

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
3.1.1 „Make or Buy“-Entscheidung	6
3.1.2 Projektkosten	7
3.1.3 Wirtschaftlichkeit / Amortisationsdauer	7
3.2 Anwendungsfälle	8
3.3 Qualitätsanforderungen	8
3.4 Lastenheft/Fachkonzept	9
4 Entwurfsphase	9
4.1 Zielplattform	9
4.2 Architekturdesign	10
4.3 Datenmodell	11

4.4	Geschäftslogik	11
4.5	Maßnahmen zur Qualitätssicherung	12
4.6	Pflichtenheft/Datenverarbeitungskonzept	12
5	Implementierungsphase	13
5.1	Implementierung der Datenstrukturen	13
5.2	Implementierung der Benutzeroberfläche	13
5.3	Implementierung der Geschäftslogik	14
6	Abnahmephase	15
7	Dokumentation	16
8	Fazit	16
8.1	Soll-/Ist-Vergleich	16
8.2	Lessons Learned	17
8.3	Ausblick	17
	Eidesstattliche Erklärung	18
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	Screenshots des Codes	vii
A.7.1	DCA	vii
A.8	Screenshots der Anwendung	viii
A.9	Entwicklerdokumentation	xiv
A.10	Testprotokoll	xvi
A.11	Klasse: TicketRoutes	xviii
A.12	Klasse: TLOrderHashes	xx
A.13	Klassendiagramm	xxii
A.14	Benutzerdokumentation	xxiii
A.14.1	Starten des Programms	xxiii
A.14.2	Scannen eines QR-Codes	xxiii
A.14.3	Entwertung des Tickets	xxiii
A.14.4	Fehlerbehebung beim Scannen	xxiii
A.14.5	Manuelle Suche für Tickets	xxiv

Abbildungsverzeichnis

1	Vereinfachtes ER-Modell	11
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 zum einlösen von Tickets	x
8	Integrierte Benutzerdokumentation	xi
9	Login Ansicht für Mitarbeiter	xii
10	Ansicht für das einscannen eines QR-Codes	xiii
11	Klassendiagramm	xxii

Tabellenverzeichnis

1	Zeitplanung	4
2	Kostenaufstellung	7
3	Soll-/Ist-Vergleich	17

Listings

1	Klasse: TicketRoutes	xviii
2	Klasse: TLOrderHashes	xx

Abkürzungsverzeichnis

API	Application Programming Interface
CSS	Cascading Style Sheets
SCSS	Sassy Cascading Style Sheets
SASS	Syntactically Awesome Style Sheets
ERM	Entity-Relationship-Modell
HTML	Hypertext Markup Language
MVC	Model View Controller
PHP	Hypertext Preprocessor
UML	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	37.5 h
Kontrolle	27.5 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

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.

3.1.1 „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.

3.1.2 Projektkosten

Rechnung Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten zusammen. Für Azubis gilt ein pauschaler Stundensatz von 50€. Für fest angestellte Mitarbeiter gilt ein Stundensatz von 89€.

Die Durchführungszeit des Projekts beträgt 80 Stunden. Die Nutzung von Ressourcen wie Räumlichkeiten, Arbeitsplatzrechner etc. sind bereits in diesem Stundensatz enthalten.

Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt 6002,50 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklung	80 h	50 €	4000 €
Entwurf	15 h	89 €	1335 €
Abnahme	2.5 h	89 €	222.50 €
Anwenderschulung	5 h	89 €	445 €
			6002.50 €

Tabelle 2: Kostenaufstellung

3.1.3 Wirtschaftlichkeit / Amortisationsdauer

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

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-Shop-Plugin können Entwickler das System schnell und mit wenig 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 Programms an mehrere Kunden können wir außerdem von wiederkehrenden Einnahmen profitieren. Wartungsverträge, Schulungen und zusätzliche Supportdienste bieten kontinuierliche Einkommensströme.

Rechnung Die Projektkosten für die Entwicklung des Programms betrugen insgesamt 6002,50€. Das Projekt wurde erfolgreich für 7200€ verkauft. Das ergibt einen direkten Umsatz nach Abnahme des Projekts, von 1197.50 €.

3.2 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 manuellen 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.

3.3 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.

3.4 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.

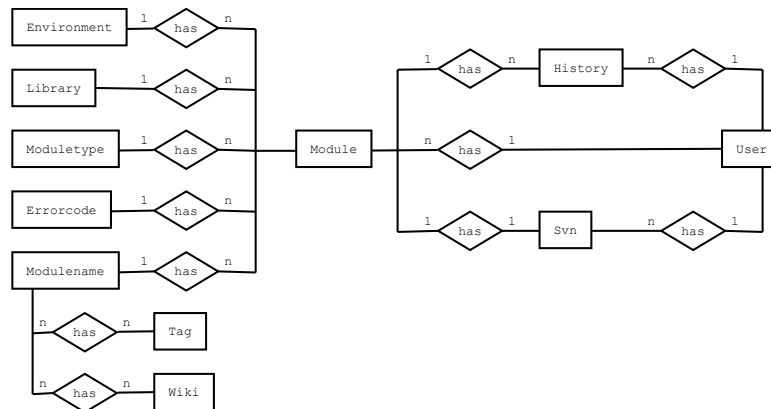


Abbildung 1: Vereinfachtes ER-Modell

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?

