

## 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**

<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>IV</b>
<b>Listings</b>	<b>V</b>
<b>Abkürzungsverzeichnis</b>	<b>VI</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Projektumfeld . . . . .	1
1.1.1 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
<b>2 Projektplanung</b>	<b>4</b>
2.1 Projektphasen . . . . .	4
2.2 Abweichungen vom Projektantrag . . . . .	4
2.3 Ressourcenplanung . . . . .	5
2.4 Entwicklungsprozess . . . . .	5
<b>3 Analysephase</b>	<b>6</b>
3.1 Ist-Analyse . . . . .	6
3.2 Wirtschaftlichkeitsanalyse . . . . .	6
3.2.1 „Make or Buy“-Entscheidung . . . . .	6
3.2.2 Projektkosten . . . . .	7
3.2.3 Amortisationsdauer . . . . .	7
3.3 Anwendungsfälle . . . . .	8
3.4 Qualitätsanforderungen . . . . .	9
3.5 Lastenheft/Fachkonzept . . . . .	10
<b>4 Entwurfsphase</b>	<b>10</b>
4.1 Zielplattform . . . . .	10
4.2 Architekturdesign . . . . .	11

4.3	Datenmodell . . . . .	11
4.4	Geschäftslogik . . . . .	11
4.5	Maßnahmen zur Qualitätssicherung . . . . .	12
4.6	Pflichtenheft/Datenverarbeitungskonzept . . . . .	13
<b>5</b>	<b>Implementierungsphase</b>	<b>13</b>
5.1	Implementierung der Datenstrukturen . . . . .	13
5.2	Implementierung der Benutzeroberfläche . . . . .	14
5.3	Implementierung der Geschäftslogik . . . . .	14
<b>6</b>	<b>Abnahmephase</b>	<b>15</b>
<b>7</b>	<b>Einführungsphase</b>	<b>15</b>
<b>8</b>	<b>Dokumentation</b>	<b>15</b>
<b>9</b>	<b>Fazit</b>	<b>16</b>
9.1	Soll-/Ist-Vergleich . . . . .	16
9.2	Lessons Learned . . . . .	16
9.3	Ausblick . . . . .	16
	<b>Eidesstattliche Erklärung</b>	<b>17</b>
<b>A</b>	<b>Anhang</b>	<b>i</b>
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 . . . . .	v
A.7	Screenshots des Codes . . . . .	vii
A.7.1	DCA . . . . .	vii
A.8	Screenshots der Anwendung . . . . .	viii
A.9	Entwicklerdokumentation . . . . .	xiv
A.10	Testfall und sein Aufruf auf der Konsole . . . . .	xvi
A.11	Klasse: ComparedNaturalModuleInformation . . . . .	xvii
A.12	Klassendiagramm . . . . .	xx
A.13	Benutzerdokumentation . . . . .	xxi

**Abbildungsverzeichnis**

1	Vereinfachtes ER-Modell . . . . .	12
2	Prozess des Einlesens eines Moduls . . . . .	12
3	Use-Case-Diagramm: Einlösung des QR-Codes durch einen Mitarbeiter . . . . .	iv
4	Datenbankmodell . . . . .	vi
5	Datenbankdefinition in iso_order_hashes.php . . . . .	vii
6	Manuelle Suche nach Datum, Bestellnummer oder Namen . . . . .	ix
7	Ansicht für das einlösen eines Tickets . . . . .	x
8	Integrierte Benutzerdokumentation . . . . .	xi
9	Login Ansicht für Mitarbeiter . . . . .	xii
10	Ansicht für das einscannen eines QR-Codes . . . . .	xiii
11	Aufruf des Testfalls auf der Konsole . . . . .	xvii
12	Klassendiagramm . . . . .	xx

## **Tabellenverzeichnis**

1	Zeitplanung . . . . .	4
2	Kostenaufstellung . . . . .	7
3	Soll-/Ist-Vergleich . . . . .	16

## Listings

1	Testfall in PHP . . . . .	xvi
2	Klasse: ComparedNaturalModuleInformation . . . . .	xvii

## **Abkürzungsverzeichnis**

<b>API</b>	Application Programming Interface
<b>ERM</b>	Entity-Relationship-Modell
<b>UML</b>	Unified Modeling 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
<b>Gesamt</b>	<b>80 h</b>

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

Der aktuelle Prozess zur Überprüfung der Echtheit und Gültigkeit von Tickets und Gutscheinen basiert auf manuellen Methoden. Dies führt zu einer hohen Fehleranfälligkeit und ist zudem äußerst ineffizient. Die Mitarbeiter sind gezwungen, viel Zeit in die Überprüfung der Tickets zu investieren, was ihre Effizienz erheblich mindert.

Die Wünsche der Mitarbeiter beinhalten eine schnellere und zuverlässigere Methode zur Überprüfung von Tickets und Gutscheinen. Es besteht ein klarer Bedarf an einer automatisierten Lösung, die menschliche Fehler minimiert und den Verifizierungsprozess beschleunigt. Die Mitarbeiter wollen ihre Zeit effektiver nutzen und sich auf wichtigere Aufgaben konzentrieren können.

Das Ziel des Projekts ist es, ein QR-Code-basiertes Einlösungsprogramm zu erstellen, das diese Probleme adressiert. Durch die Implementierung eines Systems, das QR-Codes generiert und scannt, wird die Echtheit und Gültigkeit der Tickets schnell und sicher überprüft. Dies verbessert nicht nur die Effizienz der Arbeitsabläufe, sondern erhöht auch die Zuverlässigkeit und Sicherheit des gesamten Prozesses. Insgesamt führt die Automatisierung zu einer gesteigerten Zufriedenheit sowohl der Mitarbeiter als auch der Kunden.

### 3.2 Wirtschaftlichkeitsanalyse

- Lohnt sich das Projekt für das Unternehmen?

