

Abschlussprüfung Sommer 2023 Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Depotmodellverwaltung

**Implementation einer benutzerfreundlichen Webanwendung mit
Eingabevalidierung zur Bearbeitung manuell verwalteter Depotmodelle**

Abgabedatum: 31.05.2023

**Prüfungsbewerber:** Zachariasz Jan Müller Ludwig-Thoma-Str. 19

93051 Regensburg

**Ausbildungsbetrieb:** SOBACO Betax Amberger Str. 136

93057 Regensburg



# Inhaltsverzeichnis

[Inhaltsverzeichnis I](#_bookmark0)

[Abbildungsverzeichnis III](#_bookmark1)

[Tabellenverzeichnis IV](#_bookmark2)

[Verzeichnis der Listings V](#_bookmark3)

[Abkürzungsverzeichnis VI](#_bookmark4)

1. [Einleitung 1](#_bookmark5)
	1. Das Unternehmen [1](#_bookmark6)
	2. [Projektbeschreibung 1](#_bookmark7)
	3. [Projektumfeld 1](#_bookmark8)
	4. [Projektziel 1](#_bookmark9)
	5. [Projektbegründung 1](#_bookmark10)
	6. [Projekt 1](#_bookmark10)
2. [Projektplanung 1](#_bookmark11)
	1. [Projektphasen 1](#_bookmark12)
	2. [Abweichungen vom Projektantrag 2](#_bookmark14)
	3. [Ressourcenplanung 2](#_bookmark15)
	4. [Entwicklungsprozess 2](#_bookmark16)
3. [Analysephase 2](#_bookmark17)
	1. [Ist-Analyse 2](#_bookmark18)
	2. [Wirtschaftlichkeitsanalyse 2](#_bookmark19)
		1. [Make or Buy-Entscheidung 2](#_bookmark20)
		2. [Projektkosten 2](#_bookmark21)
		3. [Amortisationsdauer 3](#_bookmark22)
	3. [Nutzwertanalyse 4](#_bookmark24)
	4. [Anwendungsfälle 4](#_bookmark25)
	5. [Qualitätsanforderungen 4](#_bookmark26)
	6. [Lastenheft/Fachkonzept 4](#_bookmark27)
4. [Entwurfsphase 4](#_bookmark28)
	1. [Zielplattform 4](#_bookmark29)
	2. [Architekturdesign 4](#_bookmark30)
	3. [Entwurf der Benutzeroberfläche 5](#_bookmark32)
	4. [Datenmodell 5](#_bookmark33)
	5. [Geschäftslogik 5](#_bookmark34)
	6. [Maßnahmen zur Qualitätssicherung 6](#_bookmark35)
	7. [Pflichtenheft/Datenverarbeitungskonzept 6](#_bookmark36)
5. [Implementierungsphase 6](#_bookmark37)
	1. [Implementierung der Datenstrukturen](#_bookmark38) 6
	2. [Implementierung der Benutzeroberfläche 6](#_bookmark39)
	3. [Implementierung der Geschäftslogik 6](#_bookmark40)
6. [Abnahmephase 7](#_bookmark41)
7. [Einführungsphase 7](#_bookmark42)
8. [Dokumentation 7](#_bookmark43)
9. [Fazit 7](#_bookmark44)
	1. [Soll-/Ist-Vergleich 7](#_bookmark45)
	2. [Lessons Learned 8](#_bookmark47)
	3. [Ausblick 8](#_bookmark48)

[Literaturverzeichnis 9](#_bookmark49)

[Eidesstattliche Erklärung 10](#_bookmark50)

[Anhang i](#_bookmark51)

[A1 Detaillierter Zeitplan i](#_bookmark52)

[A2 Lastenheft (Auszug) ii](#_bookmark54)

[A3 Use-Case-Diagramm iii](#_bookmark55)

[A4 Pflichtenheft (Auszug) iii](#_bookmark57)

[A5 Datenbankmodell v](#_bookmark58)

[A6 Ereignisgesteuerte Prozesskette vi](#_bookmark61)

[A7 Oberflächenentwürfe vii](#_bookmark63)

[A8 Screenshots der Anwendung viii](#_bookmark66)

[A9 Entwicklerdokumentation (Auszug) x](#_bookmark69)

[A10 Testfall und sein Aufruf auf der Konsole xi](#_bookmark71)

[A11 Klasse: ComparedNaturalModuleInformation xii](#_bookmark74)

[A12 Klassendiagramm xiv](#_bookmark76)

[A13 Benutzerdokumentation (Auszug) xv](#_bookmark78)

**Abbildungsverzeichnis**

# Abbildungsverzeichnis

[Abbildung 1: Use-Case-Diagramm iii](#_bookmark56)

[Abbildung 2: Entity-Relationship-Model v](#_bookmark59)

[Abbildung 3: Tabellenmodell vi](#_bookmark60)

[Abbildung 4: Prozess des Einlesens eines Moduls vi](#_bookmark62)

[Abbildung 5: Liste der Module mit Filtermöglichkeiten vii](#_bookmark64)

[Abbildung 6: Anzeige der Übersichtsseite einzelner Module viii](#_bookmark65)

[Abbildung 7: Anzeige und Filterung der Module nach Tags viii](#_bookmark67)

