



Abschlussprüfung Winter 2023/11

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

LWV-Metadaten Tool

Tool zur Verwaltung von Metadaten für Schriftstückvorlagen

Abgabetermin: Kassel, den 29.11.2023

Prüfungsbewerber:

Ahmad Darwish
Mauer Straße 8
341117 Kassel



Ausbildungsbetrieb:

Landeswohlfahrtsverband Hessen
Ständeplatz. 6 - 10
34117 Kassel

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	III
Abkürzungsverzeichnis	IV
1 Einleitung	1
1.1 Projektbeschreibung	1
1.2 Projektziel	1
1.3 Projektbegründung	2
1.4 Projektabgrenzung	2
2 Projektplanung	2
2.1 Projektphasen	2
2.2 Ressourcenplanung	3
2.3 Entwicklungsprozess	3
3 Analysephase	3
3.1 Ist-Analyse	3
3.2 Wirtschaftlichkeitsanalyse	4
3.2.1 Projektkosten	4
3.3 Lastenheft	5
4 Entwurfsphase	5
4.1 Zielplattform	5
4.2 Architekturdesign	5
4.3 Datenmodell	6
4.4 Geschäftslogik	7
4.5 Maßnahmen zur Qualitätssicherung	7
4.6 Deployment	7
4.7 Pflichtenheft	7
5 Implementierungsphase	7
5.1 Implementierung der Datenstrukturen	8
5.2 Implementierung der Geschäftslogik	8

5.3	Implementierung der Benutzeroberfläche	8
5.4	Testen der Anwendung	9
6	Abnahme- und Deploymentphase	9
6.1	Code-Review	9
7	Dokumentation	9
8	Fazit	9
8.1	Soll-/Ist-Vergleich	9
8.2	Lessons Learned	10
8.3	Ausblick.....	10
	Literaturverzeichnis	11
A	Anhang	i
A.1	Detaillierte Zeitplanung	ii
A.2	Verwendete Ressourcen.....	ii
A.3	Entity-Relationship-Modell	ii
A.4	Lastenheft	iii
A.5	Amortisation.....	iv
A.6	Gantt-Diagramm.....	v
A.7	Netzplan	v
A.8	Klassendiagramm	vi
A.9	Pflichtenheft (Auszug).....	vii
A.10	Listing von Java-Code.....	viii
A.11	Finale SQL-Datei	x
A.12	INI-Dateien.....	xi
A.13	Screenshots	xii

Abbildungsverzeichnis

1	Entity-Relationship-Modell	ii
2	Grafische Darstellung der Amortisation.....	iv
3	Gantt-Diagramm.....	v
4	Netzplan	v
5	Klassendiagramm	vi
6	Screenshot der Anzeige der SQL-Skript.....	x
7	INI-Datei für Dropdown Menu	xii
8	Screenshot der Anzeige von erste-Schnittstelle (LWV)	xii
9	Screenshot der Anzeige der Edok-Vorlage Formular	
10	Screenshot der Anzeige der Children Formular.....	xiii

Tabellenverzeichnis

1	Grobe Zeitplanung	2
2	Kostenaufstellung	5
3	Soll-/Ist-Vergleich	10

Listings

1	Funktion write_meta_query in der ToolController_Klasse	xii
2	Funktion Löschen_Könfig in der ToolController_Klasse	xiii

Abkürzungsverzeichnis

LWV	Landeswohlfahrtsverband Hessen
ANLEI	Software der Sachbearbeitung zur ANtragsannahme und LEistungsgewährung
SQL	Structured Query Language
JUnit	Open-Source Framework für Test Drive Development
CCM	Content-Communication-Management
ERM	Entity-Relationship-Modell
MoSCoW	Must, Should, Could, Would
Java SE 8	Java standard Edition 8
MVC	Model View Controller
QS	Qualitätssicherung
PT	Personentag
TDD	Test Driven Development
Swing	Grafische Benutzeroberflächen in Java
IntelliJ	Integrated Development Environment (IDE) for Java
INI	Initialisierungsdatei mit Parametern für das Programm
Edok	Die Abteilung, die für das Programm Verantwortlich ist.
Edok_Vorlage	Haupttabelle in der Edok-Datenbank
Edok_Metadatum	Child Tabelle in der Edok-Datenbank
Edok_Vorlagenbaum	Child Tabelle in der Edok-Datenbank
Edok_Folgeschreiben_Konfiguration	Child Tabelle in der Edok-Datenbank

1 Einleitung

In der folgenden Projektdokumentation wird der Ablauf des Abschlussprojektes, das durch den Autor im Rahmen seiner Abschlussprüfung zum Fachinformatiker mit der Fachrichtung Anwendungsentwicklung durchgeführt wurde, erläutert. Das Projekt wird beim Landeswohlfahrtsverband Hessen (LWV) durchgeführt, welche der Ausbildungsbetrieb des Autors ist. Der Landeswohlfahrtsverband Hessen (LWV) wird als landesweiter Kommunalverband getragen von den Landkreisen und kreisfreien Städten. In ihrem Auftrag finanziert er soziale Leistungen für behinderte, psychisch kranke sowie sozial benachteiligte Menschen und unterstützt diese in ihrem Alltag und im Beruf. Beschäftigte des LWV in Kassel, Darmstadt und Wiesbaden stellen das sicher. Bei der LWV Hessen arbeiten ca. 600 Mitarbeiter mit der Anwendung ANLEI, um die Prozesse in der Sachbearbeitung zu unterstützen (Bearbeitung von Eingliederungshilfe und damit im Zusammenhang stehenden Dinge). Ziel dieser Dokumentation ist es, die durchzuführenden Schritte des Projektes von der Planung bis zum Deployment zu erläutern und dies mit geeigneten Diagrammen und Dokumenten zu unterstützen.

1.1 Projektbeschreibung

Dieses Projekt befasst sich mit der Entwicklung eines Tools zur Verwaltung von Metadaten für Schriftstückvorlagen im Kontext des Landeswohlfahrtsverbandes Hessen (LWV Hessen). Der LWV Hessen nutzt die Anwendung ANLEI zur Unterstützung der Sachbearbeitungsprozesse für Eingliederungshilfe und verwandte Angelegenheiten. Mit rund 600 Mitarbeitern werden die Belange von etwa 60.000 Leistungsberechtigten bearbeitet.¹ was eine beträchtliche Menge an Schriftstücken erfordert. Derzeit existieren etwa 1.200 verschiedene Vorlagen im alten System.

Aktuell erfolgt die Einführung eines neuen Content-Communication-Management (CCM)-Systems, das eine verbesserte Schriftstückerstellung ermöglicht. Um die Integration von ANLEI in dieses neue System zu gewährleisten, müssen für jede neue Vorlage SQL-Skripte erstellt und auf verschiedenen Stages (Entwicklungssystem, QS-Testsystem, Anwendertestsystem, Hotfixsystem, Produktion) eingespielt werden.

Zu diesem Zweck wird im Rahmen dieses Projekts ein Tool entwickelt, das es den Vorlagenerstellern ermöglicht, die Metadaten für Schriftstückvorlagen zu erfassen und zu verwalten. Das Tool generiert automatisch die erforderlichen SQL-Skripte für die verschiedenen Stages.

1.2 Projektziel

Das Hauptziel dieses Projekts ist die Entwicklung eines benutzerfreundlichen Tools, das Vorlagenerstellern in den Leistungsfachbereichen und der IT-Abteilung die Eingabe und Verwaltung von Metadaten ermöglicht. Die automatische Generierung von SQL-Skripten für die Integration in die verschiedenen Stages soll den Prozess beschleunigen, die Fehleranfälligkeit verringern und die Notwendigkeit von Mitarbeitern mit SQL-Kenntnissen beseitigen.

¹Kennzahlen zum Stichtag 14.10.2023, vgl. Landeswohlfahrtsverband Hessen LWV [2022, S. 4].

1.3 Projektbegründung

Die Einführung des CCM-Systems erfordert eine effiziente Methode zur Verwaltung von Metadaten für Schriftstückvorlagen, um die reibungslose Integration in ANLEI zu gewährleisten. Das bisherige Verfahren, bei dem SQL-Skripte manuell erstellt wurden, ist zeitintensiv und fehleranfällig. Das neue Tool bietet die Möglichkeit, diesen Prozess zu optimieren und die Effizienz zu steigern, indem sowohl Vorlagenersteller in den Leistungsfachbereichen als auch IT-Mitarbeiter die Möglichkeit erhalten, Metadaten selbst einzugeben und SQL-Skripte automatisch zu generieren.

1.4 Projektabgrenzung

Die Anwendung interagiert mit den vier verschiedenen Tabellen des ANLEI Datenbanksystems vom LWV und dem Dateisystem auf dem Rechner des Nutzers.

Es sollen Eingabeschnittstellen für jede der vier Tabellen für die Datenbanksysteme im Projekt erstellt werden. Da die Anwender in den Leistungsfachbereichen gleichzeitig Auftraggeber und Anwender des Projektes sind, wird das Ergebnis diesem Team vorgestellt. Bei dem Projektergebnis soll es sich um einen Prototyp und nicht um ein Produktivsystem handeln, sodass das Deployment im Produktionssystem nicht Teil des Projektes ist.

2 Projektplanung

In diesem Kapitel wird die Planung des Projektes vorgestellt. Es wird sowohl die zeitliche als auch die inhaltliche Einteilung der Projektphasen dargestellt.

2.1 Projektphasen

Für die Umsetzung des Projektes standen 80 Stunden zur Verfügung, wie es die IHK Kassel vorschreibt.² Bevor mit dem Projekt gestartet wurde, fand eine Aufteilung auf verschiedene Phasen statt, die den kompletten Prozess der Softwareentwicklung abdecken. Eine grobe Zeitplanung mit den Hauptphasen lässt sich aus Tabelle 1 entnehmen. Eine detailliertere Zeitplanung mit den einzelnen Schritten der unterschiedlichen Phasen findet sich im Anhang A.1: Detaillierte Zeitplanung auf Seite i.

Und Im Anhang A.7: Netzplan auf Seite v ist einen Netzplan erstellt zu sehen., die legt die zeitliche und logische Abfolge der Vorgänge in diesem Projekt fest.

Projektphase	Geplante Zeit
Planungsphase	10 h
Entwurfsphase	12 h
Implementierungsphase	32 h
Testphase	7 h
Dokumentationsphase	19 h
Gesamt	80 h

Tabelle 1: Grobe Zeitplanung

²Vgl. IHK Kassel [2023, S. 2].

