|  |
| --- |
|  |
| Winterprüfung 2018 |
|  |
| Fachinformatiker Anwendungsentwicklung |
| Dokumentation zur praktischen Projektarbeit |
|  |
|  |
| Antrags- und Formularerstellungstool |
| Werkzeug zur modularen Antrags- und Formularerstellung mit Prozessabwicklung |
|  |
| **Abgabetermin: 07.12.2018** |
|  |
| **Kunde:**  ATIW Berufskolleg Paderborn  Riemkestraße 160  33106 Paderborn |
|  |
| **Ausbildungsbetrieb:**  Atos IT Dienstleistung und Beratung GmbH  Bruchstraße 5  45883 Gelsenkirchen |
|  |
|  |
|  |
|  |
|  |

Inhalt

[1 Einführung 1](#_Toc531161781)

[2 Auftrag 1](#_Toc531161782)

[2.1 Einsatzfeld des Produktes 1](#_Toc531161783)

[2.2 Ist-Zustand 1](#_Toc531161784)

[2.3 Soll-Zustand 1](#_Toc531161785)

[3 Analyse 2](#_Toc531161786)

[3.1 Analyse des Problems 2](#_Toc531161787)

[3.2 Abgrenzung des zu entwickelnden Systems 2](#_Toc531161788)

[3.3 Zusammenfassung der Systemarchitektur 2](#_Toc531161789)

[3.4 Beschreibung der Schnittstellen 2](#_Toc531161790)

[3.5 Gewünschtes Verhalten/ Effekte 3](#_Toc531161791)

[3.5.1 Anwendungsfalldiagramme: 3](#_Toc531161792)

[3.5.2 Flussdiagramm: 4](#_Toc531161793)

[3.5.3 XML-Tags (Beispiele) 6](#_Toc531161794)

[3.6. Produkt-/Funktionsübersicht 7](#_Toc531161795)

[3.6.1 Muss-Funktionen 7](#_Toc531161796)

[3.6.2 Soll-Funktionen 7](#_Toc531161797)

[3.6.3 Kann-Funktionen 7](#_Toc531161798)

[3.7 Praktische Umsetzung der Produkt-/Funktionsübersicht 8](#_Toc531161799)

[3.7.1 Muss-Funktionen 8](#_Toc531161800)

[3.7.2 Soll-Funktionen 9](#_Toc531161801)

[3.7.3 Kann-Funktionen 10](#_Toc531161802)

[3.8 Beurteilung der Machtbarkeit 10](#_Toc531161803)

[3.8.1 Projektstrukturplan 10](#_Toc531161804)

[3.8.2 Zeitplanung 11](#_Toc531161805)

[3.8.3 11](#_Toc531161806)

[Ablaufkontrolle und Meilensteine 11](#_Toc531161807)

[3.9 Anforderungen an die Qualität 12](#_Toc531161808)

[3.10 Einsatz von Techniken und Tools 12](#_Toc531161809)

[4 Tests 13](#_Toc531161810)

[4.1 Test Cases 13](#_Toc531161811)

[5 Anhang 14](#_Toc531161812)

[5.1 Abbildungsverzeichnis: 14](#_Toc531161813)

# 1 Einführung

Das Projekt des Antrags- und Formularerstellungstool wurde uns am 13.11.2018 vorgeschlagen. Es handelt sich um ein Tool, was die internen Prozesse in der Berufsschule ATIW in Paderborn vereinfachen soll.

# 2 Auftrag

## 2.1 Einsatzfeld des Produktes

Das Produkt soll lediglich intern im ATIW von den Lehrkräften und Schülern genutzt werden.

Die Lösung soll ausschließlich in der ATIW Berufsschule zum Einsatz kommen und den regelmäßigen bürokratischen Aufwand reduzieren. Nutzer sind Verwaltungs-, sowie Lehr- bzw. Hilfskräfte. Diese sind sowohl Nutzer als auch Verwalter der erstellten Systemumgebung.

Außerdem wird das Tool nur über das Lehrernetz erreichbar sein.

## 2.2 Ist-Zustand

Das Kollegium der ATIW Berufskolleg gGmbH Paderborn kam am 13.11.2018 auf uns zu, mit der Anfrage für einen Auftrag. Das Problem des Kollegiums ist das interne Antrags- und Formularwesen. Zurzeit müssen Anträge in Word- bzw. PDF Vorlagen ausgefüllt, ausgedruckt und unterschrieben werden, bevor Sie zu Fuß an die nächst höhere Instanz weitergeleitet werden. Anträge in Papierform erzeugen unnötigen bürokratischen Aufwand, der sich durch digitale Abläufe reduzieren ließe. Dementsprechend beauftragt uns das Berufskolleg eine Lösung für Ihre Antrags- und Formularbearbeitung zu entwickeln. Es liegen keine festen Vorstellungen über die Realisierung kundenseitig vor.

