

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 2024

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Entwicklung eines Ticketsystems

QR-Code basiertes Verifizierungs- und Entwertungssystem für
Online-Bestellungen

Abgabetermin: Rostock, den 24.05.2024

Prüfungsbewerber:

Jannick Bath
Schweriner Straße 26
18069 Rostock



Ausbildungsbetrieb:

LUPCOM media GmbH
Rahnstädter Weg 33
18069 Rostock

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Listings	V
Abkürzungsverzeichnis	VI
1 Einleitung	1
1.1 Projektumfeld	1
1.1.1 Ausbildungsbetrieb	1
1.1.2 Kunde	1
1.2 Projektziel	1
1.3 Projektbegründung	2
1.3.1 Aktuelles Problem	2
1.3.2 Vorteile des QR-Code-basierten Systems	2
1.3.3 Vorteile für die Mitarbeiter	2
1.4 Projektschnittstellen	3
1.4.1 Technische Schnittstellen	3
1.4.2 Präsentation der Ergebnisse	3
1.5 Projektabgrenzung	3
2 Projektplanung	4
2.1 Projektphasen	4
2.2 Abweichungen vom Projektantrag	4
2.3 Ressourcenplanung	5
2.4 Entwicklungsprozess	5
3 Analysephase	6
3.1 Ist-Analyse	6
3.2 Wirtschaftlichkeitsanalyse	6
3.2.1 „Make or Buy“-Entscheidung	6
3.2.2 Projektkosten	6
3.2.3 Amortisationsdauer	7
3.3 Nutzwertanalyse	7
3.4 Anwendungsfälle	8
3.5 Qualitätsanforderungen	8
3.6 Lastenheft/Fachkonzept	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
5	Implementierungsphase	11
5.1	Implementierung der Datenstrukturen	11
5.2	Implementierung der Benutzeroberfläche	11
5.3	Implementierung der Geschäftslogik	11
6	Abnahmephase	12
7	Einführungsphase	12
8	Dokumentation	12
9	Fazit	13
9.1	Soll-/Ist-Vergleich	13
9.2	Lessons Learned	13
9.3	Ausblick	13
	Eidesstattliche Erklärung	14
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Übersicht Ressourcenplanung	ii
A.3	Lastenheft (Auszug)	ii
A.4	Use Case-Diagramm	iv
A.5	Pflichtenheft (Auszug)	iv
A.6	Datenbankmodell	vi
A.7	Oberflächenentwürfe	vii
A.8	Screenshots der Anwendung	ix
A.9	Entwicklerdokumentation	xi
A.10	Testfall und sein Aufruf auf der Konsole	xiii
A.11	Klasse: ComparedNaturalModuleInformation	xiv
A.12	Klassendiagramm	xvii
A.13	Benutzerdokumentation	xviii

Abbildungsverzeichnis

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

Tabellenverzeichnis

1	Zeitplanung	4
2	Kostenaufstellung	7
3	Entscheidungsmatrix	9
4	Soll-/Ist-Vergleich	13

Listings

1	Testfall in PHP	xiii
2	Klasse: ComparedNaturalModuleInformation	xiv

Abkürzungsverzeichnis

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

1 Einleitung

1.1 Projektumfeld

1.1.1 Ausbildungsbetrieb

Die **LUPCOM media GmbH** ist eine Internetagentur mit ihrem Hauptsitz in Rostock, die sich darauf spezialisiert hat, Webseiten und Webanwendungen für eine vielfältige Kundschaft zu entwerfen und zu entwickeln. Ihr Fokus liegt darauf, maßgeschneiderte Lösungen zu bieten, die den individuellen Anforderungen und Größenordnungen ihrer Kunden entsprechen. Dabei deckt sie ein breites Spektrum von Projekten ab, von kleinen Unternehmenswebseiten bis hin zu komplexen Webanwendungen. Für die Umsetzung dieser Projekte werden Technologien wie Contao, Symfony, JavaScript sowie Docker verwendet.

1.1.2 Kunde

Der Auftraggeber dieses Projekts ist die **MS 'Ostseebad Rerik'**, ein Ausflugsschiff, das entlang der „verbotenen“ Halbinsel Wustrow fährt. Das Schiff bietet mehrmals täglich in der Hauptsaison Rundfahrten auf dem Salzhaff an, die etwa zwei Stunden dauern und unter kundiger Führung des Kapitäns stattfinden. Zusätzlich bietet die MS 'Ostseebad Rerik' gastronomische Versorgung sowie Charterfahrten für Jubiläen, Betriebsfeiern und Vereinsausflüge an.

1.2 Projektziel

Ziel dieses Projekts ist die Entwicklung eines zuverlässigen und effizienten QR-Code-basierten Einlösungsprogramms. Dieses Programm soll den Einlösungsprozess digitalisieren und automatisieren. Konkret umfasst das Projektziel die folgenden Punkte:

- **Generierung einzigartiger QR-Codes für spezifische Bestellungen:** Für jede Bestellung soll ein individueller QR-Code generiert werden, der durch einen Mitarbeiter eingescannt und entwertet werden kann.
- **Sichere und effiziente Einlösung durch Scannen der QR-Codes:** Die QR-Codes sollen von den Mitarbeitern gescannt werden können, um die Echtheit und Gültigkeit der Tickets oder Gutscheine sofort zu überprüfen.
- **Intuitive Benutzeroberfläche für die Überprüfung der Bestellinformationen und die Entwertung der Tickets/Gutscheine:** Das System soll eine benutzerfreundliche Oberfläche bieten, die es den Mitarbeitern ermöglicht, die Bestellinformationen schnell und einfach einzusehen und die Tickets oder Gutscheine nach der Einlösung zu entwerten.