2.2 Ressourcenplanung

Es wurden alle Ressourcen im Anhang [A.2: Verwendete Ressourcen](#) auf Seite [ii](#) aufgelistet. Die Planung umfasst dabei neben allen Hard- und Softwareressourcen, die im Rahmen des Projektes verwendet wurden, auch das Personal. Im Hinblick auf anfallende Kosten wurde bei der Wahl der verwendeten Software darauf geachtet, dass keine Lizenzkosten anfallen, benötigtes Know-How vorhanden ist und nicht gegen die Architekturrichtlinien der LWV verstoßen wird. Diese legen zur Entwicklung unter anderem die Nutzung von Java Standard Edition 8 ([Java SE 8](#)) mit Swing für grafische Oberflächen fest.

2.3 Entwicklungsprozess

Als letzter Schritt vor dem tatsächlichen Beginn des Projektes musste durch den Autor ein geeigneter Entwicklungsprozess gewählt werden. Durch diesen wird die Vorgehensweise bei der Umsetzung des Projektes definiert.

Grundsätzlich soll sich die Umsetzung des Projektes an den Phasen des Wasserfall-Modells orientieren, da die Anforderungen eindeutig definiert sind. Beim Wasserfall-Modell werden die einzelnen Phasen der Software-Entwicklung nacheinander durchlaufen, wobei eine Rückkehr in eine vorherige Phase jederzeit möglich ist.³ Im Bereich der Oberflächengestaltung soll es jedoch durch regelmäßige Rücksprache mit dem Fachbereich einen agilen Prozess geben. Im Anhang [A.6: Gantt-diagramm](#) auf Seite [v](#) ist ein Ganttdiagramm abgebildet, in dem dieser Prozess dargestellt wird.

Der gewählte Entwicklungsprozess soll dabei um die Praktik des Test Driven Development (TDD) erweitert werden, bei dem vor der Implementierung der fachlichen Logik jeweils ein (JUnit) Test geschrieben wird. Diese Tests sollen sicherstellen, dass die Anwendung die erwartete Funktionalität tatsächlich umsetzt und nachträgliche Änderungen am Code keine unerwarteten Effekte nach sich ziehen, die das existierende Verhalten beeinträchtigen. Dadurch wird unter anderem auch die Motivation zum Refactoring, also dem Ändern des Codes ohne eine Veränderung des Verhaltens, erhöht.⁴ Damit das Prinzip des TDD möglichst optimal durchgeführt werden kann, wurde in der zeitlichen Planung unter Punkt [2.1 \(Projektphasen\)](#) viel Zeit für die Implementierung eingerechnet.

3 Analysephase

3.1 Ist-Analyse

Wie bereits unter [1.1 \(Projektbeschreibung\)](#) beschrieben setzt sich das Projekt Metadaten-Tool aus vier verschiedenen Tabellen zusammen. In der Ist-Analyse soll jeweils herausgearbeitet werden, wie die Daten aktuell ermittelt und verarbeitet werden.

Die Erstellung von Vorlagen erfolgt sowohl durch Anwender in den Leistungsfachbereichen als auch durch IT-Mitarbeiter. In beiden Fällen wurden die Vorlagen, zusammen mit der Information welche Metadaten eingestellt werden sollen, an Mitarbeiter der IT weitergereicht, die dann SQL-Skripte erstellt haben, um die Metadaten im System einstellen zu können, was zeitaufwendig ist und oftmals zu Fehlern führt, die korrigiert werden müssen.

³Vgl. [Sommerville \[2007, S.97\]](#).

⁴Vgl. [Langr u. a. \[2015, S. 3f., S.95, S. 153f\]](#).

2 Projektplanung

Das geplante Tool ermöglicht Vorlagenerstellern in den Fachbereichen und der IT, Metadaten eigenständig einzugeben und automatisch [SQL](#)-Skripte zu generieren. Dieser automatisierte Prozess eliminiert die Notwendigkeit, manuell [SQL](#)-Skripte zu erstellen und erfordert kein [SQL](#)-Fachwissen.

Das Tool generiert [SQL](#)-Inserts für vier Tabellen: [Edok_Vorlage](#), [Edok_Metadatum](#), [Edok_Vorlagenbaum_Vorlage](#) und [Edok_Folgeschreiben_Konfiguration](#). Als Erstes, wird die [SQL](#)-Datei gelöscht, sofern sie bereits vorhanden ist. Jede Vorlage hat individuelle Inserts in eine individuelle [SQL](#)-Datei, aber für jede Vorlage können mehrere Einträge in der `edok_metadatum`, `edok_vorlagenbaum_vorlage` oder `edok_vorlage_folgeschreiben` erzeugt werden, in der gleichen [SQL](#)-Datei_. Dadurch entsteht eine 1: n-Beziehung zu `edok_vorlage`.

Das Tool bietet auch automatische Nummernvergabe für 6-stellige IDs und Dropdown-Listen, um flexible Eingabeoptionen für Felder zu ermöglichen. Alle Werte für jede Dropdown-List wird in einer [INI](#)-Dateien gespeichert. Damit ist das Tool von der Datenbank getrennt.

Mit dem Tool können auch alle Inserts zu jeder der vier Tabellen gelöscht werden. Die dafür notwendigen Statements werden erzeugt, bevor auf den Button „Fertig“ geklickt wird. Wenn auf den Button „Fertig“ am Ende des Prozesses geklickt wird, werden die Commit-Anweisungen am Ende der [SQL](#)-Datei hinzugefügt und die [SQL](#)-Datei wird für die nächsten Stages exportiert.

Die [SQL](#)-Skripte können dann zusammen mit den Vorlagen in die verschiedenen Systeme eingespielt werden. Im Anhang [A.8: Klassendiagramm](#) auf Seite [vi](#) ist ein Klassendiagramm abgebildet, in dem dieser Prozess dargestellt wird.

3.2 Wirtschaftlichkeitsanalyse

3.2.1 Projektkosten

Im Folgenden sollen die Kosten, die im Laufe des Projektes anfallen, kalkuliert werden. Neben den anfallenden Personalkosten des Entwicklers und weiterer Projektbeteiligter müssen auch die Aufwendungen für die verwendeten Ressourcen, die unter [2.2 \(Ressourcenplanung\)](#) aufgeführt sind, eingeplant werden.

Da die tatsächlichen Personalkosten nicht herausgegeben werden dürfen, wird die Kalkulation anhand von Stundensätzen durchgeführt, die durch die Personalabteilung festgelegt wurden. Die aufgeführten Stundensätze beinhalten zum Großteil das Bruttogehalt sowie die Sozialaufwendungen des Arbeitgebers. Hinzu kommen die oben erwähnten Kosten, die für die Nutzung der Ressourcen anfallen.

Für einen Mitarbeiter wird ein Stundensatz von 25,00 € eingeplant. Der Stundensatz für einen Auszubildenden ist auf 10,00 € festgelegt. Als Satz für die Ressourcennutzung werden pauschal 15,00 € angenommen.

Die Durchführungszeit des Projektes beträgt 70 Stunden. In Tabelle [2: Kostenaufstellung](#) sind die Kosten unterteilt nach den einzelnen Projektvorgängen aufgelistet, sowie summiert dargestellt, um die Gesamtkosten, die während des Projektes anfallen, zu erhalten. Diese belaufen sich auf 2240,00 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	80 h	10,00 € + 15,00 € = 25,00 €	2000,00 €
Fachgespräch	3 h	25,00 € + 15,00 € = 40,00 €	120,00 €
Abnahmetest	1 h	25,00 € + 15,00 € = 40,00 €	40,00 €
Code-Review	2 h	25,00 € + 15,00 € = 40,00 €	80,00 €
			2240,00 €

Tabelle 2: Kostenaufstellung

3.2.2 Amortisationsdauer

Im Folgenden soll die Amortisationsdauer betrachtet werden, also ab welchem Zeitpunkt sich die Entwicklung des Projektes amortisiert. Die Anschaffungskosten wurden bereits in Tabelle 2: [Kostenaufstellung](#) bestimmt. Durch die Automatisierung der Erstellung der [SQL](#)-Dateien lässt sich die Zeit dieser manuellen Arbeit einsparen. Außerdem entfällt die Versionierung und Nachbearbeitung sowie Kontrolle der Dateien wie unter [3.1 \(Ist-Analyse\)](#) beschrieben. Die daraus resultierende Zeitersparnis ist in Tabelle 3: [Zeiteinsparung pro Jahr](#) detaillierter aufgeführt. Anhand von Erfahrungen aus den letzten Durchläufen der Prozesse wurden von den jeweils zuständigen Abteilungen die Vorgangszeiten ermittelt.

In der neuen Anwendung muss durch einen Mitarbeiter die Erstellung der neuen Dateien einmalig, wenn sichergestellt wurde, dass alle Daten in der [INI](#)-Dateien hinterlegt worden sind. Für diesen Arbeitsschritt werden 10 Minuten kalkuliert.

Grundsätzlich wird wie unter [3.2.2 \(Projektkosten\)](#) ein Personentag (PT) mit 7,6 Stunden angenommen, was 456 Minuten entspricht. Es wird angenommen, dass die genannten Schritte im Durchschnitt 1,5-mal pro Jahr ausgeführt werden.

Für den Betrieb der Anwendung wurde von der Administration eine Pauschale von einem PT pro Jahr veranschlagt. Dies umfasst u. a. das Einspielen von Betriebssystem und die Wartung bei Ausfällen.

Vorgang	Anzahl pro Jahr	Zeit alt pro Vorgang	Zeit neu pro Vorgang	Einsparung pro Jahr
Erstellung der INI-Dateien	1,5	2 PT ⚡ 912 min	10 min	1353 min
Versionierung der Dateien	1,5	0,5 PT ⚡ 228 min	-	342 min
Aufbereitung der Dateien	1,5	4 PT ⚡ 1824 min	-	2736 min
Kontrolle der Daten	1,5	1 PT ⚡ 456 min	-	684 min
Betrieb der Anwendung	1	-	1 PT ⚡ 456 min	-456 min
				4659 min

Tabelle 3: Zeiteinsparung pro Jahr

Dadurch ergibt sich eine jährliche Einsparung von $\frac{4.659 \text{ min}}{60 \text{ min/h}} \cdot (25 + 15) \text{ €/h} = 3.106,00 \text{ €}$.

Die Amortisationszeit beträgt also $\frac{1990,00 \text{ €}}{3106,00 \text{ €/Jahr}} \approx 0,64 \text{ Jahre} \approx 7,5 \text{ Monate}$. Eine grafische Dar-

stellung befindet sich im Anhang [A.4: Amortisation](#) auf Seite [iv](#). Wie schon vorher beschrieben, findet dieser Prozess nur ein- bis zweimal jährlich statt. Aus der ermittelten Zeit lässt sich also ableiten, dass sich aufgrund der großen Einsparung an manueller Arbeit das Projekt schon mit der zweiten Ausführung amortisiert hat, da der Prozess durchschnittlich alle 7,5 Monate stattfindet. Daher lässt sich das Projekt als wirtschaftlich einordnen und soll umgesetzt werden.

