



Abschlussprüfung Sommer 2016

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Entwicklung einer Statistik-App

Webbasierte App für den ePages-App-Store zur statistischen
Analyse von KPIs

Abgabetermin: Gera, den 30.11.2016

Prüfungsbewerber:

Steven Hergt
Georg-Büchner-Str. 9
07749 Jena



Ausbildungsbetrieb:

ePages GmbH
Heinrich-Heine-Str. 1
07749 Jena

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	III
Tabellenverzeichnis	IV
Listings	V
Abkürzungsverzeichnis	VI
1 Einleitung	1
1.1 Projektumfeld	1
1.2 Projektziel	1
1.3 Projektbegründung	1
1.4 Projektschnittstellen	2
1.5 Projektabgrenzung	2
2 Projektplanung	3
2.1 Projektphasen	3
2.2 Abweichungen vom Projektantrag	3
2.3 Ressourcenplanung	4
2.4 Entwicklungsprozess	4
3 Analysephase	4
3.1 Ist-Analyse	4
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	6
3.4 Anwendungsfälle	7
3.5 Qualitätsanforderungen	7
3.6 Lastenheft/Fachkonzept	7
3.7 Zwischenstand	8
4 Entwurfsphase	8
4.1 Zielplattform	8
4.2 Architekturdesign	8
4.3 Entwurf der Benutzeroberfläche	9
4.4 Datenmodell	9
4.5 Geschäftslogik	9
4.6 Maßnahmen zur Qualitätssicherung	10
4.7 Pflichtenheft/Datenverarbeitungskonzept	10

Inhaltsverzeichnis

4.8	Zwischenstand	10
5	Implementierungsphase	11
5.1	Implementierung der Datenstrukturen	11
5.2	Implementierung der Benutzeroberfläche	11
5.3	Implementierung der Geschäftslogik	11
5.4	Zwischenstand	11
6	Abnahmephase	12
6.1	Zwischenstand	12
7	Einführungsphase	12
7.1	Zwischenstand	12
8	Dokumentation	13
8.1	Zwischenstand	13
9	Fazit	13
9.1	Soll-/Ist-Vergleich	13
9.2	Lessons Learned	14
9.3	Ausblick	14
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Lastenheft (Auszug)	i
A.3	Use Case-Diagramm	iii
A.4	Pflichtenheft (Auszug)	iii
A.5	Datenbankmodell	v
A.6	Oberflächenentwürfe	vi
A.7	Screenshots der Anwendung	viii
A.8	Entwicklerdokumentation	x
A.9	Testfall und sein Aufruf auf der Konsole	xii
A.10	Klasse: ComparedNaturalModuleInformation	xiii
A.11	Klassendiagramm	xvi
A.12	Benutzerdokumentation	xvii

Abbildungsverzeichnis

1	Amortisationszeit pro monatlicher Miete	7
2	Vereinfachtes ER-Modell	9
3	Prozess des Einlesens eines Moduls	10
4	Use Case-Diagramm	iii
5	Datenbankmodell	v
6	Liste der Module mit Filtermöglichkeiten	vi
7	Anzeige der Übersichtsseite einzelner Module	vii
8	Anzeige und Filterung der Module nach Tags	vii
9	Anzeige und Filterung der Module nach Tags	viii
10	Liste der Module mit Filtermöglichkeiten	ix
11	Aufruf des Testfalls auf der Konsole	xiii
12	Klassendiagramm	xvi

Tabellenverzeichnis

1	Zeitplanung	3
2	Kostenaufstellung	6
3	Zwischenstand nach der Analysephase	8
4	Entscheidungsmatrix	8
5	Zwischenstand nach der Entwurfsphase	10
6	Zwischenstand nach der Implementierungsphase	12
7	Zwischenstand nach der Abnahmephase	12
8	Zwischenstand nach der Einführungsphase	13
9	Zwischenstand nach der Dokumentation	13
10	Soll-/Ist-Vergleich	14

Listings

Listings/tests.php	xii
Listings/cnmi.php	xiii

Abkürzungsverzeichnis

API	Application Programming Interface
App	Applikation
CSV	Comma Separated Value
DOM	Document Object Model
EPK	Ereignisgesteuerte Prozesskette
ERM	Entity-Relationship-Modell
FTP	File Transfer Protocol
Git	Versionskontrollsystem
GitHub	Online-Dienst zur Verwaltung quelloffener Software
HTML	Hypertext Markup Language
MVC	Model View Controller
NatInfo	Natural Information System
PHP	Hypertext Preprocessor
QA	Quality Assurance
REST	Representational state transfer
REST-API	REST-Schnittstelle
SCP	Secure Copy
SFTP	SSH File Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
UML	Unified Modeling Language
XML	Extensible Markup Language

1 Einleitung

1.1 Projektumfeld

Die ePages GmbH ist ein deutsches Software- und Dienstleistungsunternehmen, das Produkte zur Ermöglichung eines elektronischen Handels (E-Commerce) bereitstellt, d.h. Kunden können mit der Produktsoftware, die über Hosting-Provider wie Strato AG, 1 & 1, T-Online etc. vertrieben wird, einen individualisierten Onlineshop aufsetzen und ihn gegen eine monatliche Gebühr betreiben.