- **Einfache Installation mittels composer:** Die Anwendung muss mit dem Paketmanager composer kompatibel sein, um eine einfache Installation zu gewährleisten.

Durch die Erfüllung dieser Ziele wird das Projekt erfolgreich abgeschlossen.

1.3 Projektbegründung

Die Durchführung dieses Projekts soll mehrere bestehende Probleme im aktuellen Einlösungsprozess beheben und sowohl die Effizienz als auch die Zufriedenheit der Mitarbeiter steigern.

1.3.1 Aktuelles Problem

Der aktuelle Prozess zur Überprüfung und Einlösung von online erworbenen Tickets und Gutscheinen basiert auf manuellen Methoden. Diese manuelle Verifizierung ist zeitaufwendig und fehleranfällig, was zu einer ineffizienten Nutzung der Arbeitszeit führt und die Kundenzufriedenheit negativ beeinflusst.

1.3.2 Vorteile des QR-Code-basierten Systems

Die Entwicklung eines QR-Code-basierten Einlösungsprogramms bietet zahlreiche Vorteile:

- **Reduktion von Fehlern:** Durch die Automatisierung des Verifizierungsprozesses werden menschliche Fehler minimiert, was die Zuverlässigkeit der Ticket- und Gutscheinüberprüfung erhöht.
- **Zeitersparnis:** Mitarbeiter können QR-Codes schnell scannen und sofortige Rückmeldungen über die Gültigkeit der Tickets erhalten, was den gesamten Prozess beschleunigt.
- **Erhöhte Sicherheit:** Das System gewährleistet eine sichere Einlösung von Tickets und Gutscheinen, wodurch Betrugsversuche reduziert werden.
- **Verbesserte Kundenzufriedenheit:** Ein effizienter und schneller Einlösungsprozess steigert das Kundenerlebnis und das Vertrauen in die digitalen Dienstleistungen.

1.3.3 Vorteile für die Mitarbeiter

Die Einführung eines QR-Code-basierten Einlösungsprogramms soll die Mitarbeiter vor Ort entlasten:

- **Erleichterung der Arbeitsabläufe:** Die Automatisierung reduziert den manuellen Aufwand und die Komplexität der Aufgaben, was die Arbeitslast der Mitarbeiter verringert.
- **Steigerung der Produktivität:** Mitarbeiter können ihre Zeit effizienter nutzen und sich auf andere wichtige Aufgaben konzentrieren.

- **Verbesserte Arbeitsqualität:** Durch die Minimierung von Fehlern und die Vereinfachung der Prozesse wird die Qualität der Arbeit verbessert, was zu höherer Arbeitszufriedenheit führt.

Insgesamt soll die Implementierung eines QR-Code-basierten Einlösungsprogramms zu einer Verbesserung der Effizienz und Qualität der Arbeitsabläufe sowie zu einer gesteigerten Zufriedenheit der Mitarbeiter und Kunden führen.

1.4 Projektschnittstellen

Die Anwendung soll verschiedene Schnittstellen bereitstellen, um eine nahtlose Integration und effiziente Nutzung zu gewährleisten. Im Folgenden werden die geplanten Schnittstellen beschrieben.

1.4.1 Technische Schnittstellen

- **Interaktion mit Shop-Systemen:** Die Anwendung soll sich nahtlos in die bestehenden Shop-Systeme integrieren. Hierbei soll für jede Bestellung ein einzigartiger QR-Code generiert werden, der in den Bestelldaten gespeichert und den Kunden zur Verfügung gestellt wird.
- **API-Schnittstellen:** Die Anwendung soll mehrere API-Endpunkte bereitstellen, um die Interaktion mit externen Systemen und die Verarbeitung der QR-Codes zu ermöglichen. Dies umfasst Endpunkte für die Validierung und Entwertung der QR-Codes sowie die Bereitstellung der entsprechenden Bestelldetails.

1.4.2 Präsentation der Ergebnisse

- **Benutzeroberfläche:** Die Ergebnisse der QR-Code-Verifizierung und Einlösung sollen über eine benutzerfreundliche Oberfläche präsentiert werden.
- **Darstellung für Mitarbeiter:** Die Bestellinformationen sollen den Mitarbeitern in einem klaren und strukturierten Format präsentiert werden, um eine schnelle und effiziente Überprüfung und Entwertung der Tickets zu ermöglichen. Dies umfasst die Anzeige der Bestelldetails wie Bestellnummer, Datum und Status sowie den Entwertungsstatus.

1.5 Projektabgrenzung

Die Entwicklung und Konfiguration des Onlineshops sind nicht Bestandteil dieses Projekts. Alle Arbeiten, die mit der Einrichtung, Verwaltung und Anpassung des Onlineshops selbst zu tun haben, fallen somit außerhalb des Projektumfangs.

Ebenso gehört die Erstellung der Designs für das Einlösetool nicht zum Projektumfang. Die grafischen Entwürfe und Layouts werden durch die Design-Abteilung bereitgestellt.

2 Projektplanung

2.1 Projektphasen

Für die Umsetzung des Projekts standen insgesamt 80 Stunden zur Verfügung, die vor Beginn auf verschiedene Phasen der Softwareentwicklung verteilt wurden. Die Projektarbeit erstreckte sich über zwei 4-Tage-Wochen, ergänzt durch zwei zusätzliche Arbeitstage, um Feiertage auszugleichen. Die Durchführung erfolgte vom 25.04.2024 bis 26.04.2024, sowie vom 29.04.2024 bis 03.05.2024 und vom 20.05.2024 bis 24.05.2024, jeweils mit einer täglichen Arbeitszeit von 8 Stunden.

