



Protokoll über die durchgeführte Projektarbeit

Das Protokoll ist der Dokumentation als Anhang beizufügen!

Prüfungsteilnehmer/-in

Ausbildungsberuf/ Prüfungsausschuss

1. Arbeitszeit

Das Projekt wurde von mir in der kalkulierten Zeit komplett fertiggestellt, einschließlich erforderlicher Nacharbeit ☐ ja ☐ nein

Nein, die Zeit wurde um _____ Stunden ☐ unterschritten ☐ überschritten.

Begründung

2. Ausführung

2.1 Das Projekt habe ich nach dem eingereichten Projektantrag ausgeführt

☐ ja ☐ nein

2.2 Hilfestellung war erforderlich

☐ ja ☐ nein

Begründung bei Hilfestellung

Umfang bei Hilfestellung



2.3 Das Projekt habe ich ohne Nacharbeit in einem kundengerechten Zustand übergeben

☐ ja ☐ nein

Begründung bei Nacharbeit

Umfang der Nacharbeit

2.4 Das Projekt war ein Einzelprojekt

☐ ja ☐ nein

3. Dokumentation

3.1 Die Dokumentation habe ich selbst, ohne fremde Hilfe, erstellt.

☐ ja ☐ nein

Hilfestellung

Persönliche Erklärung

Ich versichere durch meine Unterschrift, dass ich die Projektarbeit und die eingereichte Dokumentation selbstständig angefertigt, alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen, als solche kenntlich gemacht habe. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

Datum

Unterschrift Prüfungsteilnehmer/-in

Unterschrift Projektverantwortliche/-r des Auftraggebers



Abschlussprüfung Sommer 2023

Fachinformatiker für Systemintegration

Dokumentation zur betrieblichen Projektarbeit

Erweiterung eines Schulnetzwerkes um ein kabelloses Netzwerk

Implementierung einer WLAN Lösung

Abgabetermin: Berlin, den 08.06.2023

Prüfungsbewerber:

Marcel Akremi
Streitstraße 55
13587 Berlin



Ausbildungsbetrieb:

ARKTIS IT SOLUTIONS GMBH
Brunsütteler Damm 156-172
13581 Berlin

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	II
Tabellenverzeichnis	III
Abkürzungsverzeichnis	III
Literaturverzeichnis	III
1 Einleitung	1
1.1 Projektumfeld	1
1.2 Projektziel	1
1.3 Projektbegründung	2
1.4 Projektschnittstellen	2
1.4.1 Organisatorische Projektschnittstellen	2
1.4.2 Technische Projektschnittstellen	2
1.4.3 Personenelle Schnittstellen	2
1.5 Projektabgrenzung	3
2 Projektplanung	3
2.1 Projektphasen	3
2.2 Abweichungen vom Projektantrag	3
3 Analysephase	4
3.1 Kundengespräch	4
3.2 Ist-Analyse	4
3.3 Wirtschaftlichkeitsanalyse	4
3.3.1 „Make or Buy“-Entscheidung	4
3.3.2 Projektkosten	4
3.3.3 Personalkosten	5
3.3.4 Amortisationsdauer	5
3.4 Nutzwertanalyse	5
3.5 Anwendungsfälle	5
3.6 Qualitätsanforderungen	6
3.7 Schutzbedarf	6
3.8 Schutzmaßnahmen	6
4 Entwurfsphase	6
4.1 Zielplattform	6
4.2 Architekturdesign	6
4.3 Entwurf der Benutzeroberfläche	7
4.4 Datenmodell	7

Abbildungsverzeichnis

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

Abbildungsverzeichnis

1	Vereinfachtes ER-Modell	7
2	Prozess des Einlesens eines Moduls	8
3	Use Case-Diagramm	iv
4	Datenbankmodell	vi
5	Liste der Module mit Filtermöglichkeiten	vii
6	Anzeige der Übersichtsseite einzelner Module	viii

Tabellenverzeichnis

7	Anzeige und Filterung der Module nach Tags	viii
8	Anzeige und Filterung der Module nach Tags	ix
9	Liste der Module mit Filtermöglichkeiten	x
10	Aufruf des Testfalls auf der Konsole	xiv
11	Klassendiagramm	xviii

Tabellenverzeichnis

1	Zeitplanung	3
2	Entscheidungsmatrix	7
3	Soll-/Ist-Vergleich	i

Abkürzungsverzeichnis

Arktis	Arktis IT solutions GmbH	1
UG	Untergeschoss	4
VDSL	Very High Speed Digital Subscriber Line	4
LAN	Local Area Network	4
VLAN	Virtual Local Area	4
MVC	Model View Controller	6
NatInfo	Natural Information System	v
Natural	Programmiersprache der Software AG	v
PHP	Hypertext Preprocessor	6
SQL	Structured Query Language	9
SVN	Subversion	v
XML	Extensible Markup Language	9

Literaturverzeichnis

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

[lancom-systems.de] LANCOM-SYSTEMS.DE: *Infopaper - Limited Lifetime Warranty – Maximale Garantie für LANCOM Enterprise-Switches*. <https://www.lancom-systems.de/pdf/infopaper/LANCOM-Limited-Lifetime-Warranty-DE.pdf>, Abruf: 4.06.2023

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

1 Einleitung

Die folgende Projektdokumentation erläutert den Ablauf des IHK-Abschlussprojektes, das der Autor im Rahmen seiner Ausbildung zum Fachinformatiker für Systemintegration durchgeführt hat. Alle Einkaufspreise und Kalkulationen wurden abgeändert, da sie unter das Betriebsgeheimnis fallen. Aufgrund von Datenschutzbestimmungen müssen Personen und Organisationen, die im Zusammenhang mit diesem Projekt stehen, anonym bleiben und IP-Adressen und Gerätenamen aufgrund des Datenschutzes abgeändert werden. Daher werden Personen und Organisationen, die die Dienste der Arktis IT solutions GmbH ([Arktis](#)) in Anspruch nehmen im Folgenden nur als Kunde bezeichnet.

