

Über diese Vorlage

Diese L^AT_EX-Vorlage wurde von Stefan Macke¹ 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-Berufe² 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 ? abgerundet wurden.

Sämtliche verwendeten Abbildungen, Tabellen und Listings stammen von ?.

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.³



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

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.

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

²z. B. IT-Kaufleute, Fachinformatiker mit Fachrichtung Systemintegration usw.

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

⁴Die Namensnennung im L^AT_EX-Quelltext mit Link auf <http://fiae.link/LaTeXVorlageFIAE> reicht hierfür aus.

Inhalt der Projektdokumentation

Grundsätzlich definiert die ?, S. 1746⁵ 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 ?, 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.

⁵Dieses 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 ?.

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 ?. 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 Sommer 2015

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Entwicklung von NatInfo

Webbasiertes Tool zur Unterstützung der Entwickler

Abgabetermin: Vechta, den 23.04.2015

Prüfungsbewerber:

Stefan Macke
Meine Straße 1
49377 Vechta



Ausbildungsbetrieb:

ALTE OLDENBURGER Krankenversicherung AG
Theodor-Heuss-Str. 96
49377 Vechta

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

1 Einleitung

1.1 Projektumfeld

- Die folgende Projektdokumentation schildert den Ablauf des IHK-Abschlussprojektes, welches der Autor im Rahmen seiner Ausbildung zum Fachinformatiker mit Fachrichtung Anwendungsentwicklung durchgeführt hat. Ausbildungsbetrieb ist die Fanuc Europe GmbH.

1.2 Projektziel

- Im CRM-System der Fanuc Europe befinden sich über 220.000 Kundenkonten, die Daten wie Adresse, Telefonnummer und Website des Kunden enthalten. Zur Automatisierung der Verwaltung dieser Kunden wird die Salesforce CRM-Plattform verwendet.
- Da es unmöglich ist, Änderungen an den Kundendaten manuell zu verfolgen, ist die Erstellung eines Web-Scraping-Systems geplant. Diese Anwendung automatisiert die jeweiligen Kontaktseiten im Internet herauslesen und vergleicht sie mit den aktuellen Daten im CRM-System.

1.3 Projektbegründung

- Das Unternehmen verfügt oft über Informationen, die in der gesamten Organisation verstreut sind, d.h. teilweise redundant in getrennten Systemen, aber häufig unvollständig. Das hat zur Folge, dass mehrere Unternehmensbereiche scheinbar nur mit den gleichen Daten arbeiten. Die kritische 360-Grad-Perspektive von Kunden oder Geschäftspartnern fehlt somit. Darüber hinaus besteht ein erhebliches Risiko, dass unzureichende Daten als Grundlage für strategische Geschäftsentscheidungen verwendet werden, die dann zur Festlegung von Unternehmenszielen und zur Modellierung von Geschäftsprozessen herangezogen werden. Folglich sind qualitativ hochwertige Daten erforderlich, damit das Unternehmen auf dem Markt erfolgreich sein kann. Der Hauptzweck besteht darin, die manuelle Arbeit zu verringern, die häufig dazu führt, dass ungenaue Daten im System gespeichert werden. Dies führt zu Problemen bei der Datenqualität im Unternehmen. Stattdessen kann ein Administrator einmal pro Woche einen automatisierten Vorgang ausführen, um über Änderungen an den Kundendaten informiert zu werden, was die Produktivität erhöht und den Zeitaufwand für die Suche nach den relevanten Informationen verringert. Aus den oben genannten Gründen beauftragte mich die Leitung der IT-Abteilung im Rahmen meines Abschlussprojekts mit der Verbesserung der Datenqualität durch eine C#-Anwendung.

1.4 Projektschnittstellen

- Das Hauptaugenmerk liegt auf der Bestimmung von Softwarekomponenten, die sich am effektivsten in die Betriebs- und Systemumgebung des Unternehmens integrieren lassen. In jedem Fall müssen bei der Auswahl dieser Softwarekomponenten die aktuellen CRM-, ERP- oder sonstigen datenhaltenden Systeme berücksichtigt werden. Die aktuellen Daten werden über WSDL-Methoden von Salesforce in eine C-Anwendung transportiert, die WPF für die Präsentationsschicht nutzt und einem definierten MVVM-Design folgt. Damit wird eine logische Schnittstelle geschaffen, über die alle Subsysteme agieren können. Ziel ist es, die Datenverarbeitung und Web Scraping Services so einfach wie möglich zu gestalten.
- Mit welchen anderen Systemen interagiert die Anwendung (technische Schnittstellen)?
- Wer genehmigt das Projekt bzw. stellt Mittel zur Verfügung?
- Wer sind die Benutzer der Anwendung?
- Wem muss das Ergebnis präsentiert werden?