#### 3.2.1 „Make or Buy“-Entscheidung

Bei der Entscheidung, ob eine bestehende Lösung verwendet oder eine neue entwickelt werden sollte, wurde sorgfältig geprüft, ob es bereits ein fertiges Produkt gibt, das alle Anforderungen des Projekts abdeckt. Dabei stellte sich heraus, dass es keine existierende Lösung gab, die den spezifischen Anforderungen entsprach.

Ein wichtiges Kriterium war die Kompatibilität mit dem bereits genutzten Framework Contao und dem Onlineshop-Plugin Isotope. Die Suche nach einer fertigen Lösung, die als Bundle über Composer installiert werden kann und nahtlos mit Contao und Isotope zusammenarbeitet, war erfolglos. Die

vorhandenen Produkte auf dem Markt deckten entweder nicht alle funktionalen Anforderungen ab oder waren nicht vollständig kompatibel mit der bestehenden Architektur.

Aufgrund dieser speziellen Anforderungen und der fehlenden passenden Lösungen auf dem Markt wurde entschieden, das Projekt intern umzusetzen. Dies ermöglichte eine maßgeschneiderte Entwicklung, die exakt auf die Bedürfnisse des Unternehmens und der Mitarbeiter zugeschnitten ist. Durch die interne Entwicklung konnte sichergestellt werden, dass alle Anforderungen vollständig erfüllt und gleichzeitig die bestehende Architektur optimal genutzt wird.

### 3.2.2 Projektkosten

**Rechnung** 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 80 Stunden. Für die Nutzung von Ressourcen<sup>1</sup> wird ein pauschaler Stundensatz von 15 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 25 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt 2739,20 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklung	80 h	7,56 € + 15 € = 22,56 €	1804,80 €
Entwurf	15 h	25 € + 15 € = 40 €	600 €
Abnahme	2,5 h	25 € + 15 € = 40 €	100 €
Anwenderschulung	5 h	25 € + 15 € = 40 €	200 €
			<b>2704,80 €</b>

Tabelle 2: Kostenaufstellung

### 3.2.3 Amortisationsdauer

Da das System intern entwickelt wurde, entfallen Ausgaben für externe Softwarelizenzen, Updates und Wartungsverträge. Dies reduziert die langfristigen Betriebskosten erheblich und führt zu direkten Kosteneinsparungen.

---

<sup>1</sup>Räumlichkeiten, Arbeitsplatzrechner etc.

Durch die interne Entwicklung des Systems vermeiden wir zudem die Kosten für die Anpassung und Integration externer Lösungen. Die Anwendung ist exakt auf die Bedürfnisse unserer Kunden und die von uns unterstützten Plattformen abgestimmt, was zusätzliche Anpassungsarbeiten überflüssig macht.

Ein weiterer signifikanter Vorteil ist die Zeitersparnis bei der Implementierung und dem Support. Dank der engen Integration mit dem bestehenden Contao-Framework und dem Isotope-Plugin können unsere Entwickler das System schneller und mit weniger Komplikationen einrichten. Die reduzierte Implementierungszeit bedeutet, dass wir mehr Projekte in kürzerer Zeit abschließen können, was zu einer höheren Umsatzrate führt.

Zusätzlich ermöglicht uns das QR-Code-System, unsere Marktposition zu stärken und als technologischer Vorreiter wahrgenommen zu werden. Die Fähigkeit, moderne Lösungen anzubieten, verbessert unsere Wettbewerbsfähigkeit und hilft dabei, neue Kunden zu gewinnen und bestehende Kundenbeziehungen zu vertiefen.

Durch den Verkauf des QR-Code-basierten Einlösungsprogramms können wir außerdem von wiederkehrenden Einnahmen profitieren. Wartungsverträge, Schulungen und zusätzliche Supportdienste bieten kontinuierliche Einkommensströme, die die finanzielle Stabilität unseres Unternehmens unterstützen.

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

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

### 3.3 Anwendungsfälle

Der erste Anwendungsfall betrifft die Generierung eines QR-Codes für eine Bestellung. Sobald ein Kunde im Onlineshop eine Bestellung abschließt, wird automatisch ein einzigartiger QR-Code generiert. Dieser QR-Code enthält alle relevanten Bestelldaten und wird in der Datenbank gespeichert. Anschließend wird der QR-Code per E-Mail an den Kunden versendet. Dies ermöglicht eine einfache und sichere Verifizierung der Bestellung.



Ein weiterer zentraler Anwendungsfall ist die Verifizierung und Einlösung des QR-Codes durch einen Mitarbeiter. Der Kunde präsentiert den QR-Code, den er per E-Mail erhalten hat. Der Mitarbeiter scannt den QR-Code, und das System überprüft dessen Gültigkeit. Ist der QR-Code gültig, werden dem Mitarbeiter alle relevanten Informationen zur Bestellung angezeigt. Der Mitarbeiter kann dann entscheiden, ob er den QR-Code tatsächlich einlösen möchte. Nach der Bestätigung wird der QR-Code im System als eingelöst markiert und kann nicht erneut verwendet werden. Dies stellt sicher, dass jeder QR-Code nur einmal genutzt werden kann und verhindert Betrugsversuche.

Zusätzlich gibt es die Möglichkeit einer manuelle Suche nach den Tickets. Sollte es einmal nicht möglich sein, den QR-Code zu scannen, kann der Mitarbeiter nach einem Ticket suchen. Das Programm zeigt dann die Gültigkeit des Tickets an und gibt die entsprechenden Bestellinformationen aus, um die Einlösung zu ermöglichen.

**Use Case-Diagramm** Ein Use Case-Diagramm befindet sich im Anhang A.4: Use Case-Diagramm auf Seite iv.

### 3.4 Qualitätsanforderungen

Ein zentraler Qualitätsaspekt ist die Fehlerfreiheit der Anwendung. Es sollten möglichst wenig Fehler in der Benutzung auftreten. Falls dennoch Fehler auftreten, muss klar und verständlich kommuniziert werden, was falsch gelaufen ist. Die Fehlermeldungen sollten in einer Form präsentiert werden, die dem Benutzer nützliche Informationen liefert, um das Problem zu beheben oder den Support zu kontaktieren. Kritische Fehler, die das gesamte Programm lahmlegen, sind unbedingt zu vermeiden.