## 2.3 Soll-Zustand

Das Projektteam schlägt eine Stand Alone Lösung vor. Diese verwaltet alle Prozesse und Formularvorlagen an einem zentralen Ort. Die Prozesse sollen digitalisiert werden um möglichst viel Zeit und Laufaufwand einzusparen. Die Anträge und Prozesse sollen über ein XML-File geschrieben werden. XML-Files werden vorgeschlagen, da XML (Extended Markup Language) einfach und schnell erlernbar ist, sodass auch eine einfache Hilfskraft Prozesse schreiben kann. Außerdem dient es der Beschreibung des Gesamtprozesses mit all seinen benötigten Daten und Verantwortlichen und gibt gleichzeitig Namenskonventionen für die Datenbank und Vorlagenerstellung vor.

Unser Tool erlaubt es jeder Zeit weitere XML Files einzulesen. Dies bedeutet, dass immer neue Anträge und Prozesse ins Tool geladen werden können.

Zur Papierform kann durch explizite Anweisung in der XML-Datei in Form eines Belegs jederzeit zurückgekehrt werden.

Bestmöglich sollte ein Tool erstellt werden, welches das händische Ausfüllen der Vorlagen ersetzt und die zu prüfenden Daten in einer Datenbank abbilden kann. Dies sollte den Zeit und Papier bzw. Laufaufwand deutlich senken.

# 3 Analyse

## 3.1 Analyse des Problems

Das Hauptproblem liegt in dem persönlichen überbringen der Dokumente zur nächsten Instanz. Sowie dem Fehlen eines einheitlichen Bearbeitungstools.

## 3.2 Abgrenzung des zu entwickelnden Systems

Aufgrund strenger zeitlicher Begrenzungen soll das Projekt nicht dem Anspruch einer ganzeinheitlich Verwaltungslösung genügen. Vielmehr soll es als gedanklicher Ansatz betrachtet werden, der die Grundlage für eine Erweiterung der hier bereitgestellten Grundfunktionen bietet.

In diesem Sinne werden Schnittstellen bereitgestellt, sowie Programmfunktionen gekapselt um leichte Anpassung und Erweiterung möglich zu machen.

## 3.3 Zusammenfassung der Systemarchitektur

Das System ist grob in drei Komponenten unterteilt. Einen Datenbankserver auf MySQL Basis, einen Webserver zur Kommunikation mit dem HTTP-Protokoll, der in .NET entwickelt wird und alle grundlegenden Funktionalitäten bereitstellt, sowie einem Frontend für die User selbst, ebenfalls in .NET, das auf diese Funktionen zurückgreift.

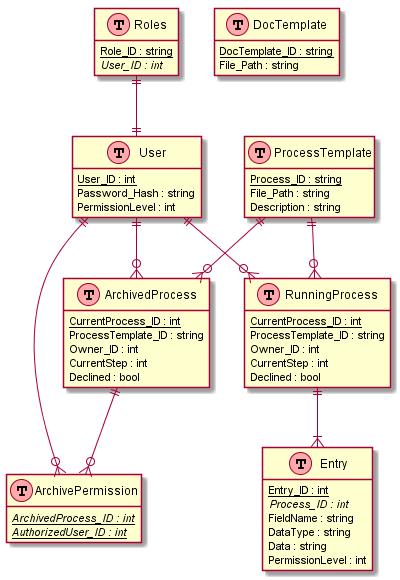


Abbildung 1 - ERD Der Datenbank

## 3.4 Beschreibung der Schnittstellen

Die Kommunikation zwischen MySQL-Server und HTTP-Server wird über die Standard .NET Bibliotheken realisiert und geschieht in Reaktion auf Interaktionen auf der Clientseite, die mit dem Webserver über eine REST-Schnittstelle kommuniziert (auch diese werden über .NET Bibliotheken umgesetzt). REST (für Representational State Transfer) bietet eine standardisierte Umsetzung einer Webschnittstelle an, die stateless ist, also ohne Sessions auskommt und stattdessen alle für eine Aktion benötigten Informationen wie Nutzernamen, Passwort und weiteres mitüberträgt. Als Vorteil ergibt sich zum Beispiel die Möglichkeit, auf die komplexe Verwaltung aktiver Usersessions zu verzichten, damit die Server zu entlasten und die Codebasis aufgeräumter zu halten. Zudem müssen weniger Ausnahmefälle berücksichtigt werden. Die .NET Standardbibliotheken reduzieren den Dokumentations- und Wartungsaufwand den externe Bibliotheken mit sich bringen würden.