1.5 Projektabgrenzung

- Eine der größten Herausforderungen bei der Erstellung dieses Projekts ist die Definition dessen, was geschehen soll. Daher ist es wichtig, den vom CRM-Entwicklungsteam benötigten Output klar zu definieren. Es wurde klargestellt, dass ein System, das Daten innerhalb der CRM-Plattform automatisch überschreibt, nicht erwünscht ist, sondern den Benutzer mit Hilfe von logisch aufgebauten Berichten benachrichtigt. Dies führt zu einer geringeren Fehleranfälligkeit, da Websites dynamisch sein oder sich ändernde Elemente aufweisen können, mit denen der Web Scraper Probleme bekommen kann.

2 Projektplanung

In der Projektplanung soll die notwendige Zeit und die benötigten Ressourcen sowie ein Ablauf der Durchführung des Projektes geplant werden.

2.1 Projektphasen

- Für die Umsetzung des Projekts standen dem Autor 70 Stunden zur Verfügung. Diese wurden vor dem Start des Projektes auf verschiedene Phasen verteilt, die während der Softwareentwicklung durchlaufen werden. Eine grobe Zeitplanung sowie die Hauptphasen können der Tabelle 1 entnommen werden: Grobzeitplanung entnommen werden. Dazu lassen sich die einzelnen Hauptphasen noch in kleinere Unterphasen untergliedern.

Beispiel Tabelle ?? zeigt ein Beispiel für eine grobe Zeitplanung.

Projektphase	Geplante Zeit
Startphase	6 h
Projektplanung	14 h
Durchführungsphase	38 h
Erstellen der Dokumentation	12 h
Gesamt	70 h

Tabelle 1: Zeitplanung

Eine detaillierte Übersicht über diese Phasen befindet sich im Anhang ??: ?? auf dieser Seite A.1: Detaillierte Zeitplanung am S. i.

2.2 Abweichungen vom Projektantrag

- Im Projektantrag wurde festgelegt, dass bestimmte Bibliotheken mit Python für das Webscraping-System zusammen mit einem einfachen Codebeispiel verwendet werden. Dies wurde jedoch geändert, da es mehrere dynamische Websites gibt, was bedeutet, dass der Benutzer zuerst mit der Website interagieren muss, um auf sie zuzugreifen. Es gibt auch Fälle, in denen die Elemente für die Webseiten auf der Serverseite verschleiert und unsichtbar sind, was es für den Webscraper schwieriger macht, sie zu lesen und fehleranfällig ist. Die Menge der verwertbaren Stammdaten hat sich aufgrund der begrenzten Komplexität des Webscrapers reduziert. Dies ist auf Probleme bei der manuellen Dateneingabe sowie auf http/https-Probleme zurückzuführen (TODO: Erweitern, Anhang)

2.3 Ressourcenplanung

- In der Übersicht, die in Anhang A.2: Verwendete Ressourcen auf S. ii zu finden ist, sind alle für das Projekt verwendeten Ressourcen aufgeführt. Dazu gehören Hard- und Software-Ressourcen sowie Personal. Bei der Auswahl der verwendeten Software wurde darauf geachtet, dass diese kostenlos (z.B. als Open Source) zur Verfügung steht oder das Unternehmen bereits Lizenzen dafür besitzt. Dadurch konnten die Projektkosten so gering wie möglich gehalten werden.

2.4 Entwicklungsprozess

- Bevor das Projekt umgesetzt wurde, musste ein bestimmter geeigneter Entwicklungsprozess gewählt werden. Dieser definiert die Prozedur, nach der die Umsetzung erfolgen soll. Im Verlauf des Projekts entschied sich der Autor für eine agile Methodik, das sogenannte Inkrementelle Vorgehensmodell. Bei der agilen Softwareentwicklung geht es darum, möglichst schnell auf sich ändernde Anforderungen reagieren zu können.

- Was ein inkrementelles Vorgehensmodell auszeichnet, ist, dass das zu entwickelnde System nicht im Voraus in allen Details genau geplant und dann in einem einzigen langen Durchgang entwickelt wird. Stattdessen stehen Teile der Software bei neuen Erkenntnissen und Entdeckungen immer wieder neu im Mittelpunkt. Die Entwicklung findet in kurzen Zeitspannen statt, nach denen jeweils ein neues Leistungsmerkmal erstellt wird, das dem Kunden gezeigt werden kann. Sollte der Kunde einen Anpassungswunsch haben oder eine neue Erkenntnis über die Identifikation von Anomalien gemacht werden, kann darauf bei der nächsten Iteration schnell reagiert werden.

3 Analysephase

3.1 Ist-Analyse

- Aufgrund der großen Anzahl von Kundenstamm Setzen ist eine manuelle Prüfung der Daten nicht möglich. Aufgrund vorgangenen Analysen geht man von Fehlerhaften Datensätzen im Umfang von 20Es bestehen Zweifel, wenn zwei Adressen vorhanden sind, da nicht klar ist, auf welche sie sich beziehen. Die Bezeichnungen SStrassenadresse 1 und SStrassenadresse 2 sind interpretationsbedürftig und sollten nach Gelegenheit geändert werden.