Die Zuverlässigkeit des Systems ist ein weiterer wichtiger Faktor. Benutzer müssen sich darauf verlassen können, dass ein als gültig angezeigtes Ticket tatsächlich gültig ist. Das System sollte sich so gut wie möglich vor gefälschten oder manipulierten Tickets schützen. Ein Fehler in der Verifizierung der Tickets kann zu erheblichen finanziellen Verlusten für den Kunden führen. Daher muss das System stets korrekte und verlässliche Ergebnisse liefern.

Die Usability der Anwendung spielt ebenfalls eine große Rolle. Die Benutzeroberfläche sollte intuitiv und benutzerfreundlich gestaltet sein, sodass auch unerfahrene Benutzer problemlos mit dem System arbeiten können.

Die Integrationsfähigkeit in die bestehende Umgebung spielt auch eine entscheidende Rolle. Das Programm muss sich nahtlos in die vorhandene Infrastruktur integrieren lassen und die vorgegebene Struktur von Symfony Bundles einhalten. Dies gewährleistet eine reibungslose Integration in bestehende Systeme und erleichtert die Wartung und Erweiterung der Anwendung.

Das Programm muss außerdem sicherstellen, dass nur Mitarbeiter mit einem hinterlegten Login Zugang zu dem Einlösetool haben. Diese Maßnahme verhindert, dass Tickets versehentlich entwertet werden oder sensible Bestellinformationen von unbefugten Personen eingesehen werden können.

### 3.5 Lastenheft/Fachkonzept

Alle spezifischen Anforderungen sowie die allgemeinen, technischen und qualitativen Ziele, die im Rahmen dieses Projekts festgelegt wurden, sind detailliert im Lastenheft beschrieben. Für eine umfassende Übersicht und weitere Details wird auf das Anhang A.3: Lastenheft (Auszug) auf Seite ii verwiesen.

## 4 Entwurfsphase

### 4.1 Zielplattform

**Programmiersprache:** Die Anwendung wird in PHP unter Verwendung des Symfony-Frameworks entwickelt. Symfony bietet eine robuste und flexible Grundlage für Webanwendungen und unterstützt die Erstellung von sicheren und skalierbaren Lösungen. Die Wahl von PHP und Symfony ermöglicht eine effiziente Entwicklung und Wartung der Anwendung.

**Datenbank:** Als Datenbank wird MySQL eingesetzt. MySQL ist eine weit verbreitete, leistungsstarke und zuverlässige relationale Datenbank, die gut mit Symfony und PHP integriert werden kann. Sie bietet die notwendige Leistung und Stabilität, um die Bestelldaten und QR-Codes sicher zu speichern und schnell abzurufen.

**Client/Server-Architektur:** Die Anwendung wird als Webanwendung mit einer Client/Server-Architektur implementiert. Dies ermöglicht eine zentrale Verwaltung und Aktualisierung der Anwendung auf dem Server, während die Benutzer über ihre Webbrowser auf die Anwendung zugreifen. Die Verwendung einer Webanwendung stellt sicher, dass keine zusätzliche Software auf den Endgeräten der Benutzer installiert werden muss.

**Hardware:** Die Zielplattform umfasst Smartphones, da diese über integrierte Kameras verfügen, die für das Scannen der QR-Codes erforderlich sind. Die Anwendung soll auf allen Smartphones mit Webbrowsern und einer Internetverbindung zugänglich sein. Dies stellt sicher, dass alle Mitarbeiter unabhängig vom verwendeten Gerät auf die Anwendung zugreifen können.

Die Entscheidung für PHP, Symfony und MySQL fiel, weil diese Technologien in unserem Betrieb gängig sind und von unseren Entwicklern routiniert verwendet werden. Dadurch kann effizient und schnell gearbeitet werden. Zudem ermöglicht diese Kombination die Erstellung von Composer-Bundles, die sich nahtlos in andere Symfony-Projekte integrieren lassen.

Besonders wichtig ist dies, weil das Kundenprojekt, in dem das Bundle eingebunden werden soll, Contao verwendet, welches auf Symfony basiert. Diese Auswahl an Technologien erleichtert die Implementierung und gewährleistet eine konsistente Benutzererfahrung.

## 4.2 Architekturdesign

Für das QR-Code-basierte Einlösungsprogramm wurde das Symfony-Framework ausgewählt, das die Grundlage für das verwendete Contao CMS bildet. Diese Entscheidung wurde sowohl aufgrund der spezifischen Anforderungen des Projekts als auch der etablierten Praktiken im Betrieb getroffen. Symfony und Contao bieten eine leistungsfähige Kombination, die auf dem Model-View-Controller (MVC) Muster basiert. Dieses Architekturmuster teilt die Anwendung in drei Hauptkomponenten: Modell, Ansicht und Controller.

Das MVC-Modell ermöglicht eine klare Trennung der Geschäftslogik von der Benutzeroberfläche. Diese Struktur erleichtert die Arbeit am Projekt, da die Logik der QR-Code-Generierung und -Verifizierung unabhängig von der Präsentation implementiert werden kann.

Ein zentraler Vorteil von Symfony und Contao liegt in der nativen Unterstützung von Composer, welches die Erstellung von Bundles ermöglicht. Ein Bundle fungiert als eine Art Plugin oder Modul, das bestimmte Funktionen kapselt und wiederverwendbar macht. Im Rahmen dieses Projekts wird das QR-Code-Tool als Bundle entwickelt, wodurch es sich problemlos in andere Contao-Projekte integrieren lässt.

Für die Authentifizierung der Benutzer kann die Mitgliedsverwaltung von Contao genutzt werden. Dies stellt sicher, dass nur autorisierte Benutzer auf das Einlösetool zugreifen können und sensible Bestellinformationen geschützt bleiben.