Die Firma wurde 1983 als “Beeck & Dahms GbR” von dem jetzigen Geschäftsführer Wilfried Beeck in Kiel gegründet und war später Teil der Intershop AG bis zur Absplitterung im Jahr 2002. Momentan arbeiten in der Firma insgesamt rund 180 Mitarbeiter. Der Hauptsitz der Firma ist in Hamburg, danach kommt Jena als Firmensitz mit rund 40 Mitarbeitern. Der Auftrag zur Erstellung der App kommt vom Produktmanagement und ist aus Kundenrückmeldungen entstanden. Innerhalb der Firma gibt es verschiedene mehr oder minder unabhängige Entwicklungsteams. Ich bin dabei Teil des ePages6-Core-Teams als Frontend-/Javascriptentwickler, das wiederum Teil der R&D-Abteilung ist. Die Projekterstellung findet halbtags während der Sprints statt, d.h. ich stehe daneben noch dem Team halbtäglich zur Unterstützung zur Verfügung und gehe in den Dailys auch immer auf den Status meines Projektes ein.

1.2 Projektziel

Ziel des Projektes ist die Erstellung einer externen WebApp für Endkunden eines ePages Onlineshops. Mit dieser App soll es für den Kunden möglich sein spezielle KPIs für deren Onlineshop berichtsmäßig dokumentiert und deren zeitliche Entwicklung angezeigt zu bekommen. Daraus sollen Hinweise zum Anpreisen bestimmter Artikel resultieren. Auf alle relevanten Bestelldaten des Onlineshops soll per REST-API zugegriffen werden. Die wichtigsten KPIs sind hierbei der Umsatz und die meistverkauften Produkte. Die Berichte sollen anpassbar an frei wählbare Zeiträume und den Bezahlstatus sein.

1.3 Projektbegründung

Für Endkunden eines ePages-Onlineshops besteht standardmäßig die Möglichkeit über das Erstellmenü ihres Shops verschiedene Analysewidgets auszuwählen, die jedoch nur die wesentlichen KPIs als Umsatz- und Artikelverkaufsstatistiken bereitstellen ohne daraus spezielleren Handlungsbedarf des Händlers abzuleiten. Durch eine Nutzerumfrage wurde festgestellt, dass von den Händlern genauere Analysen des Käuferverhaltens (Kaufabbruchrate, Herkunft, Bestellzeiten etc.) und mehr Anpassungsmöglichkeiten (KPIs pro Kunde) gewünscht werden. Die Kunden haben zwar die Möglichkeit sich bei externen Firmen wie etracker für umfangreiche Statistikauswertungen für ihren Onlineshop anzumelden, jedoch besitzt das ePages System auch einen eigenen App-Store, der sich als Verkaufsplattform für eine eigens dafür programmierte Statistik-App ebenfalls anbietet, welche auf die Nutzerwünsche zugeschnitten ist und damit höhere Nutzerbindung und Nutzerzufriedenheit als Zielsetzung hat. Das

1 Einleitung

ist auch hinsichtlich der Vermarktung des neuesten Softwareproduktes von ePages sinnvoll, welches im nächsten Jahr ausgerollt wird und eine moderne Variante des Bestandsproduktes darstellt, wodurch neuen Kunden gewonnen werden sollen und die Konkurrenzfähigkeit auf dem bestehenden Markt gewährleistet werden soll. Nicht zuletzt verdient ePages auch aktiv an der Vermarktung der App pro Nutzer.

1.4 Projektschnittstellen

Das Projekt wurde von meinem Ausbilder (Markus Höllein) und meinem direkten Disziplinarvorgesetzten (Mario Rieß) genehmigt, welcher stellvertretend für den Ausbildungsbetrieb spricht.

Die Projektmittel umfassen an Hardware einen leistungsstarken Laptop, zwei Monitore, ein externes Keyboard und eine Maus. An Software wird ein externer Zugang zu einem ePages-Developer-Shop benötigt, der exemplarisch als Anbindungsstelle für die zu entwickelnde App fungiert. Hard- und Software werden dabei vom Ausbildungsbetrieb bereitgestellt. Die App wird auf einem externen Server von "uberspace.de" gehostet, auf dem sich auch die MySQL-Datenbank der App befindet. Die monatlichen Kosten dafür übernehme ich selbst. Des Weiteren wird Git als Versionierungstool verwendet, wobei der firmeninterne GitHub-Zugang verwendet wird, um das Projekt auch extern auf dem GitHub-Server speichern zu können. Die Anbindung der App an die ePages-Software geschieht über die REST-API von ePages unter Verwendung von PHP. Dies geschieht in Zusammenarbeit mit erfahrenen Programmierern. Die finanziellen Mittel bzw. die Ausbildungsvergütung stellt die HR-Abteilung zur Verfügung. Nutzer der App können alle ePages-Endkunden sein, die sich für die kostenbehaftete Nutzung der App für ihren Onlineshop entschließen. Das Ergebnis des Projekts muss zuallererst der teaminternen QA-Abteilung zum Testen präsentiert werden. Bestehen hier keine Bedenken mehr, wird die App dem Produktmanagement vorgelegt, welches letztendlich darüber entscheidet die App in dem ePages-App-Store zur Nutzung anzubieten oder ob weitere Verbesserungen notwendig sind.

Für die Programmierung der App werden verschiedene Javascript-Frameworks und Bibliotheken verwendet wie JQuery, React.js und D3.js, die allesamt kostenlos sind.

- 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

Das Projekt ist nicht Teil der ePages-Bestandssoftware oder irgendeines anderen ePages-Softwareproduktes, sondern stellt als externe App eine eigenständige Software dar, die mittels geeigneter API nicht nur an ePages, sondern auch an andere Onlineshopsoftware angebunden werden könnte. Vorrangig wird jedoch die Anbindung an ePages sein. Durch die Eigenständigkeit der App wird zudem gewährleistet,

2 Projektplanung

dass die Bestandssoftware bei Entwicklungsfehlern/Bugs in der App nicht in Mitleidenschaft gezogen wird.