3.2 Wirtschaftlichkeitsanalyse

- Das Vertriebsteam hat festgestellt, dass es regelmäßig Rückläufer für ungültige E-Mails erhält. Außerdem erhalten einige Interessenten und Kunden nicht den richtigen Inhalt, weil sie nicht richtig qualifiziert sind. Man stellt fest, dass die E-Mail-Adresse das falsche Format hat und der Kundentyp falsch ist. Derzeit ist es erforderlich, dass ein Vertriebsmitarbeiter direkt vom Kunden über Änderungen in den Kundenkontaktdaten informiert wird, oder dass der Kunde im Rahmen der Kommunikation darauf aufmerksam gemacht wird. Dies führt jedoch zu Verzögerungen, Unannehmlichkeiten und Missverständnissen für die Kunden, was wiederum den wirtschaftlichen Faktor des Unternehmensimages beeinträchtigt. Die wirtschaftliche Berücksichtigung und die Entscheidung, ob die Realisierung des Projekts gerechtfertigt ist, wird in den folgenden Abschnitten vorgenommen.

3.2.1 „Make or Buy“-Entscheidung

- Da es sich bei den eingehenden Dokumenten um firmenspezifische Prozesse der Fanuc Europe GmbH handelt, ist eine Lösung durch ein zugekauftes Produkt nicht möglich. Daher muss eine Lösung durch die Fanuc Europe GmbH entwickelt werden.

3.2.2 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 Lehrjahr pro Monat 1000 € Brutto.

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

$$1000 \text{ €/Monat} \cdot 13,3 \text{ Monate/Jahr} = 13300 \text{ €/Jahr} \quad (2)$$

$$\frac{13300 \text{ €/Jahr}}{1760 \text{ h/Jahr}} \approx 7,56 \text{ €/h} \quad (3)$$

Es ergibt sich also ein Stundenlohn von 7,56 €. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen⁶ 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 ?? und sie betragen insgesamt 2739,20 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	70 h	7,56 € + 15 € = 22,56 €	1579,20 €
Fachgespräch	3 h	25 € + 15 € = 40 €	120 €
Abnahmetest	1 h	25 € + 15 € = 40 €	40 €
Anwenderschulung	25 h	25 € + 15 € = 40 €	1000 €
			2739,20 €

Tabelle 2: Kostenaufstellung

3.2.3 Amortisationsdauer

- Im Folgenden wird festgestellt, an welchem Punkt sich die Entwicklung der Software amortisiert hat. Durch die Implementierung dieser Software in die internen Geschäftsabläufe wird der Faktor Datenqualität erhöht. Es gibt eine große Anzahl verschiedener Dimensionen und Kriterien, mit denen die Datenqualität beschrieben werden kann. Es gibt keine Richtlinien, wie viele Dimensionen man verwenden sollte. DAMA zum Beispiel definiert sechzig Dimensionen, während die meisten DQM-Software-Anbieter in der Regel nur fünf Dimensionen vorschlagen. Durch die Art und Weise, wie die Software aufgebaut ist, wird das Potenzial für Unannehmlichkeiten für den Kunden reduziert - was den potenziellen Handel schneller und konsistenter macht. Die

⁶Räumlichkeiten, Arbeitsplatzrechner etc.

CRM-Administratoren haben außerdem einen besseren Überblick und können inkonsistente Daten besser identifizieren. Da es möglich ist, die Software automatisch im Rahmen von geplanten Arbeitsabläufen auszuführen, ist der manuelle Arbeitsaufwand sehr gering, wodurch die Personalkosten auf ein Minimum reduziert werden. Im Folgenden soll nun die Arbeitszeiterparnis in tabellarischer Form ermittelt werden. Die Anzahl der Vorgänge pro Quartal und die Zeit pro Vorgang wurden durch den Salesforce-Administrator ermittelt

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

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

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

3.3 Nutzwertanalyse

- TODO: Kriterien gewichtung, Tabelle erstellen.

Beispiel Ein Beispiel für eine Entscheidungsmatrix findet sich in Kapitel ??: ??.

3.4 Anwendungsfälle

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

Beispiel Ein Beispiel für ein Use Case-Diagramm findet sich im Anhang ??: ?? auf dieser Seite.

3.5 Qualitätsanforderungen

- Welche Qualitätsanforderungen werden an die Anwendung gestellt (z. B. hinsichtlich Performance, Usability, Effizienz etc. (siehe ?))?

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 ??: ?? auf dieser Seite.

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 ?? wurde für die Implementierung der Anwendung das **PHP!**-Framework Symfony⁷ ausgewählt.

Eigenschaft	Gewichtung	Akelos	CakePHP	Symfony	Eigenentwicklung
Dokumentation	5	4	3	5	0
Reengineerung	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

⁷Vgl. ?.

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 ??: ?? auf dieser Seite.