Die integrierten Sicherheitsmechanismen von Symfony, einschließlich der Benutzer-Authentifizierung und -Autorisierung, stellen sicher, dass die Anwendung vor unbefugtem Zugriff geschützt ist und die Integrität der Bestelldaten gewahrt bleibt.

## 4.3 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.

## 4.4 Geschäftslogik

- Modellierung und Beschreibung der wichtigsten (!) Bereiche der Geschäftslogik (z. B. mit Komponenten-, Klassen-, Sequenz-, Datenflussdiagramm, Programmablaufplan, Struktogramm, **EPK!** (**EPK!**)).
- Wie wird die erstellte Anwendung in den Arbeitsfluss des Unternehmens integriert?

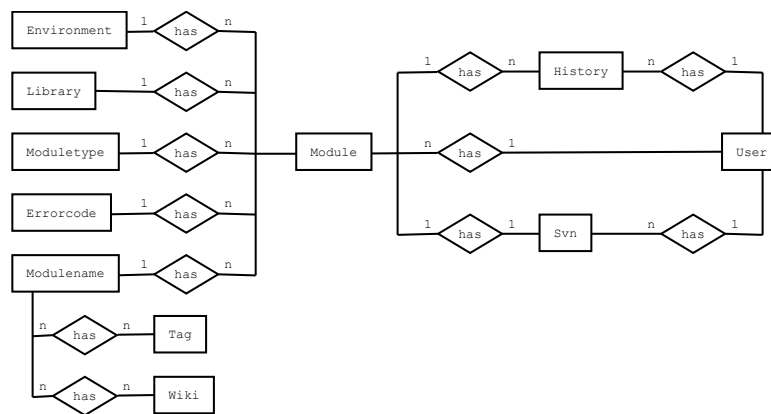


Abbildung 1: Vereinfachtes ER-Modell

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

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

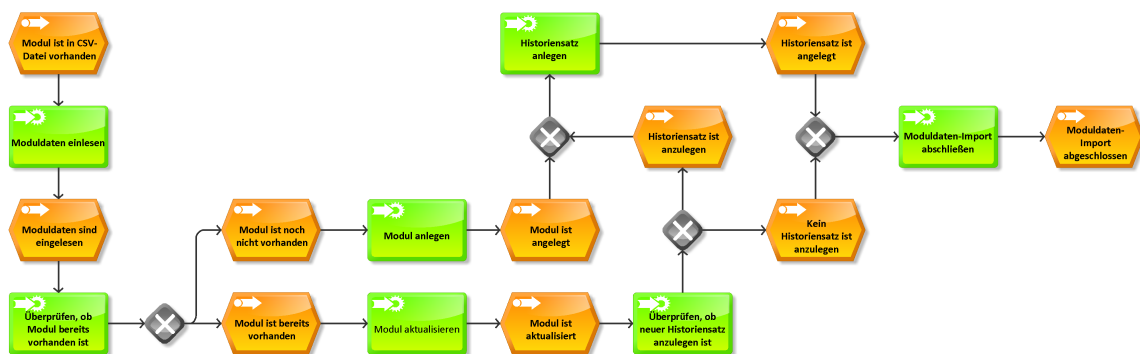


Abbildung 2: Prozess des Einlesens eines Moduls

## 4.5 Maßnahmen zur Qualitätssicherung

Die Überprüfung der korrekten Ausführung des Codes erfolgte mithilfe des Debug-Modus von Symfony und auslesen von Log-Dateien. Durch dieses Debugging konnten potenzielle Fehlerquellen identifiziert und behoben werden, bevor sie zu größeren Problemen führten.

Es wurden außerdem Frontend-Tests durchgeführt, um die Funktionalität und Benutzerfreundlichkeit der Anwendung sicherzustellen. Diese Tests beinhalteten die Simulation der typischen Benutzerinteraktionen, wie das Scannen und Verifizieren von QR-Codes, sowie die Überprüfung der Navigations- und Anzeigeelemente. Ziel war es, sicherzustellen, dass alle erforderlichen Features reibungslos funktionieren und die Anwendung intuitiv bedienbar ist.

Die Sicherheitsaspekte der Anwendung wurden ebenfalls getestet. Dies umfasste die Zugangskontrolle, bei der überprüft wurde, dass nur autorisierte Mitarbeiter mit einem hinterlegten Login auf das Einlösetool zugreifen können. Dadurch wurde gewährleistet, dass sensible Bestellinformationen geschützt und Tickets nicht versehentlich entwertet werden.

## 4.6 Pflichtenheft/Datenverarbeitungskonzept

Für die Implementierungsdetails dieses Projekts wird auf das im Anhang befindliche Pflichtenheft verwiesen. Dieses Dokument wurde im Verlauf des Projekts erarbeitet und dient als umfassende Guideline, die alle notwendigen Schritte und Vorgaben zur erfolgreichen Realisierung der Anwendung enthält. Das Pflichtenheft ist im Anhang A.5: Pflichtenheft (Auszug) auf Seite iv zu finden.

# 5 Implementierungsphase

## 5.1 Implementierung der Datenstrukturen

Contao generiert die Datenbanktabellen automatisch anhand der DCA-Dateien, die im `dca/` Ordner des jeweiligen Bundles hinterlegt sind. Eine DCA-Datei enthält die Konfiguration für eine Datenbanktabelle, einschließlich der Definition der Felder und deren Eigenschaften. Beim Installations- oder Aktualisierungsprozess liest Contao diese Konfigurationsdateien aus und erstellt oder aktualisiert die entsprechenden Tabellen in der Datenbank.

Die DCA-Datei definiert die Struktur und die Felder der Tabelle, einschließlich der SQL-Typen und anderer relevanter Einstellungen. Contao interpretiert diese Informationen und generiert die erforderlichen SQL-Befehle, um die Datenbanktabellen anzulegen oder zu aktualisieren.

