
2. Die Prozessdaten werden automatisch vom BMD-Format in BPMN-Format konvertiert
3. Das Diagramm wird auf dem Canvas dargestellt und ist sofort einsatzbereit

3.3 Oberflächenüberblick

[Screenshot: Hauptoberfläche]

Die Benutzeroberfläche besteht aus fünf Hauptbereichen:

Toolbar (oben)

- Undo-/Redobutton (Pfeil zurück und vorwärts): Erlauben dem Benutzer Änderungen schrittweise rückgängig zu machen, beziehungsweise wiederherzustellen
- Speichern Button (Disketten-Icon): Speichert den aktuellen Prozess lokal als JSON-Datei, beziehungsweise speichert den Prozess zurück in die BMD-Datenhaltung

Werkzeugpanel / Palette (links)

- Enthält die verfügbaren BPMN-Elemente (Tasks, Gateways, Events, Subprozesse, Verbindungen)
- Zum hinzufügen neuer Elemente: Element aus der Palette auf den Canvas ziehen

Zoom-Buttons (unten rechts)

- + Button: Vergrößert die Ansicht um ein vordefiniertes Level
- - Button: Verkleinert die Ansicht
- Fit-to-View Button: Zoomt so, dass das ganze Diagramm im Fenster sichtbar ist

Canvas (Mitte)

- Der Bearbeitungsbereich, in dem das BPMN-Diagramm angezeigt wird
- Hier können Sie Elemente verschieben, bearbeiten, verbinden und hinzufügen

Properties Panel (rechts)

- Zeigt Eigenschaften des aktuell gewählten Elements
- Enthält BMD-spezifische Felder (Name, Beschreibung, etc.)
- Ermöglicht die Bearbeitung von Metadaten

3.4 Zoom und Navigation

Mit der Maus

- Scroll-Rad: Die View im Canvas nach oben (hochrollen) oder nach unten schieben (hinunter). Halten Sie Strg gedrückt und zoomen Sie in (hochrollen) und aus (hinunten) dem Diagramm
- Drag & Drop: Halten Sie die linke Maustaste gedrückt (ohne ein Element auszuwählen) und bewegen Sie die Maus, um im Canvas zu navigieren

Mit Toolbar-Buttons

- + Button: Vergrößert die Ansicht um ein vordefiniertes Level
- - Button: Verkleinert die Ansicht
- Fit-to-View Button: Zoomt so, dass das ganze Diagramm im Fenster sichtbar ist

4 Prozess bearbeiten

4.1 Elemente verschieben

[Screenshot: Element wird mit Maus verschoben]

1. Klicken Sie auf das Element, das Sie verschieben möchten
2. Halten Sie die linke Maustaste gedrückt
3. Bewegen Sie die Maus an die neue Position
4. Lassen Sie die Maustaste los

Das Element wird auf dem Canvas verschoben.

4.2 Elemente umbennen und verändern

Methode 1: Inline-Bearbeitung (Schnellmethode)

[Screenshot: Doppelklick auf Elementname mit Bearbeitungsfeld]

1. Doppelklicken Sie auf das Element oder seinen Namen
2. Das Feld wird zur Bearbeitung aktiviert
3. Geben Sie den neuen Namen ein

Methode 2: Bearbeitung über das Property-Panel

[Screenshot: Auswahl Element und Aufmachen des meta data panels]

1. Klicken Sie auf das Element
2. Öffnen Sie das Property-Panel
3. Wählen Sie das gewünschte Feld aus, dass Sie bearbeiten wollen
4. Geben Sie die neue Information ein

Methode 3: Context-Menü (erweiterte Optionen)

[Screenshot: Rechtsklicken des elements]

1. Rechtsklicken Sie auf das Element
2. Im Context-Menü wird unter anderem folgende Option angezeigt:
    a. Andere Optionen: Je nach Elementtyp verfügbar.
    Erlaubt die Änderung der Art eines Elements (bspw. das Ändern eines Start-Events zu einem End-Event)

4.3 Elemente hinzufügen

Methode 1: Über das Werkzeugpanel

[Screenshot: Werkzeugpanel mit Elementen und Element wird auf Canvas gezogen]

1. Öffnen Sie das Werkzeugpanel auf der linken Seite
2. Wählen Sie das gewünschte Element aus (Task, Gateway, Event, Subprozess, Verbindung)
3. Ziehen Sie das Element mit der Maus auf den Canvas
4. Lassen Sie die Maustaste los, um das Element zu platzieren

Das neue Element wird auf dem Canvas hinzugefügt und ist sofort bearbeitbar.