4.4 Datenmodell

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

Beispiel In ?? wird ein **ERM!** (**ERM!**) dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

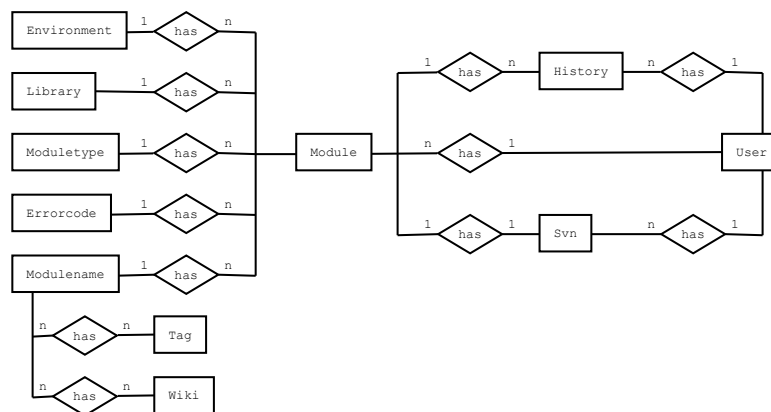


Abbildung 1: Vereinfachtes ER-Modell

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 ??: ?? auf dieser Seite eingesehen werden.

?? zeigt den grundsätzlichen Programmablauf beim Einlesen eines Moduls als **EPK!**.

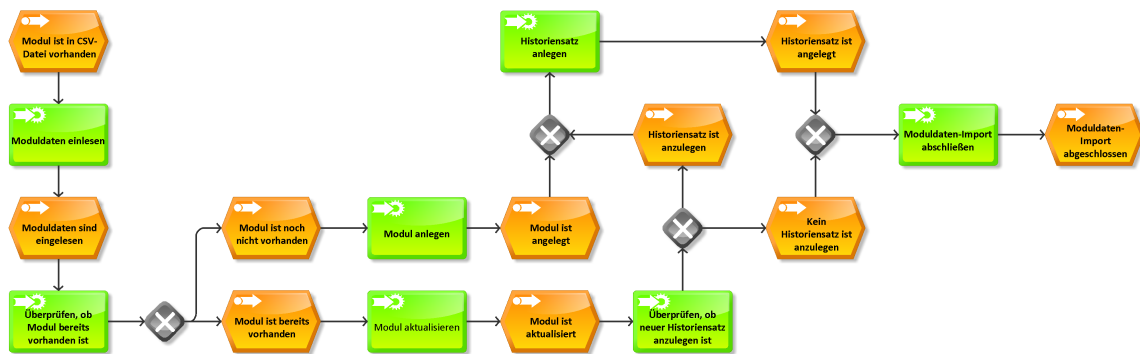


Abbildung 2: Prozess des Einlesens eines Moduls

4.6 Maßnahmen zur Qualitätssicherung

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

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 ??: ??) aufbauende Pflichtenheft ist im Anhang ??: ?? auf dieser Seite 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 ??: ?? auf dieser Seite.

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 ??: ?? 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 ??: ?? auf dieser Seite.

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 ??: ?? auf dieser Seite. Dort ist auch der Aufruf des Tests auf der Konsole des Webserverns 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 ??: ?? auf dieser Seite. Die Entwicklerdokumentation wurde mittels PHPDoc⁸ automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang ??: ?? auf dieser Seite.

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 ?? 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)?

⁸Vgl. ?

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
Gesamt	70 h	70 h	

Tabelle 4: Soll-/Ist-Vergleich

9.3 Ausblick

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

Eidesstattliche Erklärung

Ich, Stefan Macke, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler

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.

Vechta, den 23.04.2015

STEFAN MACKE

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase	9 h
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
Entwurfsphase	19 h
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
Implementierungsphase	29 h
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
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
Pufferzeit	2 h
1. Puffer	2 h
Gesamt	70 h

A.2 Lastenheft (Auszug)

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

Die Anwendung muss folgende Anforderungen erfüllen:

1. Herauslesen der Stammdaten
 - 1.1. Die Anwendung muss Kundendaten entweder als CSV oder direkt aus Salesforce verarbeiten können.
 - 1.2. Die Anwendung muss die Daten auf einheitliche Weise formatieren und Formatierungsregeln anwenden.
2. Darstellung der Daten
 - 2.1. Die Anwendung muss eine einfache Möglichkeit bieten, nach Datensegmenten zu suchen und Berichte auf der Grundlage dieser Segmente zu erstellen.
 - 2.2. Fehler und Anomalien müssen dem Nutzer deutlich angezeigt werden.
 - 2.3. Der Benutzer sollte die Möglichkeit haben, geplante Jobs zu erstellen, zu sehen und zu löschen
 - 2.4. Die Anwendung sollte die Daten in Salesforce nicht überschreiben, sondern den Benutzer auf Diskrepanzen aufmerksam machen
 - 2.5. CSV-Daten, die direkt vom Dataloader kommen, sollten ohne Probleme in der Anwendung eingelesen werden können
 - 2.6. Alle eingelesenen Daten müssen in der Benutzeroberfläche sichtbar sein
3. Webscraper
 - 3.1. Der Webscraper muss in der Lage sein, eine gegebene URL zu validieren und zu prüfen, ob die entsprechende Webseite verfügbar ist.
 - 3.2. Der Webscraper sollte nicht versuchen, ungültig formatierte URLs anzupassen.
 - 3.3. Der Benutzer sollte die Möglichkeit haben, Einstellungen für den Webscraper vorzunehmen, einschließlich der Anzahl der maximalen Seiten und ob http-Seiten (unsichere Seiten) ausgelesen werden dürfen
 - 3.4. Der Webscraper muss auf der Kommandozeile laufen können, unabhängig von der Schnittstelle.
 - 3.5. Wenn möglich, soll der Webscraper versuchen nach einer Kontaktseite zu suchen, wenn die Informationen nicht auf der Homepage aufgeführt sind.
4. Sonstige Anforderungen
 - 4.1. Die Anwendung sollte den Salesforce-Administratoren auf einem gemeinsamen Dateiserver zur Verfügung gestellt werden
 - 4.2. Die Daten der Anwendung müssen jede Nacht bzw. nach jedem **SVN!**-Commit automatisch aktualisiert werden.

