

## Über diese Vorlage

Diese L<sup>A</sup>T<sub>E</sub>X-Vorlage wurde von Stefan Macke<sup>1</sup> als Grundlage für die Projektdokumentationen der Auszubildenden zum Fachinformatiker mit Fachrichtung Anwendungsentwicklung bei der ALTE OLDENBURGER Krankenversicherung entwickelt. Nichtsdestotrotz dürfte sie ebenso für die anderen IT-Berufe<sup>2</sup> geeignet sein, da diese anhand der gleichen Verordnung bewertet werden.

Diese Vorlage enthält bereits eine Vorstrukturierung der möglichen Inhalte einer tatsächlichen Projektdokumentation, die auf Basis der Erfahrungen im Rahmen der Prüfertätigkeit des Autors erstellt und unter Zuhilfenahme von ROHRER UND SEDLACEK [2011] abgerundet wurden.

Sämtliche verwendeten Abbildungen, Tabellen und Listings stammen von GRASHORN [2010].

Download-Link für diese Vorlage: <http://fiae.link/LaTeXVorlageFIAE>

Auch verfügbar auf GitHub: <https://github.com/StefanMacke/latex-vorlage-fiae>

## Lizenz



Dieses Werk steht unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.<sup>3</sup>



**Namensnennung** Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.<sup>4</sup>

**Weitergabe unter gleichen Bedingungen** Wenn Sie das lizenzierte Werk bzw. den lizenzierten Inhalt bearbeiten oder in anderer Weise erkennbar als Grundlage für eigenes Schaffen verwenden, dürfen Sie die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

---

<sup>1</sup>Blog des Autors: <http://fachinformatiker-anwendungsentwicklung.net>, Twitter: @StefanMacke

<sup>2</sup>z. B. IT-Kaufleute, Fachinformatiker mit Fachrichtung Systemintegration usw.

<sup>3</sup><http://creativecommons.org/licenses/by-sa/4.0/>

<sup>4</sup>Die Namensnennung im L<sup>A</sup>T<sub>E</sub>X-Quelltext mit Link auf <http://fiae.link/LaTeXVorlageFIAE> reicht hierfür aus.

## Inhalt der Projektdokumentation

Grundsätzlich definiert die [REGIERUNG DER BUNDESREPUBLIK DEUTSCHLAND](#) [1997, S. 1746]<sup>5</sup> das Ziel der Projektdokumentation wie folgt:

„Durch die Projektarbeit und deren Dokumentation soll der Prüfling belegen, daß er Arbeitsabläufe und Teilaufgaben zielorientiert unter Beachtung wirtschaftlicher, technischer, organisatorischer und zeitlicher Vorgaben selbständig planen und kundengerecht umsetzen sowie Dokumentationen kundengerecht anfertigen, zusammenstellen und modifizieren kann.“

Und das [BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG](#) [2000, S. 36] ergänzt:

„Die Ausführung der Projektarbeit wird mit praxisbezogenen Unterlagen dokumentiert. Der Prüfungsausschuss bewertet die Projektarbeit anhand der Dokumentation. Dabei wird nicht das Ergebnis – z.B. ein lauffähiges Programm – herangezogen, sondern der Arbeitsprozess. Die Dokumentation ist keine wissenschaftliche Abhandlung, sondern eine handlungsorientierte Darstellung des Projektablaufs mit praxisbezogenen, d.h. betriebüblichen Unterlagen. Sie soll einen Umfang von maximal 10 bis 15 DIN A 4-Seiten nicht überschreiten. Soweit erforderlich können in einem Anhang z. B. den Zusammenhang erläuternde Darstellungen beigelegt werden.“

Außerdem werden dort die grundlegenden Inhalte der Projektdokumentation aufgelistet:

- Name und Ausbildungsberuf des Prüfungsteilnehmers
- Angabe des Ausbildungsbetriebes
- Thema der Projektarbeit
- Falls erforderlich, Beschreibung/Konkretisierung des Auftrages
- Umfassende Beschreibung der Prozessschritte und der erzielten Ergebnisse
- Gegebenenfalls Veränderungen zum Projektantrag mit Begründung
- Wenn für das Projekt erforderlich, ein Anhang mit praxisbezogenen Unterlagen und Dokumenten. Dieser Anhang sollte nicht aufgebläht werden. Die angehängten Dokumente und Unterlagen sind auf das absolute Minimum zu beschränken.

In den folgenden Kapiteln werden diese geforderten Inhalte und sinnvolle Ergänzungen nun meist stichwortartig und ggfs. mit Beispielen beschrieben. Nicht alle Kapitel müssen in jeder Dokumentation vorhanden sein. Handelt es sich bspw. um ein in sich geschlossenes Projekt, kann das Kapitel ??: ?? entfallen; arbeitet die Anwendung nur mit XML-Dateien, kann und muss keine Datenbank beschrieben werden usw.

---

<sup>5</sup>Dieses Dokument sowie alle weiteren hier genannten können unter <http://fiae.link/LaTeXVorlageFIAEQuellen> heruntergeladen werden.

## Formale Vorgaben

Die formalen Vorgaben zum Umfang und zur Gestaltung der Projektdokumentation können je nach IHK recht unterschiedlich sein. Normalerweise sollte die zuständige IHK einen Leitfaden bereitstellen, in dem alle Formalien nachgelesen werden können, wie z. B. bei der [IHK OLDENBURG \[2006\]](#).

Als Richtwert verwende ich 15 Seiten für den reinen Inhalt. Also in dieser Vorlage alle Seiten, die arabisch nummeriert sind (ohne das Literaturverzeichnis und die eidesstattliche Erklärung). Große Abbildungen, Quelltexte, Tabellen usw. gehören in den Anhang, der 25 Seiten nicht überschreiten sollte.

Typographische Konventionen, Seitenränder usw. können in der Datei `Seitenstil.tex` beliebig angepasst werden.

## Bewertungskriterien

Die Bewertungskriterien für die Benotung der Projektdokumentation sind recht einheitlich und können leicht in Erfahrung gebracht werden, z. B. bei der [IHK DARMSTADT \[2011\]](#). Grundsätzlich sollte die Projektdokumentation nach der Fertigstellung noch einmal im Hinblick auf diese Kriterien durchgeschaut werden.

## Prüfungsteil A

Prüfling (private Anschrift):	Ausbildungsbetrieb:
-------------------------------	---------------------

### Bestätigung über durchgeführte Projektarbeit

diese Bestätigung ist mit der Projektdokumentation einzureichen

Ausbildungsberuf (bitte unbedingt angeben):
---------------------------------------------

Projektbezeichnung:
---------------------

Projektbeginn: _____	Projektfertigstellung: _____	Zeitaufwand in Std.: _____
----------------------	------------------------------	----------------------------

### Bestätigung der Ausbildungsfirma:

Wir bestätigen, dass der/die Auszubildende das oben bezeichnete Projekt einschließlich der Dokumentation im Zeitraum

vom: \_\_\_\_\_ bis: \_\_\_\_\_ selbständig ausgeführt hat.

Projektverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

Ausbildungsverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

### Eidesstattliche Erklärung:

Ich versichere, dass ich das Projekt und die dazugehörige Dokumentation selbständig erstellt habe.

Ort und Datum: \_\_\_\_\_ Unterschrift des Prüflings: \_\_\_\_\_



Abschlussprüfung Winter 2023

Fachinformatikerin für Systemintegration  
Dokumentation zur betrieblichen Projektarbeit

## **Implementierung von MFA**

**Implementierung von Multi-Faktor-Authentifizierung (MFA) zur  
Erhöhung der Sicherheit bei der Zugriffskontrolle von verschiedenen  
Services in einer Cloud-Infrastruktur**

Abgabetermin: Leipzig, den 10.11.2023

**Prüfungsbewerber:**

Melissa Futtig  
Stephaniplatz 3  
04317 Leipzig

# **Deloitte.**

**Ausbildungsbetrieb:**

DELOITTE Wirtschaftsprüfungsgesellschaft GmbH

---

Dittrichring 22  
04109 Leipzig

## Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>IV</b>
<b>Listings</b>	<b>V</b>
<b>Abkürzungsverzeichnis</b>	<b>VI</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Projektumfeld . . . . .	1
1.2 Projektziel . . . . .	1
1.3 Projektbegründung . . . . .	1
1.4 Projektschnittstellen . . . . .	2
1.4.1 Technisch . . . . .	2
1.4.2 Organisatorisch . . . . .	2
1.4.3 Personell . . . . .	3
<b>2 Projektplanung</b>	<b>3</b>
2.1 Projektphasen . . . . .	3
2.2 Abweichungen vom Projektantrag . . . . .	4
2.3 Ressourcenplanung . . . . .	4
2.3.1 Sachmittelplanung . . . . .	4
2.3.2 Personalplanung . . . . .	4
2.4 Entwicklungsprozess . . . . .	5
<b>3 Analysephase</b>	<b>5</b>
3.1 Ist-Analyse . . . . .	5
3.2 Wirtschaftlichkeitsanalyse . . . . .	5
3.2.1 „Make or Buy“-Entscheidung . . . . .	5
3.2.2 Projektkosten . . . . .	5
3.2.3 Amortisationsdauer . . . . .	6
3.3 Nutzwertanalyse . . . . .	7
3.4 Anwendungsfälle . . . . .	7
3.5 Qualitätsanforderungen . . . . .	7
3.6 Lastenheft/Fachkonzept . . . . .	7
<b>4 Entwurfsphase</b>	<b>8</b>
4.1 Zielplattform . . . . .	8
4.2 Architekturdesign . . . . .	8
4.3 Entwurf der Benutzeroberfläche . . . . .	8
4.4 Datenmodell . . . . .	9