[Abbildung 8: Liste der Module mit Filtermöglichkeiten ix](#_bookmark68)

[Abbildung 9: Auszug aus der Entwicklerdokumentation mit *PHPDoc* x](#_bookmark70)

[Abbildung 10: Aufruf des Testfalls auf der Konsole xi](#_bookmark72)

[Abbildung 11: Klassendiagramm xiv](#_bookmark77)

[Abbildung 12: Auszug aus der Benutzerdokumentation xv](#_bookmark79)

**Tabellenverzeichnis**

# Tabellenverzeichnis

[Tabelle 1: Grobe Zeitplanung 2](#_bookmark13)

[Tabelle 2: Kostenaufstellung 3](#_bookmark23)

[Tabelle 3: Entscheidungsmatrix 5](#_bookmark31)

[Tabelle 4: Soll-/Ist-Vergleich 8](#_bookmark46)

[Tabelle 5: Detaillierte Zeitplanung ii](#_bookmark53)

#### Verzeichnis der Listings

# Verzeichnis der Listings

[Listing 1: Testklasse xii](#_bookmark73)

[Listing 2: Klasse ComparedNaturalModuleInformation xiv](#_bookmark75)

**Abkürzungsverzeichnis**

# Abkürzungsverzeichnis

API *Application Programming Interface*

CSS *Cascading Style Sheets*

EPK *Ereignisgesteuerte Prozesskette*

ERM *Entity Relationship Model*

GUI *Graphical User Interface*

HTML *Hypertext Markup Language*

MVC *Model View Controller*

PHP *PHP Hypertext Preprocessor*

SQL *Structured Query Language*

SVN *Subversion*

XML *Extensible Markup Language*

**Einleitung**

## Das Unternehmen

Die SOBACO Betax AG ist Teil der SOBACO Gruppe mit Sitz in Deutschland und der Schweiz. Die SOBACO Betax AG spezialisiert sich in Softwareentwicklung im Bereich der Schnittstellentechnologie und des Depotmanagements für Banken, Vermögensverwalter und Versicherungen. Das Kernprodukt der Firma ist das Portfoliomanagement-System „MyWebDepot“. Es dient zur Verarbeitung von Bestands- und Transaktionsdaten aus dem Wertpapiergeschäft.

-- insert more – Mitarbeiteranzahl,

## 1.2 Projektbeschreibung

Unser Hauptprodukt ”MyWebDepot” hat derzeit ca. 50 Vermittlerpools, welche diverse
Depotlagerstellen (Verwahrstellen von Wertpapieren) nutzen. Diese bieten eine Reihe von Depotmodellen an, welche sich durch verschiedene Dientleistungsangebote und die damit verbundenen Depotgebühren unterscheiden.

Die Spezifikationen dieser Modelle sind bei uns in einer Datenbank hinterlegt, und werden manuell verwaltet. Die dafür verwendete Software (Toad 12.12) ist jedoch eine generelle Datenbank-Software und verfügt über keinerlei Sicherheitsmechanismen für inkorrekte Eingaben und erfordert ein gewisses Maß an Fachkenntnis sowohl im Umgang mit Datenbanken als auch der Software an sich. Deshalb pflegt derzeit ausschließlich Herr Kümmel, der Projektleiter diese Modelle.

Herr Kümmel möchte diese Aufgabe in Zukunft an andere Mitarbeiter delegieren können wozu gewisse Sicherheitsmechanismen und vor allem ein intuitives User-Interface benötigt werden.

## 1.3 Projektumfeld

 Das Projektumfeld setzt sich zusammen aus einem PC-Arbeitsplatz mit dem Betriebssystem Linux-Ubuntu, der SQL-Datenbanksoftware Toad 12.12, der Entwicklungsumgebung Apache NetBeans und einer Windows10 Virtual Machine zum Testen unter Windows.

 Auftraggeber des Projekts ist der Projektleiter der Firma, Herr Klaus Kümmel.

##

## 1.4 Projektziel

Das Ziel dieses Projekts ist, eine intuitive, benutzerfreundliche Weboberfläche für die Verwaltung dieser Modelle zu erstellen, die es Mitarbeitern, welche nicht spezifisch im Umgang mit SQL-Datenbanken geschult sind, ermöglicht die Depotmodelle der Kunden zu pflegen.

## 1.5 Projektbegründung

Wie bereits in der einleitenden Projektbeschreibung dargelegt werden die Depotmodelle manuell verwaltet und erfordern ein gewisses Mindestmaß an Fachkenntnis, wodurch die Pflege derzeit nur vom Projektleiter bewerkstelligt werden kann. Diese Aufgabe zu delegieren würde die Projektleitung entlasten und eine Aufteilung der gesamten Modellverwaltung z.B. nach Kunde an mehrere Personen (z.B. an eine oder mehrere fachlich aber nicht technisch kundige Personen vom Support) ermöglichen.

Zudem würden Eingabevalidierungsmechanismen die Korrektheit der Daten zumindest in ihrer Form (Zeichenketten, Integer, Fließkommazahlen) unterstützen und zumindest

einen Teil von Fehlerquellen, wie z.B. Tippfehler eliminieren.