Beispiel Ein Klassendiagramm, welches die Klassen der Anwendung und deren Beziehungen untereinander darstellt kann im Anhang A.13: Klassendiagramm auf Seite xxii eingesehen werden.

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

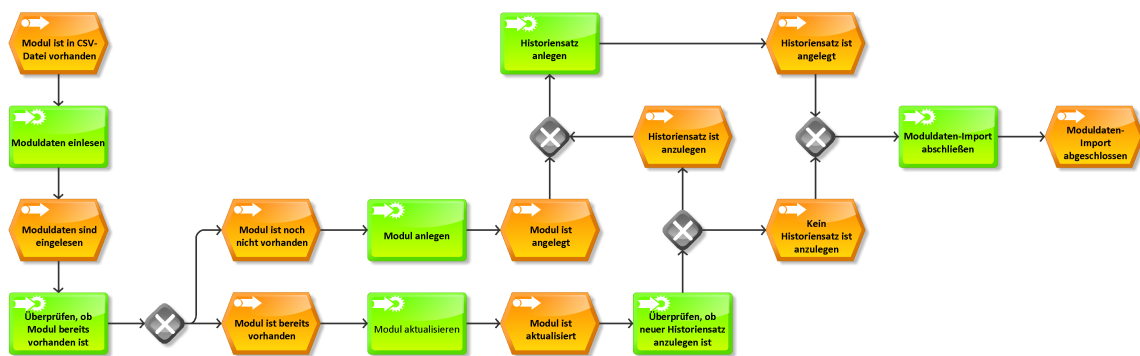


Abbildung 2: Prozess des Einlesens eines Moduls

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 befinden sich im Anhang A.8: Screenshots der Anwendung auf Seite viii.

5.3 Implementierung der Geschäftslogik

Die Generierung der QR-Codes erfolgte mithilfe der Bibliothek `'phpqrcode'`. Eine speziell dafür entwickelte PHP-Klasse war für die Erstellung der QR-Codes verantwortlich. Bei jeder abgeschlossenen Bestellung generiert diese Klasse einen eindeutigen QR-Code, der in der Datenbank gespeichert wird. Die E-Mail-Versandfunktion ist bereits in dem Isotope-Shop-Plugin vorhanden.

Für die Einlösung und Verifizierung der QR-Codes wurde eine separate PHP-Klasse entwickelt. Diese Klasse überprüft die Gültigkeit der gescannten QR-Codes und ruft die entsprechenden Bestellinformationen aus der Datenbank ab.

Im Frontend kam die Bibliothek `'html5-qrcode'` zum Einsatz, die das Scannen der QR-Codes über die Kamera eines mobilen Endgeräts ermöglicht.

Die Anwendung stellt sicher, dass nur gültige und nicht bereits eingelöste QR-Codes akzeptiert werden. Nach erfolgreicher Verifizierung markiert die Anwendung den QR-Code als eingelöst, um eine doppelte Nutzung zu verhindern.

Die Anwendung wurde so strukturiert, dass eine API-Schnittstelle die Kommunikation zwischen Frontend und Backend ermöglicht. Diese API wurde verwendet, um Anfragen vom Frontend zu verarbeiten und entsprechend Tickets zu entwerten oder Bestellungen auszugeben.

Das Frontend selbst wurde mit HTML5-Templates und SCSS für das Styling entwickelt. JavaScript steuert die interaktiven Elemente der Anwendung.

Nach der Implementierung der einzelnen Komponenten wurden diese von internen Mitarbeitern getestet, die nicht direkt am Projekt beteiligt waren. Sie führten umfangreiche Anwendertests durch.

Zusätzlich wurden im Symfony-Debug-Modus Fehlerüberprüfungen durchgeführt und die Symfony-Logdateien analysiert, um sicherzustellen, dass keine versteckten Fehler vorhanden waren.

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

6 Abnahmephase

In der Abnahmephase wurde das Programm ausführlich getestet, um sicherzustellen, dass es allen Anforderungen entspricht und zuverlässig funktioniert. Verschiedene Testarten wurden durchgeführt, wobei der Fokus auf Anwendertests lag. Diese Tests wurden von projektfremden, internen Mitarbeitern und dem Entwickler durchgeführt, die das Frontend des Programms aus der Sicht der Endbenutzer testeten.

Die Anwendertests umfassten die Überprüfung aller Hauptfunktionen des Programms, einschließlich der Generierung und Verwaltung von QR-Codes, der Verifizierung und Einlösung von Tickets sowie der Benutzerfreundlichkeit der Oberfläche. Diese Tests stellten sicher, dass das Programm intuitiv und benutzerfreundlich ist und dass alle interaktiven Elemente korrekt funktionieren. Die Testergebnisse waren insgesamt positiv, wobei es einen Verbesserungsvorschlag bezüglich der Benutzerbarkeit gab.