Eine Übersicht der groben Zeitplanung und der Hauptphasen findet sich in Tabelle 1: Grobe Zeitplanung. Diese Hauptphasen sind weiter in detaillierte Unterabschnitte gegliedert, um eine präzise Planung zu gewährleisten. Eine detaillierte Darstellung der Phasen ist im Anhang A.1: Detaillierte Zeitplanung auf Seite i zu finden.

Tabelle 1 zeigt die grobe Zeitplanung.

Projektphase	Geplante Zeit
Analysephase	5 h
Planungsphase	10 h
Implementierungsphase	40 h
Erstellen der Dokumentation	20 h
Erstellen der Benutzerdokumentation	2.5 h
Abnahme der Fachabteilung	2.5 h
Gesamt	80 h

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang A.1: Detaillierte Zeitplanung auf Seite i.

2.2 Abweichungen vom Projektantrag

Zunächst war in der Zeitplanung des Projektantrags die Erstellung eines Abnahmeprotokolls vorgesehen. Dieses Protokoll wurde jedoch nicht erstellt. Stattdessen erfolgte die Übergabe der Projektergebnisse direkt an die zuständige Abteilung, die die Abnahme informell durch ein einfaches Abnicken bestätigte.

Des Weiteren waren im Projektantrag Unit-Tests eingeplant, um die Qualität des Codes sicherzustellen. Aufgrund von Zeitmangel konnten diese Tests jedoch nicht durchgeführt werden. Stattdessen wurden manuelle Tests durchgeführt, bei denen die Arbeitsabläufe durch manuelles Durchklicken überprüft wurden.

Eine weitere Abweichung betraf die Erstellung der Entwürfe. Im Projektantrag war vorgesehen, dass diese aufgrund ihrer erwarteten geringen Anzahl vom Autor erstellt werden. Allerdings stellte sich während des Projekts heraus, dass die Anzahl der erforderlichen Benutzeroberflächen umfangreicher war als angenommen. Daher wurde beschlossen, die Entwürfe an die Design-Abteilung zu übergeben. Lediglich die Ticketansicht und die Popups wurden vom Autor entworfen.

Auch die Benutzerdokumentation fiel kleiner aus als ursprünglich geplant. Dies lag daran, dass die Abnahme durch die Fachabteilung mehr Zeit in Anspruch nahm als vorgesehen, was die Zeit für die Erstellung der Dokumentation verkürzte.

2.3 Ressourcenplanung

Die Entwicklung des Projekts wurde auf einem Desktop-Rechner mit dem Betriebssystem Ubuntu durchgeführt. Visual Studio Code (VSCode) diente als Hauptwerkzeug für die Code-Entwicklung, während Git für die Versionskontrolle und GitLab zur Sicherung und Verwaltung des Projektstands verwendet wurden.

Das Open Source Contao CMS wurde als Basis für die Entwicklung der Anwendung verwendet. Zur Integration in die Online-Shops diente das Contao-Plugin Isotope. Die Anwendung wurde in verschiedenen Browsern wie Chrome, Safari und Firefox getestet, wobei auch die Google Page-Speed Tools zur Optimierung der Leistungsfähigkeit und Ladezeiten zum Einsatz kamen.

Die personellen Ressourcen setzten sich aus dem Prüfling als Entwickler, dem Geschäftsführer als Projektverantwortlichem und einem Designer für die grafischen Entwürfe zusammen.

Anhang A.2: Übersicht Ressourcenplanung auf Seite ii

2.4 Entwicklungsprozess

Vor der Durchführung des Projekts wurde ein spezifisches Vorgehensmodell ausgewählt, das die Struktur und den Ablauf der Entwicklung vorgab. Für dieses Projekt entschied sich der Autor für das Wasserfallmodell, da es in Bezug auf die Komplexität und den Umfang des Projekts als am besten geeignet erschien.

Das Wasserfallmodell zeichnet sich durch seine lineare Herangehensweise aus, bei der jede Phase vollständig abgeschlossen sein muss, bevor die nächste beginnt. Dies ermöglichte eine klare und strukturierte Planung sowie eine präzise Definition der einzelnen Projektphasen.

Ein weiterer Punkt, der zu dieser Entscheidung führte, war die Tatsache, dass die Entwicklung hauptsächlich von einer Person, dem Prüfling, durchgeführt wurde. Dadurch konnte das Wasserfallmodell seine Stärken ausspielen, da es eine klare und übersichtliche Projektstruktur bietet, die besonders bei Einzelentwicklungen von Vorteil ist.

Die einzelnen Phasen des Wasserfallmodells, wie Anforderungsanalyse, Entwurf, Implementierung, Testen und Wartung, wurden nacheinander durchlaufen, um eine systematische und gründliche Entwicklung zu gewährleisten. Diese Methodik erlaubte es, den Projektverlauf genau zu planen und mögliche Risiken frühzeitig zu identifizieren und zu minimieren.

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

- Lohnt sich das Projekt für das Unternehmen?

3.2.1 „Make or Buy“-Entscheidung

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

3.2.2 Projektkosten

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

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 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 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

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

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

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

Dadurch ergibt sich eine jährliche Einsparung von

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

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

3.3 Nutzwertanalyse

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

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

¹Räumlichkeiten, Arbeitsplatzrechner etc.