- 4.3. Die Anwendung soll jederzeit erreichbar sein.
- 4.4. Da sich die Entwickler auf die Anwendung verlassen, muss diese korrekte Daten liefern und darf keinen Interpretationsspielraum lassen.
- 4.5. Mit dem Fortschreiten des CRM-Projekts müssen in späteren Phasen neue Datentypen implementiert werden.

A.3 Use Case-Diagramm

Use Case-Diagramme und weitere UML-Diagramme kann man auch direkt mit L^AT_EX zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/usecase-diagram.html>.

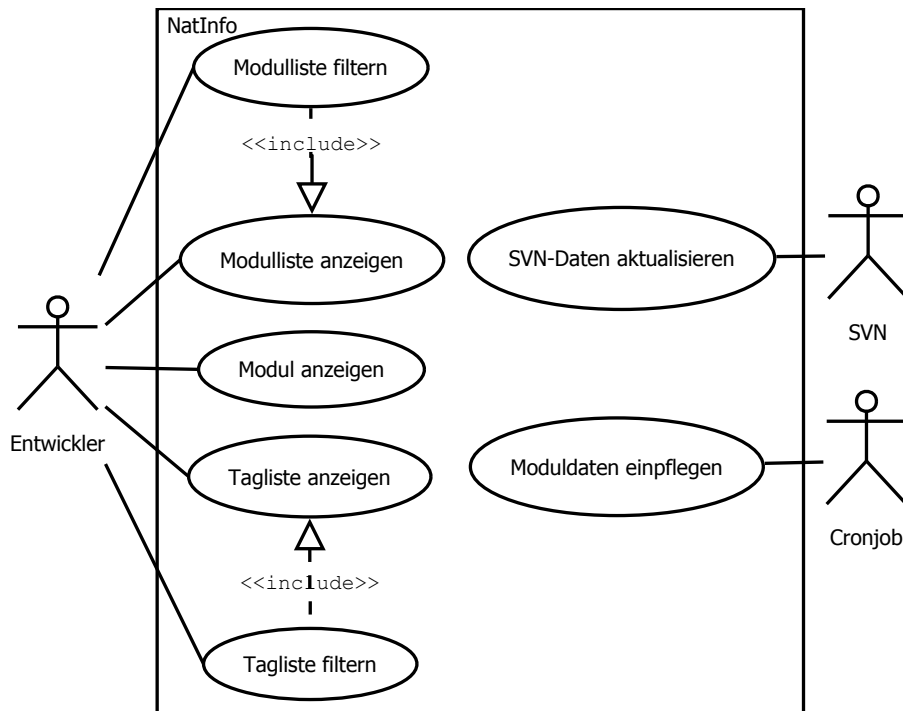


Abbildung 3: Use Case-Diagramm

A.4 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 tätigen 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 **SVN!** (**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 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.5 Datenbankmodell