## 3.5 Gewünschtes Verhalten/ Effekte

Die Software soll die vielen verschiedenen Prozesse in einer einheitlichen Oberfläche zusammenfassen, die leicht zu überblicken und Nutzen ist. Außerdem soll es obsolet machen sich den Prozessablauf für jedes Dokument einprägen zu müssen. Darüber hinaus reduziert die Verlagerung des Fokus auf digitale Verarbeitung den Papieraufwand und erhöht damit die Übersichtlichkeit.

Durch die Schnittstellen und eine ausführliche Dokumentation wird gewährleistet, dass weitere Anpassungen an die Kundenwünsche von Projektteam oder auch durch weitere externe Dienstleister umgesetzt werden können.

### 3.5.1 Anwendungsfalldiagramme:

Im Folgenden finden sich Anwendungsfalldiagramme zur Verdeutlichung des Zusammenspiels von Verantwortlichen oder Betroffenen und dem Prozessysstem.

Erstellung eines neuen Prozesses inklusive eines neuen Antrages, durch einen Administrator.



Abbildung 2 - Prozesserstellung UseCase

Nutzung eines bestehenden Prozesses durch beispielsweise einer Lehrkraft.



Abbildung 3 - Prozessnutzung UseCase

### 3.5.2 Flussdiagramm:

Die Prozessnutzung folgt einer klaren systematischen Abfolge deren technische Abläufe in der unteren Abbildung verdeutlicht werden



Abbildung 4 - Nutzung eines Prozesses

Der Prozess aus Nutzersicht hingegen ergibt sich wie folgt:



Abbildung 5 - Erstellung eines Prozesses

In diesem Zuge interagiert er mit verschiedenen Eingabedialogen in der generierten Benutzeroberfläche. Die untere Abbildung 6 zeigt exemplarisch einen solchen Dialog, für eine Unterrichtsortsverlegung.

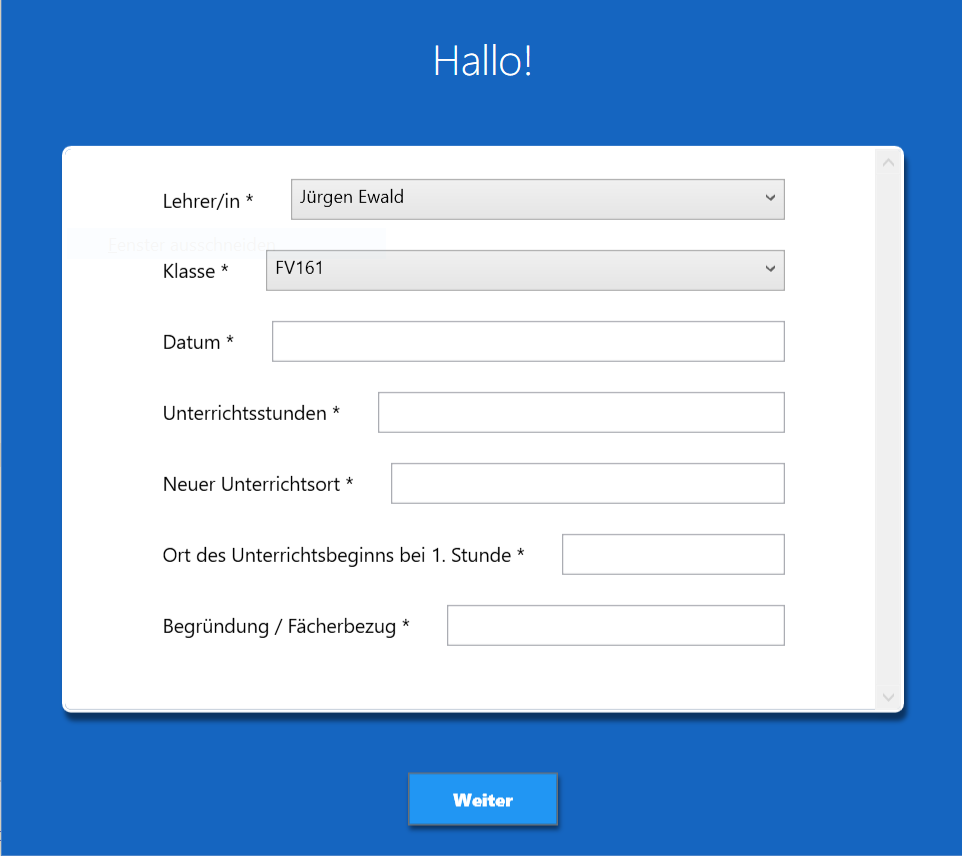


Abbildung 6 - Beispielabbildung der Formularansicht

Der in „Abbildung 6 - Beispielabbildung der Formularansicht“ dargestellten Benutzeroberfläche steht eine Prozessdatei gegenüber, in der sich auch die zu sehenden Benutzeroberflächenelemente wiederfinden. Ein Beispiel zeigt Abbildung 7.