Die Tabelle `tl_iso_order_hashes` wurde erstellt, um die QR-Codes und deren zugehörige Informationen zu verwalten.

- **id:** Dieses Feld dient als Primärschlüssel und wird automatisch hochgezählt. Es wird als `int(10) unsigned` definiert, um eine ausreichende Anzahl von Einträgen zu ermöglichen.
- **hash:** Dieses Feld speichert den aus Bestellinformationen generierten Hash-Wert. Es ist als `varchar(64)` definiert, um den Hash-Wert aufzunehmen, wobei 64 Zeichen ausreichend für einen sicheren Hash-Wert sind.
- **iso\_product\_collection\_id:** Dieses Feld speichert die ID der zugehörigen Produktkollektion aus dem Isotope-Shop. Es wird als `int(10)` definiert.
- **valid:** Dieses Feld definiert die Gültigkeit des QR-Codes. Es ist als `int(1)` definiert, wobei der Standardwert 0 ist. Ein Wert von 1 bedeutet, dass der QR-Code gültig und noch nicht eingelöst ist.

Ein Screenshot der DCA-Datei wird im Anhang bereitgestellt, um eine visuelle Darstellung der Konfiguration zu bieten. Der Screenshot ist hier zu finden: Anhang A.7.1: DCA auf Seite vii.

## 5.2 Implementierung der Benutzeroberfläche

Die Implementierung der Benutzeroberfläche für das QR-Code-basierte Einlösungsprogramm erfolgte unter Verwendung von HTML5-Templates, die sowohl HTML als auch PHP beinhalten. Diese Kombination ermöglicht eine flexible und dynamische Darstellung der Inhalte, die durch die Backend-Logik gesteuert wird.

Für das Styling der Benutzeroberfläche kamen SCSS-Dateien zum Einsatz. SCSS, als Erweiterung von CSS, bietet zusätzliche Funktionen wie Variablen, geschachtelte Regeln und Mixins, die die Erstellung und Verwaltung der Stylesheets erheblich vereinfachen. Diese SCSS-Dateien wurden mithilfe des SASS Pre-Processors in reguläres CSS kompiliert und anschließend in die HTML5-Templates eingebunden.

Die Logik im Frontend wurde mit JavaScript realisiert. JavaScript wurde verwendet, um interaktive Elemente in die Benutzeroberfläche zu integrieren. Dies umfasst unter anderem die Implementierung der QR-Code-Scanfunktion, die Überprüfung und Validierung der Eingaben sowie die Anzeige von Bestellinformationen und Fehlermeldungen.

Für das Scannen der QR-Codes wurde die externe Bibliothek `html5-qrcode` verwendet. Diese Bibliothek bietet eine leistungsfähige und benutzerfreundliche Möglichkeit, QR-Codes direkt im Browser mithilfe der Gerätekamera zu scannen.

Die Gestaltung der Benutzeroberfläche basiert auf den vorgegebenen Figma-Designs. Diese Designs wurden als Grundlage für die Entwicklung verwendet, um sicherzustellen, dass die Benutzeroberfläche sowohl ästhetisch ansprechend als auch funktional ist.

Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang A.8: 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.11: Klasse: `ComparedNaturalModuleInformation` auf Seite xvii.

## 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 xvi. 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 xxi. Die Entwicklerdokumentation wurde mittels PHPDoc<sup>2</sup> automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang A.9: Entwicklerdokumentation auf Seite xiv.

---

<sup>2</sup>Vgl. ?

## 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 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
<b>Gesamt</b>	<b>70 h</b>	<b>70 h</b>	

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)?



## **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

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

## A.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. Generierung und Verwaltung von QR-Codes
  - 1.1. Die Anwendung muss für jede abgeschlossene Bestellung einen einzigartigen QR-Code generieren.
  - 1.2. Es muss ein Eintrag für jedes Ticket in die Datenbank geschrieben werden.
  - 1.3. Die QR-Codes müssen per E-Mail an die Kunden versendet werden.
2. Einlösung und Verifizierung von QR-Codes
  - 2.1. Die Anwendung muss das Scannen der QR-Codes ermöglichen.
  - 2.2. Die Anwendung muss die Gültigkeit der gescannten QR-Codes überprüfen.
  - 2.3. Die Anwendung muss die Bestellinformationen anzeigen, wenn ein QR-Code gescannt wird.
  - 2.4. Die Anwendung muss es ermöglichen, die Einlösung der QR-Codes zu bestätigen und diese als eingelöst zu markieren.
  - 2.5. Ungültige oder bereits eingelöste QR-Codes dürfen nicht erneut akzeptiert werden.

#### 3. Benutzeroberfläche und Usability

- 3.1. Die Benutzeroberfläche muss intuitiv und benutzerfreundlich gestaltet sein.
- 3.2. Die Navigation durch die Anwendung muss einfach und klar strukturiert sein.
- 3.3. Fehlermeldungen müssen verständlich und hilfreich sein, um dem Benutzer bei der Problembehebung zu helfen.

#### 4. Sicherheit und Zugriffskontrolle

- 4.1. Die Anwendung muss vor gefälschten und manipulierten QR-Codes sicher sein.
- 4.2. Es muss sichergestellt werden, dass nur autorisierte Mitarbeiter mit einem hinterlegten Login Zugang zur Anwendung haben.
- 4.3. Die Zugriffskontrolle muss verhindern, dass unbefugte Personen sensible Bestellinformationen einsehen oder Tickets versehentlich entwerten können.

#### 5. Integrationsfähigkeit

- 5.1. Die Anwendung muss sich nahtlos in die bestehende Systemlandschaft integrieren.
- 5.2. Die Struktur der Anwendung muss den Anforderungen eines Symfony Bundles entsprechen.

#### 6. Zuverlässigkeit und Effizienz

- 6.1. Die Anwendung muss verlässlich und korrekt arbeiten.
- 6.2. Kritische Fehler, die das gesamte Programm lahmlegen, müssen unbedingt vermieden werden.