Das Projekt wird nur soweit entwickelt, dass die Grundfunktionalitäten vorhanden sind, womit alle geplanten Anwendungsfälle realisiert werden können. Der Feinschliff durch das Feedback aus der QA-Abteilung wird aus Zeitgründen in seiner Gänze nicht mehr Teil dieses Projektes sein können genauso wie die Einbeziehung des Feedbacks vom Produktmanagement oder die Integration in den ePages-App-Store.

2 Projektplanung

2.1 Projektphasen

Die Gesamtprojektbearbeitungszeit ist auf 70 Stunden festgelegt. Der Start des Projekts ist der 10.10.2016 und der Abschluss ist der 30.11.2016. Die 70 Stunden werden auf den Zeitraum relativ gleichmäßig verteilt, sodass mindestens halbtäglich noch den Tagesgeschäften nachgegangen und das Team unterstützt werden kann. Außerdem muss gewährleistet werden, dass ich trotz des Projektes an den wichtigsten Scrum- und Team-Meetings teilnehmen kann, die die Arbeitszeit regelmäßig und unregelmäßig unterbrechen.

Beispiel Tabelle 1 zeigt ein Beispiel für eine grobe Zeitplanung.

Projektphase	Geplante Zeit
Analysephase	8 h
Entwurfsphase	6 h
Implementierungsphase	50 h
Erstellen der Dokumentation	6 h
Gesamt	70 h

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang [A.1: Detaillierte Zeitplanung](#) auf Seite [i](#).

2.2 Abweichungen vom Projektantrag

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

2.3 Ressourcenplanung

An Ressourcen ist ein PC-Arbeitsplatz (Schreibtisch, Rollcontainer, Schreibtischstuhl, Schreibtischlampe) in einem Firmenbüro vonnöten. Dazu kommt an Hardware ein LAN- und WLAN-fähiger COREi7-Laptop mit 16 GByte Arbeitsspeicher und mindestens 100 GByte großer Festplatte sowie zwei Monitore, eine Maus und ein Headset. An Software wird ein Windows 10 Betriebssystem, Sublime Text 3 als Text-Editor, ein Internetbrowser (Google Chrom und Firefox), Putty als [SSH](#)-Client, WinSCP als [SFTP](#) und [FTP](#)-Client-Software ein externer Server, auf dem die App „gehostet“ wird bereitgestellt von „uberspace.de“, phpMyAdmin zur Verwaltung der Datenbank, HipChat als internes Chattool der Firma, JIRA von Atlassian als Verwaltungssoftware der agilen Entwicklungsprozesse unter Scrum sowie Confluence von Atlassian als interne Kollaborationssoftware für Teams. Dazu kommt die Verwendung von [Git](#) und [GitHub](#) zur Versionierungskontrolle. Darüberhinaus werden auch personelle Ressourcen beansprucht, die einen Backendentwickler zum Einrichten des epages Developer-Shops und des Slim-Frameworks sowie eine [QA](#)-Kollegin zum Testen der App-Funktionen und meinen Ausbilder zur generellen Überwachung meines Projektes umfassen.

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

2.4 Entwicklungsprozess

Für die Entwicklung wird das Wasserfallmodell verwendet, in dem bestimmte Meilensteine gesetzt und abgeschlossene Phasen definiert werden können. Dadurch werden insbesondere eine gut definierte und ausgereifte Planungs- und Entwurfsphase ermöglicht, die zur eigentlichen Implementierung notwendig sind.

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

3 Analysephase

3.1 Ist-Analyse

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

3.2 Wirtschaftlichkeitsanalyse

3.2.1 „Make or Buy“-Entscheidung

Für jeden Endkunden einen ePages-Onlineshops besteht bereits die Möglichkeit über interne kostenlose Widgets den täglichen, wöchentlichen oder monatlichen Umsatz oder die für diese Zeiträume getätigten Bestellungen angezeigt zu bekommen. Zusätzlich können letzte Kundenregistrierungen und letzte Bestellungen angezeigt werden. Was fehlt sind hierbei tiefergehende statistische Analysen und grafische Darstellungen der wichtigsten und interessantesten KPIs für den Kunden, die sich dynamisch über REST-Calls aktualisieren lassen. Darunter fallen die Möglichkeit der Berichterstattung von KPIs pro Kunde und genauere Analysen des Käuferverhaltens (Kaufabbruchrate, Herkunft, Bestellzeiten etc.). Der Wunsch nach einer auf das ePages-System zugeschnittenen App für den ePages-App-Store wurde direkt vom Produktmanagement geäußert und sollte bereits schon mal während eines zweitägigen Hackathons entwickelt werden, was jedoch aufgrund des geschätzten zu hohem Zeitaufwands fallen gelassen wurde. Durch diese App verspricht sich das Produktmanagement eine höhere Kundenzufriedenheit und damit eine bessere Kundenbindung an die ePages Softwareprodukte sowie längerfristig eine Anwerbung neuer Kunden, die sich durch die Existenz dieser speziellen App vorzugsweise für ePages anstatt für eine andere Onlineshopsoftware entscheiden würden.

3.2.2 Projektkosten

Die Kosten für die Durchführung des Projektes setzen sich für die 70 Stunden Bearbeitungszeit sowie aus Personal- als auch Ressourcenkosten zusammen.