3.3 Lastenheft

Am Ende der Analysephase wurde gemeinsam mit der für [Edok](#)-Abteilung ein Lastenheft erstellt. In dem Lastenheft sind alle zu berücksichtigenden Anforderungen des Auftraggebers an die zu implementierende Anwendung sortiert nach der Wichtigkeit der Umsetzung festgehalten. Die Formulierung wurde mit Hilfe der Must, Should, Could, Would ([MoSCoW](#))-Methode umgesetzt. Das heißt, dass in jeder Anforderung die Wichtigkeit aufgeschlüsselt nach „muss“, „soll“, „kann“ oder „würde gerne“ festgehalten wird.⁵ Anhand dieser Methodik wurde auch die Priorisierung vorgenommen. Das vollständige Lastenheft befindet sich im Anhang [A.4: Lastenheft](#) auf Seite [iii](#)

4 Entwurfsphase

4.1 Zielplattform

Wie bereits unter [1.1 \(Projektbeschreibung\)](#) erwähnt, soll das Abschlussprojekt als eigenständige Desktop Anwendung entwickelt werden. Die Daten, auf die zugegriffen werden soll, sind in (.ini) Dateien abgelegt. Als Programmiersprache soll Java eingesetzt werden. Die Auswahl dieser Sprache ergibt sich aus den Architekturrichtlinien, die durch die [LWV](#) vorgegeben sind. Als Spezifikation zur Entwicklung der Desktop-Anwendung wird [Java SE 8](#) benutzt, da diese auch schon in anderen Projekten in der [LWV](#) zum Einsatz gekommen ist, wobei sich dieser als geeignet erwiesen hat. Und dabei wird [Swing](#) für die grafische Oberfläche verwendet

4.2 Architekturdesign

Als Basis für das architekturelle Design soll das Model-View-Controller Architekturmuster ([MVC](#)) dienen. Dieses soll aber durch Komponenten aus dem [Java SE 8](#)-Standard erweitert werden.

Grundsätzlich besteht die Anwendung aus drei Schichten. Die Model-Schicht enthält die Entitäten, die Tabellen in einer [Edok](#)-Datenbank persistiert. In der Controller-Schicht sind neben dem Controller auch andere Klassen (wie Validation) vorhanden, in denen die fachliche Logik abgebildet wird. So können die einzelnen Entitäten-Klassen so miteinander verbunden werden, dass die Use-Cases umgesetzt werden können.

In der View-Schicht wird die Schnittstelle zur Oberfläche bereitgestellt. Die View-Schicht beinhaltet lediglich Klassen, um eine Schnittstelle für eine Anbindung an die Controller-schicht vorzugeben.

⁵Vgl. [Haughey, Duncan \[2014\]](#).

4.3 Datenmodell

Das Datenmodell besteht aus den Entitäten, die Tabellen in einer [Edok](#)-Datenbank persistieren.

Zu jeder Vorlage können mehrere Metadatum-Einträge gehören, weswegen hier eine 1: n-Beziehung besteht zwischen den Tabellen [Edok_Vorlage](#) und [Edok_Metadatum](#). Zu jeder Vorlage können mehrere Vorlagenbaum-Einträge gehören, weswegen hier auch eine 1: n-Beziehung besteht zwischen den Tabellen [Edok_Vorlage](#) und [Edok_Vorlagenbaum_Vorlage](#). Für jede Vorlage können mehrere folgeschreiben_konfiguration-Einträge gehören, weswegen auch hier eine 1: n-Beziehung besteht zwischen den Tabellen [Edok_Vorlage](#) und [Edok_Folgeschreiben_Konfiguration](#).

Diese Beziehungen sind im Anhang [A.3: Entity-Relationship-Modell](#) auf Seite [ii](#) als Entity-Relationship-Modell (ERM) dargestellt. Anhand dieses Modells sollen später u. a. die Klassen der Domäne implementiert werden.

4.4 Geschäftslogik

Das Projekt soll sich in die Schichten Model, View und Controller aufteilen. Die Geschäftslogik wird in ToolController-Klasse implementiert, deren Methoden von den Controller-Klasse aufgerufen werden können. Dabei soll für jeden Anwendungsfall, z. B. das Speichern der [Edok](#)-Vorlage Query im Dateisystem(.SQL), eine eigene Controller-Klasse implementiert werden. Controller-Klassen stellen Methoden bereit, die für mehrere Anwendungsfälle genutzt werden können.

Für eine nachträgliche Dokumentation der Geschäftslogik wurde ein Klassendiagramm aus dem Programmcode heraus generiert, das sich als Ausschnitt im Anhang [A.8: Klassendiagramm](#) auf Seite [vi](#) befindet. Dort lässt sich die schon unter dem Architekturdesign beschriebene Trennung der einzelnen Komponenten und deren Abhängigkeiten untereinander erkennen. Das Model hat keine externen Abhängigkeiten.

4.5 Maßnahmen zur Qualitätssicherung

Während der Durchführung des Projektes wurden Maßnahmen zur Qualitätssicherung ergriffen. Das Projekt wurde testgetrieben ([JUnit](#)) entwickelt, um fachliche Fehler möglichst von Beginn an zu vermeiden und sicherzustellen, dass auch bei nachträglichen Änderungen die Qualität der Anwendung nicht abnimmt. Zusätzlich wurde ein Code-Review mit dem Ausbilder durchgeführt, wobei neben der fachlichen Richtigkeit auch die technische Umsetzung überprüft werden sollte.

4.6 Pflichtenheft

Auf Grundlage der ausgearbeiteten Entwürfe wurde am Ende der Entwurfsphase ein Pflichtenheft erstellt. Dieses baut auf dem Lastenheft auf und erfasst die konkrete Umsetzung zu den in [3.4 \(Lastenheft\)](#) ermittelten Anforderungen. Es dient dazu, am Ende des Projektes überprüfen zu können, ob alle Anforderungen umgesetzt wurden und kann auch schon während der Entwicklung als Leitfaden dienen. Abgebildet ist das Pflichtenheft im Anhang [A.9: Pflichtenheft \(Auszug\)](#) auf Seite [vii](#).

5 Implementierungsphase

In der Implementierungsphase wurden die Ergebnisse der Entwurfsphase implementiert. Dazu gehört die Datenstruktur, die Benutzeroberfläche und die Geschäftslogik.

5.1 Implementierung der Datenstrukturen

Die entworfene Datenstruktur der Models Connection und Table wurde als [ERM](#) im Anhang [A.3: Entity-Relationship-Modell](#) auf Seite [ii](#) dargestellt.

Diese vier Models wurden als eigene Java-Klassen angelegt, die die Attribute der Entitäten enthalten. Die Klasse des Table-Models enthält zusätzlich die Getter- und Setter-Methoden für jedes Attribut.

Bei der Speicherung der Values von Dropdown-Menü wurde sich für die Speicherung der Objekte in [INI](#)-Dateien im Dateisystem auf dem lokalen Rechner des Nutzers entschieden. Beispiele für diese [INI](#)-Dateien finden sich im Anhang [A.12: INI-Dateien](#) auf Seite [xi](#).

5.2 Implementierung der Geschäftslogik

Die Implementierung der Oberfläche und der Datenstrukturen ermöglicht die Implementierung der Geschäftslogik. Diese ist dabei hauptsächlich in Controller-Klasse implementiert worden.

Für jeden Use Case, wie die Speicherung einer SQL Query in eine [INI](#)-Datei im Dateisystem auf dem lokalen Rechner oder Löschen dieser Query aus der gleichen Datei, wurden je eine Funktion in der Controller-Klasse erstellt. Außerdem gibt es in der Controller-Klasse verschiedene Funktionen für die Berechnung minimal und maximal ID. Diese kann von dem User vorgegeben werden. Außerdem gibt es eine Funktion `write_meta_query`, die für das Schreiben [Edok_Metadatum](#)-Query in eine lokale [SQL](#)-Datei verantwortlich ist. Es gibt auch eine Funktion `Löschen_Konfig`, die für das Löschen [Edok_Folgeschreiben_Konfiguration](#) Query aus der [SQL](#)-Datei, bevor sie abgeschlossen wurde. Im Anhang [A.10: Listing von Java-Code](#) auf Seite [viii](#) ist der entsprechende Code zu sehen, und auch die finale [SQL](#)-Datei, die das Tool am Ende des Prozesses erstellt hat, findet sich im Anhang [A.11: Finale SQL-Datei](#) auf Seite [x](#).

5.3 Implementierung der Benutzeroberfläche

Die Benutzeroberfläche der Applikation wurde mit der unter Architekturdesign genannten [Swing](#)-Technologie aus dem [Java SE 8](#)-Standard. Um die Einheitlichkeit der einzelnen Schnittstellen zu gewährleisten, haben wir in [Swing](#) eine Vorlage anzulegen und diese dann zu verwenden. Diese Vorgehensweise wurde bei allen implementierten Oberflächen genutzt. Im Anhang [A.13: Screenshots](#) auf Seite [xi](#) befinden sich Screenshots der Anwendung, die nach der Implementierungsphase entstanden sind.

Bei der Gestaltung der Oberfläche wie auch während der gesamten Entwicklung wurden die Software-Ergonomie-Richtlinien beachtet. Dazu zählt vor allem die Aufgabenangemessenheit durch eine Minimierung der Interaktionen und geeigneter Funktionalität und die Fehlertoleranz. So werden z. B. bei Fehlern während der Validierung der ID-Textfelder dem Benutzer sprechende Fehlermeldungen angezeigt.

5.4 Testen der Anwendung

Wie schon unter 2.3 ([Entwicklungsprozess](#)) erwähnt, wurde die Anwendung testgetrieben entwickelt. So wurden während der Implementierungsphase stetig Tests geschrieben. Diese Tests lassen sich in drei verschiedene Kategorien einteilen. Die Unit-Tests überprüfen die Funktionalität einer einzelnen Klasse. In den Oberflächen-Tests wird abschließend die Interaktion des Benutzers mit der Anwendung simuliert und auf mögliche Fehler getestet. Um nicht die Daten der Datenbank zu manipulieren, wird dafür [INI](#)-Dateisystem als Test-Datenbank verwendet, da diese sehr leichtgewichtig und ohne Installation nutzbar ist.

6 Abnahme- und Deploymentphase

6.1 Code-Review

Bevor die Anwendung abschließend zum Test dem Fachbereich vorgeführt wurde, fand ein Code-Review durch den Ausbilder statt. Dabei wurde neben fachlichen und technischen Fehlern vor allem auch auf die Verständlichkeit des Codes in Bezug auf die Benennung von Variablen und Komplexität der Methoden geachtet. Da der Ausbilder mit dem Thema des Projektes vertraut war, war kein Vorgespräch zur Erläuterung der Anwendung notwendig. Die Anmerkungen wurden als TODO-Kommentare direkt in den Code geschrieben. Nach dem Review konnten in einer Nachbesprechung offene Fragen bezüglich der Anmerkungen geklärt werden. Ziel des Reviews war es, die Qualität des Codes zu steigern und diesen für andere Entwickler verständlicher zu machen.