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

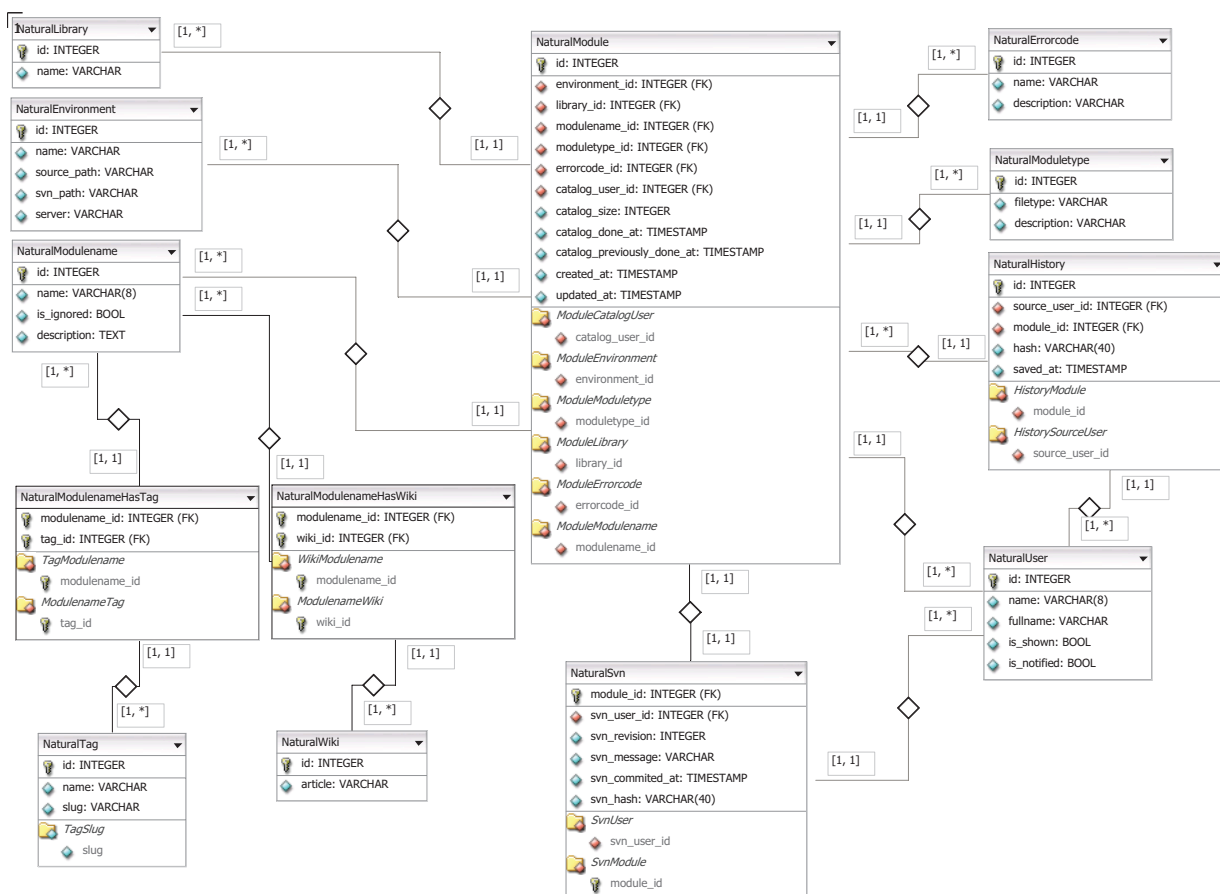


Abbildung 4: Datenbankmodell

A.6 Oberflächenentwürfe

The screenshot shows a web browser window with the address `http://natinfo.intranet/module/`. The interface includes a filter section at the top with three main categories: 'Source- and Catalog-Information from Environment', 'Source', and 'Catalog'. Each category has a 'Select' dropdown menu and a 'Filter' button. Below the filter section is a table listing modules with columns for Module, Library, Sourcen, Catalog, Source-User, Source-Date, Catalog-User, and Catalog-Date. The table contains several rows of data, including DGTEST, EINTTEST, NOCHEINS, MANUEL, RESET, and TESTER, each with associated library, source, and catalog information.

Module	Library	Sourcen	Catalog	Source-User	Source-Date	Catalog-User	Catalog-Date
DGTEST	UTILITY	+	+	Grashorn	01.04.2010	Grashorn	02.04.2010
EINTTEST	SYSTEM	-	+	Mustermann	01.04.2010	Grashorn	02.04.2010
NOCHEINS	UTILITY	o	-	Grashorn	05.04.2010	Grashorn	05.04.2010
MANUEL	SYSTEM	o	+	Grashorn	01.03.2010	Grashorn	01.03.2010
RESET	CON	-	+	Mustermann	02.03.2010	Mustermann	02.03.2010
TESTER	SYSTEM	+	-	Grashorn	01.02.2010	Grashorn	01.02.2010
...