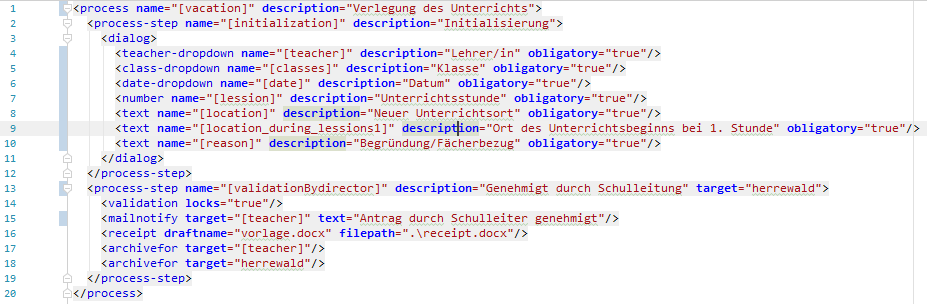


Abbildung 7 - XML - Prozessdatei (Vorläufig)

### 3.5.3 XML-Tags (Beispiele)

Die folgenden Befehle stellen ein vorläufiges Konzept dar. Syntax sowie Wording können sich im Laufe des Entwicklungsprozesses ändern.

|  |  |  |
| --- | --- | --- |
| Tags | Attribute | Beschreibung |
| <process> |  | Äußerer Tag zur Festlegung eines neuen Prozesses. |
| <process-step> |  | Ein einzelner Prozessschritt mit einem genauen Ziel. |
| <dialog> |  | Eine einzelne Ansicht mit Eingabefeldern. |
|  | name | Der technische Name der Eingabemethode. Datenbanktabellen und Platzhalter in Formularen werden an diesem Namen erkannt. |
|  | description | Der in der UI erscheinende Bezeichner. |
|  | obligatory | Erlaubt leere Eingaben. |
| Textboxes: |  |  |
| <number> |  | Erlaubt ganzzahlige Eingaben. |
| <decimal> |  | Erlaubt dezimale Eingaben. |
| <text> |  | Erlaubt beliebige Eingaben. |
| Dropdownmenue |  |  |
| <teacher-dropdown> |  | Lädt eine Liste von Lehrern aus der Datenbank und gibt standardmäßig Vor- und Nachname zurück. |
| <student-dropdown> |  | Lädt eine Liste von Schülern aus der Datenbank und gibt standardmäßig Vor- und Nachname zurück. |
| Other input types |  |  |
| <date-picker> |  | Zeigt eine Datumsauswahl an und gibt ein Datum im Format DD.MM.JJJJ. |
| Process Elements |  |  |
| <validation> |  | Zeigt einen Button zur Genehmigung des aktuellen Standes an. |
| <mailnotify> |  | Sendet eine Benachrichtigungsemail. |
|  | target | Muss vom Typ Teacher oder ein Standardtyp sein und gibt das Ziel an, dass die E-Mail erhalten soll. |
|  | text | Der Inhalt der Email. Darf Platzhalter beinhalten. |

## 3.6. Produkt-/Funktionsübersicht

### 3.6.1 Muss-Funktionen

* XML-Prozessdateien überprüfen
  + Auf Struktur
  + Auf Syntax
  + Allgemeine Korrektheit
* XML-Prozessdateien interpretieren
  + Daraus eine DB-Tabelle erzeugen
  + Daraus Objekte erzeugen
* UI dynamisch aus Objekten erzeugen
  + Einträge aus der UI in die erzeugte DB-Tabelle füllen
* Ermöglichung allgemeiner Prozessgestaltung
  + Genehmigung
  + Zielperson eines Prozessschrittes festlegen
  + Belege erzeugen
    - Automatische Befüllung von Word-Vorlagen nach der XML-Prozessdatei
  + E-Mail Bestätigung

### 3.6.2 Soll-Funktionen

* Prozessfortschritt in DB hinterlegen
* Benachrichtigungssystem (Programmintern)
* Rollensystem
* Standardrollen (Sekretärin, Schulleiter etc.)
* Upload und Aktualisierung von XML-Prozessdateien von Serverseite

### 3.6.3 Kann-Funktionen

* Einfache Rechenoperationen
* Eingabebedingungen

## 3.7 Praktische Umsetzung der Produkt-/Funktionsübersicht

### 3.7.1 Muss-Funktionen

#### 3.7.1.1 XML-Prozessdateien überprüfen

Um sicherzustellen, dass im weiteren Programmablauf keine Fehler geschehen, muss sichergestellt sein, dass Prozessdateien in sich korrekt und schlüssig sind. Zum Beispiel sollten Benutzeroberflächen-Elemente immer nur in ihren zugehörigen „<Dialog>“ Tags gefasst werden, statt frei im Dokument zu stehen.