3.4 Anwendungsfälle

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

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

3.5 Qualitätsanforderungen

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

3.6 Lastenheft/Fachkonzept

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

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

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 3 wurde für die Implementierung der Anwendung das PHP-Framework Symfony² 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

4.3 Entwurf der Benutzeroberfläche

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

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

4.4 Datenmodell

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

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

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?

²Vgl. ?.

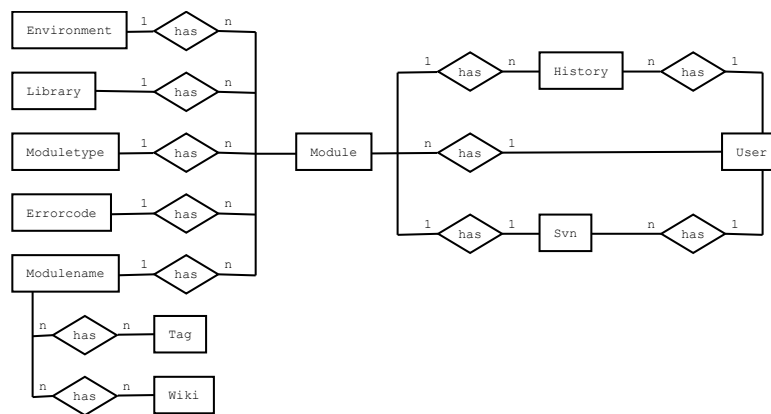


Abbildung 1: Vereinfachtes ER-Modell

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

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

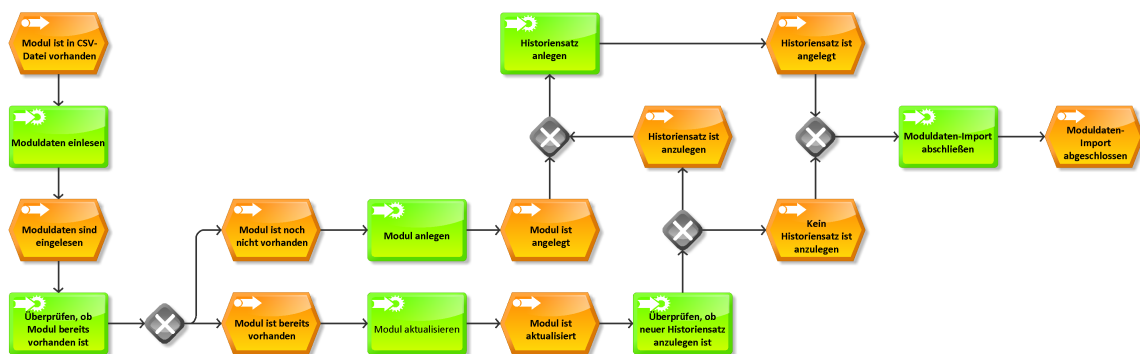


Abbildung 2: Prozess des Einlesens eines Moduls

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.5: Pflichtenheft (Auszug) auf Seite iv zu finden.

5 Implementierungsphase

5.1 Implementierung der Datenstrukturen

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

5.2 Implementierung der Benutzeroberfläche

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

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

5.3 Implementierung der Geschäftslogik

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

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

6 Abnahmephase

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

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

7 Einführungsphase

- Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?
- Wurden ggfs. Altdaten migriert und wenn ja, wie?
- Wurden Benutzerschulungen durchgeführt und wenn ja, Wie wurden sie vorbereitet?

8 Dokumentation

- Wie wurde die Anwendung für die Benutzer/Administratoren/Entwickler dokumentiert (z. B. Benutzerhandbuch, API-Dokumentation)?
- Hinweis: Je nach Zielgruppe gelten bestimmte Anforderungen für die Dokumentation (z. B. keine IT-Fachbegriffe in einer Anwenderdokumentation verwenden, aber auf jeden Fall in einer Dokumentation für den IT-Bereich).

Beispiel Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang A.13: Benutzerdokumentation auf Seite xviii. Die Entwicklerdokumentation wurde mittels PHPDoc³ automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang A.9: Entwicklerdokumentation auf Seite xi.

³Vgl. ?

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 4 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 4: 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)?

Eidesstattliche Erklärung

Ich, Jannick Bath, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

Entwicklung eines Ticketsystems – QR-Code basiertes Verifizierungs- und Entwertungssystem für Online-Bestellungen

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.

Rostock, den 24.05.2024

JANNICK BATH

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 geparsten 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 Übersicht Ressourcenplanung

Kategorie	Details
Hardware	Desktop-Rechner: Rechner mit Ubuntu Betriebssystem für die Entwicklung
Texteditor	Visual Studio Code (VSCode): Hauptwerkzeug für die Code-Entwicklung
Versionsverwaltung	Git: Tool zur Versionskontrolle
Projektmanagement	GitLab: Plattform zur Sicherung und Verwaltung des Projektstands
CMS	Contao CMS: Open Source CMS als Framework
E-Commerce-Integration	Isotope Contao-Plugin: Für die Integration mit Online-Shops
Browser	Chrome, Safari, Firefox: Browser zur Überprüfung und Testung der Anwendung
Test-Tools	Google Page-Speed Tools: Werkzeuge zur Optimierung der Leistungsfähigkeit
Entwickler	Prüfling: Verantwortlich für die Entwicklung und Implementierung
Projektverantwortlicher	Geschäftsführer: Überwachung des Fortschritts und Entscheidungsunterstützung
Designer	Grafische Entwürfe: Erstellung der Designentwürfe für die Benutzeroberflächen

A.3 Lastenheft (Auszug)