Berechnung der Entwicklungskosten für ePages Laut Arbeitsvertrag liegt meine Ausbildungsvergütung im aktuellen Lehrjahr bei 680 € pro Monat. Damit verursache ich der Firma jährliche Kosten in Höhe von

$$680 \text{ €/Monat} \cdot 12 \text{ Monate/Jahr} = 8160 \text{ €/Jahr.} \quad (1)$$

Die Anzahl der Arbeitstage 2016 belaufen sich auf 252 Tage in Thüringen (Jena). Davon stehen mir 25 Urlaubstage zu. Es verbleiben $(252 - 25) \text{ Tage} = 227 \text{ Tage}$ vollwertige achtstündige Arbeitstage. Meine Stundenlohn berechnet sich damit zu

$$8 \text{ h/Tag} \cdot 227 \text{ Tage/Jahr} = 1816 \text{ h/Jahr.} \quad (2)$$

$$\frac{8160 \text{ €/Jahr}}{1816 \text{ h/Jahr}} \approx 4,49 \text{ €/h.} \quad (3)$$

3 Analysephase

Für die Nutzung der Ressourcen¹ 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 2 und sie betragen insgesamt 1844,30 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	70 h	4,49 € + 15 € = 19,49 €	1364,30 €
Unterstützung durch Mitarbeiter	10 h	25 € + 15 € = 40 €	400,00 €
Abnahmetest durch QA	2 h	25 € + 15 € = 40 €	80,00 €
			1844,30 €

Tabelle 2: Kostenaufstellung

3.2.3 Amortisationsdauer

Geplant ist die fertige App den Nutzern der ePages-Software zur monatlichen Miete zur Verfügung zu stellen. Der Mietpreis wird in dem Bereich [5 €, 50 €] liegen. Die angenommene Zahl an Nutzern liegt in dem Bereich [1, 20000].

Berechnung der Amortisationszeit Für den Umsatz ($=U$), die diese App abhängig von der Zeit in Monaten ($=t_m$), den Nutzern ($=N$) und von dem Mietpreis pro Monat ($=p_m$) erwirtschaften soll wird die Formel $U = p_m \cdot N \cdot t_m$ zugrunde gelegt. Es wird angenommen, dass die App nach einem Monat 10 Nutzer hat. Dieser Wert wird modellhaft als linear ansteigend mit der Zeit t_m angenommen, d.h. $N = 10 \cdot t_m$. Daraus lässt eine Formel zur Berechnung der Amortisationszeit (t_m^A) ableiten:

$$t_m^A = \frac{U}{p_m \cdot N}, \quad \text{wobei } U = 1844,30 \text{ €, } N = 10 \cdot t_m^A \quad (4)$$

$$\Rightarrow t_m^A = \sqrt{\frac{1844,30 \text{ €}}{10 \cdot p_m}} = 13,58 \cdot p_m^{-\frac{1}{2}}. \quad (5)$$

Aus dem Funktionsgraphen aus [Abbildung 1](#) lässt sich die Amortisationszeit t_m^A für jeden möglichen Mietpreis pro Monat ablesen, wobei die Unsicherheit über das Ergebnis bei kleinen Monatsmietpreisen am geringsten ist, so liegt beispielsweise die Amortisationszeit für 5 € Monatsmiete bei ca. 6 Monaten, was durchaus im akzeptablen Budget- bzw. Investitionsrahmen der Firma ist.

3.3 Nutzwertanalyse

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

¹Laptop, Monitore, Servernutzung, Büro- und Firmenräumlichkeiten, Stromverbrauch, Heizkosten etc.

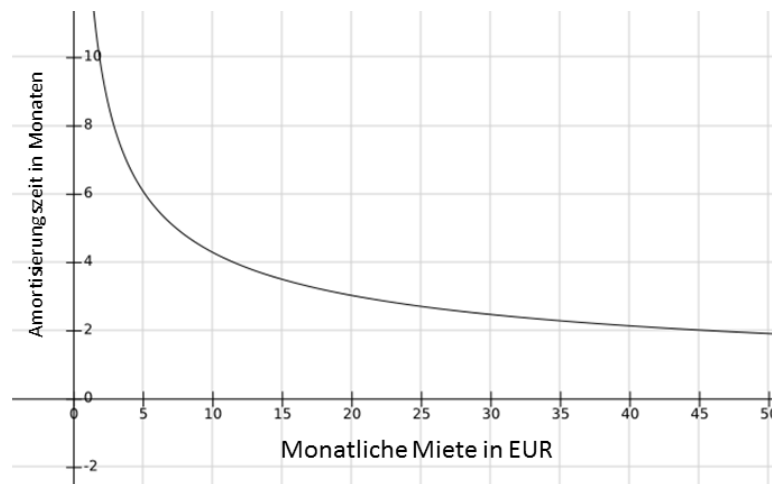


Abbildung 1: Amortisationszeit pro monatlicher Miete

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.3: Use Case-Diagramm](#) auf Seite [iii](#).

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 [A.2: Lastenheft \(Auszug\)](#) auf Seite [i](#).

3.7 Zwischenstand

Tabelle 3 zeigt den Zwischenstand nach der Analysephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Analyse des Ist-Zustands	3 h	4 h	+1 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h	1 h	
3. Erstellen eines „Use-Case“-Diagramms	2 h	2 h	
4. Erstellen des Lastenhefts	3 h	3 h	

Tabelle 3: Zwischenstand nach der Analysephase

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](#)² ausgewählt.

Eigenschaft	Gewichtung	Akelos	CakePHP	Symfony	Eigenentwicklung
Dokumentation	5	4	3	5	0
Reenginierung	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 4: Entscheidungsmatrix

²Vgl. ?.

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.6: Oberflächenentwürfe](#) auf Seite [vi](#).