Da die eingegebenen Daten täglich für bestimmte Kalkulationen (wie z.B. Performanceberechnungen bestimmter Wertpapiere) sowie für die Buchung von Gebühren verwendet werden, sind Fehler hier kritisch.

Somit kann das Projekt, auch wenn es nicht direkt einen monäteren Gewinn bringen kann, schwere wirtschaftliche Schäden mindern oder sogar verhindern, die durch die Delegation der Pflege dieser Daten an Mitarbeiter, die weniger oder noch keine Erfahrung im Umgang damit haben entstehen könnten.

## 1.6 Projektschnittstellen

Da das Projekt nahtlos in die bereits vorhandene, interne Sammlung von Verwaltungstools (WfeAdmin) integriert werden soll, entstehen natürlich Schnittstellen mit dem bereits vorhandenem System.

Diese sind:

* ein einheitliches Design der Oberfläche wird durch einheitliche CSS-Dateien gesteuert
* der Aufbau der jeweiligen Seiten und Datenbankverbindungen (Oracle SQL) für jedes Tool wird zentral von einer Klasse gesteuert, die für dieses Tool geschriebenen Klassen müssen also diese erweitern (in PHP)
* die Datenübertragung zwischen Front- und Backend muss via AJAX (Javascript) erfolgen um die Einheitlichkeit der Tool-Sammlung zu gewährleisten

Die endgültigen User der Anwendung, außer dem Projektleiter selbst, stehen noch nicht fest und werden in Zukunft vom Projektleiter bestimmt.

Das Projekt wurde vom Projektleiter Klaus Kümmel in Auftrag gegeben, genehmigt und wird am Ende von ihm abgenommen.

## 1.7 Projektabgrenzung

Da die Daten bereits in einer Oracle-Datenbank vorhanden sind, sind weder das Anlegen entsprechender Tabellen, noch deren Population mit Daten Teil dieses Projekts. Die Schulung im Umgang darin sowie die Bestimmung der letztendlichen Benutzer wird zum entsprechenden Zeitpunkt vom Projektleiter durchgeführt.

**Projektplanung**

## 2.1 Projektphasen

Da für das Projekt laut Vorgaben der IHK 70 Stunden zur Verfügung stehen, musste der Prozess zunächst in Phasen eingeteilt werden. Ein grober Zeitplan kann der nachfolgenden Tabelle entnommen werden. Ein detaillierter Zeitplan befindet sich als Anlage A1 auf Seite i.

|  |  |
| --- | --- |
| **Projektphase** | **Zeit** |
| Planungs- und Analysephase | 8h |
| Entwurfsphase | 14h |
| Implementierungsphase | 34h |
| Qualitätsmanagement | 16h |
| Dokumentation | 8h |
| **Gesamt** | **80h** |

Tabelle 1: grober Zeitplan

## 2.2 Ressourcenplanung

## Projektphasen

* + - In welchem Zeitraum und unter welchen Rahmenbedingungen (z.B. Tagesarbeitszeit) findet das Projekt statt?
		- Verfeinerung der Zeitplanung, die bereits im Projektantrag vorgestellt wurde.

#### Beispiel

[Tabelle 1](#_bookmark13) zeigt ein Beispiel für eine grobe Zeitplanung.1

Eine detailliertere Zeitplanung ist in [Tabelle 5](#_bookmark53) in Anhang [A1](#_bookmark52) zu sehen.

1 Die Beispiele in dieser Dokumentation stammen aus (Grashorn, 2010).

|  |  |
| --- | --- |
| **Projektphase** | **Geplante Zeit** |
| Analyse | 9 h |
| Entwurf | 20 h |
| Implementierung | 30 h |
| Abnahme | 1 h |
| Einführung | 1 h |
| Dokumentation | 9 h |
| **Gesamt** | **70 h** |

**Tabelle 1: Grobe Zeitplanung**

## Abweichungen vom Projektantrag

* + - Sollte es Abweichungen zum Projektantrag geben (z.B. Zeitplanung, Inhalt des Pro- jekts, neue Anforderungen), müssen diese explizit aufgeführt und begründet werden.

## Ressourcenplanung

* + - Detaillierte Planung der benötigten Ressourcen (Hard-/Software, Räumlichkeiten usw.).
		- Ggfs. sind auch personelle Ressourcen einzuplanen (z.B. unterstützende Mitarbeiter).
		- Hinweis: Häufig werden hier Ressourcen vergessen, die als selbstverständlich ange- sehen werden (z.B. PC, Büro).

## Entwicklungsprozess

* + - Welcher Entwicklungsprozess wird bei der Bearbeitung des Projekts verfolgt (z.B. Wasserfall, agiler Prozess)?

## Ist-Analyse

* + - Wie ist die bisherige Situation (z.B. bestehende Programme, Wünsche der Mitarbei- ter)?
		- Was gilt es zu erstellen/verbessern?

## Wirtschaftlichkeitsanalyse

* + - Rentiert sich das Projekt für das Unternehmen?

### Make or Buy-Entscheidung

* + - * Gibt es vielleicht schon ein fertiges Produkt, das alle Anforderungen des Projekts ab- deckt?
			* Wenn ja, wieso wird das Projekt trotzdem umgesetzt?

### Projektkosten

* + - * Welche Kosten fallen bei der Umsetzung des Projekts im Detail an (z.B. Entwicklung, Einführung/Schulung, Wartung)?