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

Die Anwendung muss folgende Anforderungen erfüllen:

1. Verarbeitung der Moduldaten
 - 1.1. Die Anwendung muss die von Subversion und einem externen Programm bereitgestellten Informationen (z.B. Source-Benutzer, -Datum, Hash) verarbeiten.
 - 1.2. Auslesen der Beschreibung und der Stichwörter aus dem Sourcecode.
2. Darstellung der Daten
 - 2.1. Die Anwendung muss eine Liste aller Module erzeugen inkl. Source-Benutzer und -Datum, letztem Commit-Benutzer und -Datum für alle drei Umgebungen.
 - 2.2. Verknüpfen der Module mit externen Tools wie z.B. Wiki-Einträgen zu den Modulen oder dem Sourcecode in Subversion.
 - 2.3. Die Sourcen der Umgebungen müssen verglichen und eine schnelle Übersicht zur Einhaltung des allgemeinen Entwicklungsprozesses gegeben werden.
 - 2.4. Dieser Vergleich muss auf die von einem bestimmten Benutzer bearbeiteten Module eingeschränkt werden können.

- 2.5. Die Anwendung muss in dieser Liste auch Module anzeigen, die nach einer Bearbeitung durch den gesuchten Benutzer durch jemand anderen bearbeitet wurden.
- 2.6. Abweichungen sollen kenntlich gemacht werden.
- 2.7. Anzeigen einer Übersichtsseite für ein Modul mit allen relevanten Informationen zu diesem.
- 3. Sonstige Anforderungen
 - 3.1. Die Anwendung muss ohne das Installieren einer zusätzlichen Software über einen Webbrowser im Intranet erreichbar sein.
 - 3.2. Die Daten der Anwendung müssen jede Nacht bzw. nach jedem SVN-Commit automatisch aktualisiert werden.
 - 3.3. Es muss ermittelt werden, ob Änderungen auf der Produktionsumgebung vorgenommen wurden, die nicht von einer anderen Umgebung kopiert wurden. Diese Modulliste soll als Mahnung per E-Mail an alle Entwickler geschickt werden (Peer Pressure).
 - 3.4. Die Anwendung soll jederzeit erreichbar sein.
 - 3.5. Da sich die Entwickler auf die Anwendung verlassen, muss diese korrekte Daten liefern und darf keinen Interpretationsspielraum lassen.
 - 3.6. Die Anwendung muss so flexibel sein, dass sie bei Änderungen im Entwicklungsprozess einfach angepasst werden kann.

A.4 Use Case-Diagramm

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

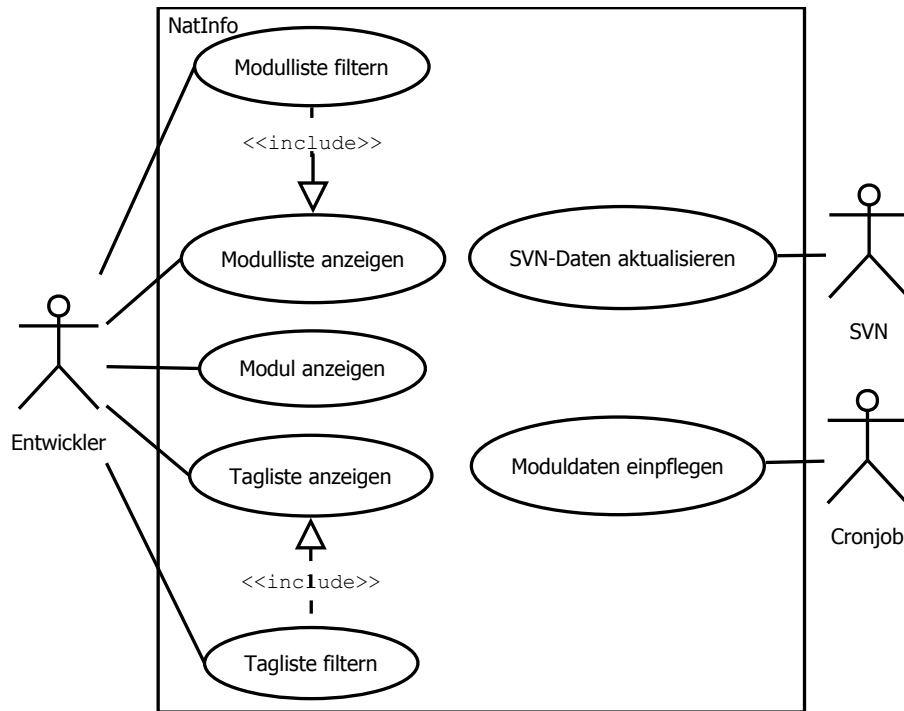


Abbildung 3: Use Case-Diagramm

A.5 Pflichtenheft (Auszug)

Zielbestimmung

1. Musskriterien

1.1. Modul-Liste: Zeigt eine filterbare Liste der Module mit den dazugehörigen Kerninformationen sowie Symbolen zur Einhaltung des Entwicklungsprozesses an

- In der Liste wird der Name, die Bibliothek und Daten zum Source und Kompilat eines Moduls angezeigt.
- Ebenfalls wird der Status des Moduls hinsichtlich Source und Kompilat angezeigt. Dazu gibt es unterschiedliche Status-Zeichen, welche symbolisieren in wie weit der Entwicklungsprozess eingehalten wurde bzw. welche Schritte als nächstes getan werden müssen. So gibt es z. B. Zeichen für das Einhalten oder Verletzen des Prozesses oder den Hinweis auf den nächsten zu tätigenden Schritt.
- Weiterhin werden die Benutzer und Zeitpunkte der aktuellen Version der Sourcen und Kompilate angezeigt. Dazu kann vorher ausgewählt werden, von welcher Umgebung diese Daten gelesen werden sollen.