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 2](#) wird ein Entity-Relationship-Modell ([ERM](#)) dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

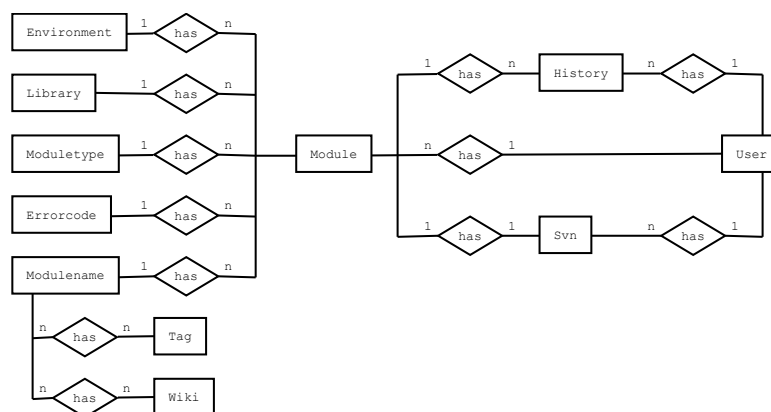


Abbildung 2: Vereinfachtes ER-Modell

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.11: Klassendiagramm](#) auf Seite [xvi](#) eingesehen werden.

[Abbildung 3](#) zeigt den grundsätzlichen Programmablauf beim Einlesen eines Moduls als [EPK](#).

4 Entwurfsphase

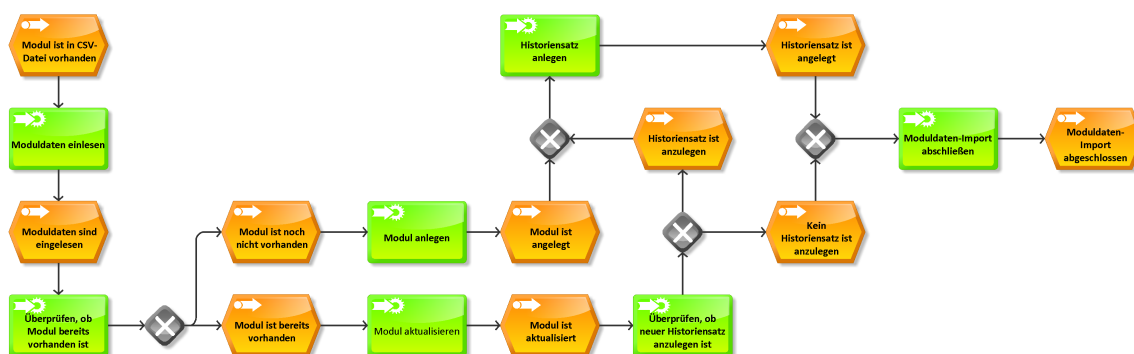


Abbildung 3: Prozess des Einlesens eines Moduls

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).

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.4: [Pflichtenheft \(Auszug\)](#) auf Seite iii zu finden.

4.8 Zwischenstand

Tabelle 5 zeigt den Zwischenstand nach der Entwurfsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Prozessentwurf	2 h	3 h	+1 h
2. Datenbankentwurf	3 h	5 h	+2 h
3. Erstellen von Datenverarbeitungskonzepten	4 h	4 h	
4. Benutzeroberflächen entwerfen und abstimmen	2 h	1 h	-1 h
5. Erstellen eines UML-Komponentendiagramms	4 h	2 h	-2 h
6. Erstellen des Pflichtenhefts	4 h	4 h	

Tabelle 5: Zwischenstand nach der Entwurfsphase

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.7: Screenshots der Anwendung](#) auf Seite [viii](#).

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.10: Klasse: ComparedNaturalModuleInformation](#) auf Seite [xiii](#).

5.4 Zwischenstand

Tabelle [6](#) zeigt den Zwischenstand nach der Implementierungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Anlegen der Datenbank	1 h	1 h	
2. Umsetzung der HTML-Oberflächen und Stylesheets	4 h	3 h	-1 h
3. Programmierung der PHP-Module für die Funktionen	23 h	23 h	
4. Nächtlichen Batchjob einrichten	1 h	1 h	

Tabelle 6: Zwischenstand nach der Implementierungsphase

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.9: Testfall und sein Aufruf auf der Konsole](#) auf Seite [xii](#). Dort ist auch der Aufruf des Tests auf der Konsole des Webserver zu sehen.

6.1 Zwischenstand

Tabelle 7 zeigt den Zwischenstand nach der Abnahmephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Abnahmetest der Fachabteilung	1 h	1 h	

Tabelle 7: Zwischenstand nach der Abnahmephase

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?

7.1 Zwischenstand

Tabelle 8 zeigt den Zwischenstand nach der Einführungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Einführung/Benutzerschulung	1 h	1 h	

Tabelle 8: Zwischenstand nach der Einführungsphase

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.12: Benutzerdokumentation](#) auf Seite [xvii](#). Die Entwicklerdokumentation wurde mittels PHPDoc³ automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang [A.8: Entwicklerdokumentation](#) auf Seite [x](#).

8.1 Zwischenstand

Tabelle 9 zeigt den Zwischenstand nach der Dokumentation.

Vorgang	Geplant	Tatsächlich	Differenz
1. Erstellen der Benutzerdokumentation	2 h	2 h	
2. Erstellen der Projektdokumentation	6 h	8 h	+2 h
3. Programmdokumentation	1 h	1 h	

Tabelle 9: Zwischenstand nach der Dokumentation

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?

³Vgl. ?