*Inhaltsverzeichnis*


---

4.5	Geschäftslogik . . . . .	9
4.6	Maßnahmen zur Qualitätssicherung . . . . .	9
4.7	Pflichtenheft/Datenverarbeitungskonzept . . . . .	10
<b>5</b>	<b>Implementierungsphase</b>	<b>10</b>
5.1	Implementierung der Datenstrukturen . . . . .	10
5.2	Implementierung der Benutzeroberfläche . . . . .	10
5.3	Implementierung der Geschäftslogik . . . . .	11
<b>6</b>	<b>Abnahmephase</b>	<b>11</b>
<b>7</b>	<b>Einführungsphase</b>	<b>11</b>
<b>8</b>	<b>Dokumentation</b>	<b>12</b>
<b>9</b>	<b>Fazit</b>	<b>12</b>
9.1	Soll-/Ist-Vergleich . . . . .	12
9.2	Lessons Learned . . . . .	12
9.3	Ausblick . . . . .	13
	<b>Literaturverzeichnis</b>	<b>14</b>
	<b>Eidesstattliche Erklärung</b>	<b>15</b>
<b>A</b>	<b>Anhang</b>	<b>ii</b>
A.1	Gantt . . . . .	ii
A.2	Detaillierte Zeitplanung . . . . .	ii
A.3	Lastenheft (Auszug) . . . . .	iii
A.4	Use Case-Diagramm . . . . .	iv
A.5	Pflichtenheft (Auszug) . . . . .	iv
A.6	Datenbankmodell . . . . .	vi
A.7	Oberflächenentwürfe . . . . .	vii
A.8	Screenshots der Anwendung . . . . .	ix
A.9	Entwicklerdokumentation . . . . .	xi
A.10	Testfall und sein Aufruf auf der Konsole . . . . .	xiii
A.11	Klasse: ComparedNaturalModuleInformation . . . . .	xiv
A.12	Klassendiagramm . . . . .	xvii
A.13	Benutzerdokumentation . . . . .	xviii



Abbildungsverzeichnis

1	Vereinfachtes ER-Modell . . . . .	9
2	Prozess des Einlesens eines Moduls . . . . .	10
3	Use Case-Diagramm . . . . .	iv
4	Datenbankmodell . . . . .	vi
5	Liste der Module mit Filtermöglichkeiten . . . . .	vii
6	Anzeige der Übersichtsseite einzelner Module . . . . .	viii
7	Anzeige und Filterung der Module nach Tags . . . . .	viii
8	Anzeige und Filterung der Module nach Tags . . . . .	ix
9	Liste der Module mit Filtermöglichkeiten . . . . .	x
10	Aufruf des Testfalls auf der Konsole . . . . .	xiv
11	Klassendiagramm . . . . .	xvii

**Tabellenverzeichnis**

1	Zeitplanung . . . . .	3
2	Personalplanung . . . . .	4
3	Kostenaufstellung . . . . .	6
4	Entscheidungsmatrix . . . . .	8
5	Soll-/Ist-Vergleich . . . . .	13

Listings

1	Testfall in PHP . . . . .	xiii
2	Klasse: ComparedNaturalModuleInformation . . . . .	xiv

## Abkürzungsverzeichnis

<b>API</b>	Application Programming Interface
<b>CSV</b>	Comma Separated Value
<b>EPK</b>	Ereignisgesteuerte Prozesskette
<b>ERM</b>	Entity-Relationship-Modell
<b>HTML</b>	Hypertext Markup Language
<b>MVC</b>	Model View Controller
<b>NatInfo</b>	Natural Information System
<b>Natural</b>	Programmiersprache der Software AG
<b>PHP</b>	Hypertext Preprocessor
<b>SQL</b>	Structured Query Language
<b>SVN</b>	Subversion
<b>UML</b>	Unified Modeling Language
<b>XML</b>	Extensible Markup Language

## 1 Einleitung

### 1.1 Projektumfeld

Die Deloitte Wirtschaftsprüfungsgesellschaft GmbH ist ein internationales Unternehmen für Wirtschaftsprüfung, Steuer-, Unternehmens-, Risiko- und Finanzberatung. Die Deloitte hat Niederlassungen in vielen Ländern, darunter auch Deutschland. Mit Vertreter:innen in über 150 Ländern weltweit und 415.000 Mitarbeiter:innen bietet das Unternehmen eine breite Palette von Dienstleistungen für Unternehmen und Organisationen in verschiedenen Branchen. Im B&TCL, auch dem Business & Technology Center Leipzig, erbringt die Deloitte GmbH mit ihren 100 Mitarbeiter:innen eine Vielfalt an Business Services, mit und ohne IT-Bezug und treibt Transformationsprojekte rund um die Themen Cyber Security, Digitalisierung, Prozessoptimierung oder Automatisierung voran.

In dem Projekt arbeiten interne Mitarbeiter:innen aktiv mit, um die Entwicklung des Projektes voranzutreiben. Dabei stellen diese die Zielgruppe dar.

### 1.2 Projektziel

Die Deloitte Wirtschaftsprüfungsgesellschaft GmbH verwendet die Cloud-Infrastruktur von OVH-Cloud, um einen Business Hosting Service bereitzustellen.

Ziel ist es, die Sicherheit des Zugriffs auf verschiedene Dienste innerhalb dieser Cloud-Infrastruktur zu verbessern, indem Multi-Faktor-Authentifizierung (MFA) eingeführt wird. Dadurch soll eine erhöhte Sicherheit für firmeninterne und kundenbezogene Daten gewährleistet werden, indem nur eine Anmeldung mit MFA möglich ist dadurch Risiken, wie Password-Leaks und Phishing vermieden werden können. Um das Ziel zu erreichen wird Authelia für die Implementierung von MFA auf verschiedene Services in der Cloud-Infrastruktur eingeführt, um die Sicherheit der Zugriffskontrollen zu verbessern. Dabei ist Authelia eine Identity- und Access-Management-Lösung (IAM), die den Zugriff auf verschiedene Dienste und Ressourcen in der Cloud-Infrastruktur verwalten und sichern soll.

### 1.3 Projektbegründung

Die Einführung der MFA erhöht die Sicherheit und stellt sicher, dass der Zugriff auf Dienste und Daten der Cloud-Infrastruktur nicht allein durch ein gestohlenes Passwort gefährdet ist. Benutzer müssen zusätzlich zur Eingabe des Passworts einen weiteren Authentifizierungsfaktor, wie zum Beispiel ein Einmalpasswort, bereitstellen, was die Sicherheit erheblich erhöht. Dabei werden nicht nur firmeninterne und kundenbezogene Daten geschützt, sondern auch unsere Kundenzufriedenheit und das Vertrauen erhöht. Dies hat hohe Priorität und verhindert unbefugten Zugriff auf sensible Informationen.

## 1 Einleitung

---

### 1.4 Projektschnittstellen

#### 1.4.1 Technisch

Die in Kapitel 1.2 besagte Cloud-Infrastruktur wird in vier bzw. mit Authelia in fünf Hauptbereiche unterteilt, sodass in jedem dieser Bereiche verschiedene Services bzw. Dienste zur Verfügung werden.

##### *Compliance and Security Stack*

Dieser Bereich umfasst den Einsatz von Docker-Containern für Dienste, wie die OPNsense Firewall, den Nginx Proxy Manager, Guacamole Server und Infection Monkey zur Sicherheitsüberprüfung. Zusammenfassend ist zu sagen, dass der Nginx Proxy Manager als Reverse Proxy fungiert und den HTTP-Verkehr umleitet und andere Ports für Streaming-Anforderungen bedient. Das Open-Source-Sicherheitstesttool Infection Monkey überprüft die Sicherheit der Infrastruktur.

##### *Monitoring*

Im Überwachungsbereich werden alle vorhandenen Dienste mittels Docker-Containern der Cloud-Infrastruktur auditiert und beinhaltet dieser die Open-Source-Software Grafana zur Visualisierung und Überwachung von Leistungsdaten, Uptime Kuma als Webserver für Statusseiten und Healthchecks, um die Verfügbarkeit der Dienste zu kontrollieren.

##### *DevOps*

Der Fokus dieses Bereiches liegt bei den Anwendungen in der Entwicklung und Bereitstellung dieser und enthält die kollaborationsplattform GitLab, um Projekte zu verwalten. Dabei werden diese Dienste mittels Docker-Containern hochgefahren. Nexus kommt als Verwaltungstool der Repositories für die Anwendungsabhängigkeiten zum Einsatz. Sonarqube ermöglicht die statische Analyse und Bewertung der Quelltextqualität. Zusätzlich wird SFTPgo verwendet, um sichere Authentifizierungsmethoden, wie zum Beispile öffentliche Schlüssel, SSH-Schlüssel und Passwörter zu verwalten.

##### *E-Mail-Kommunikation*

In diesem Bereich werden Docker-Container eingesetzt, der den Mail-Server Mailcow verwendet, um E-Mail zu senden und zu empfangen, SOGO als Groupware-Lösung und ermöglicht somit eine effiziente E-Mail-Kommunikation und Zusammenarbeit innerhalb und außerhalb der Organisation.

##### *Authelia*

Authelia, als Freeware, stellt eine wichtige Schnittstelle für die Benutzerauthentifizierung und -autorisierung bereit und kann von verschiedenen Anwendungen über Docker-Container genutzt werden, um Authentifizierungsmechanismen einzurichten.