7 Dokumentation

Im Rahmen des Projektes wurden eine Projektdokumentation erstellt, in der die während der Umsetzung des Projektes durchlaufenden Phasen beschrieben werden. Die Projektdokumentation wurde mit Microsoft Office Word verfasst.

Diese Dokumentation soll bei der Weiterentwicklung der Anwendung oder einer Anpassung durch einen anderen Entwickler als Übersicht und Nachschlagewerk dienen.

Zusätzlich wurde für Dokumentationszwecke ein Klassendiagramm generiert, um einem anderen Entwickler die Möglichkeit zu geben, sich schnell einen Überblick über die Anwendung zu schaffen. Ein Ausschnitt von diesem Diagramm befindet sich im Anhang [A.8: Klassendiagramm](#) auf Seite [vi](#).

8 Fazit

8.1 Soll-/Ist-Vergleich

Abschließend soll die Planung des Projektes rückblickend überprüft werden. Dazu wird die benötigte mit der im Vorfeld kalkulierten Zeit verglichen. Wie in Tabelle [4](#) zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden. Das gesamte Projekt konnte in der von der IHK Kassel vorgegebenen Zeit von 80 Stunden umgesetzt werden, da die geringfügigen Abweichungen von der Planung stets kompensiert werden konnten.

5 Implementierungsphase

Die Abnahmephase fiel um eine halbe Stunde kürzer aus, da das für zwei Stunden eingeplante Code-Review durch den Ausbilder schneller durchgeführt und besprochen werden konnte. Zudem wurde für die Dokumentation eine halbe Stunde weniger Zeit benötigt. Für die Implementierung hingegen war die eingeplante Zeit nicht ausreichend und wurde um 1,5 Stunden überschritten.

Phase	Geplant	Tatsächlich	Differenz
Planungsphase	10 h	10 h	0 h
Entwurfsphase	12 h	12 h	0 h
Implementierungsphase	32 h	33,5 h	+1,5 h
Testphase	7 h	6 h	-1 h
Dokumentationsphase	19 h	18,5 h	-0,5 h
Gesamt	80 h	80 h	0h

Tabelle 4: Soll-/Ist-Vergleich

8.2 Lessons Learned

Während der Umsetzung des Projektes konnte ich meine Fähigkeiten im Umgang mit der Programmiersprache Java verbessern. Ich habe gelernt, dass der Einsatz verschiedener Entwurfsmuster wie [MVC](#) sinnvoll ist und die Arbeit erleichtert. Durch die Code-Reviews und das Feedback meines Betreuers habe ich gelernt, dass es bei der Entwicklung größerer Programme vieles zu beachten gibt und eine Planung der Handlungsschritte im Voraus sinnvoll ist. Für mein Projekt eignete sich das Wasserfallmodell sehr gut. Dennoch war es sinnvoll, dass ich in Zusammenarbeit mit den Betreuern Aspekte eines inkrementellen Vorgehens eingebracht habe, damit einige wechselnde Anforderungen und zusätzliche Funktionen beachtet werden konnten. Gerade im Rahmen größerer Projekte erkenne ich, dass es sinnvoller ist inkrementelle, agile Vorgehensweisen einzusetzen, um dem Kunden regelmäßig eine erweiterte, verbesserte Software zur Verfügung zu stellen. Dadurch kann flexibler auf Anforderungen eingegangen werden.

8.3 Ausblick

Auch wenn die an dieses Projekt gestellten Anforderungen umgesetzt wurden, ist nicht auszuschließen, dass zukünftig neue Themenfelder, welche dem Metadaten-Tool zugeordnet werden können, entstehen. Diese könnten dann sehr einfach mit in diese Anwendung integriert werden, da der Aufbau modular ist.

Literaturverzeichnis

Landeswohlfahrtsverband Hessen LWV 2022

Landeswohlfahrtsverband Hessen LWV: *Jahresbericht über das Jahr 2022.*
https://www.lwv-hessen.de/fileadmin/user_upload/daten/Dokumente/Broschueren_barrierefr/Jahresbericht_2022_der_Schulen_und_angegliederten_Einrichtungen_barr.pdf. Version: 2022.– Zugriff am: 14.10.2023

Haughey, Duncan 2014

Haughey, Duncan: *MoSCoW Method.* <https://www.projectsmart.co.uk/moscow-method.php>. Version: 2014. – Zugriff am: 14.10.2023

IHK Kassel 2023

IHK Kassel: *Merkblatt zur Abschlussprüfung der IT-Berufe.*
<https://www.ihk.de/blueprint/servlet/resource/blob/5797456/26716e5bcbe00e51a906d778e3cd41e7/umsetzungshilfe-it-berufe-data.pdf>. Version: 2023. – Zugriff am: 14.10.2023

Langr u. a. 2015

Langr, Jeff ; Thomas, Dave ; Hunt, Andy: *Pragmatic Unit Testing: in Java 8 with JUnit.* USA : The Pragmatic Programmers, 2015. – ISBN 9781941222591

Sommerville 2007

Sommerville, Ian: *Software Engineering.* Pearson Studium, 2007. – ISBN 9783868940992

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase	10 h
1. Durchführen der Ist-Analyse	2 h
2. Durchführen der Wirtschaftlichkeitsanalyse	2 h
3. Ermitteln von Use-Cases	3 h
4. Unterstützen des Fachbereiches bei der Erstellung des Lastenheftes	3 h
Entwurfsphase	12 h
1. Entwerfen der Benutzeroberfläche	2 h
2. Entwerfen der Datenbankstruktur (ERM)	3 h
3. Ableiten des Tabellen- und Domänenmodells aus dem ERM	3 h
4. Planen der Architektur	1 h
5. Erstellen des Pflichtenheftes	3 h
Implementierungsphase	32 h
1. Anlegen des Java-Projekts	1 h
2. Einrichten des Builds und der statischen Code-Analyse	2 h
3. Implementieren der Speicherung der Beitragsverläufe in einem File System in INI -Datei	3 h
4. Implementieren der Domänenklassen inkl. Tests	3 h
5. Herstellen der File System-Verbindung in Java	2 h
6. Implementieren der Datenverarbeitung in Java	16 h
6.1. Implementieren der Verarbeitung der erst Einträge inkl. Tests	6 h
6.2. Implementieren der Verarbeitung der Kosten inkl. Tests	3 h
6.3. Implementieren der Verarbeitung der Eintragsverläufe inkl. Tests	3 h
6.4. Implementieren der Ermittlung der Klassen Diagramm -Daten	4 h
7. Implementieren der Oberfläche inkl. Tests	5 h
Abnahme und Deployment	7 h
1. Code-Review mit dem Ausbilder	2 h
2. Abnahme durch die Fachabteilung	1 h
3. Test Der Anwendung	2 h
4. Deployment der Anwendung	2 h
Erstellen der Dokumentation	19 h
1. Erstellen des Benutzerhandbuchs	19 h
	80 h

A.2 Verwendete Ressourcen

Hardware

- Büroarbeitsplatz mit Laptop

Software

- [IntelliJ](#) für Java SE Developers – Entwicklungsumgebung [Java SE 8](#)
- [IntelliJ](#) mit [Swing](#) für die grafische Oberfläche.
- Draw.io – Programm zum Erstellen verschiedener Modelle und Diagramme
- [JUnit](#) – Framework zur Durchführung von Unit-Tests
- FileSystem ([.ini](#)) – Datenbanksystem
- Windows 10 Enterprise– Betriebssystem

Personal

- Anwendungsentwickler – Review des Codes
- Auszubildender – Umsetzung des Projektes
- Mitarbeiter der [Edok](#)_Abteilung
Festlegung der Anforderungen, Abstimmung der Oberfläche und Abnahme des Projektes

A.3 Entity-Relationship-Modell

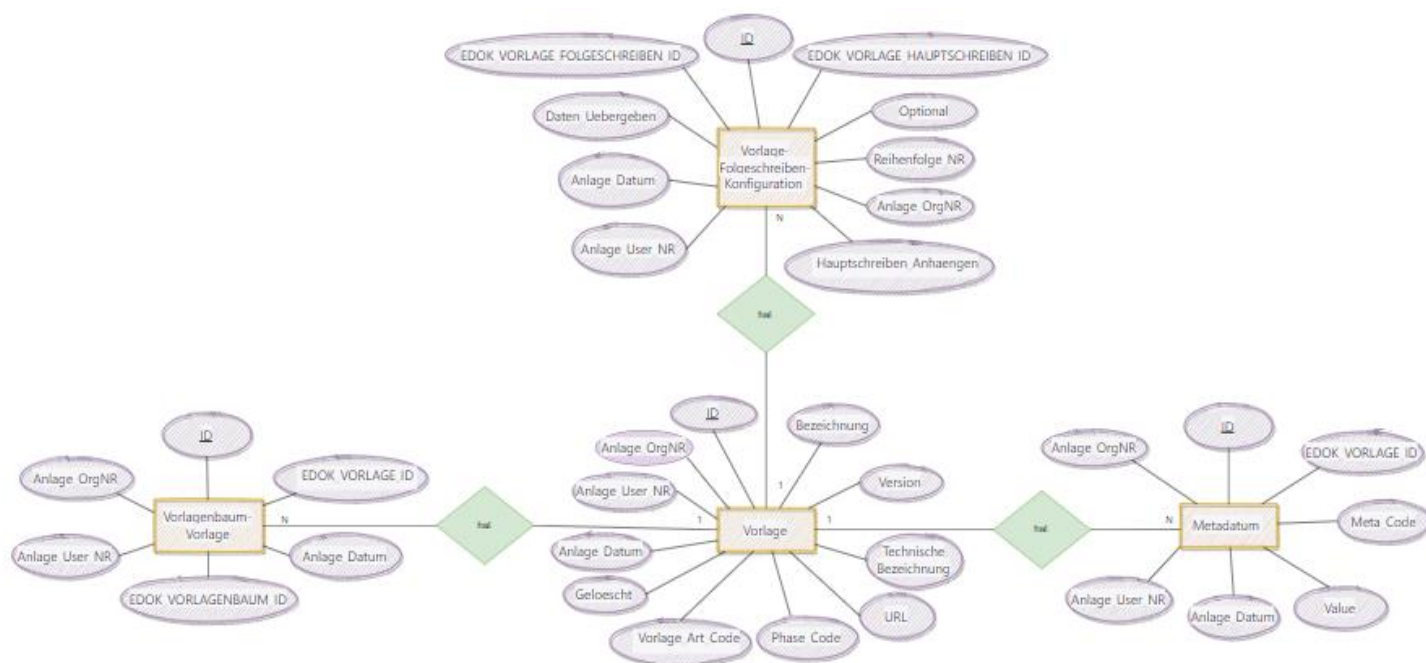


Abbildung 1: Entity-Relationship-Modell

A.4 Lastenheft

Die Anwendung muss folgende Anforderungen erfüllen:

1. Muss eine Benutzeroberfläche zur Eingabe und Verwaltung von Metadaten bieten.
2. Muss die Möglichkeit bieten, Vorlagen mit den erforderlichen Metadaten zu erstellen.
3. Muss die Generierung von [SQL](#)-Skripten ermöglichen.
4. Muss Inserts für die folgenden Datenbanktabellen erzeugen können:
 - 4.1. [Edok_Vorlage](#)
 - 4.2. [Edok_Metadatum](#)
 - 4.3. [Edok_Vorlagenbaum_Vorlage](#)
 - 4.4. [Edok_Folgeschreiben_Konfiguration](#)
5. Muss eine eindeutige und 6-stellige ID für jede Vorlage vergeben können
6. Muss Bezeichnungen, Versionen und technische Bezeichnungen für Vorlagen verwalten können.
7. Muss URLs basierend auf auswählbaren Pfaden und technischer Bezeichnung generieren können.
8. Muss Dropdown-Felder für Vorlage_Art_Code, Phase_Code, Geloescht, Anlage_User_NR und Anlage_OrgNR bereitstellen, die konfigurierbar sind.
9. Muss eine 1: n-Beziehung zwischen Vorlagen und Metadaten herstellen können.
10. Muss eine 1: n-Beziehung zwischen Vorlagen und Vorlagenbäumen ermöglichen.
11. Muss eine 1: n-Beziehung zwischen Vorlagen und Folgeschreiben-Konfigurationen unterstützen.
12. Muss jeden erzeugten [SQL](#)-Script mit einem Commit abschließen.
13. Muss am Anfang jedes erstellten [SQL](#)-Scripts ein Delete durchführen, wenn Daten zur Vorlage in der Datenbank vorhanden sind.
14. Muss Scripts für jede Vorlage mit allen zugehörigen Inserts ablegen können, benannt nach der Vorlagenbezeichnung (edok_vorlage.bezeichnung).
15. Muss die Möglichkeit bieten, Konfigurationsmöglichkeiten (z.B., Dropdown-Werte) in der Datenbank oder im Filesystem zu hinterlegen.
16. Muss die Möglichkeit bieten, Nummernvergabe für IDs zu implementieren.