* + Auf Struktur

Es muss sichergestellt sein, dass die allgemeine Struktur eines Dokuments eingehalten wird. Die allgemeine Struktur ist wie folgt:

***<process …>***

***<process-step …>***

***<dialog …>***

***<[inputelements] …/>***

***</dialog>***

***<[processelements] …/>***

***</process-step>***

***</process>***

* + Auf Syntax

Die Syntax muss eingehalten werden. Ein Beispiel wäre die Syntax für die Verwendung von Platzhaltern. Platzhalter werden so formuliert: „[Placeholder]“ und rühren her von technischen Namen in anderen Teilen der XML-Datei, die aber ohne Klammern formuliert werden.

Ein Beispiel wäre ein Eingabetag im ersten Prozessschritt, der später weiterverwendet wird:

***<process …>***

***<process-step …>***

***<dialog …>***

***<teacher-dropdown name=”teacher”…/>***

***</dialog>***

***</processstep>***

***<process-step …>***

***<mail-notify target=”[teacher]” …/>***

***</process-step>***

***</process>***

* + Allgemeine Korrektheit

An dieser Stelle wird die Einhaltung des allgemeinen Konzeptes geprüft. Das heißt, dass zum Beispiel kein Prozessschritt ohne Interaktion bestehen darf. Gibt es zum Beispiel weder ein Eingabefeld (Dialog -> Teacherdropdown) noch eine Genehmigungsaufforderung (Validate) sondern nur eine E-Mailbenachrichtigung (Mail-Notify) ist das kein gültiger Prozessschritt, da er obsolet wäre.

#### 3.7.1.2 XML-Prozessdateien zu Objekten interpretieren

Zu diesem Zweck wird die XML-Datei Tag für Tag ausgelesen. Zu Oberst liegt ein Process Objekt, dem dann eine Liste aus Process-Step Objekten zugeordnet wird. Jeder Process-Step hält dann eine Liste aus Dialog-Objekten, denen dann zuletzt Input-Elemente zugeordnet werden. Ebenfalls den Process-Steps untergeordnet, ist eine Auflistung an Prozessinteraktionen (Validate, etc.) und an Automatisierungen (Mail-Notify etc.). Diese Struktur ermöglicht die komplette Abbildung des Prozesses in .NET Objekten. Somit muss die XML-Datei nur einmal eingelesen werden und danach kann der komplette Prozess davon unabhängig abgearbeitet werden. Dadurch ist der Quelltext besser separiert und lesbar.

#### 3.7.1.3 UI dynamisch aus Objekten erzeugen

Die Benutzeroberfläche wird in Dialogverarbeitung gestaltet sein. Der Dialog Tag in einer Prozessdatei beschreibt einen eigenen Block mit Benutzeroberflächenelementen. Es wird immer nur ein Block auf einmal angezeigt.

Es gibt verschiedene Elemente zur Benutzereingabe, doch alle gehen auf die gleichen Basisklassen zurück. Das ermöglicht eine einheitliche Listenführung. In diesem Sinne haben alle Benutzereingabe-objekte mehrere Eigenschaften. Die genaue Struktur ist dem Klassendiagramm zu entnehmen. Gesagt sei aber, dass jedem Objekt ein WPF Control hinterlegt ist, dass in der Benutzeroberfläche erscheinen soll. Das Eingabe-Objekt dient als Container, der dieses Control erzeugt und mit Daten befüllt. Das kann zum Beispiel durch Anbindung an eine Datenbank geschehen.

Beispielsweise ist „teacher-dropdown“ einer normalen Combobox zugeordnet. Diese wird von der Containerklasse mit einer Lehrerliste aus der Datenbank befüllt.

##### 3.7.1.4Ermöglichung allgemeiner Prozessgestaltung

Die Orientierung dieser Softwarelösung am Prozessdenken macht es notwendig, für jeden Prozess-Schritt festlegen zu können, wer diesen durchführen soll. Dazu muss die Möglichkeit bestehen, einen Prozess in der Mitte abzubrechen, wenn Situationen eintreten, in denen er nicht mehr genehmigt werden kann. Tritt ein solcher Fall ein, dann muss der Nutzer informiert werden können, nicht nur in der Software selbst, die die notwendigen Informationen aus der Datenbank beziehen kann, sondern auch per E-Mail, sollte er derzeit keinen Zugriff haben.