#### 1.4.2 Organisatorisch

Nach der Implementierung Authelias in der Cloud-Umgebung erfolgen erste Tests und Validierungen in den erstellten Docker-Containern, um sicherzustellen, dass die MFA ordnungsgemäß funktioniert und den Sicherheitsanforderungen entspricht. Im Anschluss erfolgen Schulungen der Benutzer, in dem Fall des Entwicklerteams, zur genauen Verwendung von Authelia mit MFA. Nach den Test und Schulungen wird die Authelia-MFA-Implementierung in der Cloud-Umgebung bereitgestellt und kann in die kontinuierliche Überwachung und Wartung der Docker-Container sichergestellt werden.

## 2 Projektplanung

### 1.4.3 Personell

Das Projektteam besteht aus folgenden Personen:

- Projektauftraggeber/ Director: Herr Dr. Volker Stroetmann
- Projektleiter/ Manager/ Projektentwickler: Herr Edgar Johann Kapler
- Projektentwicklerin/ Auszubildende: Frau Melissa Futtig

Der Projektauftraggeber und -leiter sind die Verantwortlichen für die Projektleitung und -finanzierung. Sie genehmigen das Projekt und stellen die notwendigen Ressourcen bereit.

Die Projektentwickler: innen sind die allgemeinen Benutzer und das Entwicklerteam, die für die Umsetzung und den reibungslosen Betrieb der Cloud-Infrastruktur verantwortlich und benötigen sichere Zugriffsmöglichkeiten zu den bereitgestellten Anwendungen.

## 2 Projektplanung

### 2.1 Projektphasen

Das Projekt findet in zwei Wochen unter maximal vier Stunden Arbeitszeit pro Tag statt, sodass zur gleichen Zeit die anderen Aufgaben und das Brainstorming für dieses Projekt erledigt werden können. Die im Projektantrag festgelegten Projektphasen lassen sich chronologisch in die 5-stündige Planungsphase, die 16-stündige Implementierungsphase, die 6-stündige Testphase, die 4-stündige Phase, in der die Einführung und Übergabe erfolgt und die 7-stündige Dokumentation mit der 2-stündigen Pufferzeit im Anschluss. Dabei änderte sich die Zeit in der Implementierungsphase von 14 auf 16 Stunden, die Zeit in der Dokumentation von 6 auf 7 Stunden und die Pufferzeit, welche hinzugefügt wurde, um zeitliche Konflikte mit den anderen Phasen zu vermeiden.

**Beispiel** Tabelle 1 zeigt ein Beispiel für eine grobe Zeitplanung.

Projektphase	Geplante Zeit
Planungsphase	5 h
Implementierungsphase	16 h
Testphase	6 h
Einführung und Übergabe	4 h
Dokumentation	7 h
Pufferzeit	2 h
<b>Gesamt</b>	<b>40 h</b>

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang A.2: Detaillierte Zeitplanung auf Seite ii.

## 2.2 Abweichungen vom Projektantrag

Die im Projektantrag mit "in Auflage genehmigt" Inhalte, erfordern Änderungen in der Projektdokumentation.

Die ursprüngliche Zeitplanung von 35 Stunden änderte sich auf 40 Stunden.

Zu Beginn erfolgen die Änderungen in der Zeitplanung, im Schritt Planungsphase, in welcher der Punkt "Klärung der Projektziele" in den Start dieser Phase geschoben werden.

Des Weiteren erfolgt in der Zeitplanung, Dokumentation, in welchem geplant keine Entwicklerdokumentation stattfindet, sondern gewünscht eine Benutzerdokumentation.

Die technische Umsetzung von Sitecars sollte sich in der Implementierungsphase nach der Auswahl einer MFA-Lösung ereignen, was bedingt der Nutzwertanalyse auf der Seite 6 in Kapitel 3 Analysephase, Punkt 3.3 Nutzwertanalyse, sich auf Authelia änderte.

## 2.3 Ressourcenplanung

### 2.3.1 Sachmittelplanung

Um die Umsetzung des Projektes zu ermöglichen, wurden folgende Hard- und Software verwendet:

- Notebook - Lenovo ThinkPad T15 Gen 1 (für die Entwicklung)
- Betriebssystem - Microsoft Windows 10 Enterprise auf dem Lenovo-Notebook
- IDE - Visual Studio Code 1.83.1 (user setup)
- Docker-Containerisierung in einer Linux-Umgebung
- OVHCloud-Server - Ext-Net-Baremetal
- OVHCloud-Instanz - firewall\_instance\_dev (flavor name: b2-7)
- OVHCloud-Instanz - tal\_cloud\_infra (flavor name: r2-60)
- OVHCloud-Instanz - rev\_prox-dev (flavor name: b2-15)

### 2.3.2 Personalplanung

Tabelle 2 zeigt die Personalplanung des Projektes.

Name	Rolle/ Berufsbezeichnung	Zeitaufwand
Dr. Volker Stroetmann	Director	
Edgar Johann Kapler	Manager	10h
Melissa Futtig	Auszubildende	40h

Tabelle 2: Personalplanung



## 2.4 Entwicklungsprozess

Das Projekt unterteilt sich in einem überschaubaren, zeitlich und inhaltlich begrenzten Entwicklungsprozess mit einzelnene begrenzten Phasen, die nach- und voneinander aufbauen. So wird eine Sicherstellung der Schritt für Schritt-Beendigung der jeweiligen Phasen und Übersicht gewährleistet. Bedingt dessen, dass in diesem Projekt wenig Projektteilnehmer: innen zur Verfügung stehen, kann auf die agile Methodik verzichtet werden.

## 3 Analysephase

### 3.1 Ist-Analyse

Das Projekt, operiert durch die Deloitte Wirtschaftsprüfungsgesellschaft GmbH, verwendet die Cloud-Infrastruktur von OVHCloud, um einen Business Hosting Service bereitzustellen. Dabei enthält diese Konfiguration der Cloud-Umgebung eine Firewall, welcher eine öffentliche IP-Adresse zugewiesen bekommen hat. Zusätzlich sollte erwähnt werden, dass dieses Netzwerk privat ist und andere Services und Instanzen enthält, welche von der Firewall geschützt werden. Um einen Zugriff auf die Instanzen zu ermöglichen, wird den Command-Line-Interface-Usern die Möglichkeit geboten, über die Kontrollinstanz "control\_node" die Instanzen hoch- und runterzufahren und die grundlegenden Einstellungen, wie zum Beispiel Portzuweisungen, vorzunehmen. Die Graphical-User-Interface-User haben des Weiteren die Chance die Services über den Reverse Proxy Manager zu erreichen, indem der Zugriff klassisch mittels einer einfachen Nutzer- und Passwordeingabe, ohne weiterer Schutzebene erfolgt.

### 3.2 Wirtschaftlichkeitsanalyse

#### 3.2.1 „Make or Buy“-Entscheidung

- Gibt es vielleicht schon ein fertiges Produkt, dass alle Anforderungen des Projekts abdeckt?
- Wenn ja, wieso wird das Projekt trotzdem umgesetzt?

#### 3.2.2 Projektkosten

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

## 3 Analysephase

**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 Lehrjahr pro Monat 1400 € Brutto.

$$8 \text{ h/Tag} \cdot 220 \text{ Tage/Jahr} = 1760 \text{ h/Jahr} \quad (1)$$

$$1400 \text{ €/Monat} \cdot 12 \text{ Monate/Jahr} = 16800 \text{ €/Jahr} \quad (2)$$

$$\frac{16800 \text{ €/Jahr}}{1760 \text{ h/Jahr}} \approx 9,55 \text{ €/h} \quad (3)$$

Es ergibt sich also ein Stundenlohn von 7,56 €. Die Durchführungszeit des Projekts beträgt 40 Stunden. Für die Nutzung von Ressourcen<sup>6</sup> wird ein pauschaler Stundensatz von 15 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 25 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 3 und sie betragen insgesamt 2739,20 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Arbeitskosten	40 h	9,55 € + 15,00 € = 24,55 €	982,00 €
Unterstützungskosten	8 h	35,00 € + 15,00 € = 40,00 €	320,00 €
Abnahmetest	2 h	25,00 € + 15,00 € = 40,00 €	80 €
			<b>1.382 €</b>

Tabelle 3: Kostenaufstellung

## 3.2.3 Amortisationsdauer

- Welche monetären Vorteile bietet das Projekt (z. B. Einsparung von Lizenzkosten, Arbeitszeiterparnis, 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 Arbeitstagen im Jahr ergibt sich eine gesamte Zeiteinsparung von

$$25 \cdot 220 \text{ Tage/Jahr} \cdot 10 \text{ min/Tag} = 55000 \text{ min/Jahr} \approx 917 \text{ h/Jahr} \quad (4)$$

Dadurch ergibt sich eine jährliche Einsparung von

$$917 \text{ h} \cdot (25 + 15) \text{ €/h} = 36680 \text{ €} \quad (5)$$

<sup>6</sup>Räumlichkeiten, Arbeitsplatzrechner etc.

### 3 Analysephase

---

Die Amortisationszeit beträgt also  $\frac{2739,20 \text{ €}}{36680 \text{ €/Jahr}} \approx 0,07 \text{ Jahre} \approx 4 \text{ Wochen}$ .

### 3.3 Nutzwertanalyse

- Darstellung des nicht-monetären Nutzens (z. B. Vorher-/Nachher-Vergleich anhand eines Wirtschaftlichkeitskoeffizienten).