1.1 Projektumfeld

Der Ausbildungsbetrieb [Arktis](#) ist ein mittelständischer IT-Dienstleister mit Hauptsitz in Berlin. Die [Arktis](#) beschäftigt zur Zeit ca. 158 MitarbeiterInnen und bietet Technologielösungen mit den Schwerpunkten IT Security, IT-Infrastrukturmanagement, Integrierte Kommunikationslösungen, Intelligente Gebäudetechnik und Digitalisierung an. Beim Kunden handelt es sich um eine Gesamtschule. Der gesamte Schulcampus umfasst ca. 34.000 Quadratmeter und das Schulgebäude hat 30 Klassenräume, welche auf 3 Stockwerke verteilt sind. Der Kunde hat aktuell für die festen Arbeitsplätze ein kabelgebundenes Netzwerk mit Zugang zum Internet über deren Internetprovider.

Zum Umfeld gehören auch die Schnittstellen, s. [1.4](#).

1.2 Projektziel

Seit einiger Zeit werden im Unterricht verstärkt mobile Endgeräte, wie z.B. Tablets und Laptops, verwendet. Das Ziel dieses Projektes ist, die im Schulgebäude eingesetzten mobilen Endgeräte an das interne Schulnetzwerk und das Internet anzubinden. Hierfür soll im Schulgebäude ein kabelloses Netzwerk ausgestrahlt werden, welches die Anbindung der mobilen Endgeräte an das Schulnetz und Internet gewährleistet. Für diesen Zweck soll das bestehende kabelgebundene Netzwerk erneuert und um ein kabelloses Netzwerk erweitert werden. Die Kommunikation in diesem Netzwerk soll durch Switches und Access Points ermöglicht werden. Die Access Points sollen von einem WLAN-Controller zentral gesteuert werden. Die Switches, Access Points und der WLAN-Controller sollen den Kundenanforderungen entsprechend gewählt, konfiguriert und montiert werden. Das Netzwerk soll ausreichend Kapazität haben um den mobilen Endgeräten eine schnelle und hochverfügbare Verbindung zum Schulnetz und Internet bereitzustellen. Die eingesetzten Access Points müssen gleichzeitig im 2.4GHz -und 5GHz Frequenzband senden und empfangen. Es sollen mehrere Netze ausgestrahlt werden (z.B. Gast, Schüler, Lehrer). Abhängig davon, über welches Netz man sich im WLAN anmeldet, soll man verschiedene Berechtigungen im Schulnetz bekommen. Die Kommunikation innerhalb des WLANs soll nach aktuellen Sicherheitsstandards verschlüsselt werden.

1.3 Projektbegründung

Da der Kunde bereits mobile Endgeräte als alternatives Unterrichtsmedium verwendet, soll das Projekt das digitale Lernen erleichtern, indem es die Bereitstellung zusätzlicher Dienste wie z.B. einem Moodle Server und Netzwerklauferwerken ermöglicht. Da die Anbindung von so vielen Geräten, die zum Teil keine Ethernet-Schnittstelle haben, mit einer kabelgebundenen Verbindung wirtschaftlich nicht rentabel ist, wird eine Alternative benötigt. Ein kabelloses Netzwerk ist hierfür sehr gut geeignet, da es im Vergleich zum kabelgebundenen Netzwerk sehr flexibel und skalierbar ist und durch weniger benötigte passive Verkabelung Kosten bei der Anschaffung eingespart werden können.

1.4 Projektschnittstellen

1.4.1 Organisatorische Projektschnittstellen

Das Projekt wurde in dem Team IT-Infrastruktur durchgeführt, welche auch die Räumlichkeiten und nötigen Arbeitsmittel zur Verfügung gestellt hat. Da es sich bei dem Projekt um eine Erweiterung und Teilerneuerung eines bestehenden Netzwerkes handelt, kam es während des Projektes zu einer engen Zusammenarbeit mit der IT Abteilung des Kunden um sicherzustellen, dass die bestehenden Netzwerkkomponenten wie z.B. Webserver und Netzwerkdrucker nach der Durchführung komplett funktionstüchtig sind. Die Bestellung der benötigten Komponenten und den Kontakt mit den Lieferanten hat die Einkaufsabteilung der [Arktis](#) übernommen. Die Personalzuweisung der benötigten Mitarbeiter fand in Koordination mit dem PMO (Projekt Management Office) der Abteilung IT-Infrastruktur statt.

1.4.2 Technische Projektschnittstellen

Die eingesetzten Netzwerkkomponenten sollen mit den VM-Servern des Kunden kommunizieren, auf welchen die IT Abteilung vor Ort in Zukunft ihre Domäne und Services betreiben will. Darüber hinaus melden sich die mobilen Endgeräte im Schulgebäude über die Access Points im Schulnetzwerk an. Die benutzten Netzwerkkomponenten wurden über das Webinterface mit den Tools LANconfig, WLANmonitor und LANmonitor konfiguriert.