#### Beispielrechnung (verkürzt)

Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten zusammen. Laut Tarifvertrag verdient ein Auszubildender im dritten Lehr- jahr pro Monat *1.000 €* (brutto).

h

8

Tag

· 220

Tage Jahr

= 1.760

h Jahr

1.000

€ Monat

· 13, 3

Monate Jahr

= 13.300

€ Jahr

13.300 €

𝐽𝑎ℎ𝑟 ≈ 7,56 €

1.760 h h Jahr

Es ergibt sich also ein Stundensatz von *7,56 EUR*. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen2 wird ein pauschaler Stundensatz von *15 EUR* angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundensatz von *25 EUR* ange- nommen. Eine Aufstellung der Kosten befindet sich in [Tabelle 2](#_bookmark23) und sie betragen insgesamt *2.739,20 EUR*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Vorgang** | **Zeit** | **Kosten / Stunde** | **Kosten** |
| Entwicklung | 70 h | *7,56 €* + *15 €* = *22,56 €* | *1.579,20 €* |
| Fachgespräch | 3 h | *25 €* + *15 €* = *40,00 €* | *120,00 €* |
| Abnahme | 1 h | *25 €* + *15 €* = *40,00 €* | *40,00 €* |
| Schulung | 25 h | *25 €* + *15 €* = *40,00 €* | *1.000,00 €* |
|  |  | ***Gesamt*** | ***2.739,20 €*** |

### Amortisationsdauer

**Tabelle 2: Kostenaufstellung**

* + - * Welche monetären Vorteile bietet das Projekt (z.B. Einsparung von Lizenzkosten, Ar- beitszeitersparnis, bessere Usability, Korrektheit)?
			* Wann hat sich das Projekt amortisiert?

#### Beispielrechnung (verkürzt)

Bei einer Zeiteinsparung von 10 Minuten am Tag für jeden der 25 Anwender und 220 Arbeits- tagen im Jahr ergibt sich eine gesamte Zeiteinsparung von:

25 · 220

Tage Jahr

· 10

min Tag

= 55.000

min Jahr

≈ 917

h Jahr

Dadurch ergibt sich eine jährliche Einsparung von:

€

917h · (25 + 15)

h

= 36.680 €

Die Amortisationszeit beträgt also:

2 Räumlichkeiten, Arbeitsplatzrechner etc.

2.739,20 €

 € ≈ 0, 07 𝐽𝑎ℎ𝑟𝑒 ≈ 4 𝑊𝑜𝑐ℎ𝑒𝑛

36.680 𝐽𝑎ℎ𝑟

## Nutzwertanalyse

* + - Darstellung des nicht-monetären Nutzens (z.B. Vorher-/Nachher-Vergleich anhand ei- nes Wirtschaftlichkeitskoeffizienten).

#### Beispiel

Ein Beispiel für eine Entscheidungsmatrix findet sich in Kapitel [4.2](#_bookmark30) ([Architekturdesign](#_bookmark30)).

## Anwendungsfälle

* + - Welche Anwendungsfälle soll das Projekt abdecken?
		- Einer oder mehrere interessante (!) Anwendungsfälle könnten exemplarisch durch ein Aktivitätsdiagramm oder eine EPK detailliert beschrieben werden.

#### Beispiel

Ein Beispiel für ein Use-Case-Diagramm findet sich im Anhang [A3](#_bookmark55).

## Qualitätsanforderungen

* + - Welche Qualitätsanforderungen werden an die Anwendung gestellt, z.B. hinsichtlich Performance, Usability, Effizienz etc. (siehe (ISO/IEC 9126-1, 2001))?

## Lastenheft/Fachkonzept

* + - Auszüge aus dem Lastenheft/Fachkonzept, wenn es im Rahmen des Projekts erstellt wurde.
		- Mögliche Inhalte: Funktionen des Programms (Muss/Soll/Wunsch), User Stories, Be- nutzerrollen

#### Beispiel

Ein Beispiel für ein Lastenheft findet sich im Anhang [A2](#_bookmark54).

## Zielplattform

* + - Beschreibung der Kriterien zur Auswahl der Zielplattform (u.a. Programmiersprache, Datenbank, Client/Server, Hardware).

## Architekturdesign

* + - Beschreibung und Begründung der gewählten Anwendungsarchitektur (z.B. MVC).
		- Ggfs. Bewertung und Auswahl von verwendeten Frameworks sowie ggfs. eine kurze Einführung in die Funktionsweise des verwendeten Frameworks.

#### Beispiel

Anhand der Entscheidungsmatrix in [Tabelle 3](#_bookmark31) wurde für die Implementierung der Anwendung das PHP-Framework *Symfony* ausgewählt.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Eigenschaft** | **Gewich-****tung** | ***Akelos*** | ***CakePHP*** | ***Symfony*** | **Eigenent- wicklung** |
| Dokumentation | 5 | 4 | 3 | 5 | 0 |
| Reengineering | 3 | 4 | 2 | 5 | 3 |
| Generierung | 3 | 5 | 5 | 5 | 2 |
| Testfälle | 2 | 3 | 2 | 3 | 3 |
| Standardaufgaben | 4 | 3 | 3 | 3 | 0 |
| Gesamt | 17 | 65 | 52 | 73 | 21 |
| **Nutzwert** |  | **3,82** | **3,06** | **4,29** | **1,24** |

**Tabelle 3: Entscheidungsmatrix**

## Entwurf der Benutzeroberfläche

* + - Entscheidung für die gewählte Benutzeroberfläche (z.B. GUI, Webinterface).
		- Beschreibung des visuellen Entwurfs der konkreten Oberfläche (z.B. Mockups, Menü- führung).
		- Ggfs. Erläuterung von angewendeten Richtlinien zur Usability und Verweis auf Corpo- rate Design.