**Beispiel** Ein Beispiel für eine Entscheidungsmatrix findet sich in Kapitel 4.2: Architekturdesign.

### 3.4 Anwendungsfälle

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

**Beispiel** Ein Beispiel für ein Use Case-Diagramm findet sich im Anhang A.4: Use Case-Diagramm auf Seite iv.

### 3.5 Qualitätsanforderungen

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

### 3.6 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, Benutzerrollen

**Beispiel** Ein Beispiel für ein Lastenheft findet sich im Anhang A.3: Lastenheft (Auszug) auf Seite iii.

## 4 Entwurfsphase

### 4.1 Zielplattform

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

### 4.2 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 4 wurde für die Implementierung der Anwendung das PHP-Framework Symfony<sup>7</sup> ausgewählt.

Eigenschaft	Gewichtung	MFA, SSO	Authelia	Sitecar
Benutzerfreundlichkeit	20	17	18	14
Sicherheit	25	22	25	18
Implementierung	20	17	16	15
Dokumentation	15	14	14	13
Skalierbarkeit	20	19	17	16
<b>Gesamt:</b>	<b>100</b>	<b>89</b>	<b>90</b>	<b>76</b>
<b>Nutzwert:</b>		<b>18,42</b>	<b>18,55</b>	<b>15,45</b>

Tabelle 4: Entscheidungsmatrix

### 4.3 Entwurf der Benutzeroberfläche

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

**Beispiel** Beispielentwürfe finden sich im Anhang A.7: Oberflächenentwürfe auf Seite vii.

<sup>7</sup>Vgl. [SENSIO LABS \[2010\]](#).

## 4.4 Datenmodell

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

**Beispiel** In Abbildung 1 wird ein Entity-Relationship-Modell (ERM) dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

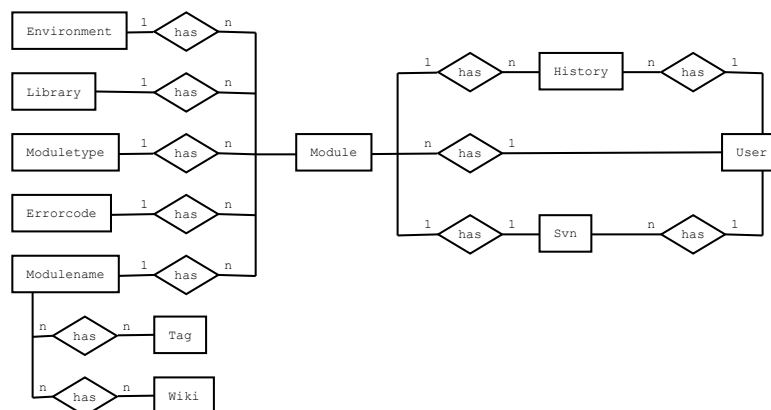


Abbildung 1: Vereinfachtes ER-Modell

## 4.5 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?

**Beispiel** Ein Klassendiagramm, welches die Klassen der Anwendung und deren Beziehungen untereinander darstellt kann im Anhang A.12: Klassendiagramm auf Seite xvii eingesehen werden.

Abbildung 2 zeigt den grundsätzlichen Programmablauf beim Einlesen eines Moduls als EPK.

## 4.6 Maßnahmen zur Qualitätssicherung

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

## 5 Implementierungsphase

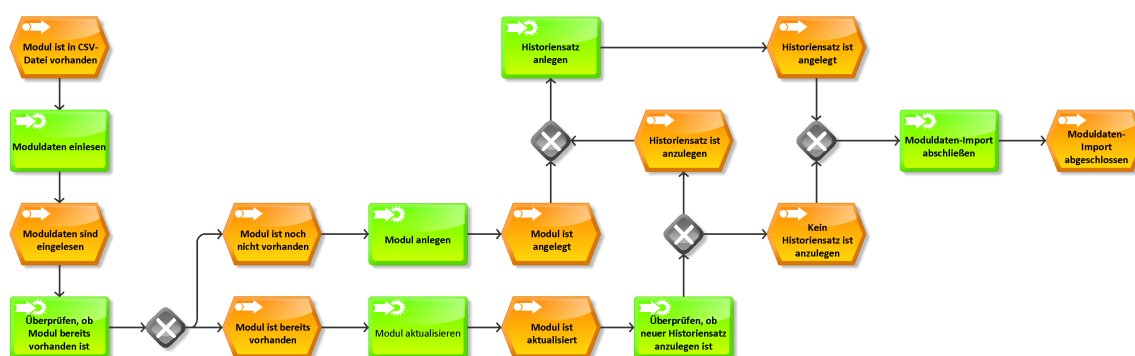


Abbildung 2: Prozess des Einlesens eines Moduls

### 4.7 Pflichtenheft/Datenverarbeitungskonzept

- Auszüge aus dem Pflichtenheft/Datenverarbeitungskonzept, wenn es im Rahmen des Projekts erstellt wurde.

**Beispiel** Ein Beispiel für das auf dem Lastenheft (siehe Kapitel 3.6: Lastenheft/Fachkonzept) aufbauende Pflichtenheft ist im Anhang A.5: Pflichtenheft (Auszug) auf Seite iv zu finden.

## 5 Implementierungsphase

### 5.1 Implementierung der Datenstrukturen

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

### 5.2 Implementierung der Benutzeroberfläche

- Beschreibung der Implementierung der Benutzeroberfläche, falls dies separat zur Implementierung der Geschäftslogik erfolgt (z. B. bei HTML-Oberflächen und Stylesheets).
- Ggfs. Beschreibung des Corporate Designs und dessen Umsetzung in der Anwendung.
- Screenshots der Anwendung

**Beispiel** Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang A.8: Screenshots der Anwendung auf Seite ix.

### 5.3 Implementierung der Geschäftslogik

- Beschreibung des Vorgehens bei der Umsetzung/Programmierung der entworfenen Anwendung.
- Ggfs. interessante Funktionen/Algorithmen im Detail vorstellen, verwendete Entwurfsmuster zeigen.
- Quelltextbeispiele zeigen.
- Hinweis: Wie in Kapitel 1: Einleitung zitiert, wird nicht ein lauffähiges Programm bewertet, sondern die Projektdurchführung. Dennoch würde ich immer Quelltextausschnitte zeigen, da sonst Zweifel an der tatsächlichen Leistung des Prüflings aufkommen können.

**Beispiel** Die Klasse `ComparedNaturalModuleInformation` findet sich im Anhang A.11: Klasse: `ComparedNaturalModuleInformation` auf Seite xiv.

## 6 Abnahmephase

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

**Beispiel** Ein Auszug eines Unit Tests befindet sich im Anhang A.10: Testfall und sein Aufruf auf der Konsole auf Seite xiii. Dort ist auch der Aufruf des Tests auf der Konsole des Webservers zu sehen.

## 7 Einführungsphase

- 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?

## 8 Dokumentation

- 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 jeden Fall in einer Dokumentation für den IT-Bereich).

**Beispiel** Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang A.13: Benutzerdokumentation auf Seite xviii. Die Entwicklerdokumentation wurde mittels PHPDoc<sup>8</sup> automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang A.9: Entwicklerdokumentation auf Seite xi.

## 9 Fazit

### 9.1 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 Abweichungen sogar als normal anzusehen. Sie müssen nur vernünftig begründet werden (z. B. durch Änderungen an den Anforderungen, unter-/überschätzter Aufwand).

**Beispiel (verkürzt)** Wie in Tabelle 5 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

### 9.2 Lessons Learned

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

---

<sup>8</sup>Vgl. [PHPDOC.ORG](http://PHPDOC.ORG) [2010]



## 9 Fazit

Phase	Geplant	Tatsächlich	Differenz
Entwurfsphase	19 h	19 h	
Analysephase	9 h	10 h	+1 h
Implementierungsphase	29 h	28 h	-1 h
Abnahmetest der Fachabteilung	1 h	1 h	
Einführungsphase	1 h	1 h	
Erstellen der Dokumentation	9 h	11 h	+2 h
Pufferzeit	2 h	0 h	-2 h
<b>Gesamt</b>	<b>70 h</b>	<b>70 h</b>	

Tabelle 5: Soll-/Ist-Vergleich

## 9.3 Ausblick

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

## Literaturverzeichnis

### Bundesministerium für Bildung und Forschung 2000

BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG: Umsetzungshilfen für die neue Prüfungsstruktur der IT-Berufe / Bundesministerium für Bildung und Forschung. Version: Juli 2000. <http://fiae.link/UmsetzungshilfenITBerufe>. Bonn, Juli 2000. – Abschlussbericht. – 476 S.

### Grashorn 2010

GRASHORN, Dirk: Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler / Alte Oldenburger Krankenversicherung AG. Vechta, April 2010. – Dokumentation zur Projektarbeit

### IHK Darmstadt 2011

IHK DARMSTADT: *Bewertungsmatrix für Fachinformatiker/innen Anwendungsentwicklung*. <http://fiae.link/BewertungsmatrixDokuDarmstadt>. Version: März 2011

### IHK Oldenburg 2006

IHK OLDENBURG: *Merkblatt zur Abschlussprüfung der IT-Berufe*. <http://fiae.link/MerkblattDokuOldenburg>. Version: Mai 2006

### ISO/IEC 9126-1 2001

ISO/IEC 9126-1: *Software-Engineering – Qualität von Software-Produkten – Teil 1: Qualitätsmodell*. Juni 2001

### phpdoc.org 2010

PHPDOC.ORG: *phpDocumentor-Website*. Version: 2010. <http://www.phpdoc.org/>, Abruf: 20.04.2010

### Regierung der Bundesrepublik Deutschland 1997

REGIERUNG DER BUNDESREPUBLIK DEUTSCHLAND: *Verordnung über die Berufsausbildung im Bereich der Informations- und Telekommunikationstechnik*. <http://fiae.link/VerordnungITBerufe>. Version: Juli 1997

### Rohrer und Sedlacek 2011

ROHRER, Anselm ; SEDLACEK, Ramona: *Cleverer Tipps für die Projektarbeit - IT-Berufe: Abschlussprüfung Teil A*. 5. Solingen : U-Form-Verlag, 2011 <http://fiae.link/ClevererTippsFuerDieProjektarbeit>. – ISBN 3882347538

### Sensio Labs 2010

SENSIO LABS: *Symfony - Open-Source PHP Web Framework*. Version: 2010. <http://www.symfony-project.org/>, Abruf: 20.04.2010

## Eidesstattliche Erklärung

Ich, Melissa Futtig, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

*Implementierung von MFA – Implementierung von Multi-Faktor-Authentifizierung (MFA)  
zur Erhöhung der Sicherheit bei der Zugriffskontrolle von verschiedenen Services in einer  
Cloud-Infrastruktur*

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.