A.5 Amortisation

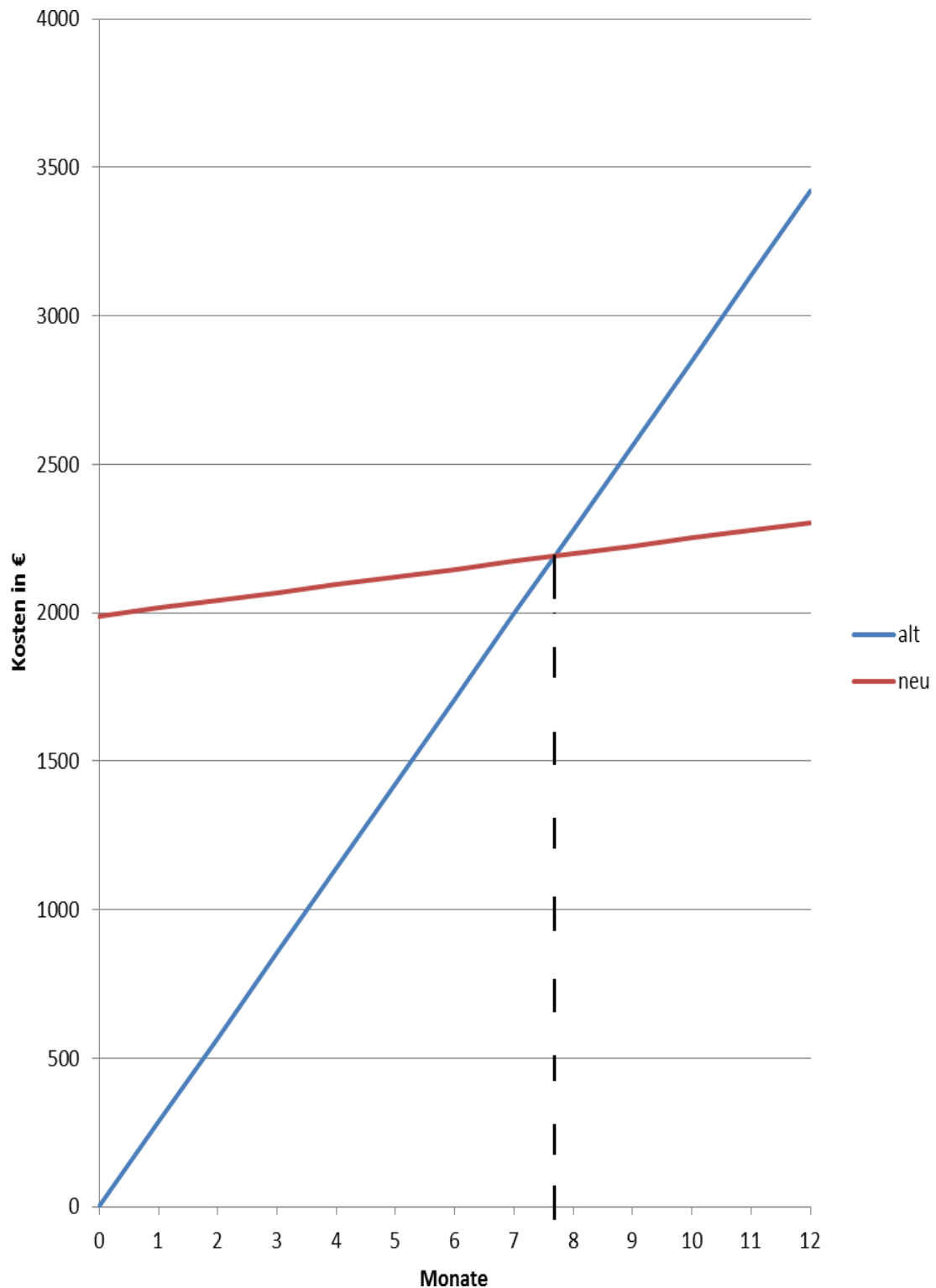


Abbildung 2: Grafische Darstellung der Amortisation

A Anhang

A.6 GANTT-Diagramm

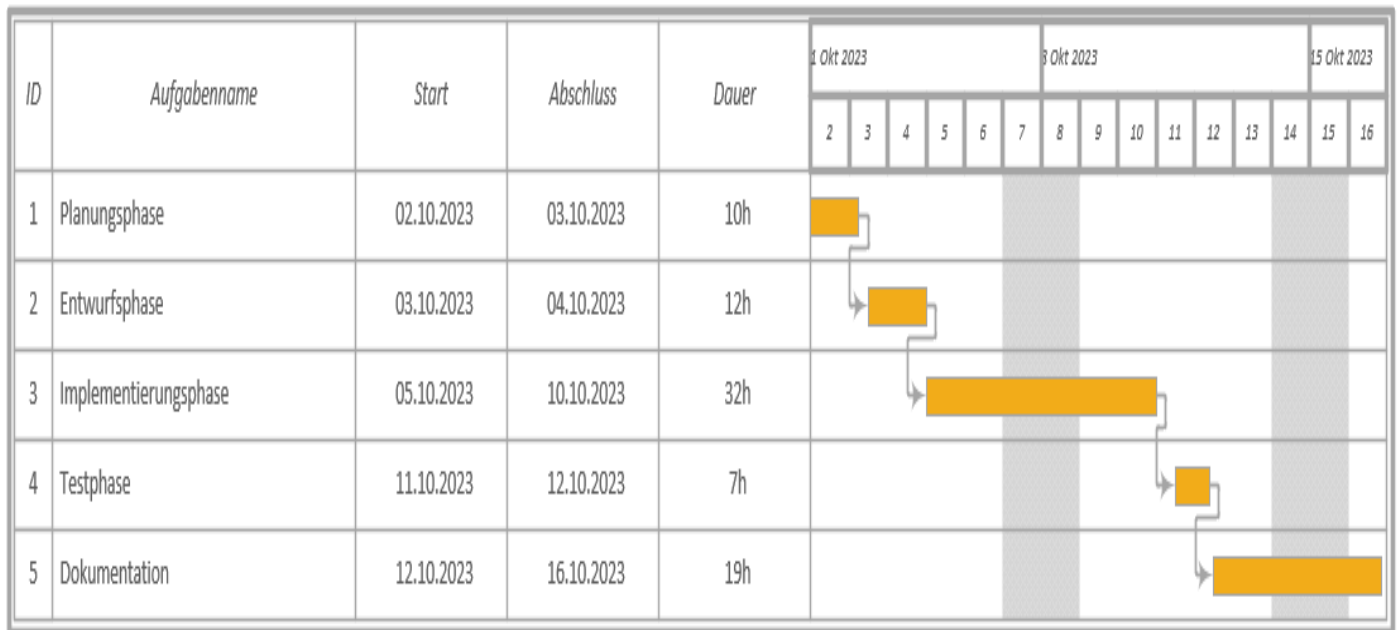


Abbildung 3: Gantt-Diagramm

A.7 Netzplan

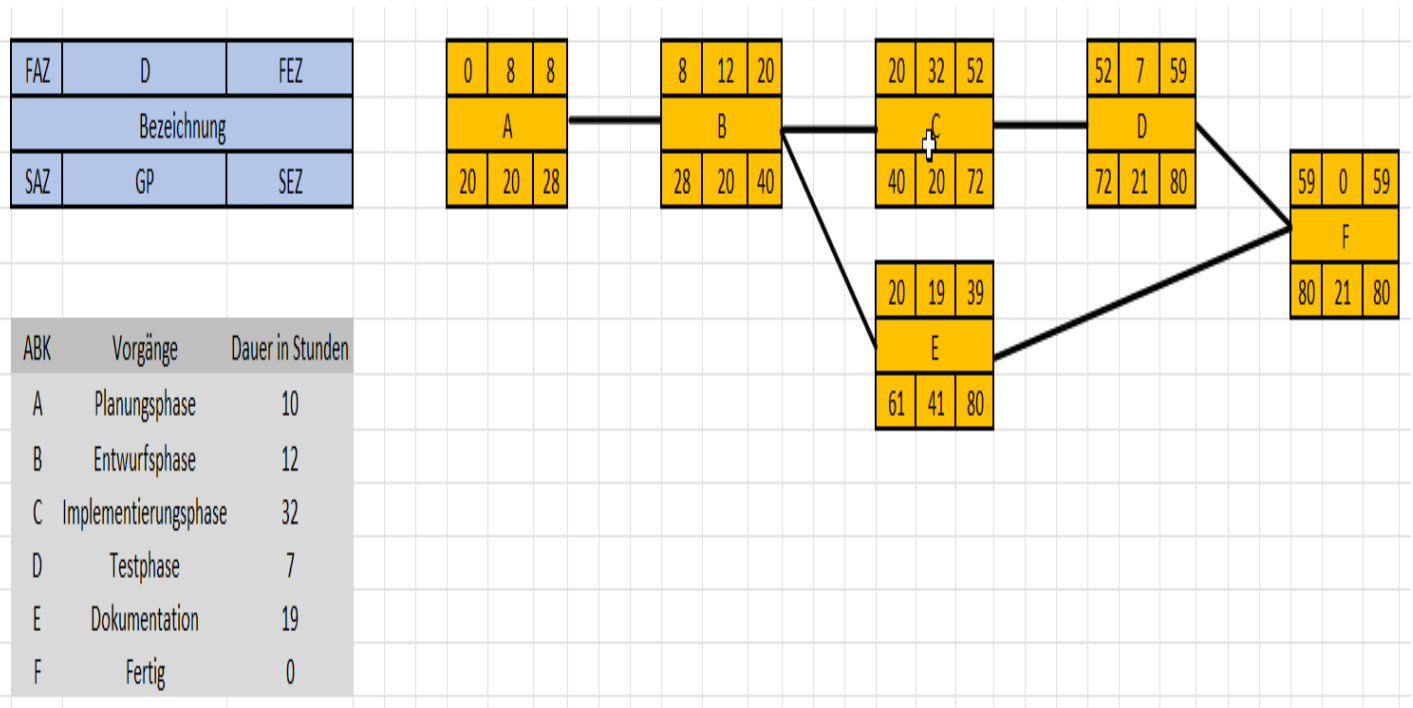


Abbildung 4: Netzplan

A.8 Klassen-Diagramm

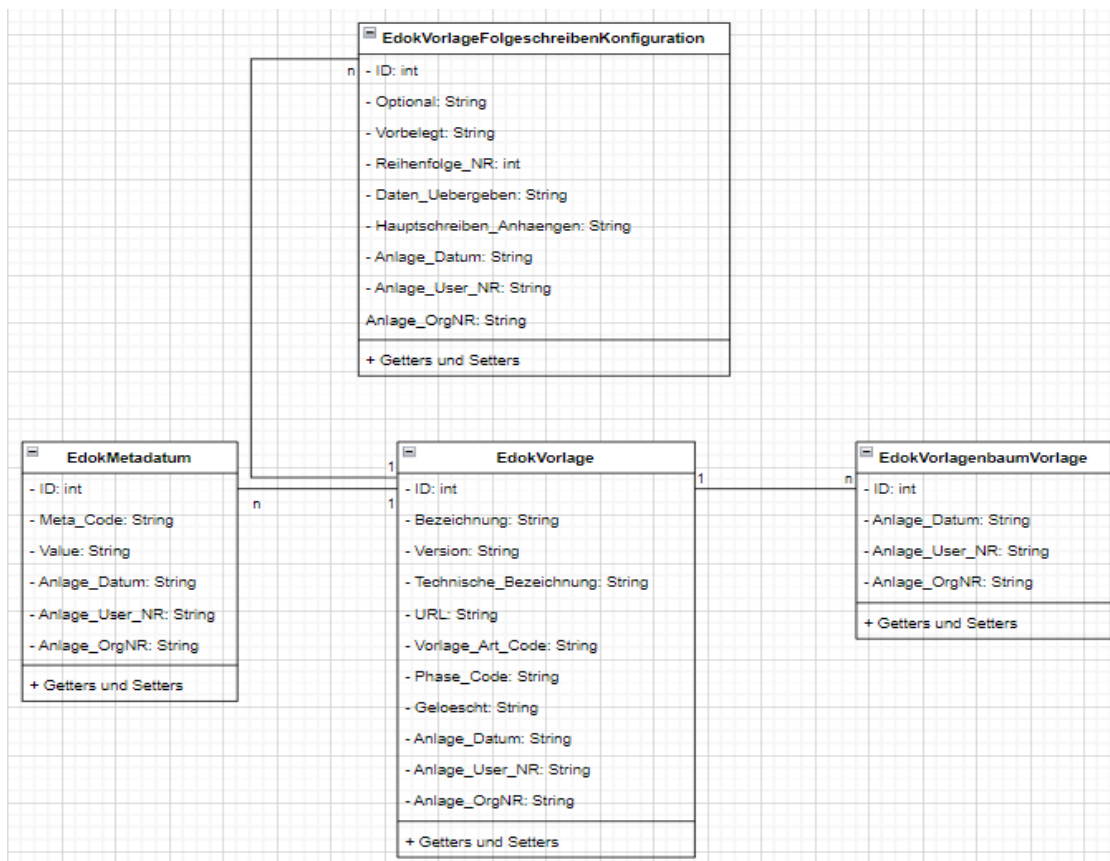
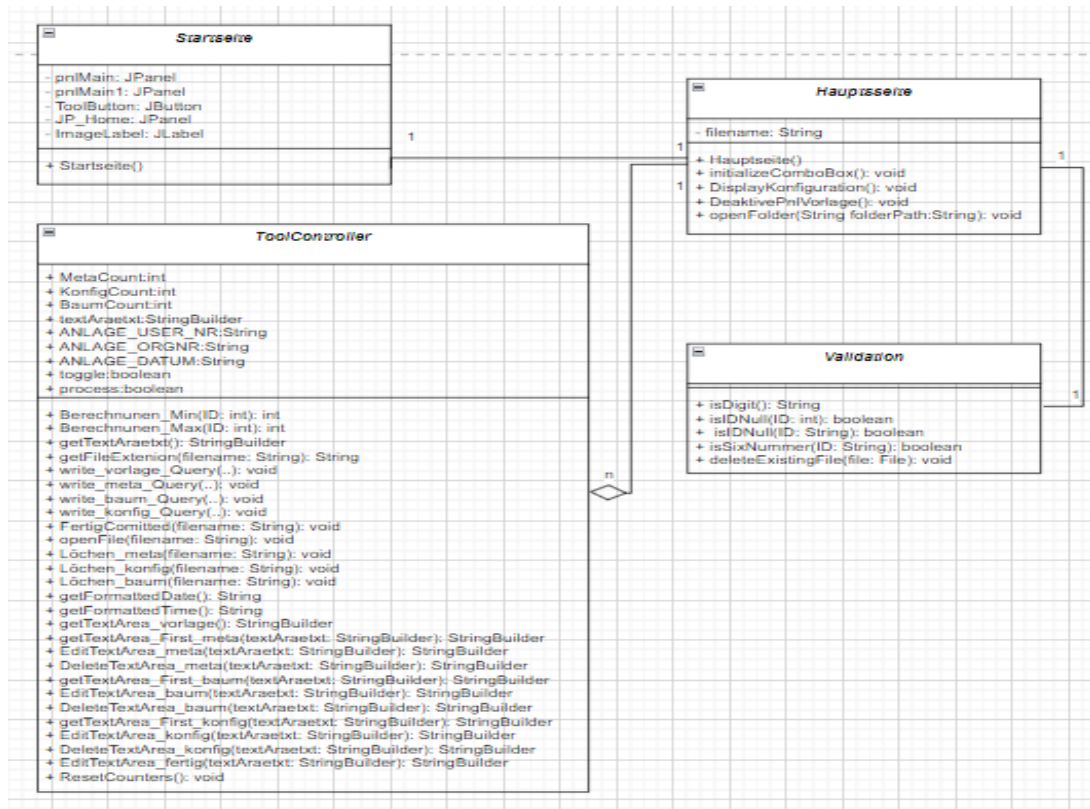


Abbildung 5: Klassen-Diagramm

A.9 Pflichtenheft (Auszug)

In folgendem Auszug aus dem Pflichtenheft wird die geplante Umsetzung der im Lastenheft definierten Anforderungen beschrieben:

1. Plattform

- 1.1. Zur Entwicklung der Anwendung wird [IntelliJ](#) für Java SE Developers eingesetzt.
- 1.2. Die Anwendung wird mit der Programmiersprache Java in der Version 8 programmiert.
- 1.3. Weitergehend wird die Spezifikation [Java SE 8](#) eingesetzt.

2. Oberfläche

- 2.1. Die Oberfläche wird mit [Swing](#) umgesetzt werden.

3. Geschäftslogik

- 3.1. Die Text-Dateien werden mit der Java-Bibliothek erstellt.
- 3.2. Die Generierung von [SQL](#)-Skripten für jede Schriftstückvorlage basierend auf den eingegebenen Metadaten.
- 3.3. Die Verwaltung von eindeutigen 6-stelligen IDs für jede Vorlage, die nicht geändert werden können.
- 3.4. Das Merken und Verwalten bereits vergebener IDs.
- 3.5. Die Erstellung von [SQL](#)-Skripten, die am Anfang ein Delete für vorhandene Daten zur Vorlage in der jeweiligen Datenbank durchführen.
- 3.6. Die Unterstützung einer flexiblen Konfiguration für Dropdown-Werte und andere Einstellungen, die in der Datenbank oder im Filesystem hinterlegt werden können.
- 3.7. Die Möglichkeit zur Anpassung der Nummernvergabe für IDs.
- 3.8. Die Implementierung der Commit-Funktion am Ende jedes erstellten [SQL](#)-Scripts

4. Anforderungen an [SQL](#)-Skript-Generierung:

- 4.1. Das Tool generiert [SQL](#)-Skripte für die folgenden Tabellen:

[Edok_Vorlage](#),[Edok_Metadatum](#),[Edok_Vorlagenbaum_Vorlage](#),[Edok_Folgeschreiben_Konfiguration](#).

- 4.2. Jedes [SQL](#)-Skript beginnt mit einem Delete, um vorhandene Daten zur Vorlage in der jeweiligen Datenbank zu löschen.
- 4.3. Für jede Vorlage wird ein [SQL](#)-Skript mit allen zugehörigen Inserts erstellt und unter der Bezeichnung der Vorlage (`edok_vorlage.bezeichnung`) abgelegt.

5. Konfigurationsmöglichkeiten:

- 5.1. Die Konfiguration von Dropdown-Werten und anderen Einstellungen kann im Filesystem ([.ini](#)) hinterlegt werden.