Gesetzliche Vorgaben können es zuweilen nötig machen, zur Papierform zurückzukehren. Deshalb muss die Software eine Möglichkeit bieten, Belege in Form von Word Vorlagen vorzubereiten, diese um Platzhalter anzureichern (Name, Adresse, etc.), wobei diese auch dem bereits bekannten Platzhalterformat „[name]“ folgen und diese auf den Server hochzuladen. In der XML Datei kann dann der Name der Vorlage angegeben und diese automatisch befüllt werden. Platzhalter für die es keine Äquivalente gibt werden durch Leerzeichen ersetzt.

### 3.7.2 Soll-Funktionen

#### 3.7.2.1 Prozessfortschritt verfolgen

Die Datenbank muss Tabellen beinhalten, mit denen es möglich ist den Fortschritt eines angestoßenen Prozesses zu überprüfen. Die Software zeigt das dann im unteren Teil des Fensters als Fortschrittsanzeige an. In der XML-Prozessdatei erhalten die Prozess-Schritte auch Beschreibungstexte, diese werden dann für den aktuellen Schritt als Text in der UI angezeigt.

#### 3.7.2.2 Benachrichtigungssystem

Die Client-Software stellt in regelmäßigen Abständen Anfragen an den Server, ob neue Prozessinstanzen bereitstehen, die ein Mitarbeiter abarbeiten muss, sowie ob es für, durch einen Mitarbeiter, erstellte Prozesse Aktualisierungen gibt, z.B. diese abgelehnt wurden.

Der Webserver übermittelt den Status der aktuell für den User hinterlegten Prozesse und nach Abgleich der Fortschrittsdaten, erfragt der Client alle weiteren benötigten Informationen.

#### 3.7.2.3 Rollensystem

Dem bisher vorgestellten Konzept muss ein Rollensystem zu Grunde liegen um datenschutzrechtlichen Anforderungen zu genügen. In diesem Sinne wird jedoch darauf verzichtet für jeden Lehrer eigene Rollen zuzuweisen. Stattdessen, erhalten nur die Teilnehmer eines Prozesses die übermittelten Informationen. Zuletzt gibt es Standardrollen, die in der Datenbank leicht austauschbar sind. Zu diesen gehören zum Beispiel Schulleiter und Sekretariat. Mitglieder dieser Rolle können in den XML Dateien über diese Bezeichnungen angesprochen werden.

Beispiel:

***<process-step target=“Secretary“ …>***

Es ist also keine Platzhalternotation von Nöten. Findet sich in einem target Attribut ein Text ohne Klammern wird davon ausgegangen, dass es sich um eine Standardrolle handelt. Diese wird über die Datenbank abgeglichen.

#### 3.7.2.4 Verwaltung und Erreichbarkeit von Prozessdateien

Um die verschiedenen Prozesse allgemein Verfügbar zu machen, werden diese über einen Teil des Clienttools auf den HTTP-Server hochgeladen und werden von dort bei Anfragen bezogen. So können die Prozesse zentral verwaltet und aktualisiert werden, was Verwirrung vermeiden und Standardisierung erleichtern kann.

### 3.7.3 Kann-Funktionen

Bei Zeiten kann es ebenfalls nötig sein, Eingaben auf Bedingungen zu prüfen oder einfache Berechnungen durch Formeln in der XML-Datei durchzuführen. Diese würden .NET seitig als Strings aufgeschlüsselt, in Formeln umgewandelt, Platzhalter aufgelöst und dann berechnet werden.