Methode 2: Über das Contextmenü (erweiterte Optionen)

[Screenshot: Rechtslicken des Elements und offenes Contextmenü]

1. Rechtsklicken Sie das Element, dass dem zu erstellenden Element vorausgeht
2. Im sich öffnenden Context-Menü gibt es die folgende Option:
    a. Klicken Sie auf das gewünschte Element im Context-Menü und es wird automatisch mit einer Verbindung an das ausgewählte Element angefügt

4.4 Element löschen

Methode 1: Mit Tastatur

[Screenshot: Element wird mit Entf-Taste gelöscht]

1. Klicken Sie auf das Element, um es auszuwählen
2. Drücken Sie die Entf-Taste (oder Delete)
3. Das Element wird sofort gelöscht

Methode 2: Mit Context-Menü

[Screenshot: Mit Context-Menü]

1. Rechtsklicken Sie auf das Element
2. Wählen Sie Delete aus dem Menü
3. Das Element wird gelöscht

Hinweis: Gelöschte Elemente können mit Strg + Z (Undo) wiederhergestellt werden.

4.5 Metadaten bearbeiten

[Screenshot: Properties Panel mit BMD-spezifischen Feldern]

Das Properties Panel auf der rechten Seite zeigt alle Metadaten des aktuell ausgewählten Elements. Beispielsweise Name, Beschreibung und weitere BMD-Felder.

Bearbeitung:

1. Klicken Sie auf das Element im Diagramm
2. Im Properties Panel werden dessen Eigenschaften angezeigt
3. Klicken Sie auf ein Feld, um es zu bearbeiten
4. Änderungen werden automatisch übernommen

5 Neuen Prozess erstellen

5.1 URL zum Erstellen eines neuen Prozesses

Um einen neuen, leeren Prozess zu erstellen, öffnen Sie die folgende URL:

```
/Visualisieren/new
```

5.2 Was geschieht beim Öffnen

[Screenshot: Leerer Editor mit leerem Canvas]

- Ein leerer BPMN-Editor öffnet sich
- Die Canvas ist leer und bereit für neue Elemetne
- Sie können sofort anfangen, Elemente hinzuzufügen und den Prozess zu gestalten

5.3 Erste Schritte

1. Nutzen Sie die Werkzeugpalette (links in der Weboberfläche), um Elemente hinzuzufügen
2. Verschieben, benennen und bearbeiten Sie Elemente wie in Kapitel 3 beschrieben

6 Exportieren und Speichern

6.1 PNG-Export

PNG-Exports werden über spezielle URLs durchgeführt und ermöglichen es, das Diagramm als Bild zu speichern.

Client-seitige Vorschau

Rufen Sie folgende URL auf:
`/download?MCA_PRO_PROJEKTNR=<ID>`

Ersetzen Sie <ID> durch die Projektnummer.

Was Sie sehen:

1. Das Diagramm wird als PNG gerendert
2. Eine Vorschau wird angezeigt
3. Über Rechts -> Bild speichern unter können Sie das PNG auf Ihrem Computer speichern.

Server-seitiger API-Export (für Integration)

Für Systeme, die das PNG direkt abrufen möchten, steht folgende API zur Verfügung:

```
GET /api/export/<PROJEKTNUMMER>
```

Diese API wird typischerweise direkt von BMD/NTCS abgerufen und gibt das PNG als binären Response zurück.