Abbildung 5: Liste der Module mit Filtermöglichkeiten

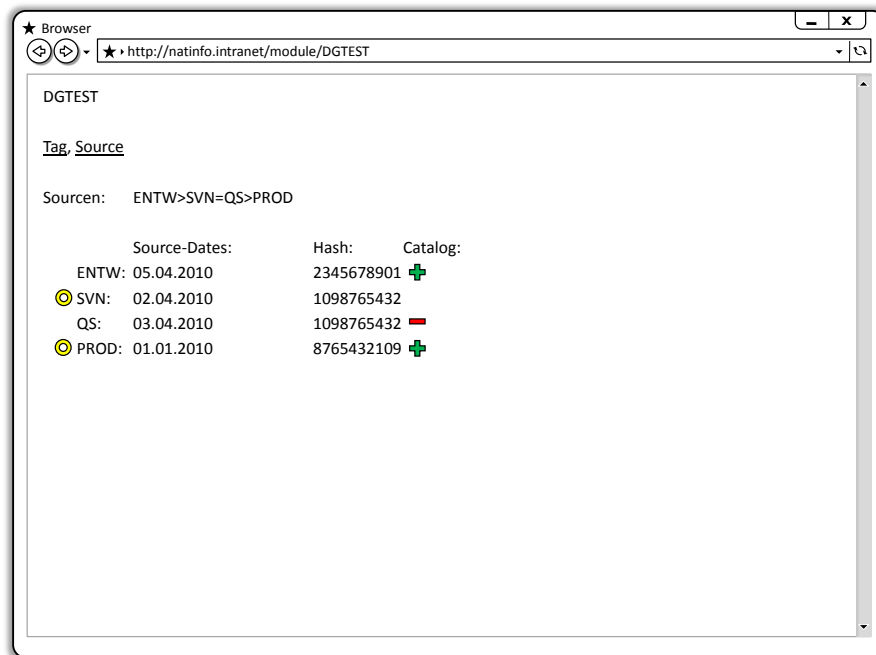


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

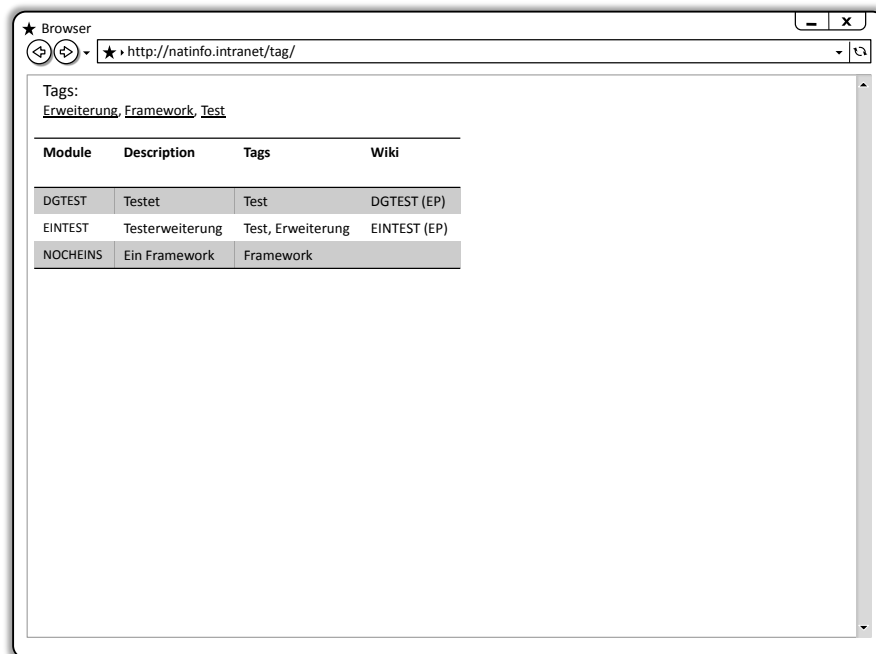


Abbildung 7: Anzeige und Filterung der Module nach Tags

A.7 Screenshots der Anwendung



Tags

Project, Test

Modulename	Description	Tags	Wiki
DGTEST	Macht einen ganz tollen Tab.	HGP	SMTAB_(EP), b
MALWAS		HGP, Test	
HDRGE		HGP, Project	
WURAM		HGP, Test	
PAMIU		HGP	

Abbildung 8: Anzeige und Filterung der Module nach Tags



Modules

Environment	ENTW
Library	Select
Catalog user	Select
Catalog date	Select
Source user	Select
Source date	Select
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.8 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.

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](#)