Leipzig, den 10.11.2023

---

MELISSA FUTTIG



## A Anhang

### A.1 Gantt

### A.2 Detaillierte Zeitplanung

<b>Analysephase</b>	<b>9 h</b>
1. Analyse des Ist-Zustands	3 h
1.1. Fachgespräch mit der EDV-Abteilung	1 h
1.2. Prozessanalyse	2 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h
3. Erstellen eines „Use-Case“-Diagramms	2 h
4. Erstellen des Lastenhefts mit der EDV-Abteilung	3 h
<b>Entwurfsphase</b>	<b>19 h</b>
1. Prozessentwurf	2 h
2. Datenbankentwurf	3 h
2.1. ER-Modell erstellen	2 h
2.2. Konkretes Tabellenmodell erstellen	1 h
3. Erstellen von Datenverarbeitungskonzepten	4 h
3.1. Verarbeitung der CSV-Daten	1 h
3.2. Verarbeitung der SVN-Daten	1 h
3.3. Verarbeitung der Sourcen der Programme	2 h
4. Benutzeroberflächen entwerfen und abstimmen	2 h
5. Erstellen eines UML-Komponentendiagramms der Anwendung	4 h
6. Erstellen des Pflichtenhefts	4 h
<b>Implementierungsphase</b>	<b>29 h</b>
1. Anlegen der Datenbank	1 h
2. Umsetzung der HTML-Oberflächen und Stylesheets	4 h
3. Programmierung der PHP-Module für die Funktionen	23 h
3.1. Import der Modulinformationen aus CSV-Dateien	2 h
3.2. Parsen der Modulquelltexte	3 h
3.3. Import der SVN-Daten	2 h
3.4. Vergleichen zweier Umgebungen	4 h
3.5. Abrufen der von einem zu wählenden Benutzer geänderten Module	3 h
3.6. Erstellen einer Liste der Module unter unterschiedlichen Aspekten	5 h
3.7. Anzeigen einer Liste mit den Modulen und geparsen Metadaten	3 h
3.8. Erstellen einer Übersichtsseite für ein einzelnes Modul	1 h
4. Nächtlichen Batchjob einrichten	1 h
<b>Abnahmetest der Fachabteilung</b>	<b>1 h</b>
1. Abnahmetest der Fachabteilung	1 h
<b>Einführungsphase</b>	<b>1 h</b>
1. Einführung/Benutzerschulung	1 h
<b>Erstellen der Dokumentation</b>	<b>9 h</b>
1. Erstellen der Benutzerdokumentation	2 h
2. Erstellen der Projektdokumentation	6 h
3. Programmdokumentation	1 h
3.1. Generierung durch PHPdoc	1 h
<b>Pufferzeit</b>	<b>2 h</b>
1. Puffer	2 h
<b>Gesamt</b>	<b>70 h</b>

### A.3 Lastenheft (Auszug)

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

Die Anwendung muss folgende Anforderungen erfüllen:

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

## A.4 Use Case-Diagramm

Use Case-Diagramme und weitere UML-Diagramme kann man auch direkt mit  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/usecase-diagram.html>.

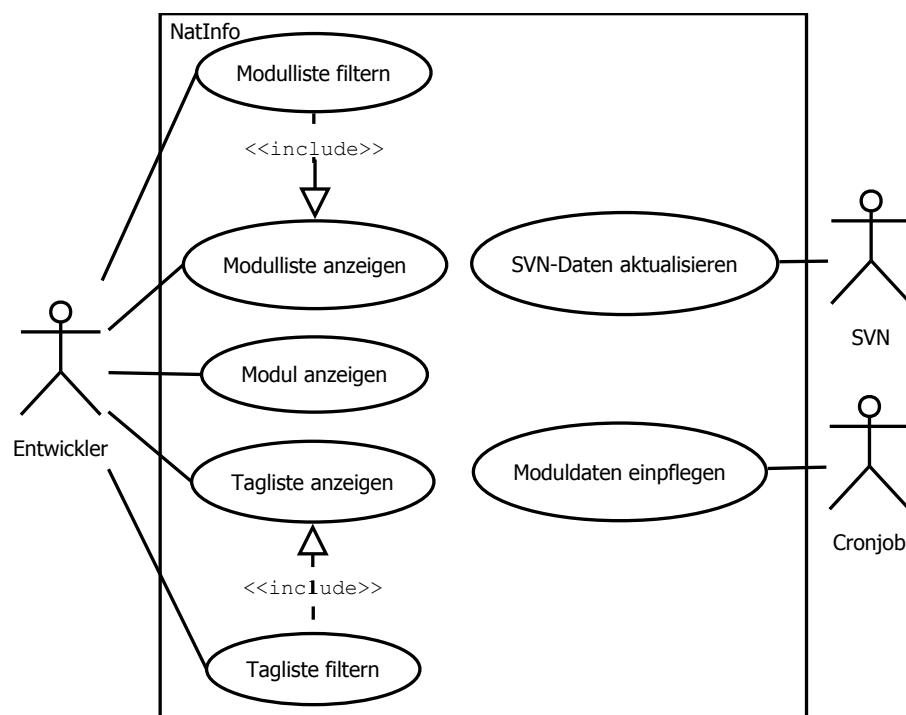


Abbildung 3: Use Case-Diagramm

## A.5 Pflichtenheft (Auszug)

### Zielbestimmung

#### 1. Musskriterien

1.1. Modul-Liste: Zeigt eine filterbare Liste der Module mit den dazugehörigen Kerninformationen 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 Prozesses 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 Daten 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.

#### 1.2. 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 einem Tag versehen sind.

#### 1.3. 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 Programmtyp, 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 Datenbank aktualisiert. Die Veränderungen am Source werden dabei aber nicht ersetzt, sondern historisiert.

#### 1.4. Import der Informationen aus Subversion (SVN). Durch einen „post-commit-hook“ wird nach jedem Einchecken eines Moduls ein PHP-Script auf der Konsole aufgerufen, welches die Informationen, die vom SVN-Kommandozeilentool geliefert werden, an NATINFO übergibt.

#### 1.5. 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 gesetzte Tags/Wikiartikel entfernt.

#### 1.6. Sonstiges

- Das Programm läuft als Webanwendung im Intranet.
- Die Anwendung soll möglichst leicht erweiterbar sein und auch von anderen Entwicklungsprozessen ausgehen können.
- Eine Konfiguration soll möglichst in zentralen Konfigurationsdateien erfolgen.

## Produkteinsatz

### 1. Anwendungsbereiche

Die Webanwendung dient als Anlaufstelle für die Entwicklung. Dort sind alle Informationen



## A Anhang

für die Module an einer Stelle gesammelt. Vorher getrennte Anwendungen werden ersetzt bzw. verlinkt.

### 2. Zielgruppen

NatInfo wird lediglich von den NATURAL-Entwicklern in der EDV-Abteilung genutzt.

### 3. Betriebsbedingungen

Die nötigen Betriebsbedingungen, also der Webserver, die Datenbank, die Versionsverwaltung, das Wiki und der nächtliche Export sind bereits vorhanden und konfiguriert. Durch einen täglichen Cronjob werden entsprechende Daten aktualisiert, die Webanwendung ist jederzeit aus dem Intranet heraus erreichbar.

## A.6 Datenbankmodell

ER-Modelle kann man auch direkt mit  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://www.texample.net/tikz/examples/entity-relationship-diagram/>.