#### Beispiel

Beispielentwürfe finden sich im Anhang [A7](#_bookmark63).

## Datenmodell

* + - Entwurf/Beschreibung der Datenstrukturen (z.B. ERM und/oder Tabellenmodell, XML- Schemas) mit kurzer Beschreibung der wichtigsten (!) verwendeten Entitäten.

#### Beispiel

In Anhang [A5](#_bookmark58) wird ein ERM dargestellt, welches lediglich Entitäten, Relationen und die dazu- gehörigen Kardinalitäten enthält.

## Geschäftslogik

* + - Modellierung und Beschreibung der wichtigsten (!) Bereiche der Geschäftslogik (z.B. mit Komponenten-, Klassen-, Sequenz-, Datenflussdiagramm, Programmablaufplan, Struktogramm, EPK).
		- Wie wird die erstellte Anwendung in den Arbeitsfluss des Unternehmens integriert?

#### Implementierungsphase

**Beispiel**

Ein Klassendiagramm, welches die Klassen der Anwendung und deren Beziehungen unterei- nander darstellt, kann im Anhang [A12](#_bookmark76) eingesehen werden.

Die EPK in Anhang [A6](#_bookmark61) zeigt den grundsätzlichen Ablauf beim Einlesen eines Moduls.

## Maßnahmen zur Qualitätssicherung

* + - Welche Maßnahmen werden ergriffen, um die Qualität des Projektergebnisses (siehe Kapitel [3.5](#_bookmark26)) zu sichern (z.B. automatische Tests, Anwendertests)?
		- Ggfs. Definition von Testfällen und deren Durchführung (durch Programme/Benutzer).

## Implementierung der Datenstrukturen

* + - Beschreibung der angelegten Datenbank (z.B. Generierung von SQL aus Modellie- rungswerkzeug oder händisches Anlegen), XML-Schemas usw.

## Implementierung der Benutzeroberfläche

* + - Beschreibung der Implementierung der Benutzeroberfläche, falls dies separat zur Im- plementierung der Geschäftslogik erfolgt (z.B. bei HTML-Oberflächen und Styles- heets).
		- Ggfs. Beschreibung des Corporate Designs und dessen Umsetzung in der Anwen- dung.
		- Screenshots der Anwendung

#### Beispiel

Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang [A8](#_bookmark66).

## Implementierung der Geschäftslogik

* + - Beschreibung des Vorgehens bei der Umsetzung/Programmierung der entworfenen Anwendung.
		- Ggfs. interessante Funktionen/Algorithmen im Detail vorstellen, verwendete Entwurfs- muster zeigen.
		- Quelltextbeispiele zeigen.
		- Hinweis: Es wird nicht ein lauffähiges Programm bewertet, sondern die Projektdurch- führung. Dennoch würde ich immer Quelltextausschnitte zeigen, da sonst Zweifel an der tatsächlichen Leistung des Prüflings aufkommen können.

#### Abnahmephase

**Beispiel**

Die Klasse ComparedNaturalModuleInformation findet sich im Anhang [A11](#_bookmark74).

* + - Welche Tests (z.B. Unit-, Integrations-, Systemtests) wurden durchgeführt und welche Ergebnisse haben sie geliefert (z.B. Logs von Unit Tests, Testprotokolle der Anwen- der)?
		- Wurde die Anwendung offiziell abgenommen?

#### Beispiel

Ein Auszug eines Unit Tests befindet sich im Anhang [A10](#_bookmark71). Dort ist auch der Aufruf des Tests auf der Konsole des Webservers zu sehen.

* + - Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?
		- Wurden Ggfs. Altdaten migriert und wenn ja, wie?
		- Wurden Benutzerschulungen durchgeführt und wenn ja, Wie wurden sie vorbereitet?
		- Wie wurde die Anwendung für die Benutzer/Administratoren/Entwickler dokumentiert (z.B. Benutzerhandbuch, API-Dokumentation)?
		- Hinweis: Je nach Zielgruppe gelten bestimmte Anforderungen für die Dokumentation (z.B. keine IT-Fachbegriffe in einer Anwenderdokumentation verwenden, aber auf je- den Fall in einer Dokumentation für den IT-Bereich).

#### Beispiel

Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang [A13](#_bookmark78).

Die Entwicklerdokumentation wurde mittels *PHPDoc* automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang [A9](#_bookmark69).