1.4.3 Personenelle Schnittstellen

Bei der Lieferung und Installation der Netzwerkkomponenten hat mir ein Mitarbeiter der Abteilung It-Infrastruktur geholfen. Er ist bei der [Arktis](#) der Experte bezüglich LANCOM-Lösungen und ist für das Hauptprojekt (Netzwerkerneuerung für Schulen in einem Landkreis in Brandenburg) verantwortlich. Bei der Inventarisierung der benutzten Netzwerkkomponenten haben mich die anderen Auszubildenden der Abteilung IT-Infrastruktur unterstützt.

1.5 Projektabgrenzung

Das Projekt ist zwar ein Teilprojekt der Netzwerkerneuerung für Schulen in einem Landkreis in Brandenburg, betrachtet aber nur die Arbeiten an einer Schule. Außerdem beschränkt sich das Projekt auf die Planung des Netzwerklayouts, die Konfiguration der Netzwerkkomponenten und die Ausarbeitung eines Sicherheitskonzepts. Die passive Verkabelung vor Ort für die benötigten Netzwerkdosen hat eine andere Firma übernommen. Die Montage der Access Points hat nach Kundenwunsch das Personal der Schule übernommen, um Kosten einzusparen. Die Access Points wurden demnach bei uns inventarisiert und danach zum Kunden geschickt. Die VM-Server, die der Kunde über das Netzwerk betreiben will werden von uns nur angebunden, die Einrichtung und Konfiguration ist Aufgabe der IT-Abteilung der Schule. Begehungen und WLAN-Ausleuchtungen würden über den Rahmen dieses Projekts hinaus gehen und wurden deswegen nicht betrachtet.

2 Projektplanung

2.1 Projektphasen

Das Projekt wurde in 5 Arbeitstagen im Zeitraum vom 14.04.2023 bis zum 31.05.2023 durchgeführt. Das Projekt ist in die vier Phasen „Planungsphase“, „Implementierungsphase“, „Deploymentphase“ und „Projektabschluss“ eingeteilt, wie in Tabelle 1 zu sehen ist.

Projektphase	Geplante Zeit
Planungsphase	14 h
Implementierungsphase	9 h
Deploymentphase	9 h
Projektabschluss	8 h
Gesamt	40 h

Tabelle 1: Zeitplanung

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

2.2 Abweichungen vom Projektantrag

Der detailliertere Projektplan hat sich seit dem Projektantrag etwas verändert, da sich manche Anforderungen geändert haben. Der Kunde hatte angemerkt, dass zusätzlich zu der im Lieferumfang enthaltenen „Limited Lifetime Warranty“, welche Geräte bis zum offiziellen „End of Life“ absichert (siehe [LANCOM-SYSTEMS.DE](#)), auch noch eine Garantieverweiterung, welche die Lieferung von Ersatzgeräten vom Hersteller zum nächsten Werktag beinhaltet, bestellt werden sollte. Damit diese Garantieverweiterung aktiv wird, mussten die Netzwerkgeräte zusätzlich inventarisiert und bei LANCOM auf der Webseite aktiviert werden.

3 Analysephase

3.1 Kundengespräch

Zu Beginn des Projekts wurde ein Kundengespräch per Telefonkonferenz durchgeführt. Anwesend war der Leiter der IT-Abteilung der Schule, ein weiterer Mitarbeiter der Arktis, und ich. In diesem Gespräch wurden die Anforderungen und der aktuelle Zustand des Schulnetzwerkes konkretisiert und dokumentiert. Im Anschluss überreichte der Kunde uns die Ergebnisse der Funkausleuchtung, welche vor dem Beginn des Projektes abgeschlossen wurde. Bei der Funkausleuchtung wurden die idealen Montagepunkte für die Access Points und die benötigte Anzahl der Access Points ermittelt, indem an verschiedenen möglichen Installationspunkten die Signalstärke gemessen wurde, um sicherzustellen dass von jedem Teil des Schulgebäudes ein störungsfreier Netzwerkzugang möglich ist. Mit Hilfe der zusätzlichen Informationen konnte ein Ist- und Soll-Konzept im Anschluss an das Gespräch ausgearbeitet werden.

3.2 Ist-Analyse

Die Schule hat für ihre festen Arbeitsplätze ein kabelgebundenes Netzwerk mit Zugang zum Internet über deren Internet Provider. Die Leitung vom Internet Provider kommt in einem Serverraum im 1. Untergeschoss (UG) an und geht dort auf einen Router Very High Speed Digital Subscriber Line (VDSL)-Router, welcher als Gateway zum Internet agiert. Von der Local Area Network (LAN)-Schnittstelle des Routers ist der Router per Kupferkabel mit einem Switch verbunden, welcher über ein Patchfeld mit den festen Arbeitsplätzen verbunden ist (siehe NetzwerkplanIST). Auf dem Switch gibt es keine Virtual Local Area (VLAN)

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

3.3 Wirtschaftlichkeitsanalyse

- Lohnt sich das Projekt für das Unternehmen?

3.3.1 „Make or Buy“-Entscheidung

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

3.3.2 Projektkosten

Die Projektkosten für die Durchführung des Projektes setzen sich aus den Personal- und den Ressourcenkosten zusammen.

3 Analysephase

3.3.3 Personalkosten

Laut Arbeitsvertrag verdient ein Auszubildener bei der Arktis GmbH im dritten Lehrjahr pro Monat 1000 € brutto. Wenn man den errechneten Stundensatz von 8,25 € mal die Durchführungszeit von 40 Stunden nimmt, kommt man auf 330,13 € Gesamtpersonalkosten für meinen Arbeitsaufwand. Die komplette Rechnung der Personalkosten für meine Arbeit befindet sich im Anhang [A.2: Personalkosten](#) auf Seite [iii](#). Der Mitarbeiter der mit mir die Installation vor Ort durchgeführt hat, wurde mit einem Stundensatz von 25 € die Stunde berechnet.