## 3.8 Beurteilung der Machtbarkeit

### 3.8.1 Projektstrukturplan



Abbildung 8 - Projektstrukturplan

### 3.8.2 Zeitplanung

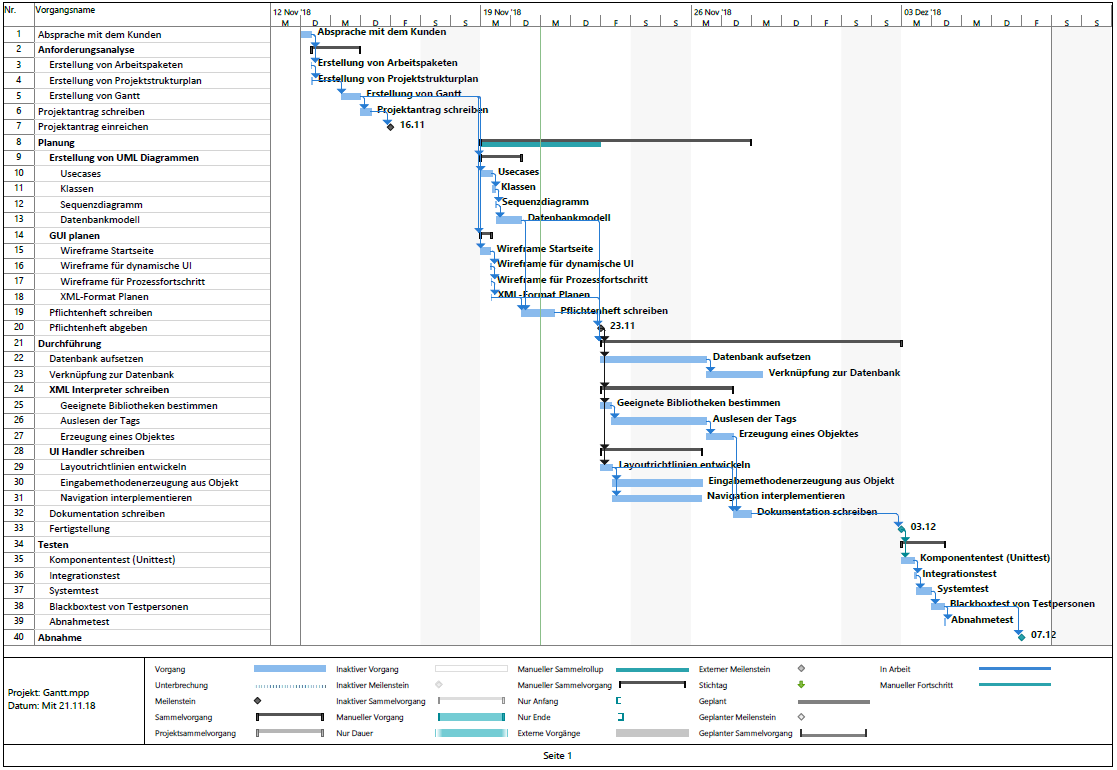


Abbildung 9 - GANTT - Diagramm

### 3.8.3 Ablaufkontrolle und Meilensteine

|  |  |
| --- | --- |
| Meilenstein | Datum |
| Projektantrag einreichen | 16.11.2018 |
| Pflichtenheft einreichen | 23.11.2018 |
| Fertigstellung eines lauffähigen Programmes | 03.12.2018 |
| Abnahme | 07.12.2018 |

## 3.9 Anforderungen an die Qualität

|  |  |
| --- | --- |
| Qualitätskriterium | Beschreibung |
| Korrektheit | Die Software sollte die Muss-Ziele erfüllen und verlässlich und fehlerfrei ausführen. |
| Robustheit | Die Software sollte auch bei fehlerhaften Eingaben und Benutzerinteraktionen geeignete Maßnahmen zur Fehlerbehebung ergreifen |
| Erweiterbarkeit | Die Software sollte gut erweiterbar sein, da der Grundgedanke des Projektes ist, eine Lösung zu erschaffen, die von jedem anderen Entwickler erweiter- und anpassbar ist. |
| Wiederverwendbarkeit | Die Software wird nur im Umfeld der ATIW Berufsschule zum Einsatz kommen, somit entfällt diese Anforderung. |
| Kompatibilität | Es bestehen wenige bis keine Schnittstellen zu anderen Programmen. Die Kompatibilität ist folglich zu vernachlässigen. |
| Portabilität | Die Software muss nicht auf andere Systeme übertragen werden, die Kompatibilität zwischen Windows-Versionen garantiert das .NET Framework. |
| Benutzerfreundlichkeit | Benutzerfreundlichkeit steht an zweiter Stelle hinter Erweiterbarkeit und soll nicht nur für den Nutzer, sondern auch für den Administrator gegeben sein. |