- Es kann eine Filterung nach allen angezeigten Daten vorgenommen werden. Die Daten zu den Sourcen sind historisiert. Durch die Filterung ist es möglich, auch Module zu finden, die in der Zwischenzeit schon von einem anderen Benutzer editiert wurden.

1.2. Tag-Liste: Bietet die Möglichkeit die Module anhand von Tags zu filtern.

- Es sollen die Tags angezeigt werden, nach denen bereits gefiltert wird und die, die noch der Filterung hinzugefügt werden könnten, ohne dass die Ergebnisliste leer wird.
- Zusätzlich sollen die Module angezeigt werden, die den Filterkriterien entsprechen. Sollten die Filterkriterien leer sein, werden nur die Module angezeigt, welche mit einem Tag versehen sind.

1.3. Import der Moduldaten aus einer bereitgestellten CSV-Datei

- Es wird täglich eine Datei mit den Daten der aktuellen Module erstellt. Diese Datei wird (durch einen Cronjob) automatisch nachts importiert.
- Dabei wird für jedes importierte Modul ein Zeitstempel aktualisiert, damit festgestellt werden kann, wenn ein Modul gelöscht wurde.
- Die Datei enthält die Namen der Umgebung, der Bibliothek und des Moduls, den Programmtyp, den Benutzer und Zeitpunkt des Sourcecodes sowie des Kompilats und den Hash des Sourcecodes.
- Sollte sich ein Modul verändert haben, werden die entsprechenden Daten in der Datenbank aktualisiert. Die Veränderungen am Source werden dabei aber nicht ersetzt, sondern historisiert.

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

1.5. Parsen der Sourcen

- Die Sourcen der Entwicklungsumgebung werden nach Tags, Links zu Artikeln im Wiki und Programmbeschreibungen durchsucht.
- Diese Daten werden dann entsprechend angelegt, aktualisiert oder nicht mehr gesetzte Tags/Wikiartikel entfernt.

1.6. Sonstiges

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

Produkteinsatz

1. Anwendungsbereiche

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

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

2. Zielgruppen

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

3. Betriebsbedingungen

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

A.6 Datenbankmodell

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

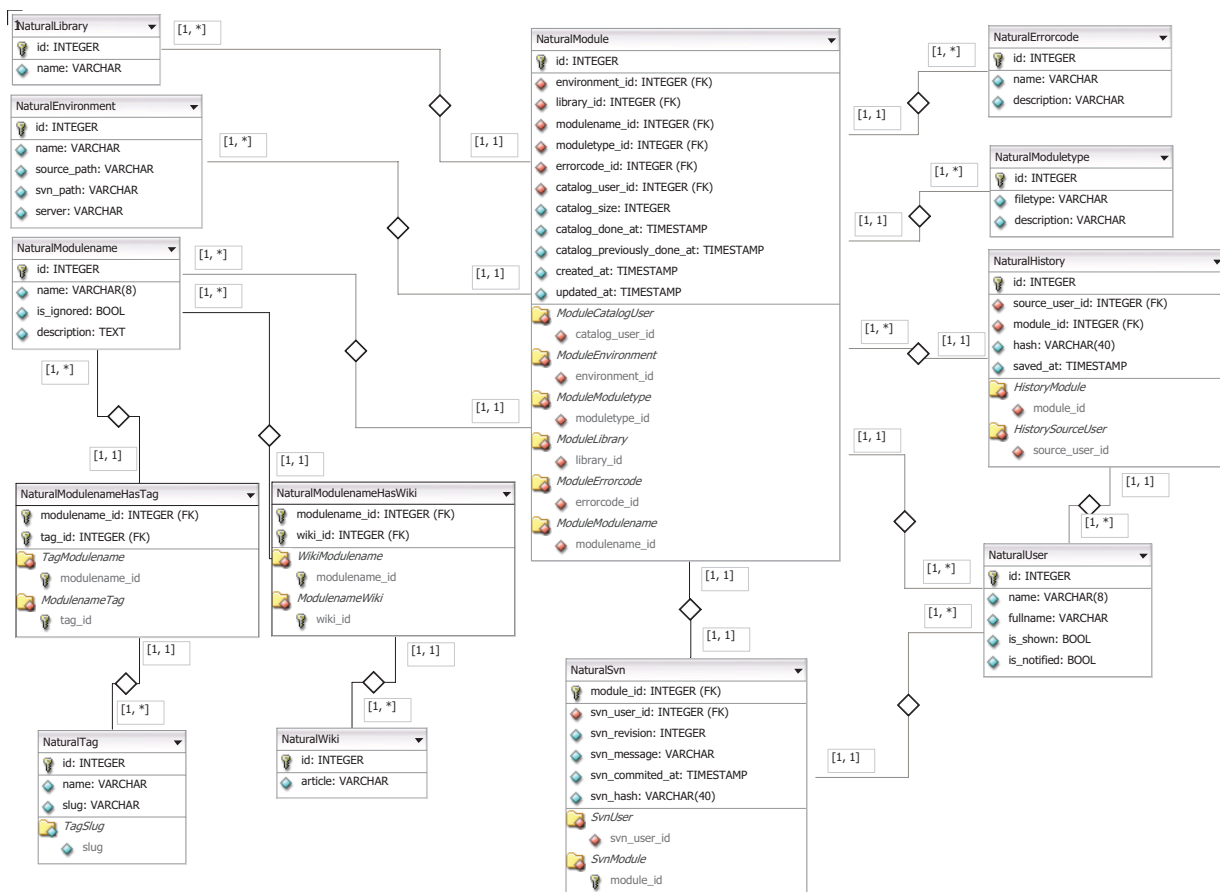


Abbildung 4: Datenbankmodell

A.7 Oberflächenentwürfe

Filter:

Source- and Catalog-Information from Environment:

Library:

Filter

Source:

Date:

Username:

Catalog:

Date:

Username:

Module	Library	Sourcen	Catalog	Source-User	Source-Date	Catalog-User	Catalog-Date
DGTEST	UTILITY	+	+	Grashorn	01.04.2010	Grashorn	02.04.2010
EINTEST	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

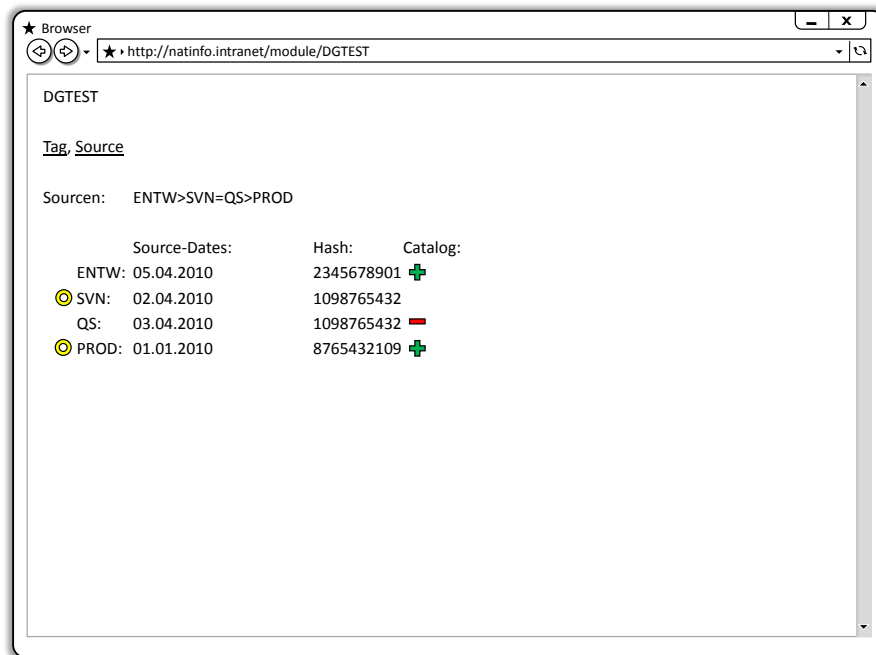


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

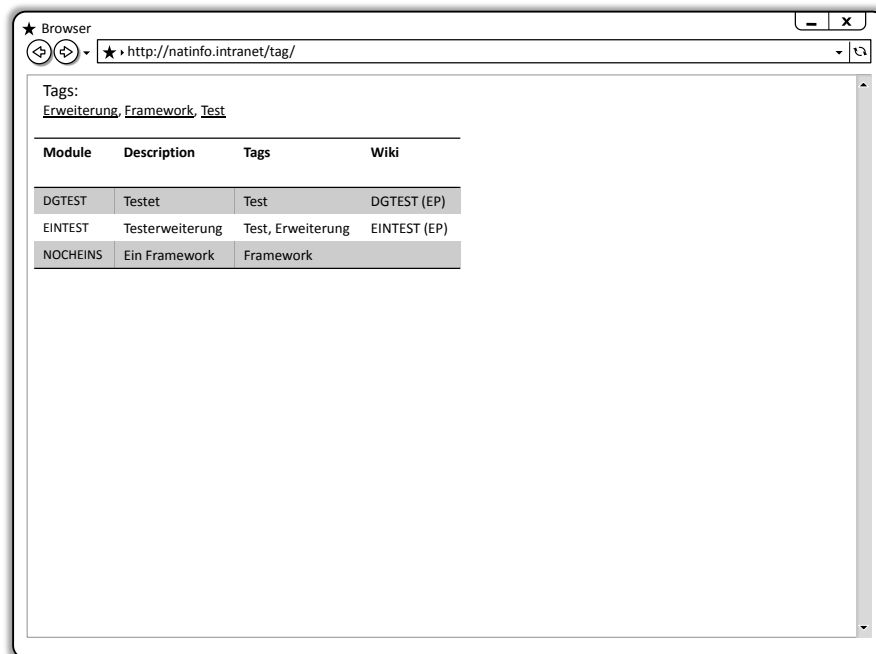


Abbildung 7: Anzeige und Filterung der Module nach Tags

A.8 Screenshots der Anwendung

The screenshot shows the homepage of the 'ALTE OLDENBURGER NATINFO' website. The header features the logo and a family walking on a grassy dune. On the right, there are links for 'Häufig benötigte Seiten', 'Ein-Klick-Menü', 'Direktaufruf', and 'Inhaltssuche'. The main content area is titled 'Tags' and shows a filter for 'Project, Test'. Below this is a table listing modules with their descriptions and associated tags.

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.9 Entwicklerdokumentation

lib-model

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

Packages:
lib-model

Files:
Naturalmodulename.php

Classes:
Naturalmodulename

Class: Naturalmodulename

Source Location: /Naturalmodulename.php

Class Overview

```

BaseNaturalmodulename
|
--Naturalmodulename

```

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

Methods

- [__construct](#)
- [getNaturalTags](#)
- [getNaturalWikis](#)
- [loadNaturalModuleInformation](#)
- [__toString](#)

Class Details

[line 10]
Subclass for representing a row from the 'NaturalModulename' table.