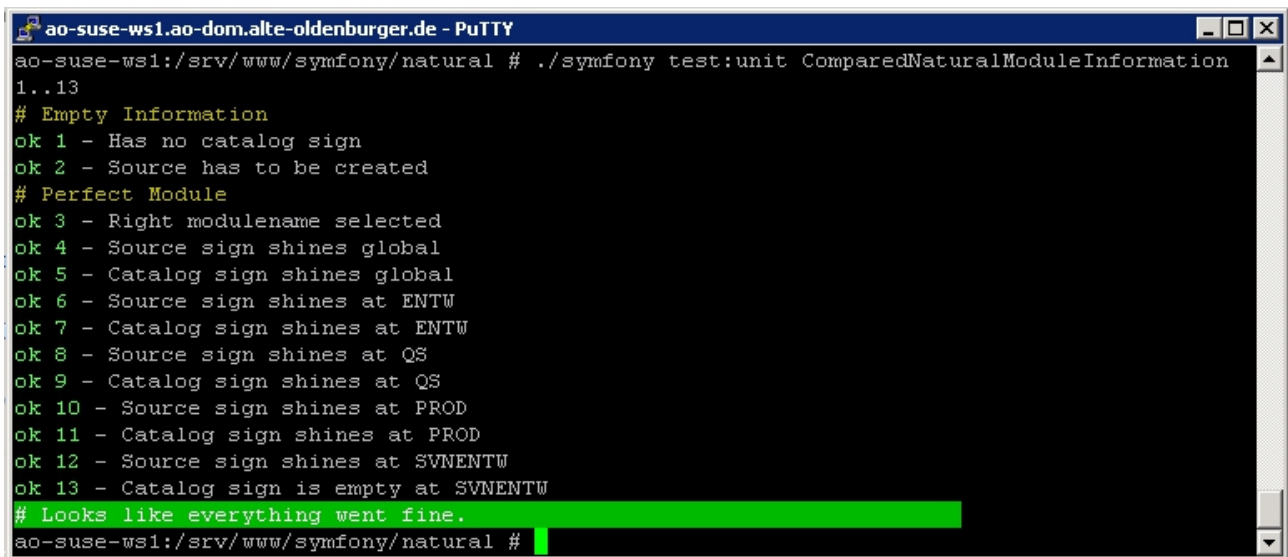
A.9 Testfall und sein Aufruf auf der Konsole

```

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.10 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);
```

```

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.11 Klassendiagramm

Klassendiagramme und weitere UML-Diagramme kann man auch direkt mit L^AT_EX zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/class-diagram.html>.

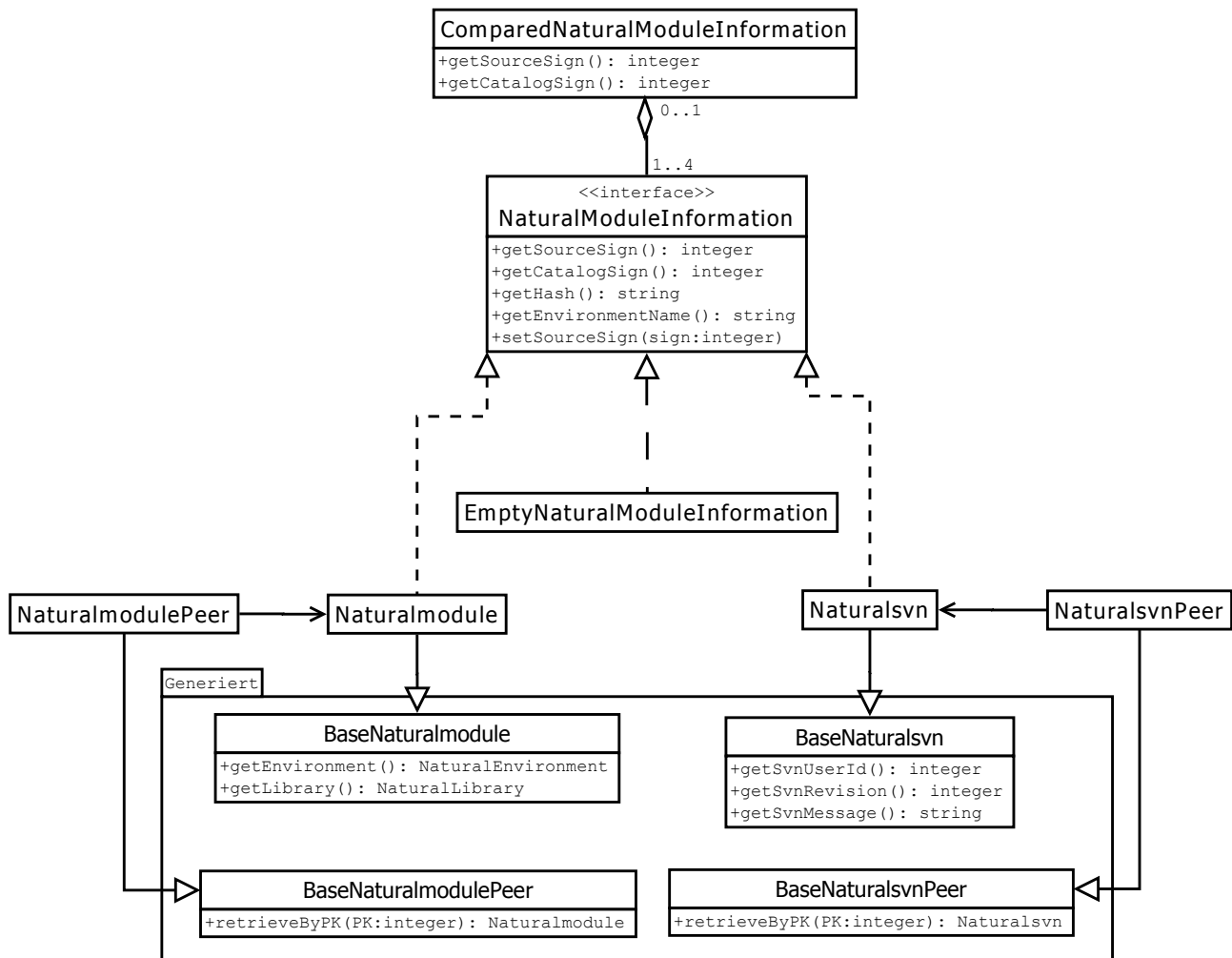







Abbildung 11: Klassendiagramm

A.12 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.