A.10 Listing von Java-Code

```

public static void write_meta_Query (
    String filePath, int IDMETA,
    String EDOK_VORLAGE_ID,
    String META_CODE,
    String VALUE,
    String ANLAGE_DATUM_META,
    String ANLAGE_USER_NR Meta,
    String ANLAGE_ORGNR_META)
{
    Löschen_comitted(filePath);
    String MetaQuery = "insert into `edok_metadatum` (`id`, `edok_vorlage_id`,
`meta_code`, 'value', 'anlage_datum', 'anlage_user_nr', 'anlage_orgnr') \n" +
        "values ('" + IDMETA + "', '" + EDOK_VORLAGE_ID + "', '" + META_CODE
+ "', '" + VALUE + "', '" + ANLAGE_DATUM_META + "', '" + ANLAGE_USER_NR_Meta + "', '"
+ ANLAGE_ORGNR_META + "')";
    try{
        // Write the text to the file
        if (MetaCount == 0) {
            // Create a BufferedWriter to write to the file
            BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true));
            writer.newLine();
            writer.write("-- Metadaten der Vorlage in Tabelle edok_metadatum hinzufügen ");
            writer.newLine();
            writer.write(MetaQuery);
            writer.newLine();
            addTextArea_First_meta(textAraetxt);
            // Close the writer
            writer.close();
            MetaCount++;
        } else {
            String targetWord = "Metadaten";
            FileReader fileReader = new FileReader(filePath);
            BufferedReader reader = new BufferedReader(fileReader);
            StringBuilder fileContent = new StringBuilder();
            String line;

            while ((line = reader.readLine()) != null) {
                fileContent.append(line).append("\n");

                if (line.contains(targetWord)) {
                    // Insert the new text under the target word
                    fileContent.append(MetaQuery).append("\n");
                }
            }
            reader.close();
            // Write the updated content back to the file
            FileWriter fileWriter = new FileWriter(filePath, false);
            fileWriter.write(fileContent.toString());
            fileWriter.close();
            MetaCount++;
            EditTextArea_meta(textAraetxt);
        }
    } catch (IOException ex){
        JOptionPane.showMessageDialog(null, "Error saving text to file: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

Listing 1: Funktion write_meta_query in der ToolController Klasse


```
public static void Löschen_konfig(String filename) {

    Löschen_comitted(filename);
    String startWord1 = "Folgeschreibenkonfiguration";
    String startWord2 = "Folgeschreiben";
    String startWord3 = "insert into `edok_vorlage_folgeschreiben_konfiguration";
    String endWord = ");";

    StringBuilder text = new StringBuilder();

    try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
        String line;
        String previousLine = null;

        while ((line = reader.readLine()) != null) {

            if (line.contains(startWord1) || (line.contains(startWord2) ||
                (line.contains(startWord3))) {

                if (line.contains(startWord3)) {
                    KonfigCount--;
                }
                previousLine = null;
                toggle =true;

            } else {
                if (previousLine != null) {
                    text.append(previousLine).append("\n");
                }

                if (line.contains(endWord) && toggle ==true) {
                    previousLine=null;
                    toggle =false;
                    continue;
                }
                previousLine = line;
            }
        }
        // Append the last line if it wasn't skipped
        if (previousLine != null ) {
            text.append(previousLine).append("\n");
        }

        //reader.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
        writer.write(text.toString());

        // writer.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    DeleteTextArea_konfig(textAraetxt);
}
```

Listing 2: Funktion Löschen_Konfig in der ToolController Klasse

A.11 Finale SQL-Datei

```

6000wtb1.sql - Editor
Datei Bearbeiten Format Ansicht Hilfe
spool P:\V-Modell\CCM\Betrieb\Vorlagenkonfigurationen\logs\6000wtb1.LOG

declare
  lc_anlage_user_nr   edok_vorlage.anlage_user_nr%type := 'CCM-ADMIN';
  lc_anlage_orgrnr    edok_vorlage.anlage_orgrnr%type := 'CCM-ADMIN';

begin
  -- Vorlagenkonfiguration für diese Vorlage löschen
  delete from edok_vorlage_folgeschreiben_konfiguration where edok_vorlage_folgeschreiben_id between 300700 and 300799;
  delete from edok_vorlagenbaum_vorlage where id between 300700 and 300799;
  delete from edok_metadatum where id between 300700 and 300799;
  delete from edok_vorlage where id between 300700 and 300799;

  -- Vorlage in Tabelle edok_vorlage hinzufügen
  insert into `edok_vorlage` (`id`, `bezeichnung`, `version`, `url`, `vorlage_art_code`, `phase_code`, `geloescht`, `anlage_datum`, `anlage_user_nr`, `anlage_
values ('300700', 'Aktendeckblatt - 6000wtb1', '1', 'LWV-Ressourcen/Vorlagen/Integrationsamt/2_Begleitende_Hilfe/2_1_Sonstige_Schriftstuecke/2_1_1_Allgemein

  -- Metadaten der Vorlage in Tabelle edok_metadatum hinzufügen
  insert into `edok_metadatum` (`id`, `edok_vorlage_id`, `meta_code`, `value`, `anlage_datum`, `anlage_user_nr`, `anlage_orgrnr`)
values ('300700', '300700', 'ARBGE_DATA', 'BEZUG', 'sysdate', 'CCM-ADMIN', 'CCM-ADMIN');
  insert into `edok_metadatum` (`id`, `edok_vorlage_id`, `meta_code`, `value`, `anlage_datum`, `anlage_user_nr`, `anlage_orgrnr`)
values ('300700', '300700', 'ARBGE_DATA', 'BEZUG', 'sysdate', 'CCM-ADMIN', 'CCM-ADMIN');

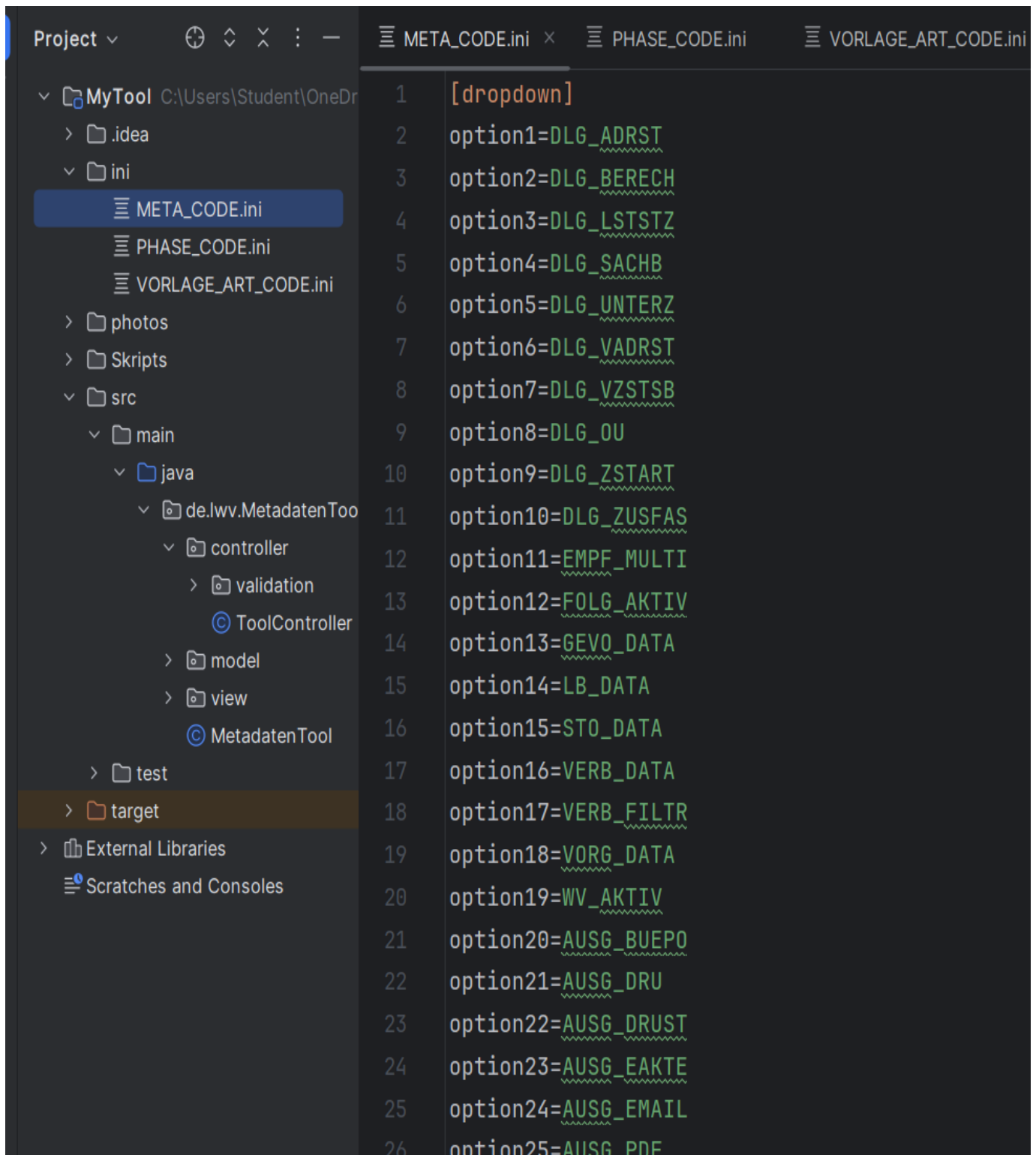
  -- Folgeschreibenkonfiguration in Tabelle edok_vorlage_folgeschreiben_konfiguration hinzufügen
  -- Folgeschreiben = 6050itb3 b
  insert into `edok_vorlage_folgeschreiben_konfiguration` (`id`, `edok_vorlage_hauptschreiben_id`, `edok_vorlage_folgeschreiben_id`, `optional`, `vorbelegt`,
values ('300700', '300700', '300004', 'J', 'N', '12', 'J', 'N', 'sysdate', 'CCM-ADMIN', 'CCM-ADMIN');
  -- Folgeschreiben = 6050itb3 b
  insert into `edok_vorlage_folgeschreiben_konfiguration` (`id`, `edok_vorlage_hauptschreiben_id`, `edok_vorlage_folgeschreiben_id`, `optional`, `vorbelegt`,
values ('300700', '300700', '300004', 'J', 'N', '12', 'J', 'N', 'sysdate', 'CCM-ADMIN', 'CCM-ADMIN');

  -- Vorlagenbaumeintrag in Tabelle edok_vorlagenbaum_vorlage hinzufügen
  -- Ordner = HePAS
  insert into `edok_vorlagenbaum_vorlage` (`id`, `edok_vorlagenbaum_id`, `edok_vorlage_id`, `anlage_datum`, `anlage_user_nr`, `anlage_orgrnr`)
values ('300700', '300004', '300700', 'sysdate', 'CCM-ADMIN', 'CCM-ADMIN');

  commit;
end;
/
spool off

```

Abbildung 6: Screenshot der Anzeige der SQL-Skript

A.12 INI-Dateien