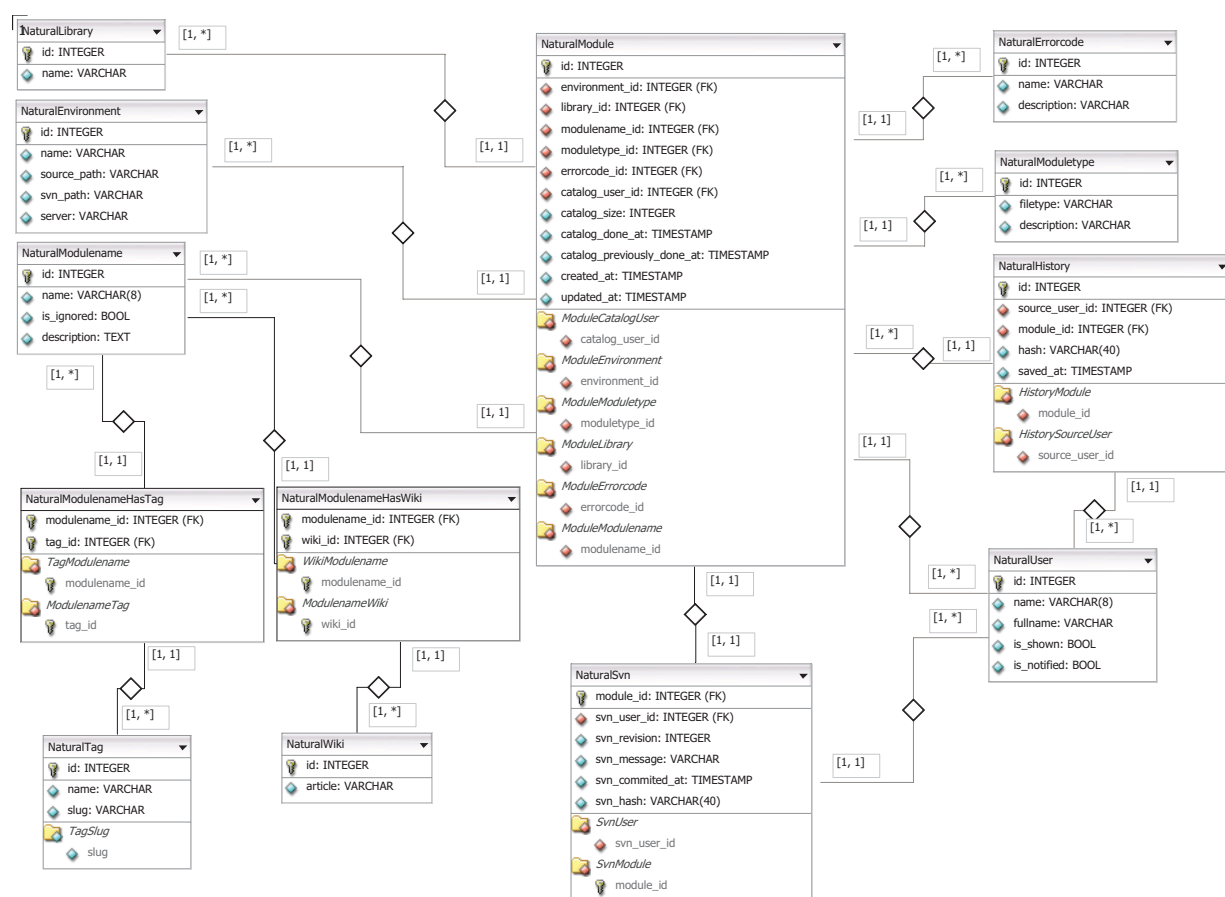


Abbildung 4: Datenbankmodell

A.7 Oberflächenentwürfe

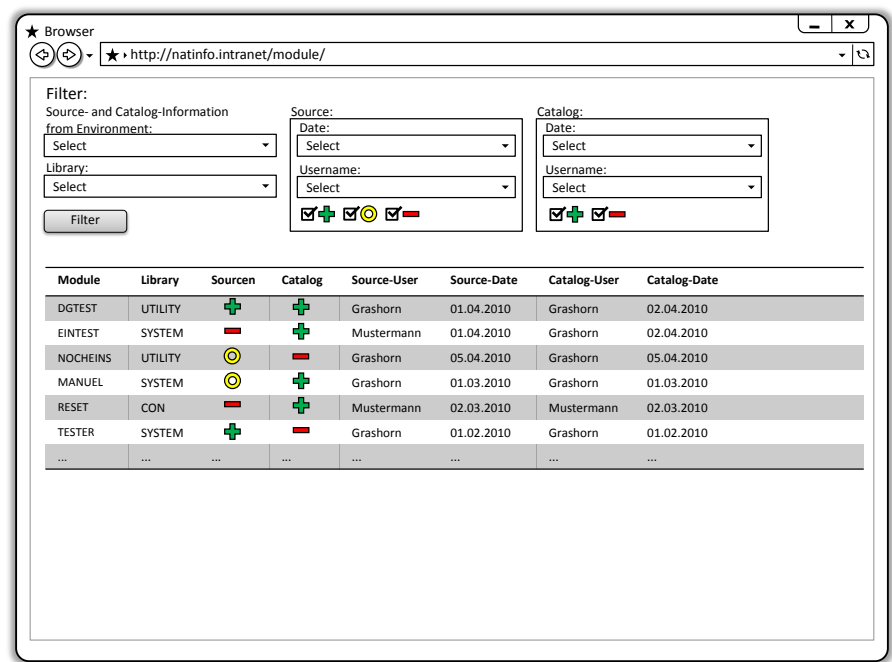


Abbildung 5: Liste der Module mit Filtermöglichkeiten

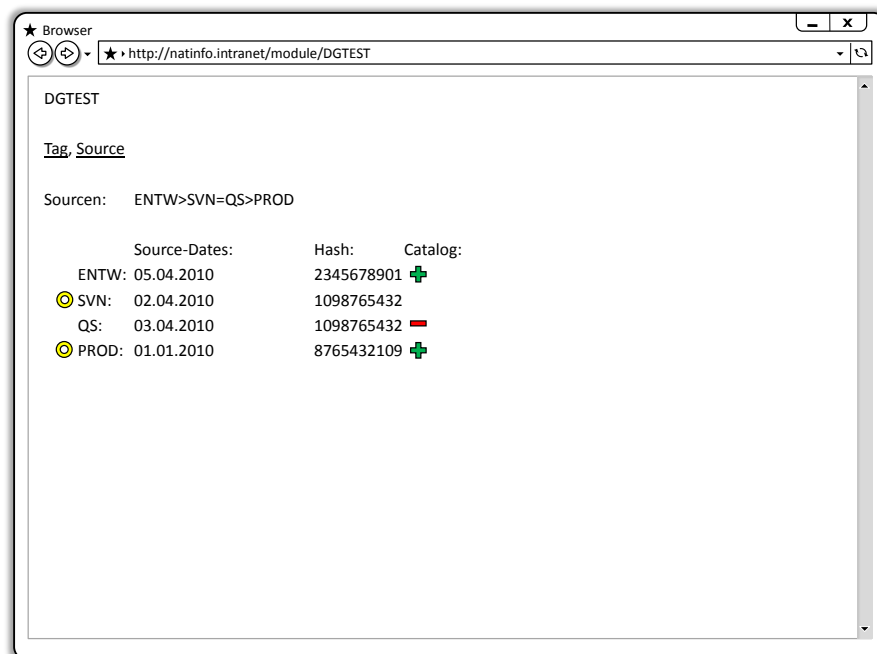


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

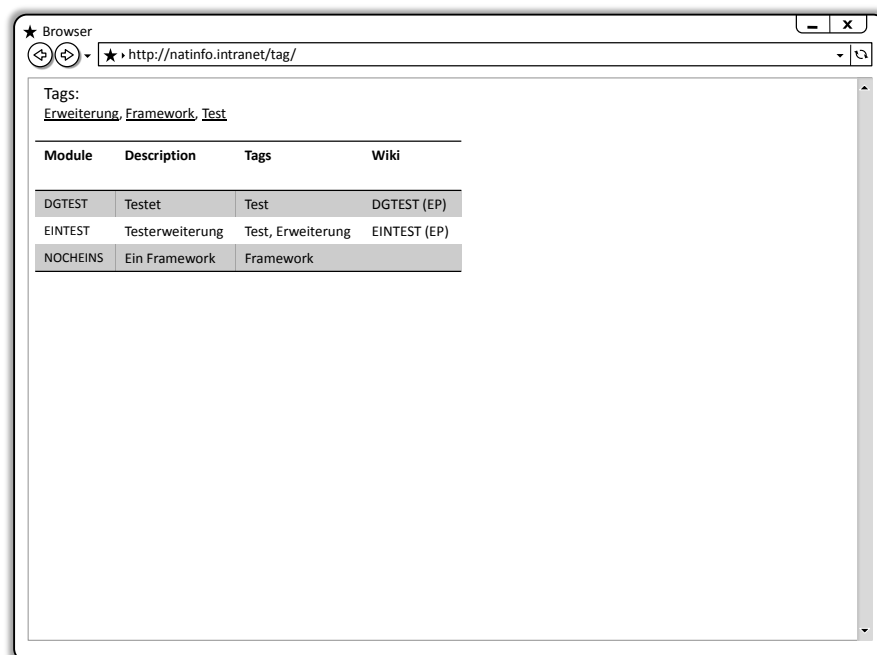


Abbildung 7: Anzeige und Filterung der Module nach Tags

A.8 Screenshots der Anwendung

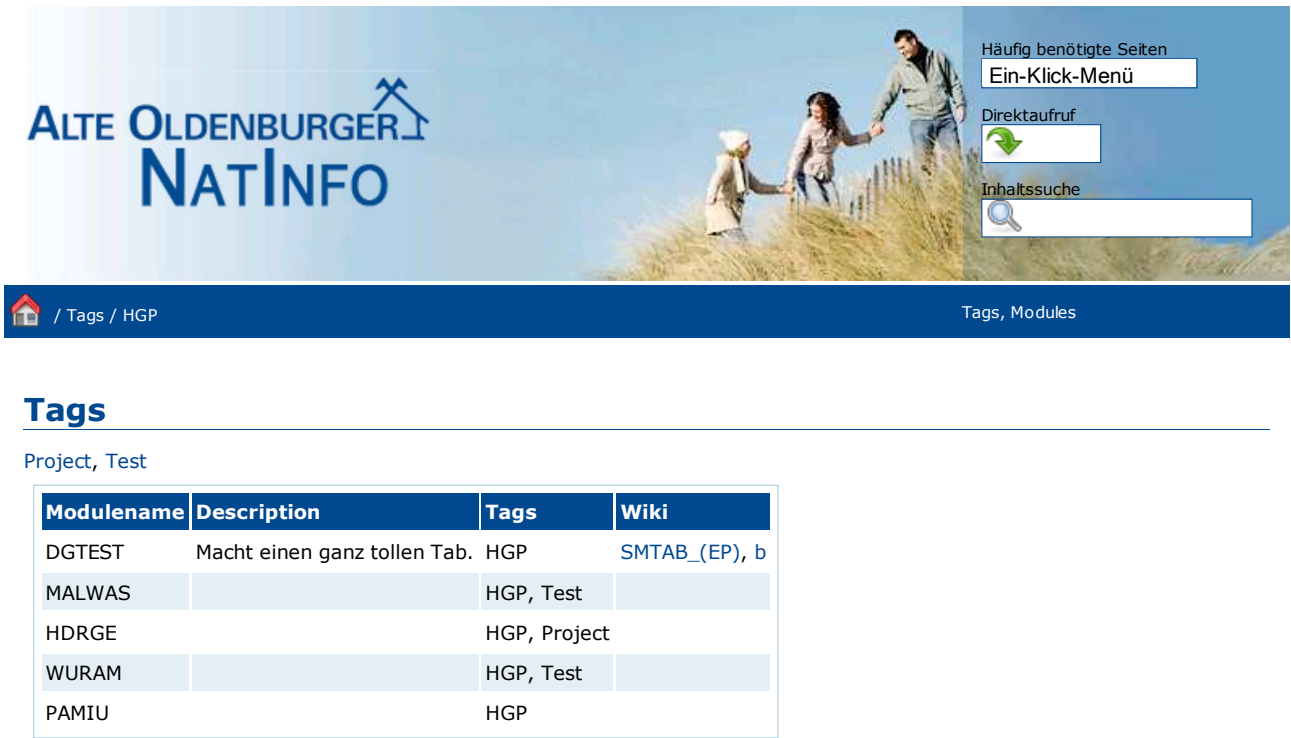


Abbildung 8: Anzeige und Filterung der Module nach Tags

ALTE OLDENBURGER  
NATINFO

Häufig benötigte Seiten

Ein-Klick-Menü

Direktaufruf

Inhaltssuche

/ Modules / ENTW

Tags, Modules

Environment

Library

Catalog user

Catalog date

Source user

Source date

Reset

Filter

Name	Library	Source	Catalog	Source-User	Source-Date	Catalog-User	Catalog-Date
SMTAB	UTILITY			MACKE	01.04.2010 13:00	MACKE	01.04.2010 13:00
DGTAB	CON			GRASHORN	01.04.2010 13:00	GRASHORN	01.04.2010 13:00
DGTEST	SUP			GRASHORN	05.04.2010 13:00	GRASHORN	05.04.2010 13:00
OHNETAG	CON			GRASHORN	05.04.2010 13:00	GRASHORN	01.04.2010 15:12
OHNEWIKI	CON			GRASHORN	05.04.2010 13:00	MACKE	01.04.2010 15:12

Abbildung 9: Liste der Module mit Filtermöglichkeiten

A.9 Entwicklerdokumentation

lib-model

[ class tree: lib-model ] [ index: lib-model ] [ all elements ]

Packages:

lib-model

Files:

Naturalmodulename.php

Classes:

Naturalmodulename

Class: Naturalmodulename

Source Location: /Naturalmodulename.php

Class Overview

BaseNaturalmodulename  
|  
--Naturalmodulename

Subclass for representing a row from the 'NaturalModulename' table.

Methods

- [\\_\\_construct](#)
- [getNaturalTags](#)
- [getNaturalWikis](#)
- [loadNaturalModuleInformation](#)
- [\\_\\_toString](#)

Class Details

[line 10]

Subclass for representing a row from the 'NaturalModulename' table.

Adds some business logic to the base.

[ Top ]

Class Methods

constructor **\_\_construct** [line 56]

Naturalmodulename \_\_construct ( )

Initializes internal state of Naturalmodulename object.

Tags:

**see:** parent::\_\_construct()

**access:** public

[ Top ]

method **getNaturalTags** [line 68]

array getNaturalTags ( )

Returns an Array of NaturalTags connected with this Modulename.

Melissa Futtig

xi

**Tags:**

**return:** Array of NaturalTags  
**access:** public

[\[ Top \]](#)

---

**method getNaturalWikis** [line 83]

```
array getNaturalWikis( )
```

Returns an Array of NaturalWikis connected with this Modulename.

**Tags:**

**return:** Array of NaturalWikis  
**access:** public

[\[ Top \]](#)

---

**method loadNaturalModuleInformation** [line 17]

```
ComparedNaturalModuleInformation  
loadNaturalModuleInformation( )
```

Gets the ComparedNaturalModuleInformation for this NaturalModulename.