## Soll-/Ist-Vergleich

* + - Wurde das Projektziel erreicht und wenn nein, warum nicht?
		- Ist der Auftraggeber mit dem Projektergebnis zufrieden und wenn nein, warum nicht?
		- Wurde die Projektplanung (Zeit, Kosten, Personal, Sachmittel) eingehalten oder haben sich Abweichungen ergeben und wenn ja, warum?
		- Hinweis: Die Projektplanung muss nicht strikt eingehalten werden. Vielmehr sind Ab- weichungen sogar als normal anzusehen. Sie müssen nur vernünftig begründet wer- den (z.B. durch Änderungen an den Anforderungen, unter-/überschätzter Aufwand).

#### Beispiel (verkürzt)

Wie in [Tabelle 4](#_bookmark46) zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehal- ten werden.

|  |  |
| --- | --- |
| **Fazit** |  |
|  |
|  | **Phase** | **Geplant Tatsächlich** | **Differenz** |
|  | Analyse | 9 h 10 h | +1 h |
|  | Entwurf | 20 h 20 h |  |
|  | Implementierung | 30 h 27 h | -3 h |
|  | Abnahme | 1 h 1 h |  |
|  | Einführung | 1 h 1 h |  |
|  | Dokumentation | 9 h 11 h | +2 h |
|  | **Gesamt** | **70 h 70 h****Tabelle 4: Soll-/Ist-Vergleich** |  |

## Lessons Learned

* + - Was hat der Prüfling bei der Durchführung des Projekts gelernt (z.B. Zeitplanung, Vor- teile der eingesetzten Frameworks, Änderungen der Anforderungen)?

## Ausblick

* + - Wie wird sich das Projekt in Zukunft weiterentwickeln (z.B. geplante Erweiterungen)?

**Literaturverzeichnis**

# Literaturverzeichnis

Grashorn, D., 2010. *Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler,* Vechta: s.n.

ISO/IEC 9126-1, 2001. *Software-Engineering – Qualität von Software-Produkten – Teil 1: Qualitätsmodell.* s.l.:s.n.

**Eidesstattliche Erklärung**

# Eidesstattliche Erklärung

Ich, Der Autor, versichere hiermit, dass ich meine Dokumentation zur betrieblichen Projektar- beit mit dem Thema

*Der Kurztitel – Der Langtitel der Projektdokumentation*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Abgabeort, den 03.10.2016

DER AUTOR

## A1 Detaillierte Zeitplanung

|  |  |  |  |
| --- | --- | --- | --- |
| **1. Planungs- und Analysephase** |  |  | **8h** |
| 1.1 Durchführung der Ist-Analyse |  | 2h |  |
| 1.2 Analysieren der Anforderungen |  | 2h |  |
| 1.3 Erstellen eines Soll-Konzepts |  | 2h |  |
| 1.4 Wirtschaftlichkeitsanalyse |  | 2h |  |
| **2. Entwurfsphase** |  |  | **14h** |
| 2.1 Einlesen in die Strukturen des internen Verwaltungssystems  |  | 3h |  |
| 2.2 Einlesen in die Datenbankstrukturen und die Datenverarbeitung  |  | 3h |  |
| 2.3 Erstellen eines Backend-Konzepts zur Kommunikation mit der Datenbank  |  | 2h |  |
| 2.4 Entwurf der Validierungsmechanismen |  | 4h |  |
| 2.5 Entwurf der Benutzeroberfläche |  | 4h |  |
| **3. Implementierungsphase** |  |  | **34h** |
| 3.1 Anlegen eines neuen Web-Projekts |  | 1h |  |
| 3.1 Erstellen einer übersichtlichen Grundstruktur  |  | 1h |  |
| 3.3 Herstellen einer Datenbankverbindung |  | 1h |  |
| 3.4 Erstellen der SQL-Anweisungen |  | 4h |  |
| 3.5 Implementation ... |  | 27h |  |
| 3.5.1 … des Backend-Konzepts | 5h |  |  |
| 3.5.2 … der Oberfläche | 5h |  |  |
| 3.5.3 … der Eingabevalidierung | 6h |  |  |
| 3.5.4 … der Erstellungs-, Bearbeitungs- und Lösch-Funktionen  | 6h |  |  |
| 3.5.5 … der Sicherheitsmechanismen bei Bearbeitung und Löschung  | 2h |  |  |
| 3.5.6 … der Kopierfunktion | 3h |  |  |
| **4. Qualitätsmanagement** |  |  | **16h** |
| 4.1 Sammlung aller Testfälle |  | 2h |  |
| 4.2 Durchführung der gesammelten Testfälle |  | 5h |  |
| 4.3 Fehlerbehebung |  | 8h |  |
| 4.4 Soll-Ist-Vergleich |  | 1h |  |
| **5. Dokumentation** |  |  | **8h** |
| 5.1 Erstellen der Benutzerdokumentation |  | 4h |  |
| 5.2 Erstellen der Entwicklerdokumentation |  | 4h |  |
| **Gesamt** |  |  | **80h** |

|  |  |  |
| --- | --- | --- |
| **Abnahmetest der Fachabteilung** |  | **1 h** |
| 1. Abnahmetest der Fachabteilung | 1 h |  |
| **Einführungsphase** |  | **1 h** |
| 1. Einführung/Benutzerschulung | 1 h |  |
| **Erstellen der Dokumentation** |  | **9 h** |
| 1. Erstellen der Benutzerdokumentation | 2 h |  |

|  |  |  |
| --- | --- | --- |
| 2. Erstellen der Projektdokumentation |  | 6 h |
| 3. Programmdokumentation |  | 1 h |
| 3.1. Generierung durch *PHPdoc* | 1 h |  |
| **Gesamt** |  | **70 h** |