3.3.4 Amortisationsdauer

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

Beispielrechnung (verkürzt) Bei einer Zeiteinsparung von 10 Minuten am Tag für jeden der 25 Anwender und 220 Arbeitstagen im Jahr ergibt sich eine gesamte Zeiteinsparung von

$$25 \cdot 220 \text{ Tage/Jahr} \cdot 10 \text{ min/Tag} = 55000 \text{ min/Jahr} \approx 917 \text{ h/Jahr} \quad (1)$$

Dadurch ergibt sich eine jährliche Einsparung von

$$917 \text{ h} \cdot (25 + 15) \text{ €/h} = 36680 \text{ €} \quad (2)$$

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

3.4 Nutzwertanalyse

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

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

3.5 Anwendungsfälle

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

4 Entwurfsphase

Beispiel Ein Beispiel für ein Use Case-Diagramm findet sich im Anhang [A.3: Use Case-Diagramm](#) auf Seite [iii](#).

3.6 Qualitätsanforderungen

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

3.7 Schutzbedarf

- Welcher Schutzbedarf wird an die Anwendung gestellt (z. B. hinsichtlich Sicherheit, Wichtigkeit, ... etc. (siehe ?))?

3.8 Schutzmaßnahmen

- Welche Schutzmaßnahmen werden unternommen um das System abzusichern (z. B. gegenüber fremden Zugriff, Sicherheitslücken, Updates, Ausfall des Systems etc.)?

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. Model View Controller ([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 [2](#) wurde für die Implementierung der Anwendung das Hypertext Preprocessor ([PHP](#))-Framework [Symfony](#)¹ ausgewählt.

¹Vgl. ja

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 2: Entscheidungsmatrix

4.3 Entwurf der Benutzeroberfläche

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

Beispiel Beispielentwürfe finden sich im Anhang [A.6: Oberflächenentwürfe](#) auf Seite [vii](#).

4.4 Datenmodell

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

Beispiel In [Abbildung 1](#) wird ein **ERM!** (**ERM!**) dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

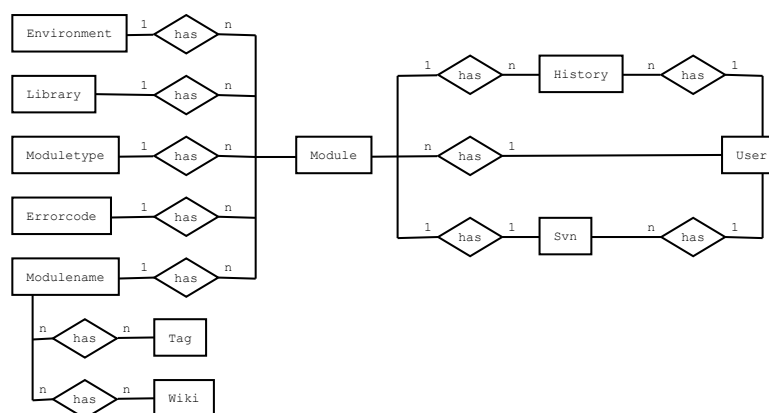


Abbildung 1: Vereinfachtes ER-Modell

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

[Abbildung 2](#) zeigt den grundsätzlichen Programmablauf beim Einlesen eines Moduls als **EPK!**.

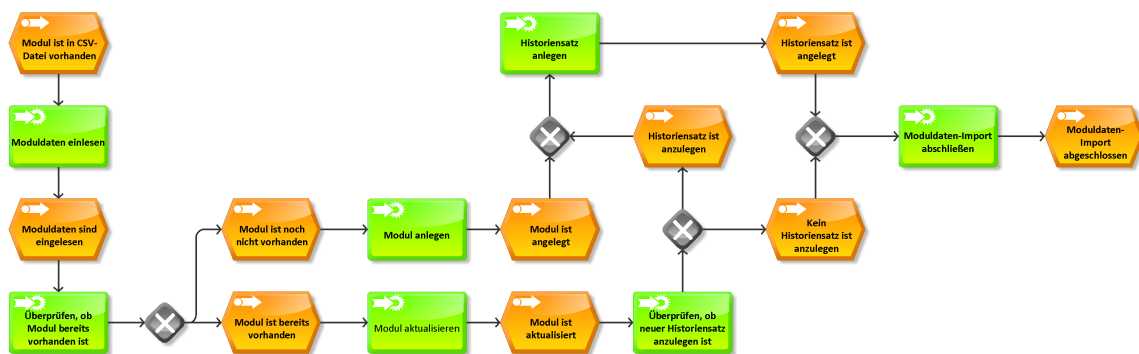


Abbildung 2: Prozess des Einlesens eines Moduls

4.6 Maßnahmen zur Qualitätssicherung

- Welche Maßnahmen werden ergriffen, um die Qualität des Projektergebnisses (siehe Kapitel [3.6: 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.

5 Implementierungsphase

5.1 Implementierung der Datenstrukturen

- Beschreibung der angelegten Datenbank (z. B. Generierung von Structured Query Language ([SQL](#)) aus Modellierungswerkzeug oder händisches Anlegen), Extensible Markup Language ([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!** ([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 [ix](#).