Adds some business logic to the base.

[\[Top \]](#)

Class Methods

constructor [__construct](#) [line 56]

```
Naturalmodulename __construct( )
```

Initializes internal state of Naturalmodulename object.

Tags:

see: parent::__construct()
access: public

[\[Top \]](#)

method [getNaturalTags](#) [line 68]

```
array getNaturalTags( )
```

Returns an Array of NaturalTags connected with this Modulename.

Tags:

return: Array of NaturalTags
access: public

[\[Top \]](#)

method getNaturalWikis [line 83]

```
array getNaturalWikis( )
```

Returns an Array of NaturalWikis connected with this Modulename.

Tags:

return: Array of NaturalWikis
access: public

[\[Top \]](#)

method loadNaturalModuleInformation [line 17]

```
ComparedNaturalModuleInformation  
loadNaturalModuleInformation( )
```

Gets the ComparedNaturalModuleInformation for this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

method __toString [line 47]

```
string __toString( )
```

Returns the name of this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

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

A.10 Testfall und sein Aufruf auf der Konsole

```

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

```

Listing 1: Testfall in PHP

```

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

```

Abbildung 10: Aufruf des Testfalls auf der Konsole

A.11 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

```

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

```

```

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

```

```

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

```

Listing 2: Klasse: ComparedNaturalModuleInformation

A.12 Klassendiagramm

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

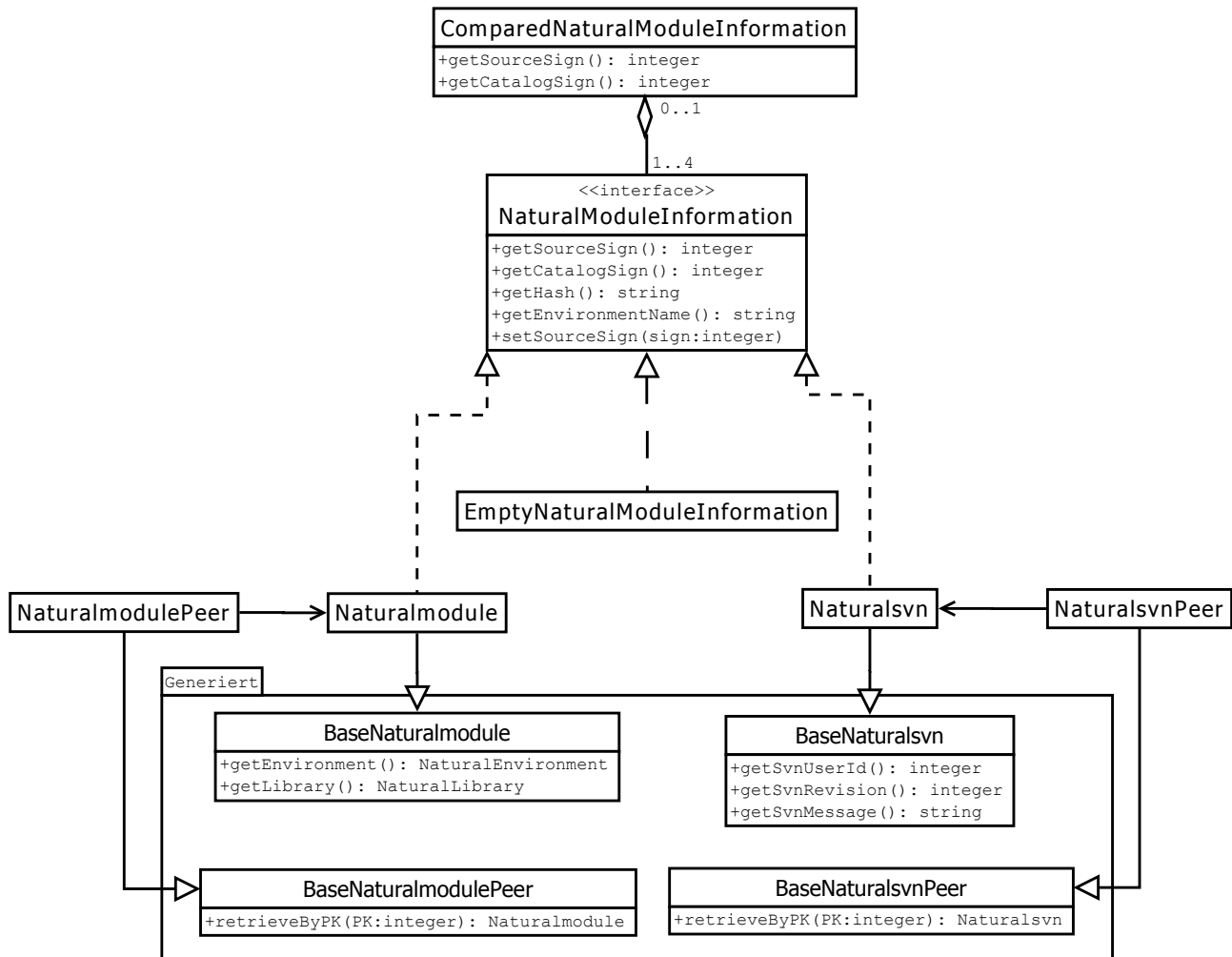







Abbildung 11: Klassendiagramm

A.13 Benutzerdokumentation

Ausschnitt aus der Benutzerdokumentation:

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