**Tabelle 5: Detaillierte Zeitplanung**

## A2 Lastenheft (Auszug)

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen:

#### Die Anwendung muss folgende Anforderungen erfüllen.

1. Verarbeitung der Moduldaten
	1. Die Anwendung muss die von Subversion und einem externen Programm bereitgestell- ten Informationen (z.B. Source-Benutzer, -Datum, Hash) verarbeiten.
	2. Auslesen der Beschreibung und der Stichwörter aus dem Sourcecode.
2. Darstellung der Daten
	1. Die Anwendung muss eine Liste aller Module erzeugen inkl. Source-Benutzer und - Datum, letztem Commit-Benutzer und -Datum für alle drei Umgebungen.
	2. Verknüpfen der Module mit externen Tools wie z.B. Wiki-Einträgen zu den Modulen oder dem Sourcecode in Subversion.
	3. Die Sourcen der Umgebungen müssen verglichen und eine schnelle Übersicht zur Ein- haltung des allgemeinen Entwicklungsprozesses gegeben werden.
	4. Dieser Vergleich muss auf die von einem bestimmten Benutzer bearbeiteten Module eingeschränkt werden können.
	5. Die Anwendung muss in dieser Liste auch Module anzeigen, die nach einer Bearbei- tung durch den gesuchten Benutzer durch jemand anderen bearbeitet wurden.
	6. Abweichungen sollen kenntlich gemacht werden.
	7. Anzeigen einer Übersichtsseite für ein Modul mit allen relevanten Informationen zu die- sem.
3. Sonstige Anforderungen
	1. Die Anwendung muss ohne das Installieren einer zusätzlichen Software über einen Webbrowser im Intranet erreichbar sein.
	2. Die Daten der Anwendung müssen jede Nacht bzw. nach jedem SVN-Commit automa- tisch aktualisiert werden.
	3. Es muss ermittelt werden, ob Änderungen auf der Produktionsumgebung vorgenom- men wurden, die nicht von einer anderen Umgebung kopiert wurden. Diese Modulliste soll als Mahnung per E-Mail an alle Entwickler geschickt werden (Peer Pressure).
	4. Die Anwendung soll jederzeit erreichbar sein.
	5. Da sich die Entwickler auf die Anwendung verlassen, muss diese korrekte Daten liefern und darf keinen Interpretationsspielraum lassen.
	6. Die Anwendung muss so flexibel sein, dass sie bei Änderungen im Entwicklungspro- zess einfach angepasst werden kann.

## A3 Use-Case-Diagramm

****

**Abbildung 1: Use-Case-Diagramm**

## A4 Pflichtenheft (Auszug)

#### Zielbestimmung

1. Musskriterien
	1. Modul-Liste: Zeigt eine filterbare Liste der Module mit den dazugehörigen Kerninforma- tionen sowie Symbolen zur Einhaltung des Entwicklungsprozesses an
		* In der Liste wird der Name, die Bibliothek und Daten zum Source und Kompilat eines Moduls angezeigt.
		* Ebenfalls wird der Status des Moduls hinsichtlich Source und Kompilat angezeigt. Dazu gibt es unterschiedliche Status-Zeichen, welche symbolisieren in wie weit der Entwicklungsprozess eingehalten wurde bzw. welche Schritte als nächstes getan werden müssen. So gibt es z. B. Zeichen für das Einhalten oder Verletzen des Pro- zesses oder den Hinweis auf den nächsten zu tätigenden Schritt.
		* Weiterhin werden die Benutzer und Zeitpunkte der aktuellen Version der Sourcen und Kompilate angezeigt. Dazu kann vorher ausgewählt werden, von welcher Umgebung diese Daten gelesen werden sollen.
		* Es kann eine Filterung nach allen angezeigten Daten vorgenommen werden. Die Da- ten zu den Sourcen sind historisiert. Durch die Filterung ist es möglich, auch Module zu finden, die in der Zwischenzeit schon von einem anderen Benutzer editiert wurden.
	2. Tag-Liste: Bietet die Möglichkeit die Module anhand von Tags zu filtern.
		* Es sollen die Tags angezeigt werden, nach denen bereits gefiltert wird und die, die noch der Filterung hinzugefügt werden könnten, ohne dass die Ergebnisliste leer wird.
		* Zusätzlich sollen die Module angezeigt werden, die den Filterkriterien entsprechen. Sollten die Filterkriterien leer sein, werden nur die Module angezeigt, welche mit ei- nem Tag versehen sind.
	3. Import der Moduldaten aus einer bereitgestellten CSV-Datei
		* Es wird täglich eine Datei mit den Daten der aktuellen Module erstellt. Diese Datei wird (durch einen Cronjob) automatisch nachts importiert.
		* Dabei wird für jedes importierte Modul ein Zeitstempel aktualisiert, damit festgestellt werden kann, wenn ein Modul gelöscht wurde.
		* Die Datei enthält die Namen der Umgebung, der Bibliothek und des Moduls, den Pro- grammtyp, den Benutzer und Zeitpunkt des Sourcecodes sowie des Kompilats und den Hash des Sourcecodes.
		* Sollte sich ein Modul verändert haben, werden die entsprechenden Daten in der Da- tenbank aktualisiert. Die Veränderungen am Source werden dabei aber nicht ersetzt, sondern historisiert.
	4. Import der Informationen aus Subversion (SVN). Durch einen „post-commit-hook“ wird nach jedem Einchecken eines Moduls ein PHP-Script auf der Konsole aufgerufen, wel- ches die Informationen, die vom SVN-Kommandozeilentool geliefert werden, an NatInfo übergibt.
	5. Parsen der Sourcen
		* Die Sourcen der Entwicklungsumgebung werden nach Tags, Links zu Artikeln im Wiki und Programmbeschreibungen durchsucht.
		* Diese Daten werden dann entsprechend angelegt, aktualisiert oder nicht mehr ge- setzte Tags/Wikiartikel entfernt.
	6. Sonstiges
		* Das Programm läuft als Webanwendung im Intranet.
		* Die Anwendung soll möglichst leicht erweiterbar sein und auch von anderen Entwick- lungsprozessen ausgehen können.
		* Eine Konfiguration soll möglichst in zentralen Konfigurationsdateien erfolgen.