5.3 Implementierung der Geschäftslogik

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

Beispiel Die Klasse `ComparedNaturalModuleInformation` findet sich im Anhang [A.10: Klasse: ComparedNaturalModuleInformation](#) auf Seite [xiv](#).

6 Abnahmephase

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

Beispiel Ein Auszug eines Unit Tests befindet sich im Anhang [A.9: Testfall und sein Aufruf auf der Konsole](#) auf Seite [xiii](#). 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!** (**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 [xix](#). Die Entwicklerdokumentation wurde mittels PHPDoc² automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang [A.8: Entwicklerdokumentation](#) auf Seite [xi](#).

9 Fazit

9.1 Soll-/Ist-Vergleich

- Projektziel erreicht?
- 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?

²Vgl. PHPDOC.ORG [2010]

9 Fazit

- 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 3 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 3: 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

Planungsphase	14 h
1. Analyse des Ist-Zustands	2 h
1.1. Fachgespräch mit der IT-Abteilung vom Kunden	1 h
1.2. Analyse und Zusammenfassung der Mitschriften	1 h
2. Ausarbeitung eines Soll-Konzepts	2 h
3. Auswahl der Hardware entsprechend den Anforderungen des Kunden	4 h
4. Erstellen der Netzwerkpläne	2 h
5. Prüfung der Wirtschaftlichkeit	2 h
6. Ausarbeitung eines Sicherheitskonzepts	2 h
Implementierungsphase	9 h
1. Einrichtung eines Testaufbaus und Installieren von Firmwareupdates	1 h
2. Konfiguration der Switches	2 h
3. Konfiguration des WLAN Controllers	4 h
4. Aktivierung der Garantieerweiterungen und Lizenzen	1 h
5. Konfiguration der Access Points	1 h
Deploymentphase	9 h
1. Montage der Netzwerkkomponenten	3 h
2. Qualitätssicherung	3 h
3. Funktionstests mit Kunden durchführen	3 h
Projektabschluss	8 h
1. Soll-Ist-Vergleich	1 h
2. Erstellen der Projektdokumentation	6 h
3. Reflexion und Ausblick	1 h
Gesamt	40 h

A.2 Personalkosten

Personalkosten für Marcel Akremi	
Brutto pro Monat	1000 €
Arbeitgeberanteile	
Krankenversicherung (7,3%)	73,00
Rentenversicherung (9,3%)	93,00€
Arbeitslosenversicherung (1,3%)	13,00€
Unfallversicherung (1,6%)	16,00€
Pflegeversicherung (1,525%)	15,25€
Gesamtpersonalkosten pro Monat	1210,25€
Arbeitszeit	
Stunden pro Tag	8 h
Tage pro Jahr	220
Tage pro Monat	18,33
Gesamtarbeitszeit pro Monat	146,64 h
Stundensatz	
Gesamtpersonalkosten pro Monat	1210,25€
Gesamtarbeitszeit pro Monat	146, h
Stundensatz	8,25 €
Personalkosten des Projektes	
Stundensatz	8,25 €
Arbeitszeit	40 h
Gesamtpersonalkosten	330,13 €
Personalkosten für Marcel Akremi	330,13 €

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>.

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

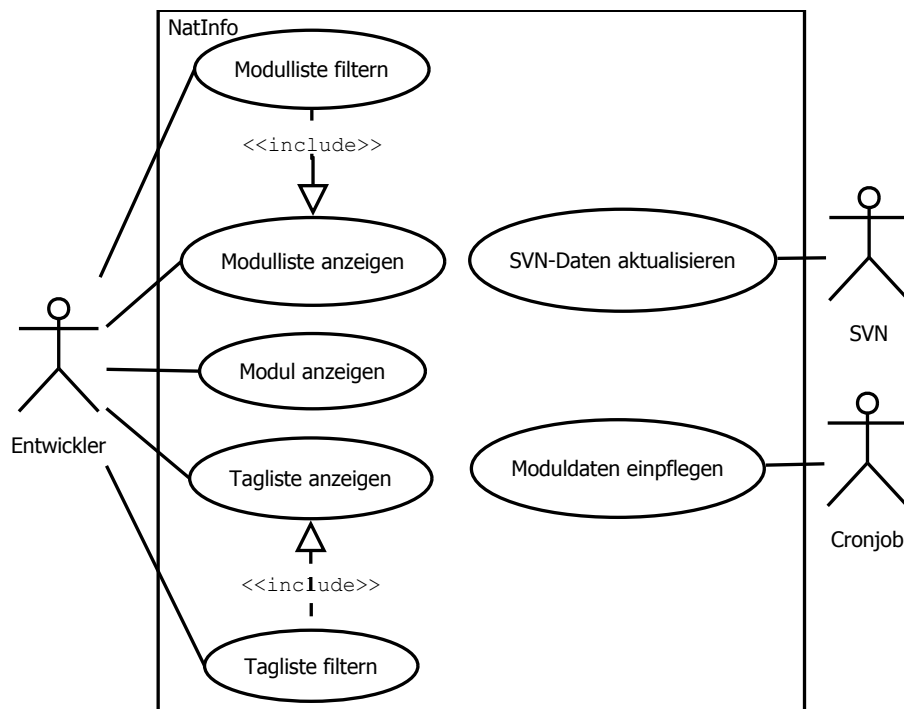


Abbildung 3: Use Case-Diagramm

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! (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.