## A.4 Use Case-Diagramm

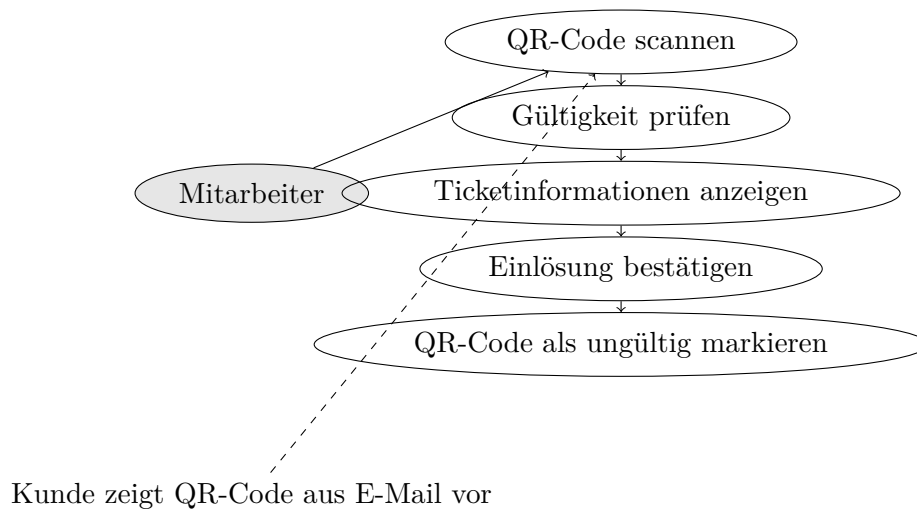


Abbildung 3: Use-Case-Diagramm: Einlösung des QR-Codes durch einen Mitarbeiter

## A.5 Pflichtenheft (Auszug)

### Zielbestimmung

#### 1. Musskriterien

##### 1.1. Generierung und Verwaltung von QR-Codes

- Das Programm muss für jede abgeschlossene Bestellung automatisch einen einzigartigen QR-Code erstellen.
- Jeder generierte QR-Code muss in der Datenbank erfasst und gespeichert werden.
- Die QR-Codes müssen per E-Mail an die Kunden verschickt werden.

##### 1.2. Einlösung und Verifizierung von QR-Codes

- Das Programm muss das Scannen der QR-Codes mittels der Kamera eines mobilen Endgeräts ermöglichen.
- Das Programm muss die Gültigkeit der gescannten QR-Codes überprüfen.
- Beim Scannen eines QR-Codes müssen die relevanten Bestellinformationen angezeigt werden.
- Das Programm muss die Einlösung der QR-Codes bestätigen und diese als eingelöst markieren.
- Ungültige oder bereits eingelöste QR-Codes dürfen nicht erneut akzeptiert werden.

##### 1.3. Benutzeroberfläche und Usability

- Die Benutzeroberfläche des Programms muss intuitiv und benutzerfreundlich gestaltet sein.
- Die Navigation innerhalb des Programms muss klar und strukturiert sein.

- Fehlermeldungen müssen verständlich und hilfreich formuliert sein.

#### 1.4. Sicherheit und Zugriffskontrolle

- Das Programm muss vor gefälschten und manipulierten QR-Codes sicher sein.
- Es muss sichergestellt werden, dass nur autorisierte Mitarbeiter Zugang zum Programm haben.
- Die Zugriffskontrolle muss verhindern, dass unbefugte Personen sensible Bestellinformationen einsehen oder Tickets versehentlich entwerten können.

#### 1.5. Integrationsfähigkeit

- Das Programm muss sich nahtlos in die bestehende Systemlandschaft integrieren.
- Die Struktur des Programms muss den Anforderungen eines Symfony Bundles entsprechen.

#### 1.6. Zuverlässigkeit und Effizienz

- Das Programm muss zuverlässig und korrekt arbeiten.
- Kritische Fehler, die das gesamte Programm lahmlegen, müssen unbedingt vermieden werden.

### Produkteinsatz

#### 1. Anwendungsbereiche

Das QR-Code-basierte Einlösungsprogramm dient zur Verwaltung und Verifizierung von Online-Tickets. Es soll eine effiziente und sichere Abwicklung der Ticketverkäufe und deren Einlösung vor Ort ermöglichen.

#### 2. Zielgruppen

Die Hauptnutzer der Anwendung sind unsere Kunden, die für die Verifizierung und Einlösung der Tickets ihrer Online-Shops verantwortlich sind. Dazu gehören insbesondere Mitarbeiter in der Veranstaltungsbranche, im Einzelhandel sowie im Tourismusbereich.

#### 3. Betriebsbedingungen

Die Anwendung muss auf Smartphones mit Webbrowser und Internetverbindung lauffähig sein. Das Programm wird als Webanwendung bereitgestellt. Eine ständige Verfügbarkeit der Anwendung ist durch den Betrieb auf einem Webserver sichergestellt. Updates und Wartungen werden zentral durchgeführt, um die kontinuierliche Verfügbarkeit zu gewährleisten.