#### Produkteinsatz

1. Anwendungsbereiche
	1. Die Webanwendung dient als Anlaufstelle für die Entwicklung. Dort sind alle Informati- onen für die Module an einer Stelle gesammelt. Vorher getrennte Anwendungen werden ersetzt bzw. verlinkt.
2. Zielgruppen
	1. NatInfo wird lediglich von den Natural-Entwicklern in der EDV-Abteilung genutzt.
3. Betriebsbedingungen
	1. Die nötigen Betriebsbedingungen, also der Webserver, die Datenbank, die Versions- verwaltung, das Wiki und der nächtliche Export sind bereits vorhanden und konfiguriert. Durch einen täglichen Cronjob werden entsprechende Daten aktualisiert, die Weban- wendung ist jederzeit aus dem Intranet heraus erreichbar.

## A5 Datenbankmodell

**Abbildung 2: Entity-Relationship-Model**



**Abbildung 3: Tabellenmodell**

## A6 Ereignisgesteuerte Prozesskette

****

**Abbildung 4: Prozess des Einlesens eines Moduls**

## A7 Oberflächenentwürfe

****

**Abbildung 5: Liste der Module mit Filtermöglichkeiten**



**Abbildung 6: Anzeige der Übersichtsseite einzelner Module**

## A8 Screenshots der Anwendung

**Abbildung 7: Anzeige und Filterung der Module nach Tags**



**Abbildung 8: Liste der Module mit Filtermöglichkeiten**

## A9 Entwicklerdokumentation (Auszug)

**Abbildung 9: Auszug aus der Entwicklerdokumentation mit *PHPDoc***

## A10 Testfall und sein Aufruf auf der Konsole

**Abbildung 10: Aufruf des Testfalls auf der Konsole**

$t->comment('Empty Information');

$emptyComparedInformation = new ComparedNaturalModuleInformation(array());

$t->is($emptyComparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY\_SIGN, 'Has no catalog sign');

$t->is($emptyComparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN\_CREATE, 'Source has to be created');

$t->comment('Perfect Module');

$criteria = new Criteria();

$criteria->add(NaturalmodulenamePeer::NAME, 'SMTAB');

$moduleName = NaturalmodulenamePeer::doSelectOne($criteria);

$t->is($moduleName->getName(), 'SMTAB', 'Right modulename selected');

$comparedInformation = $moduleName->loadNaturalModuleInformation();

$t->is($comparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN\_OK, 'Source sign shines global');

$t->is($comparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::SIGN\_OK, 'Catalog sign shines global');

$infos = $comparedInformation->getNaturalModuleInformations(); foreach($infos as $info) {

**Listing 1: Testklasse**

## A11 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht gezeigt.

private function allocateModulesToEnvironments(array

$naturalInformations) {

foreach ($naturalInformations as $naturalInformation) {

$env = $naturalInformation->getEnvironmentName(); if(in\_array($env, self::environments())) {

$this->naturalModuleInformations[array\_search($env, self::environments())] = $naturalInformation;

}

}

}

private function allocateEmptyModulesToMissingEnvironments() { if(array\_key\_exists(0, $this->naturalModuleInformations)) {

$this->naturalModuleInformations[0]-

>setSourceSign(self::SIGN\_OK);

}

for($i = 0;$i < count(self::environments());$i++) { if(!array\_key\_exists($i, $this-

>naturalModuleInformations)) {

$environments = self::environments();

$this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation($environments[$i]);

$this->naturalModuleInformations[$i]-

>setSourceSign(self::SIGN\_CREATE);

}

}

}

private function containsSourceSign($sign) {

foreach($this->naturalModuleInformations as $information) { if($information->getSourceSign() == $sign) {

return true;

}

}

return false;

**Listing 2: Klasse ComparedNaturalModuleInformation**

## A12 Klassendiagramm

****

**Abbildung 11: Klassendiagramm**

## A13 Benutzerdokumentation (Auszug)

**Abbildung 12: Auszug aus der Benutzerdokumentation**