A Anhang

- Die Datei enthält die Namen der Umgebung, der Bibliothek und des Moduls, den Programmtyp, den Benutzer und Zeitpunkt des Sourcecodes sowie des Kompilats und den Hash des Sourcecodes.
- Sollte sich ein Modul verändert haben, werden die entsprechenden Daten in der Datenbank aktualisiert. Die Veränderungen am Source werden dabei aber nicht ersetzt, sondern historisiert.

1.4. Import der Informationen aus Subversion (SVN). Durch einen „post-commit-hook“ wird nach jedem Einchecken eines Moduls ein PHP-Script auf der Konsole aufgerufen, welches die Informationen, die vom SVN-Kommandozeilentool geliefert werden, an Natural Information System (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 Programmiersprache der Software AG (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 L^AT_EX zeichnen, siehe z. B. <http://www.texample.net/tikz/examples/entity-relationship-diagram/>.

A Anhang

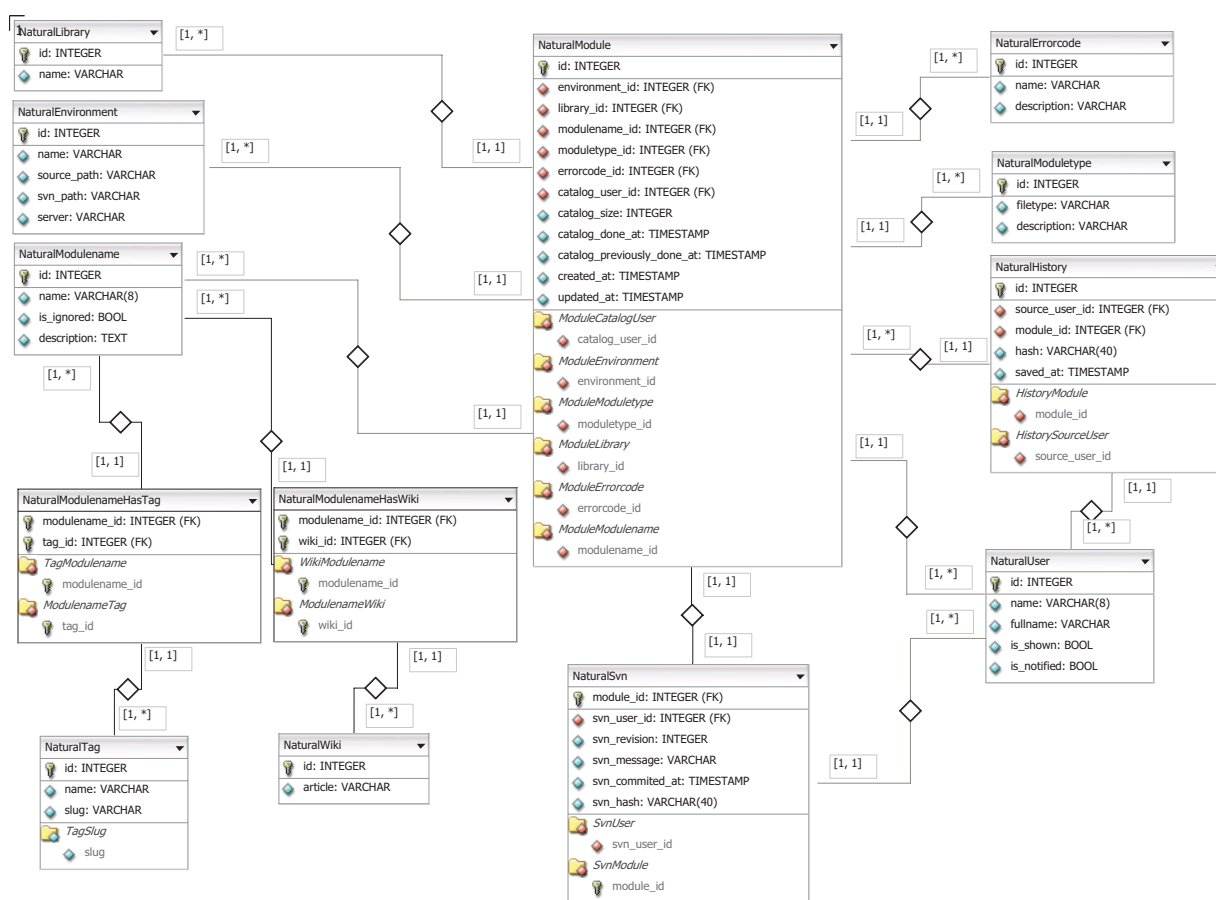


Abbildung 4: Datenbankmodell