**Tags:**

**access:** public

[\[ Top \]](#)

---

**method \_\_toString** [line 47]

```
string __toString( )
```

Returns the name of this NaturalModulename.

**Tags:**

**access:** public

[\[ Top \]](#)

---

Documentation generated on Thu, 22 Apr 2010 08:14:01 +0200 by [phpDocumentor 1.4.2](#)

## A.10 Testfall und sein Aufruf auf der Konsole

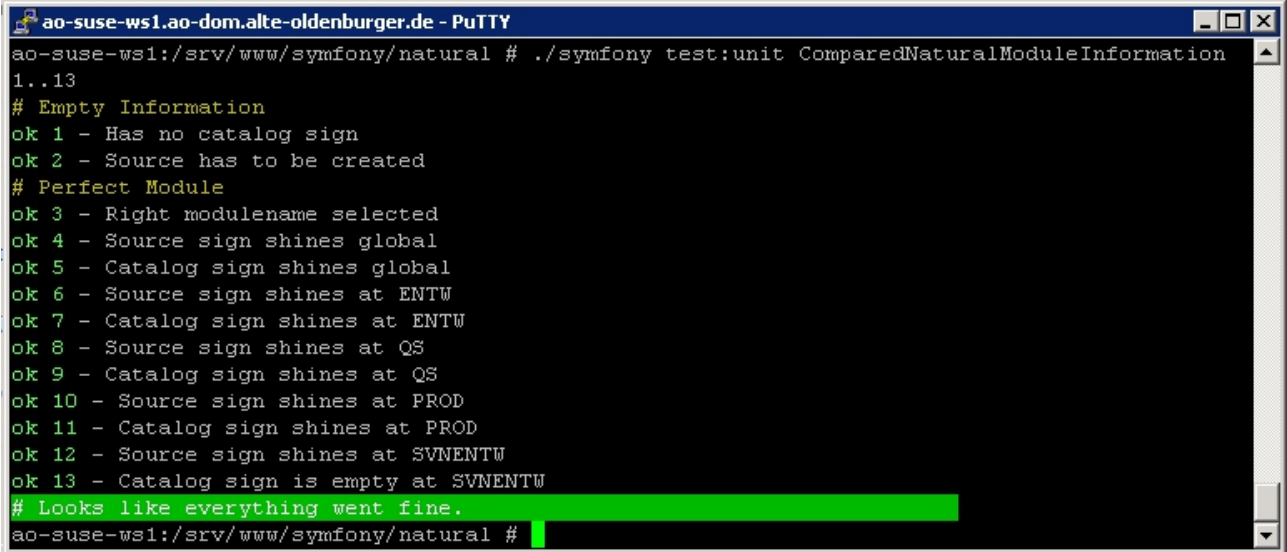
```

1 <?php
2 include(dirname(__FILE__).'/../bootstrap/Propel.php');
3
4 $t = new lime_test(13);
5
6 $t->comment('Empty Information');
7 $emptyComparedInformation = new ComparedNaturalModuleInformation(array());
8 $t->is($emptyComparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, '
    Has no catalog sign');
9 $t->is($emptyComparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_CREATE, '
    Source has to be created');
10
11 $t->comment('Perfect Module');
12 $criteria = new Criteria();
13 $criteria->add(NaturalmodulePeer::NAME, 'SMTAB');
14 $moduleName = NaturalmodulePeer::doSelectOne($criteria);
15 $t->is($moduleName->getName(), 'SMTAB', 'Right module name selected');
16 $comparedInformation = $moduleName->loadNaturalModuleInformation();
17 $t->is($comparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign
    shines global');
18 $t->is($comparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign
    shines global');
19 $infos = $comparedInformation->getNaturalModuleInformations();
20 foreach($infos as $info)
21 {
22     $env = $info->getEnvironmentName();
23     $t->is($info->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign shines at ' . $env);
24     if ($env != 'SVNENTW')
25     {
26         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign shines at ' .
            $info->getEnvironmentName());
27     }
28     else
29     {
30         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, 'Catalog sign is empty
            at ' . $info->getEnvironmentName());
31     }
32 }
33 ?>

```

Listing 1: Testfall in PHP





```

ao-suse-ws1.ao-dom.alte-oldenburger.de - PuTTY
ao-suse-ws1:/srv/www/symfony/natural # ./symfony test:unit ComparedNaturalModuleInformation
1..13
# Empty Information
ok 1 - Has no catalog sign
ok 2 - Source has to be created
# Perfect Module
ok 3 - Right modulename selected
ok 4 - Source sign shines global
ok 5 - Catalog sign shines global
ok 6 - Source sign shines at ENTW
ok 7 - Catalog sign shines at ENTW
ok 8 - Source sign shines at QS
ok 9 - Catalog sign shines at QS
ok 10 - Source sign shines at PROD
ok 11 - Catalog sign shines at PROD
ok 12 - Source sign shines at SVNENTW
ok 13 - Catalog sign is empty at SVNENTW
# Looks like everything went fine.
ao-suse-ws1:/srv/www/symfony/natural #