9 Fazit

- 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 10 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

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 10: Soll-/Ist-Vergleich

9.2 Lessons Learned

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

9.3 Ausblick

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

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase	8 h
1. Analyse des Ist-Zustands	1 h
1.1. Studium des Blogeintrags zur gewünschten App und Verschriftlichung	1 h
2. Wirtschaftlichkeitsprüfung und Amortisationsrechnung	2 h
3. Erstellen eines „Use-Case“-Diagramms	2 h
4. Erstellung eines Lastenheftes	3 h
Entwurfsphase	6 h
1. Erstellung eines Pflichtenheftes	3 h
2. Auswahl eines geeigneten Designs	2 h
3. Erstellung eines UML-Klassendiagramms	1 h
Implementierungsphase	50 h
1. Einrichtung eines ePages-Developer-Shops für die App-Entwicklung	0,5 h
2. Einrichtung des Slim-Frameworks zur Anbindung an die ePages REST-API	1,5 h
3. Implementierung des DOM-Gerüsts mit React.js	16 h
3.1. Routing der REST-Calls von Slim zu React	1 h
4. Datenbankerstellung	4 h
4.1. Erstellung der MySQL-Datenbank	1 h
4.2. Datenbankmodellerstellung zur Speicherung von Kunden- und Shopdaten	2 h
4.3. Umsetzung des Datenbankmodells	1 h
5. Implementierung serverseitiger php-Skripte	6 h
6. Implementierung der Javascript UI-Interaktionen	18 h
6. Tests der App-Funktionalitäten	4 h
6.1. Anlegen von Kunden- und Shopdaten im epages Developershop	2 h
6.1. Durchführung der Test	2 h
Erstellen der Dokumentation	6 h
1. Erstellen der Projektdokumentation	4 h
2. Erstellen der Entwicklerdokumentation	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. 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.3 Use Case-Diagramm

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

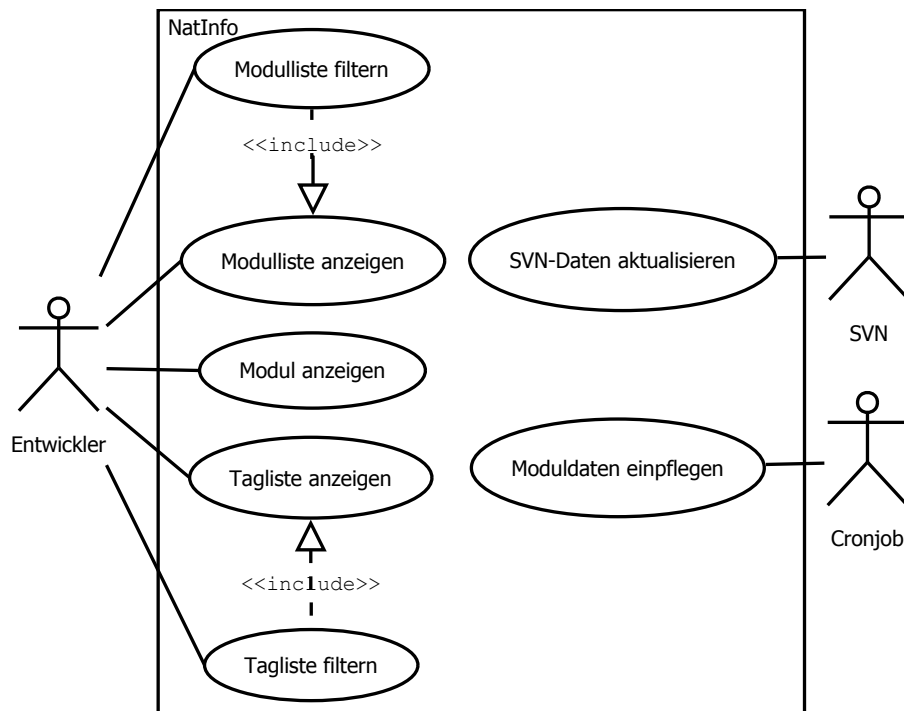


Abbildung 4: Use Case-Diagramm

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 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 **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

A Anhang

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

2. Zielgruppen

wird lediglich von den 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/>.