A.6 Oberflächenentwürfe

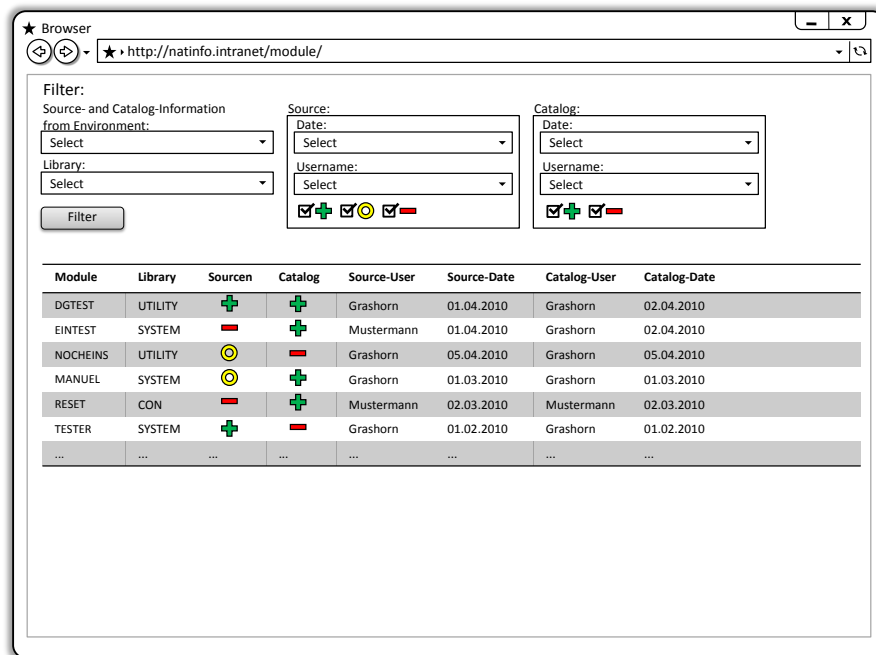


Abbildung 5: Liste der Module mit Filtermöglichkeiten

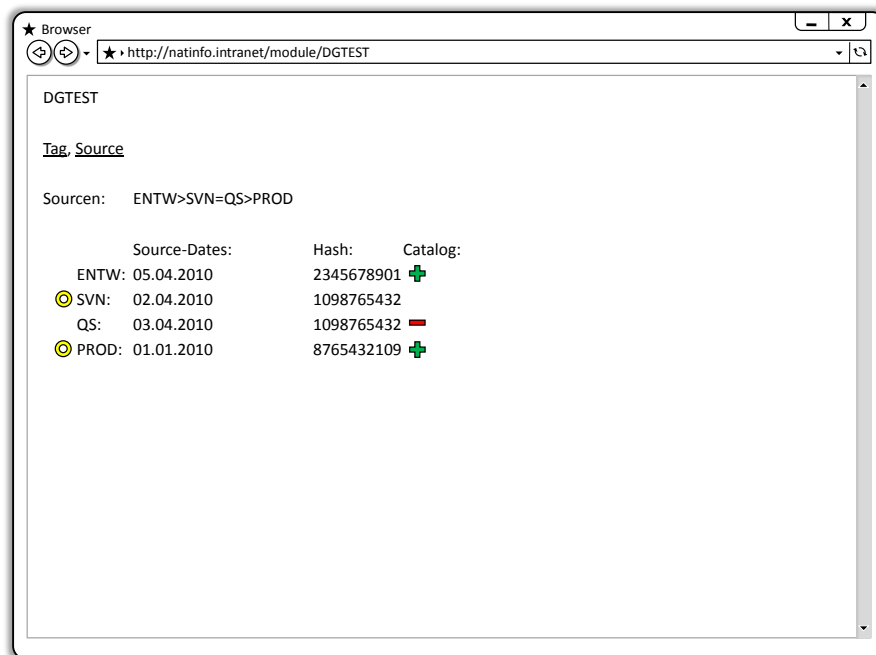


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

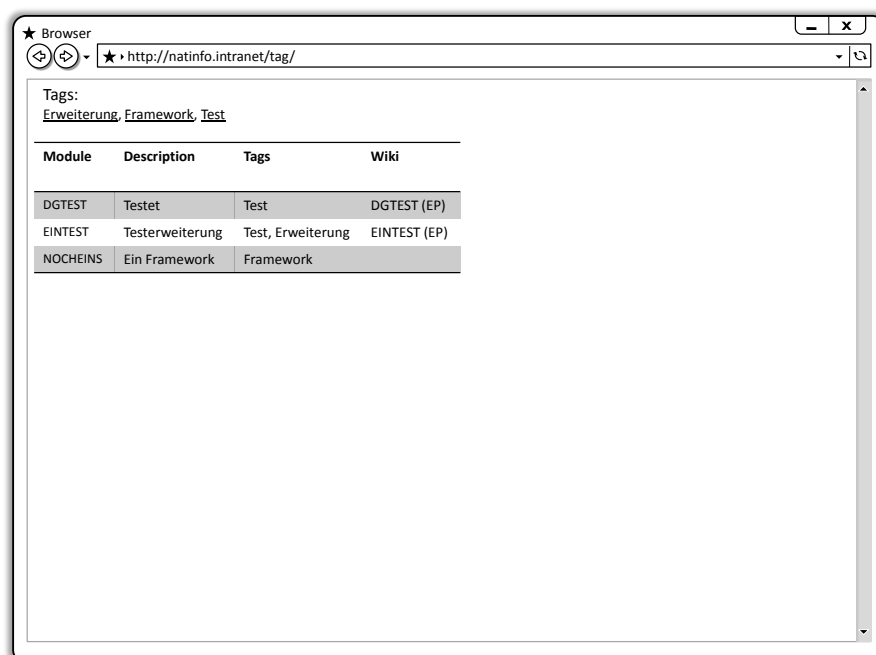


Abbildung 7: Anzeige und Filterung der Module nach Tags

A.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

Listing 1: Testfall in PHP

```
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(NaturalmodulenamePeer::NAME, 'SMTAB');
14 $moduleName = NaturalmodulenamePeer::doSelectOne($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 ?>
```

Abbildung 10: Aufruf des Testfalls auf der Konsole

A.10 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

Listing 2: Klasse: ComparedNaturalModuleInformation

```
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,
21                     self::SIGN_CREATE, self::SIGN_OK);
22     }
23
24     public function __construct(array $naturalInformations)
```