6.2 Speichern (JSON lokal)

[Screenshot: Speichern-Dialog mit JSON-Datei]

Speichern mit dem Toolbar-Button

1. Klicken Sie auf das Disketten-Icon in der Toolbar
2. Ein „Speichern unter“-Dialog öffnet sich
3. Wählen Sie einen Speicherort auf Ihrem Computer
4. Wählen Sie einen Dateinamen und klicken Sie auf Speichern um die Datei zu speichern.

Was wird gespeichert

Die Datei enthält das BPMN-Diagramm im JSON-Format, einschließlich:

- Aller Elemente (Tasks, Gateways, Events, etc.)
- Ihrer Positionen und Größen
- Aller Metadaten, die im Vornherein geladen werden (Namen, Beschreibungen, etc.)
- Der Verbindungen zwischen Elementen

Hinweis: Die Speicherung in das NTCS-System ist technisch vorbereitet, aber noch nicht aktiv. Sie wird erst mit der vollständigen NTCS-Integration verfügbar sein. Derzeit können Sie Ihre Arbeit lokal speichern.

6.3 SVG und PDF-Export (ausgeblendet)

Die Exportformate SVG (Vektorgrafik) und PDF sind technisch bereits implementiert, aber zur Zeit in der Benutzeroberfläche nach Absprechung mit dem Auftraggeber ausgeblendet.

7 Häufige Probleme und Lösungen

Problem	Mögliche Ursache	Lösung
Prozess lädt nicht	URL / ID falsch oder BMD-Server nicht erreichbar	1. Überprüfen Sie die URL und Projektnummer 2. Kontaktieren Sie IT, falls BMD nicht erreichbar ist
PNG-Export schlägt fehl	API-Server ist nicht aktiv oder nicht erreichbar	Kontaktieren Sie IT zur Überprüfung des API-Servers

Speichern in NTCS fehlgeschlagen	NTCS-Integration nicht eingerichtet oder noch in Entwicklung	Dies ist eine bekannte Einschränkung. Siehe Hinweis in Kapitel 5.2
Elemente lassen sich nicht verschieben	Zoom-Level sehr klein oder Browser-Cache veraltet	1. Nutzen Sie den „Fit-to-View“ Button 2. Laden Sie die Seite neu (F5) und leeren Sie ggf. den Browser-Cache
Diagramm oder Canvas wird falsch dargestellt	Browser-Cache ist veraltet oder Browser wird nicht vollständig unterstützt	1. Leeren Sie Ihren Browser-Cache 2. Versuchen Sie einen anderen, aktuellen Browser
Änderungen gehen verloren	Seite wurde geschlossen ohne zu speichern	Speichern Sie regelmäßig mit dem Disketten-Icon, um Ihre Arbeit zu schützen
Zoom-Buttons funktionieren nicht	Browser-Kompatibilität oder JavaScript ist deaktiviert	Überprüfen Sie, dass JavaScript im Browser aktiviert ist

8 Änderungsverzeichnis

Autor	**Datum**	**Inhalt**	**VisuPro – Version**
Alexander Vorreither	18.03.2026	Erstellung	1. v. 2.0

F Abnahmeprotokoll

Abnahmeprotokoll	**HTBLA Perg**
Projektname : VisuPro	
Auftraggeber : *BMD Systemhaus GesmbH*	**Empfänger :** *Hannes Großauer, Andreas Fuchs*

Das Werk „VisuPro“ Version 2.0

basierend auf dem Vertrag vom: 03.07.2025

wurde am 17.03.2026 zur Abnahme angemeldet.

Zu dem Werk gehören folgende **Komponenten**:

- ☑ Dieses Protokoll
- ☑ Quellcode
- ☑ Benutzerhandbuch
- ☑ Datenbankmodell und Datendokumentation

Das Werk „VisuPro“ Version 2.0

basierend auf dem Vertrag vom: 03.07.2025

wurde am 19.03.2026 abgenommen und zur Umsetzung freigegeben.

Mit seiner Unterschrift bestätigt der Auftraggeber die Abnahme des Werks.

Datum: 19.03.26		
	Unterschrift Hannes Großauer	Unterschrift Mathias Grubbauer

VisuPro-Abnahmeprotokoll.doc 17.03.2026 1 von 1