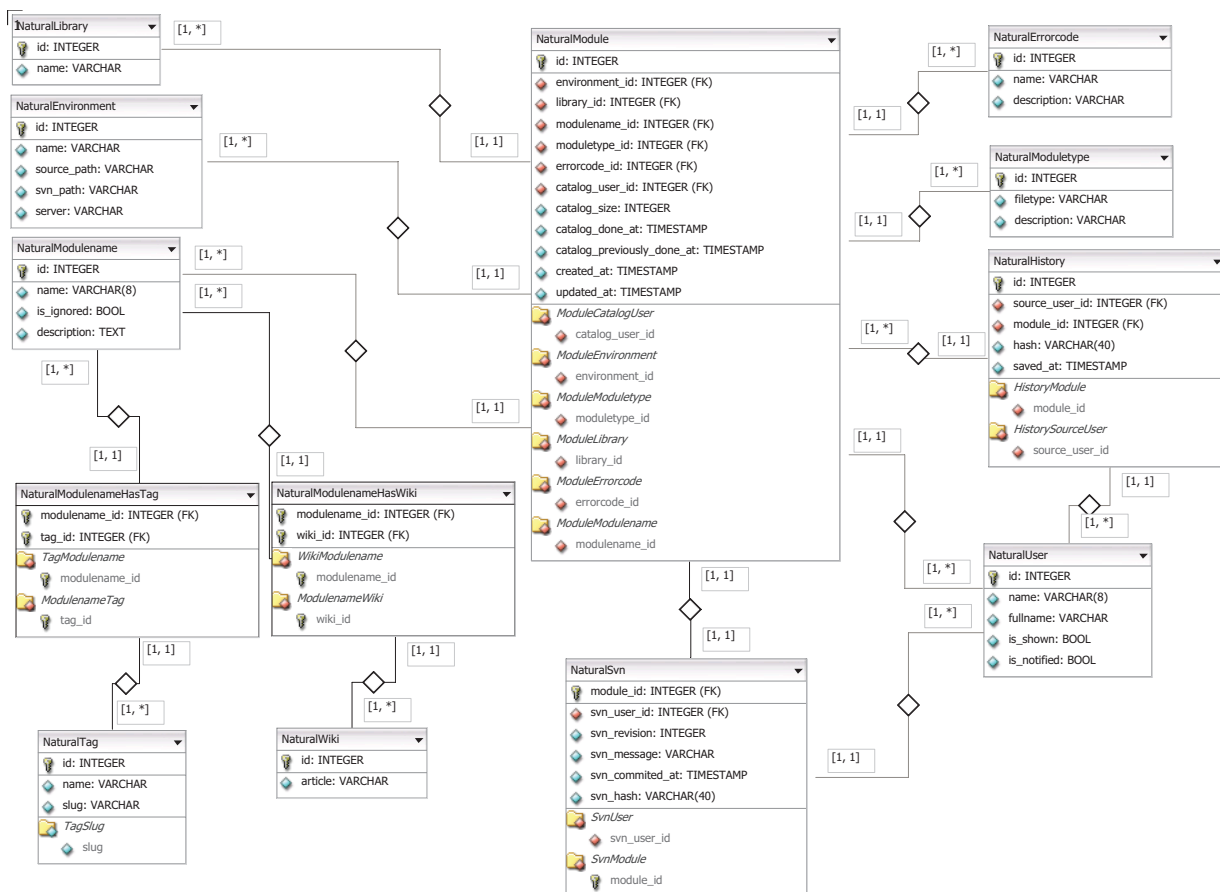


Abbildung 5: Datenbankmodell

A.6 Oberflächenentwürfe

Browser: <http://natinfo.intranet/module/>

Filter:

Source- and Catalog-Information from Environment:

Library:

Filter

Source:

Username:

Catalog:

Username:

☒ + ☒ - ☒ ☐

Module	Library	Sourcen	Catalog	Source-User	Source-Date	Catalog-User	Catalog-Date
DGTEST	UTILITY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Grashorn	01.04.2010	Grashorn	02.04.2010
EINTST	SYSTEM	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Mustermann	01.04.2010	Grashorn	02.04.2010
NOCHEINS	UTILITY	<input type="radio"/>	<input checked="" type="checkbox"/>	Grashorn	05.04.2010	Grashorn	05.04.2010
MANUEL	SYSTEM	<input type="radio"/>	<input checked="" type="checkbox"/>	Grashorn	01.03.2010	Grashorn	01.03.2010
RESET	CON	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Mustermann	02.03.2010	Mustermann	02.03.2010
TESTER	SYSTEM	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Grashorn	01.02.2010	Grashorn	01.02.2010
...