### A.6 Datenbankmodell

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

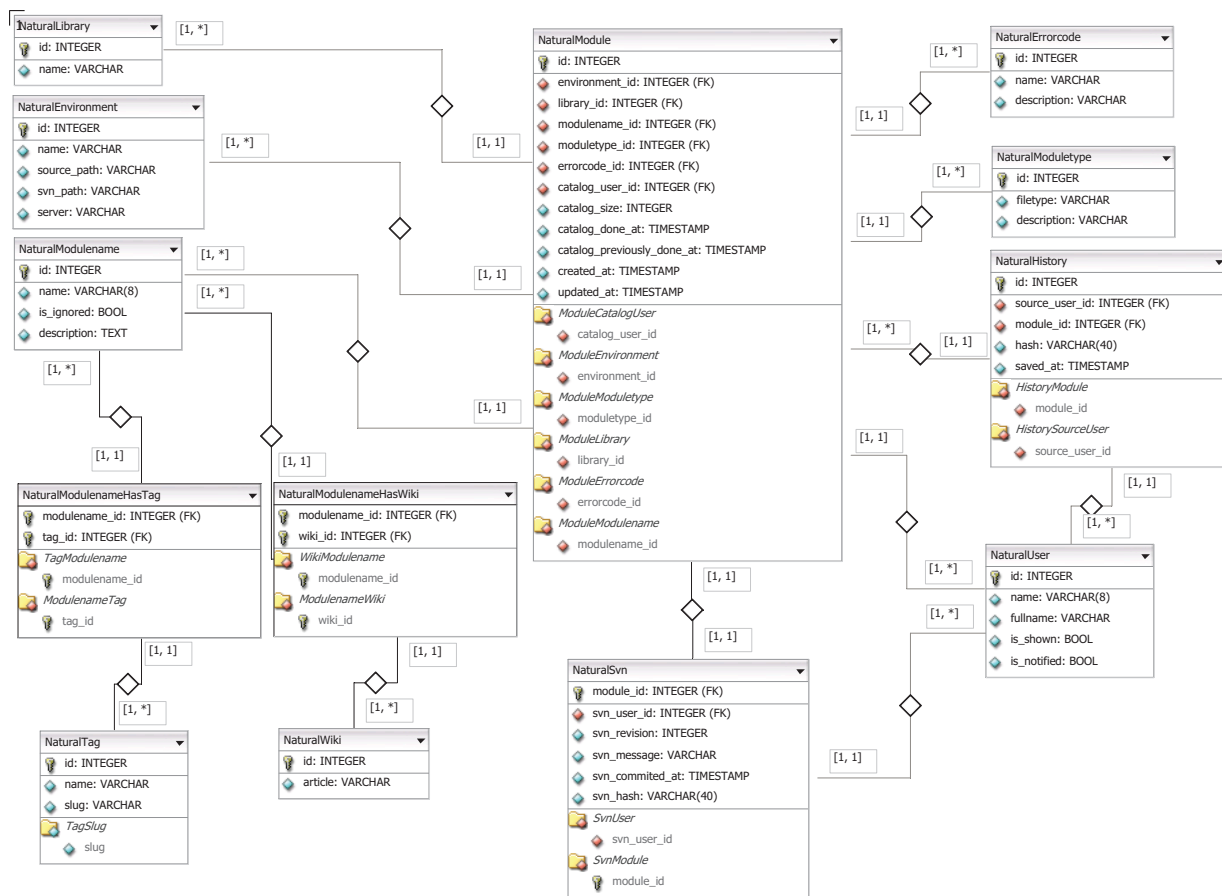


Abbildung 4: Datenbankmodell

## A.7 Screenshots des Codes

### A.7.1 DCA

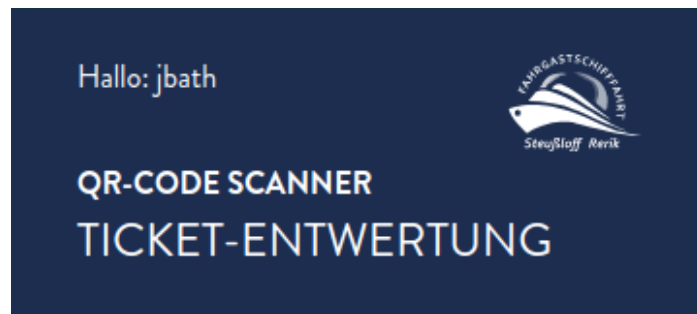
A screenshot of a code editor with a dark background and light-colored text. The code is PHP, defining a database table structure. It starts with a line number 1 and ends with 33. The code defines a table named 'tl\_iso\_order\_hashes' and sets up its configuration, palettes, and fields. The 'fields' section includes an 'id' field with a primary key constraint, a 'hash' field, an 'iso\_product\_collection\_id' field, and a 'valid' field.

```
1 $strTable = 'tl_iso_order_hashes';
2
3 $GLOBALS['TL_DCA'][$strTable] = array(
4     // Config
5     'config' => array(
6         'dataContainer' => 'Table',
7         'enableVersioning' => 'true',
8         'sql' => array(
9             'keys' => array(
10                'id' => 'primary'
11            )
12        ),
13    ),
14    //Palettes
15    'palettes' => array(
16        'default' => ''
17    ),
18    //Fields
19    'fields' => array(
20        'id' => array(
21            'sql' => "int(10) unsigned NOT NULL auto_increment PRIMARY KEY"
22        ),
23        'hash' => array(
24            'sql' => "varchar(64) NULL"
25        ),
26        'iso_product_collection_id' => array(
27            'sql' => "int(10) NULL",
28        ),
29        "valid" => array(
30            'sql' => "int(1) NOT NULL DEFAULT '0'",
31        )
32    )
33 );
```

Abbildung 5: Datenbankdefinition in iso\_order\_hashes.php





## A.8 Screenshots der Anwendung




Ticket scannen







**QR-CODE SCANNER**

**TICKET-ENTWERTUNG**


 Ticketnummer, Name oder Datum

**Ticketliste**

**23.05.2024**

Jannick Bath	0008	
Jannick Bath	0009	

**22.05.2024**

Jannick Bath	0007	
--------------	------	---

**03.05.2024**



Jannick Bath	0005	
Jannick Bath	0006	

Abbildung 6: Manuelle Suche nach Datum, Bestellnummer oder Namen



**QR-CODE SCANNER**  
**TICKET-ENTWERTUNG**

● **GÜLTIG:**  
Gekauft am: 23.05.2024  
Abfahrt: 13:00 Uhr

**Haffrundfahrt**  
**Jannick Bath**  
Ticketnummer: 0008

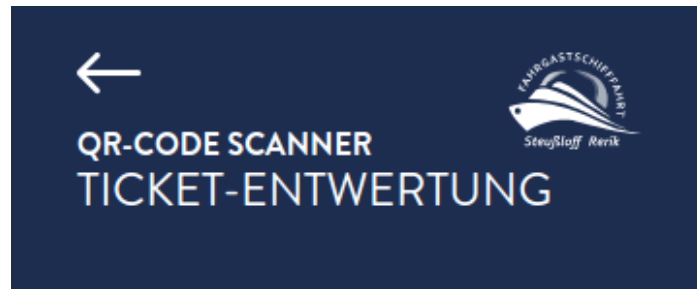
-----

**Tickets:**

Kind	x 3
------	-----

**TICKET ENTWERTEN**

Abbildung 7: Ansicht für das einlösen eines Tickets



## 1. Scan von Tickets


Scanner aktivieren und QR-Code anvisieren.