## 3.10 Einsatz von Techniken und Tools

|  |  |  |
| --- | --- | --- |
|  | **GitHub** | GitHub ist ein Onlinedienst, der Software-Entwicklungsprojekte auf seinen Servern bereitstellt (Filehosting). Es bietet die Versionsverwaltung von quelloffener Software über ein Repository (Verzeichnis) an und erleichtert damit das gemeinsame Arbeiten an einem Projekt |
|  | **Rider** | JetBrains Rider ist eine Cross-Plattform-IDE, welche die Ansätze von ReSharper und der Java Entwicklungsumgebung IntelliJ IDEA kombiniert. Es erleichtert die Arbeit mit Live Code Inspection, Refactorings, Navigationshilfen, einem Runner für Unit-Tests, Debugging-Werkzeugen und Programmierhilfen. |
|  | **PlantUML** | PlantUML ist ein quelloffenes Projekt zur Erstellung von UML Diagrammen in einer eigenen DSL (Domain Specific Language). Damit sind UML Diagramme leicht über Versionsverwaltungen austauschbar und der Formatierung wird automatisch Sorge getragen. |
|  | **.NET** | Die Software wird auf Basis des .NET Frameworks entwickelt. Einem Framework, dass mit einer weitreichenden Windows-Integration und hoher Kompatibilität aufwartet. Die zugehörige Programmiersprachen sind VC++, C#, VB und F#, wobei wir uns an dieser Stelle auf C# beschränken. |
|  | **REST** | Representational State Transfer bezeichnet ein Programmierparadigma für insbesondere Webservices. REST hat das Ziel, einen Architekturstil zu schaffen, der die Anforderungen des modernen Web besser darstellt. So sind kommt REST „Stateless“ aus. Es werden also alle für eine Anfrage erforderlichen Informationen (Username, Passwort etc.) jedes Mal mitübertragen, um eine Sessionverwaltung auf Serverseite zu vermeiden. |
|  | **MySQL** | MySQL ist eines der weltweit verbreitesten relationalen Datenbankverwaltungssysteme. |
|  | **Visio 2016** | Microsoft Visio ist ein Visualisierungsprogramm, welches mit verschiedenen Vorlagen, passenden Werkzeugen und Symbolen grafische Darstellungen erzeugt. Besonders geeignet ist es für Ablaufdiagramme oder zur Modellierung von Geschäftsprozessen. |

# 4 Tests

|  |  |
| --- | --- |
| Komponententest | Der Komponententest prüft die einzelnen Funktionalitäten innerhalb abgrenzbarer Teile der Software |
| Integrationstest | Der Integrationstest testet die Zusammenarbeit abhängiger Komponenten, wie zum Beispiel dem XML Interpreter und der dynamisch erzeugten UI. |
| Systemtest | Der Systemtest prüft unsere Muss-, Kann- und Soll-Ziele, bestmöglich in der Produktivumgebung |
| Blackboxtest | Der Blackboxtest wird von Testpersonen durchgeführt, die keinen Einblick auf den Quellcode haben. |
| Abnahmetest | Der Abnahmetest erfolgt durch den Kunden, um letzte Anpassungen oder Fehlerbehebungen zu ermöglichen. |

## 4.1 Test Cases

|  |  |  |
| --- | --- | --- |
| Test Case ID | Testart | Test Beschreibung |
| 01 | Komponententest | Test der Netzwerkkomponente auf dem Client. |
| 02 | Komponententest | Test der Objekterzeugung des XML-Interpreters. |
| 03 | Komponententest | Test der Anfragenverarbeitung der REST Schnittstelle . |
| 04 | Komponententest | Test der Sicherheitsmechanismen der REST Schnittstelle. |
| 05 | Komponententest | Test der UI Erzeugung aus Objekten. |
| 06 | Integrationstest | Upload und Download von Dokumenten auf dem Fileserver testen. |
| 07 | Integrationstest | Interpretation der ausgelesenen XML-Prozessdatei zur dynamisch erzeugten UI. |
| 08 | Integrationstest | Erzeugung eines Beleges aus der Datenbank. |
| 09 | Integrationstest | Erzeugung von Tabellen entsprechend der XML-Prozessdatei |
| 10 | Systemtest | Überprüfen der oben genannten Funktionen. |
| 11 | Blackboxtest | Potentielle Nutzer sollen einen Beispielprozess erzeugen und ausführen. |
| 12 | Abnahmetest | Kunde testet das System nach seinen Testkriterien |

# 5 Anhang

## 5.1 Abbildungsverzeichnis:

|  |  |
| --- | --- |
| Abbildung | URL |
|  | <https://github.com/> |
|  | <https://twitter.com/jetbrainsrider> |
|  | <https://meetup.toast.com/posts/117> |
|  | <https://docs.microsoft.com/de-de/dotnet/azure/> |
|  | <https://de.wikipedia.org/wiki/MySQL> |
|  | https://de.wikipedia.org/wiki/Datei:Visio\_2016.png |

[Abbildung 1 - ERD Der Datenbank 3](#_Toc530730423)

[Abbildung 2 - Prozesserstellung UseCase 4](#_Toc530730424)

[Abbildung 3 - Prozessnutzung UseCase 4](#_Toc530730425)

[Abbildung 4 - Nutzung eines Prozesses 5](#_Toc530730426)

[Abbildung 5 - Erstellung eines Prozesses 6](#_Toc530730427)

[Abbildung 6 - Beispielabbildung der Formularansicht 6](#_Toc530730428)

[Abbildung 7 - XML - Prozessdatei (Vorläufig) 7](#_Toc530730429)

[Abbildung 8 - Projektstrukturplan 11](#_Toc530730430)

[Abbildung 9 - GANTT - Diagramm 12](#_Toc530730431)