Abbildung 6: Liste der Module mit Filtermöglichkeiten

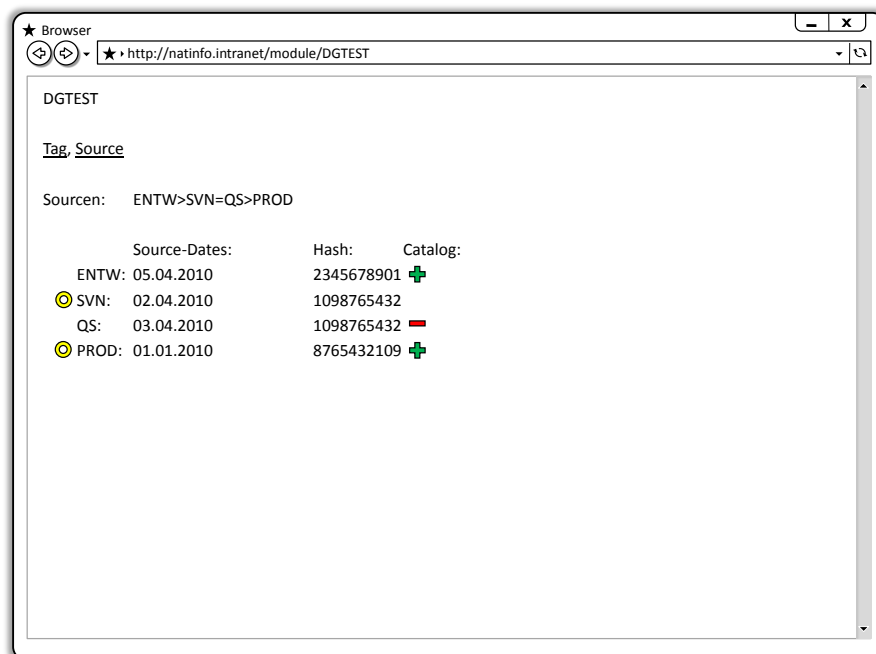


Abbildung 7: Anzeige der Übersichtsseite einzelner Module

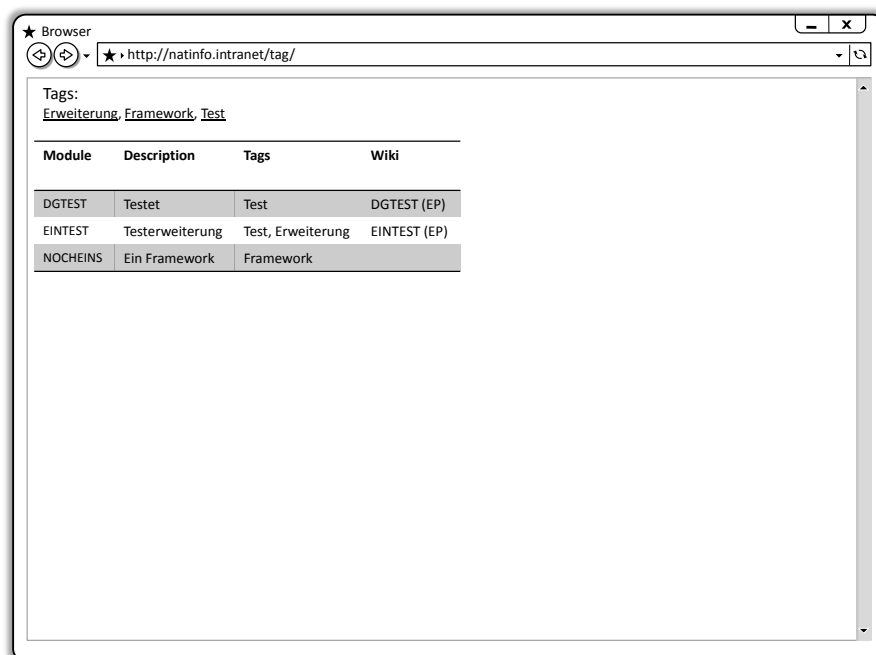


Abbildung 8: Anzeige und Filterung der Module nach Tags

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 9: 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 10: 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 ?>

```

```

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 11: 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);

```

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 }

```

A Anhang

```

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 ?>

```

A.11 Klassendiagramm

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

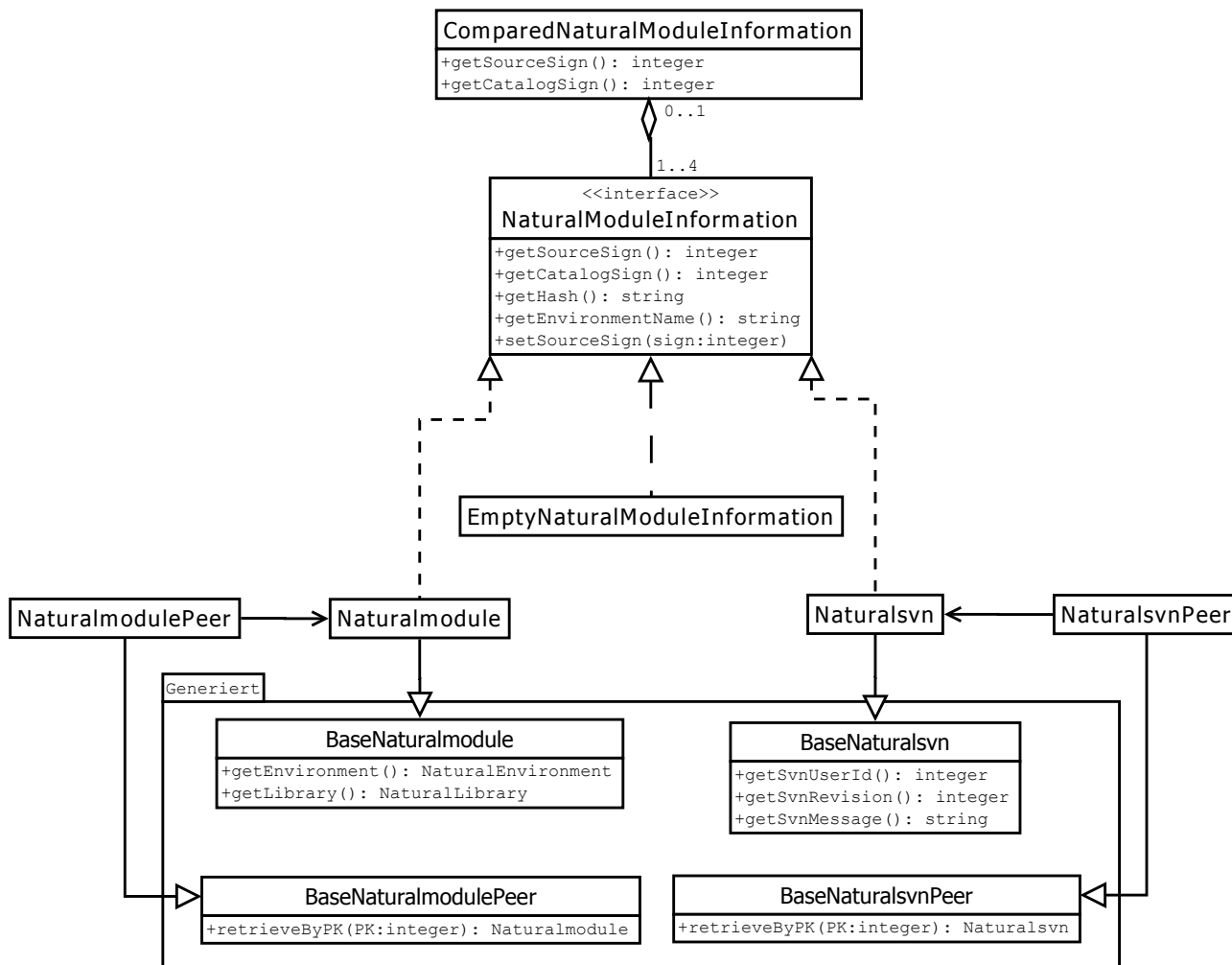







Abbildung 12: Klassendiagramm

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.