```

Abbildung 10: Aufruf des Testfalls auf der Konsole

## A.11 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

```

1 <?php
2 class ComparedNaturalModuleInformation
3 {
4     const EMPTY_SIGN = 0;
5     const SIGN_OK = 1;
6     const SIGN_NEXT_STEP = 2;
7     const SIGN_CREATE = 3;
8     const SIGN_CREATE_AND_NEXT_STEP = 4;
9     const SIGN_ERROR = 5;
10
11     private $naturalModuleInformations = array();
12
13     public static function environments()
14     {
15         return array("ENTW", "SVNENTW", "QS", "PROD");
16     }
17
18     public static function signOrder()
19     {
20         return array(self::SIGN_ERROR, self::SIGN_NEXT_STEP, self::SIGN_CREATE_AND_NEXT_STEP, self::SIGN_CREATE, self::SIGN_OK);
21     }
22
23     public function __construct(array $naturalInformations)
24     {
25         $this->allocateModulesToEnvironments($naturalInformations);

```

## A Anhang

```

26     $this->allocateEmptyModulesToMissingEnvironments();
27     $this->determineSourceSignsForAllEnvironments();
28 }
29
30 private function allocateModulesToEnvironments(array $naturalInformations)
31 {
32     foreach ($naturalInformations as $naturalInformation)
33     {
34         $env = $naturalInformation->getEnvironmentName();
35         if (in_array($env, self::environments()))
36         {
37             $this->naturalModuleInformations[array_search($env, self::environments())] = $naturalInformation;
38         }
39     }
40 }
41
42 private function allocateEmptyModulesToMissingEnvironments()
43 {
44     if (array_key_exists(0, $this->naturalModuleInformations))
45     {
46         $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
47     }
48
49     for ($i = 0; $i < count(self::environments()); $i++)
50     {
51         if (!array_key_exists($i, $this->naturalModuleInformations))
52         {
53             $environments = self::environments();
54             $this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation($environments[$i]);
55             $this->naturalModuleInformations[$i]->setSourceSign(self::SIGN_CREATE);
56         }
57     }
58 }
59
60 public function determineSourceSignsForAllEnvironments()
61 {
62     for ($i = 1; $i < count(self::environments()); $i++)
63     {
64         $currentInformation = $this->naturalModuleInformations[$i];
65         $previousInformation = $this->naturalModuleInformations[$i - 1];
66         if ($currentInformation->getSourceSign() <> self::SIGN_CREATE)
67         {
68             if ($previousInformation->getSourceSign() <> self::SIGN_CREATE)
69             {
70                 if ($currentInformation->getHash() <> $previousInformation->getHash())
71                 {
72                     if ($currentInformation->getSourceDate('YmdHis') > $previousInformation->getSourceDate('YmdHis'))
73                     {
74                         $currentInformation->setSourceSign(self::SIGN_ERROR);
75                     }
76                 }
77             }
78         }
79     }
80 }

```

```

76         else
77         {
78             $currentInformation->setSourceSign(self::SIGN_NEXT_STEP);
79         }
80     }
81     else
82     {
83         $currentInformation->setSourceSign(self::SIGN_OK);
84     }
85 }
86 else
87 {
88     $currentInformation->setSourceSign(self::SIGN_ERROR);
89 }
90 }
91 elseif ($previousInformation->getSourceSign() <> self::SIGN_CREATE && $previousInformation->
    getSourceSign() <> self::SIGN_CREATE_AND_NEXT_STEP)
92 {
93     $currentInformation->setSourceSign(self::SIGN_CREATE_AND_NEXT_STEP);
94 }
95 }
96 }
97
98 private function containsSourceSign($sign)
99 {
100     foreach($this->naturalModuleInformations as $information)
101     {
102         if ($information->getSourceSign() == $sign)
103         {
104             return true;
105         }
106     }
107     return false ;
108 }
109
110 private function containsCatalogSign($sign)
111 {
112     foreach($this->naturalModuleInformations as $information)
113     {
114         if ($information->getCatalogSign() == $sign)
115         {
116             return true;
117         }
118     }
119     return false ;
120 }
121 }
122 ?>

```

Listing 2: Klasse: ComparedNaturalModuleInformation

## A.12 Klassendiagramm

Klassendiagramme und weitere UML-Diagramme kann man auch direkt mit  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/class-diagram.html>.

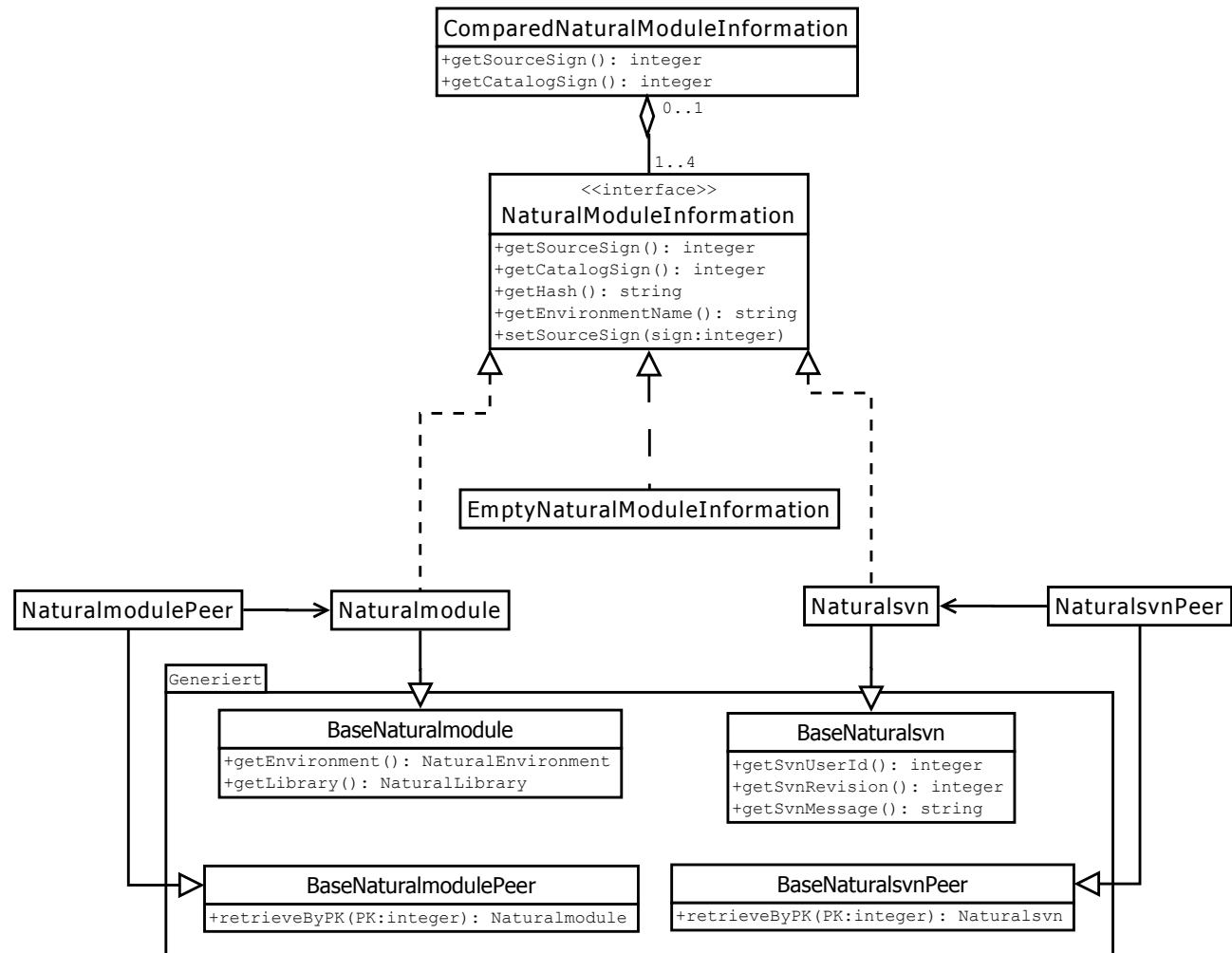







Abbildung 11: Klassendiagramm

## A.13 Benutzerdokumentation

Ausschnitt aus der Benutzerdokumentation:

Symbol	Bedeutung global	Bedeutung einzeln
	Alle Module weisen den gleichen Stand auf.	Das Modul ist auf dem gleichen Stand wie das Modul auf der vorherigen Umgebung.
	Es existieren keine Module (fachlich nicht möglich).	Weder auf der aktuellen noch auf der vorherigen Umgebung sind Module angelegt. Es kann also auch nichts übertragen werden.
	Ein Modul muss durch das Übertragen von der vorherigen Umgebung erstellt werden.	Das Modul der vorherigen Umgebung kann übertragen werden, auf dieser Umgebung ist noch kein Modul vorhanden.
	Auf einer vorherigen Umgebung gibt es ein Modul, welches übertragen werden kann, um das nächste zu aktualisieren.	Das Modul der vorherigen Umgebung kann übertragen werden um dieses zu aktualisieren.
	Ein Modul auf einer Umgebung wurde entgegen des Entwicklungsprozesses gespeichert.	Das aktuelle Modul ist neuer als das Modul auf der vorherigen Umgebung oder die vorherige Umgebung wurde übersprungen.