```
Project ▾
├── MyTool C:\Users\Student\OneDr
│   ├── .idea
│   ├── ini
│   │   ├── META_CODE.ini
│   │   ├── PHASE_CODE.ini
│   │   └── VORLAGE_ART_CODE.ini
│   ├── photos
│   ├── Skripts
│   ├── src
│   │   ├── main
│   │   │   ├── java
│   │   │   │   ├── de.lwv.MetadatenToo
│   │   │   │   │   ├── controller
│   │   │   │   │   │   ├── validation
│   │   │   │   │   │   │   ├── ToolController
│   │   │   │   │   │   ├── model
│   │   │   │   │   │   └── view
│   │   │   │   │   └── MetadatenTool
│   │   └── test
│   └── target
├── External Libraries
└── Scratches and Consoles

META_CODE.ini
1 [dropdown]
2 option1=DLG_ADRST
3 option2=DLG_BERECH
4 option3=DLG_LSTSTZ
5 option4=DLG_SACHB
6 option5=DLG_UNTERZ
7 option6=DLG_VADRST
8 option7=DLG_VZSTSB
9 option8=DLG_OU
10 option9=DLG_ZSTART
11 option10=DLG_ZUSFAS
12 option11=EMPF_MULTI
13 option12=FOLG_AKTIV
14 option13=GEVO_DATA
15 option14=LB_DATA
16 option15=STO_DATA
17 option16=VERB_DATA
18 option17=VERB_FILTR
19 option18=VORG_DATA
20 option19=WV_AKTIV
21 option20=AUSG_BUEPO
22 option21=AUSG_DRU
23 option22=AUSG_DRUST
24 option23=AUSG_EAKTE
25 option24=AUSG_EMAIL
26 option25=AUSG_PDF
```

Abbildung 7: INI-Datei für Dropdown Menu

A.13 Screenshots



Abbildung 8: Screenshot der Anzeige von erste-Schnittstelle (LWV)

The image shows a web form titled 'Tool zur Verwaltung von Metadaten für Schriftstückvorlagen'. The form is for editing an 'Edok_vorlage' and contains the following fields:

- ID**: A text input field containing '300700'.
- BEZEICHNUNG**: A text input field containing 'Aktendeckblatt - 6000wtb1'.
- VERSION**: A text input field containing '1'.
- URL**: A text input field containing 'LWV-Ressourcen/Vorlagen/Integrationsamt/2_Begleit'.
- VORLAGE_ART_CODE**: A dropdown menu with 'VORLAGE' selected.
- PHASE_CODE**: A dropdown menu with 'PRODUKTION' selected.
- GELOESCHT**: Two radio buttons, 'Ja' (unchecked) and 'Nein' (checked).
- TECHNISCHE_BEZEICHNUNG**: A text input field containing '6000wtb1'.

At the bottom of the form are three buttons: 'Zurück', 'Speichern', and 'Die Skripts'.

Abbildung 9: Screenshot der Anzeige der Edok-Vorlage Formular

Haupttabelle	Childtabelle										
Edok_vorlage	Edok_metadatum										
ID: 300700	ID: 300700										
BEZEICHNUNG: Aktendeckblatt - 6000wtb1	EDOK_VORLAGE_ID: 300700										
VERSION: 1	META_CODE: ARBGE_DATA										
URL: LWV-Ressourcen/Vorlagen	VALUE: <input type="checkbox"/> Ja <input type="checkbox"/> Nein <input checked="" type="checkbox"/> Andere										
VORLAGE_ART_CODE: VORLAGE	BEZUG:										
PHASE_CODE: PRODUKTION											
GELÖSCHT: <input type="checkbox"/> Ja <input checked="" type="checkbox"/> Nein											
TECHNISCHE_BEZEICHNUNG: 6000wtb1											
Speichern	Löschen Speichern Offen										
MiniScreen ***** Details ***** Time: 13:42:30 Date: 2023-10-23 *****											
<table border="1"> <thead> <tr> <th>Query</th> <th>Anzahl</th> </tr> </thead> <tbody> <tr> <td>1.Vorlage</td> <td>1</td> </tr> <tr> <td>2.Meta</td> <td>2</td> </tr> <tr> <td>3.Konfig</td> <td>2</td> </tr> <tr> <td>4.Baum</td> <td>1</td> </tr> </tbody> </table> ***** Comitted! *****		Query	Anzahl	1.Vorlage	1	2.Meta	2	3.Konfig	2	4.Baum	1
Query	Anzahl										
1.Vorlage	1										
2.Meta	2										
3.Konfig	2										
4.Baum	1										
Edok_vorlage_folgeschreiben_konfiguration	Edok_vorlagenbaum_vorlage										
ID: 300700	ID: 300700										
EDOK_VORLAGE_HAUPTSCHREIBEN_ID: 300700	EDOK_VORLAGENBAUM_ID: 300004										
EDOK_VORLAGE_FOLGESCHREIBEN_ID: 300004	EDOK_VORLAGE_ID: 300700										
OPTIONAL: <input checked="" type="checkbox"/> Ja <input type="checkbox"/> Nein	ORDNER: HePAS										
VORBELEGT: <input type="checkbox"/> Ja <input checked="" type="checkbox"/> Nein											
REIHENFOLGE_NR: 12											
DATEN_UEBERGEBEN: <input checked="" type="checkbox"/> Ja <input type="checkbox"/> Nein											
HAUPTSCHREIBEN_ANHAENGEN: <input type="checkbox"/> Ja <input checked="" type="checkbox"/> Nein											
FOLGESCHREIBEN: 6050itb3_b											
Löschen Speichern	Löschen Speichern										
	Zurücksetzen Fertig										

Abbildung 10: Screenshot der Anzeige der Children Formular



Industrie- und Handelskammer
Kassel-Marburg

Bestätigung über durchgeführte Projektarbeit / durchgeführten betrieblichen Auftrag

Diese Bestätigung ist mit der Dokumentation einzureichen!

Ausbildungsberuf:

Prüfungsteilnehmer/Prüfungsteilnehmerin:

Ahmad Darwish

E-Mail: ahmaddarwish2018@gmail.com

Ausbildungsbetrieb/Umschulungsträger:

Landeswohlfahrtsverband Hessen
Ständeplatz 6-10
34117 Kassel

E-Mail: maurus.eichenberg@lwv-hessen.de

Projekt-/Auftragsbezeichnung:

Tool zur Verwaltung von Metadaten für Schriftstückvorlagen

Beginn: 02.10.2023

Fertigstellung: 16.10.2023

Zeitaufwand in Stunden: 80

Bestätigung des Ausbildungs-/Praktikumsbetriebes:

Mit den Unterschriften bestätigt der Ausbildungs-/Praktikumsbetrieb, dass sich die/der bezeichnete Projektarbeit/betriebliche Auftrag nicht auf Betriebsgeheimnisse bezieht und keine datenschutzrechtlichen Bedenken bestehen. Weiterhin wird bestätigt, dass der/die Auszubildende die o.g. Projektarbeit/den o.g. betrieblichen Auftrag einschließlich der Dokumentation im Zeitraum

von 02.10.2023 bis 16.10.2023 selbständig ausgeführt hat.

Die von der Verordnung vorgesehene Richtzeit wurde eingehalten.

Projekt-/Auftragsverantwortliche/r in der Firma:

Vor- und Nachname: Maurus Eichenberg

Telefon: 0561 1004 - 2307

Unterschrift: 

Ausbildungsverantwortliche/r in der Firma:

Vor- und Nachname: Thomas Kolb

Telefon: 0561 80700 - 31

Unterschrift: 

Persönliche Erklärung des Prüfungsteilnehmers:

Ich versichere durch meine Unterschrift, dass ich das Projekt/den betrieblichen Auftrag und die dazugehörige Dokumentation selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

Ich bin darüber aufgeklärt worden, dass meine Arbeit bei Täuschungshandlungen bzw. Ordnungsverstößen mit „null Punkten“ bewertet wird und als nicht bestanden gilt.

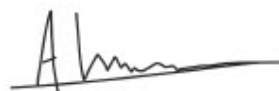
Ich bin weiter darüber aufgeklärt worden, dass dies auch dann gilt, wenn festgestellt wird, dass meine Arbeit im Ganzen oder zu Teilen mit der eines anderen Prüfungsteilnehmers übereinstimmt.

Ich nehme zur Kenntnis, dass ggf. stichprobenartige Kontrollen durchgeführt werden können.

Kassel 14.11.2023

Ort, Datum:

Unterschrift des Prüfungsteilnehmers





Industrie- und Handelskammer
Kassel-Marburg

Ausbildungsberuf: Fachinformatiker Anwendungsentwicklung

Prüfungsteilnehmer Ahmad Darwish

Ausbildungsbetrieb Landeswohlfahrtsverband Hessen Ständeplatz 6-10 34117 Kassel

Protokoll über die/den durchgeführte(n) Projektarbeit / betrieblichen Auftrag

1. Arbeitszeit

1.1 Die vom Prüfungsteilnehmer kalkulierte Zeit entspricht

der betrieblichen Kalkulation ja ☒ nein ☐

wenn nein: Sie ist um _____ % höher
 _____ % niedriger

1.2 Das Projekt wurde vom Prüfungsteilnehmer in der kalkulierten Zeit komplett fertiggestellt (einschließlich eventueller Nacharbeit).

ja ☒ nein ☐

Wenn nein: Um _____ Stunden früher fertig geworden
 _____ Stunden länger gebraucht

2. Ausführung

2.1 Wurde das Projekt entsprechend dem eingereichten Konzept ausgeführt?

ja ☒ nein ☐

Wenn nein: Welche Änderungen ergaben sich?

2.2 Wurde das Projekt selbständig und ohne fremde Hilfe ausgeführt?

ja ☒ nein ☐

Wenn nein: Begründung und Umfang der Hilfestellung

2.3 Das Projekt konnte ohne Nacharbeit in einem einwandfreien Zustand übergeben werden

ja ☒ nein ☐

Wenn nein: Begründung

3. Dokumentation

3.1 Die Dokumentation wurde vom Prüfungsteilnehmer selbständig und ohne fremde Hilfe erstellt:

ja ☒ nein ☐

Wenn nein: Welche Hilfestellung wurde gegeben?

3.2 Die Dokumentation entspricht den betrieblichen Anforderungen

ja ☒ nein ☐

Wenn nein: Worin bestehen die Abweichungen?

Ort, Datum: Kassel 14.11.2023

Unterschrift des Projektverantwortlichen: 

Unterschrift des Prüfungsteilnehmers: 