Der Nutzer wird zur Ticketinformation weitergeleitet und kann das Ticket entwerten.



Abbildung 8: Integrierte Benutzerdokumentation



**Anmelden**

Benutzername
Passwort

**Anmelden**

Abbildung 9: Login Ansicht für Mitarbeiter



Abbildung 10: Ansicht für das einscannen eines QR-Codes

## 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(NaturalmodulePeer::NAME, 'SMTAB');
14 $moduleName = NaturalmodulePeer::doSelectOne($criteria);
15 $t->is($moduleName->getName(), 'SMTAB', 'Right module name selected');
16 $comparedInformation = $moduleName->loadNaturalModuleInformation();
17 $t->is($comparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign
    shines global');
18 $t->is($comparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign
    shines global');
19 $infos = $comparedInformation->getNaturalModuleInformations();
20 foreach($infos as $info)
21 {
22     $env = $info->getEnvironmentName();
23     $t->is($info->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign shines at ' . $env);
24     if ($env != 'SVNENTW')
25     {
26         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign shines at ' .
            $info->getEnvironmentName());
27     }
28     else
29     {
30         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, 'Catalog sign is empty
            at ' . $info->getEnvironmentName());
31     }
32 }
33 ?>

```

Listing 1: Testfall in PHP

```

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

```

Abbildung 11: 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  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/class-diagram.html>.

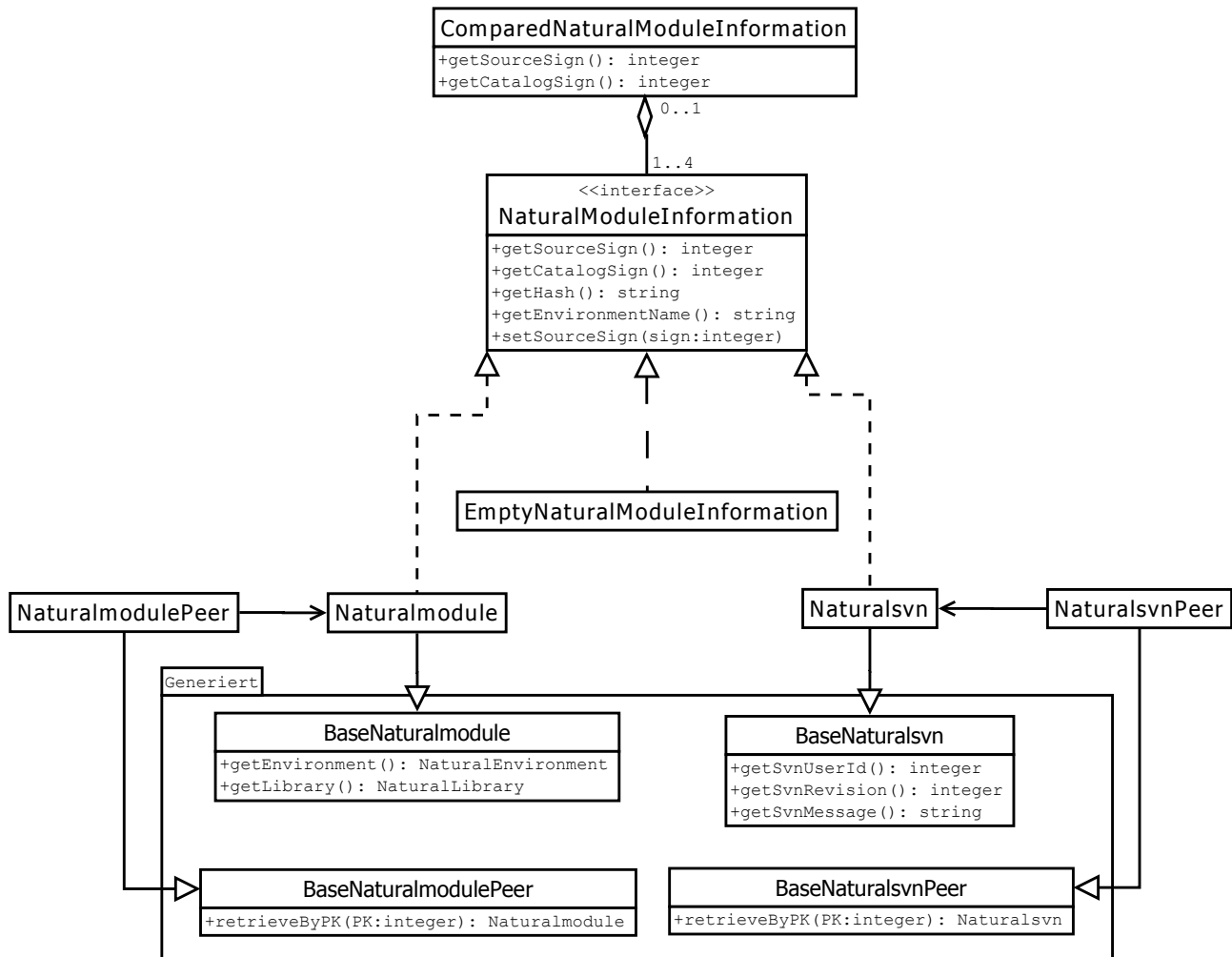







Abbildung 12: Klassendiagramm

**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.