A Anhang

```
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())] =
38                 $naturalInformation;
39         }
40     }
41 }
42
43 private function allocateEmptyModulesToMissingEnvironments()
44 {
45     if(array_key_exists(0, $this->naturalModuleInformations))
46     {
47         $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
48     }
49
50     for($i = 0; $i < count(self::environments()); $i++)
51     {
52         if(!array_key_exists($i, $this->naturalModuleInformations))
53         {
54             $environments = self::environments();
55             $this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation(
56                 $environments[$i]);
57             $this->naturalModuleInformations[$i]->setSourceSign(self::SIGN_CREATE);
58         }
59     }
60 }
61
62 public function determineSourceSignsForAllEnvironments()
63 {
64     for($i = 1; $i < count(self::environments()); $i++)
65     {
66         $currentInformation = $this->naturalModuleInformations[$i];
67         $previousInformation = $this->naturalModuleInformations[$i - 1];
68         if($currentInformation->getSourceSign() <> self::SIGN_CREATE)
69         {
70             if($previousInformation->getSourceSign() <> self::SIGN_CREATE)
71             {
72                 if($currentInformation->getHash() <> $previousInformation->getHash())
73                 {
74                     if($currentInformation->getSourceDate('YmdHis') > $previousInformation->
75                         getSourceDate('YmdHis'))
```

A Anhang

```
73     {
74         $currentInformation->setSourceSign(self::SIGN_ERROR);
75     }
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 &&
92         $previousInformation->getSourceSign() <> self::SIGN_CREATE_AND_NEXT_STEP)
93 {
94     $currentInformation->setSourceSign(self::SIGN_CREATE_AND_NEXT_STEP);
95 }
96 }
97
98 private function containsSourceSign($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>.

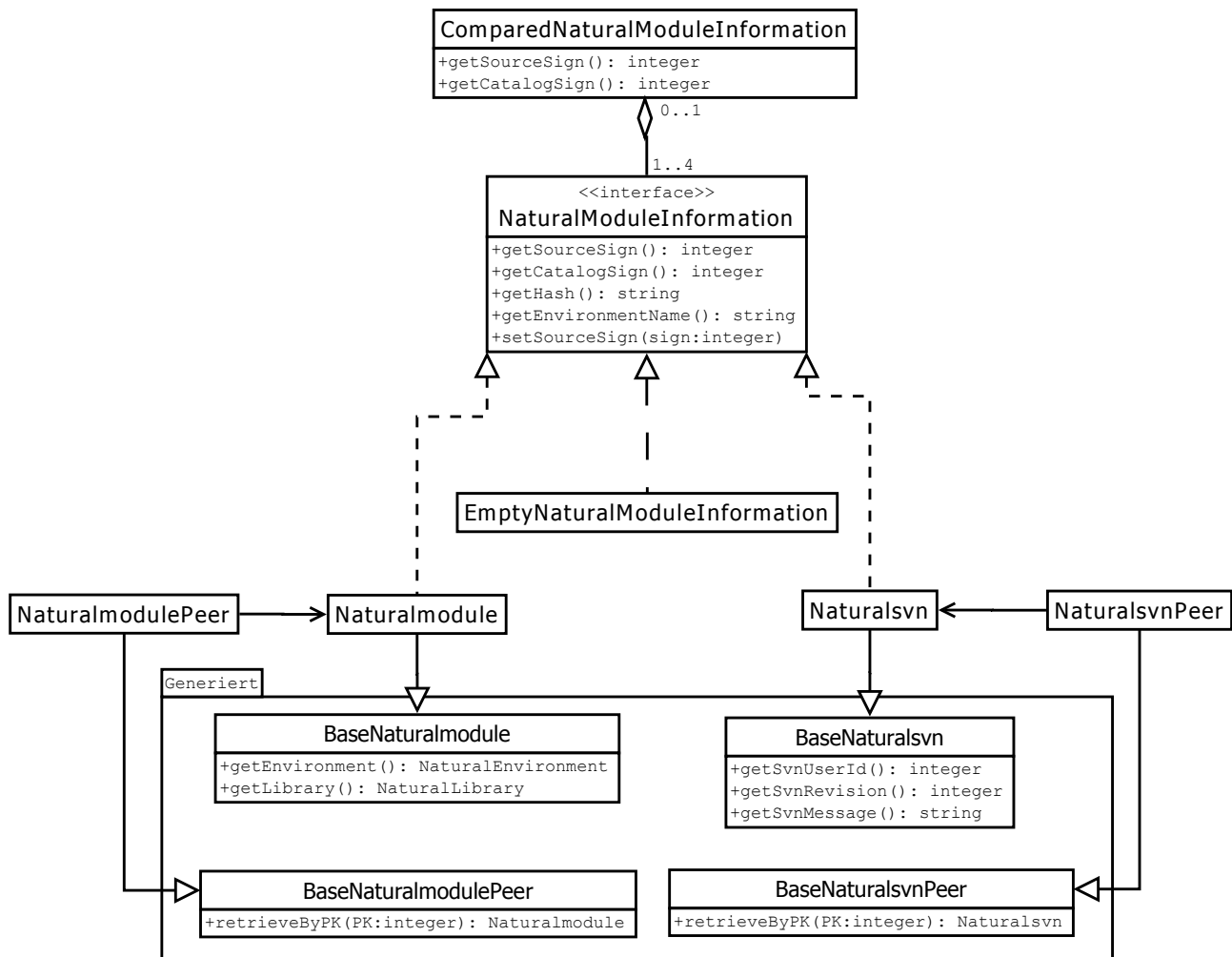







Abbildung 11: Klassendiagramm

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.