Zusätzlich zu den Anwendertests wurde die Anwendung im Symfony-Debug-Modus überprüft. Dieser Modus ermöglichte eine detaillierte Analyse und Diagnose möglicher Fehler, die im Frontend nicht sofort sichtbar sind. Die Symfony-Logdateien wurden ebenfalls sorgfältig untersucht, um sicherzustellen, dass keine schwerwiegenden Fehler oder Probleme im Backend der Anwendung vorliegen. Diese Überprüfungen bestätigten, dass die Anwendung stabil und zuverlässig ist.

Die Ergebnisse der Anwendertests wurden dokumentiert. Alle identifizierten Fehler wurden behoben, und die finalen Tests zeigten, dass das Programm die festgelegten Anforderungen erfüllt.

Nach Abschluss der Tests und Korrekturen wurde das Projekt offiziell vom Projektverantwortlichen abgenommen. Die Abnahme erfolgte mündlich und bestätigte, dass das Programm die gestellten Anforderungen erfüllt und betriebsbereit ist. Anschließend wurde das Projekt an einen anderen Entwickler zur weiteren Wartung und Erweiterung übergeben.

Das Protokoll des finalen Frontend-Tests, welcher vor Abnahme durchgeführt wurde, befindet sich im Anhang A.10: Testprotokoll auf Seite xvi.

7 Dokumentation

Für die Benutzer der Anwendung wurde eine Benutzerdokumentation erstellt. Diese Dokumentation enthält detaillierte Anweisungen zur Nutzung der verschiedenen Funktionen der Anwendung, wie das Scannen und Einlösen von QR-Codes, sowie Tipps zur Fehlerbehebung und zum Umgang mit möglichen Problemen. Ziel war es, die Benutzer schrittweise durch die Anwendung zu führen und ihnen zu ermöglichen, das Programm auch ohne großen Schulungsaufwand zu verstehen.

Ein Teil der Benutzerdokumentation ist direkt im Programm implementiert, sodass Benutzer schnell darauf zugreifen können.

Für die Entwickler wurde der Quellcode mit PHPDoc-Kommentaren versehen. Diese Kommentare bieten eine klare und präzise Beschreibung der Funktionen, Klassen und Methoden im Code. PHPDoc erleichtert es den Entwicklern, den Code zu verstehen, zu warten und bei Bedarf zu erweitern.

Es wurde keine separate Entwicklerdokumentation erstellt. Stattdessen wurde darauf geachtet, dass der Code selbst gut strukturiert und kommentiert ist.

Ein Beispiel der dokumentierten PHP-Funktionen befindet sich im Anhang A.9: Entwicklerdokumentation auf Seite xiv.

8 Fazit

8.1 Soll-/Ist-Vergleich

Das Projektziel, ein QR-Code-basiertes Einlösungsprogramm zu entwickeln, wurde in vollem Umfang erreicht. Alle funktionalen Anforderungen wurden erfolgreich umgesetzt. Das System generiert zuverlässig eindeutige QR-Codes für jede Bestellung, ermöglicht eine einfache und sichere Einlösung der Tickets und bietet eine benutzerfreundliche Oberfläche.

Der Auftraggeber ist mit dem Projektergebnis äußerst zufrieden. Die Anwendung wurde erfolgreich in Betrieb genommen und erfüllt alle gestellten Anforderungen und Erwartungen.

In Bezug auf die Projektplanung gab es nur minimale Abweichungen. Die Dokumentation des Projekts nahm etwas mehr Zeit in Anspruch als ursprünglich geplant. Dieser zusätzliche Zeitaufwand war notwendig, um sicherzustellen, dass alle Aspekte des Projekts umfassend und verständlich dokumentiert wurden.

Insgesamt kann das Projekt als erfolgreich abgeschlossen betrachtet werden. Die entwickelten Lösungen entsprechen den Anforderungen und Erwartungen des Auftraggebers, und die wenigen Abweichungen in der Zeitplanung haben das Gesamtergebnis nicht negativ beeinflusst.

Wie in Tabelle 3 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

Phase	Geplant	Tatsächlich	Differenz
Analysephase	5 h	5 h	
Planungsphase	10 h	10 h	
Implementierungsphase	40 h	37.5 h	-2.5 h
Kontrolle	25 h	27.5 h	+2.5 h
Gesamt	80 h	80 h	

Tabelle 3: Soll-/Ist-Vergleich

8.2 Lessons Learned

Es hat sich gezeigt, dass die Dokumentation eines Projekts sehr zeitintensiv ist und sorgfältige Planung erfordert, um alle Aspekte umfassend und präzise darzustellen.

Diese Erfahrung hat den Prüfling dazu veranlasst, zukünftig mehr Zeit für die Dokumentationsphase einzuplanen und diese bereits in der frühen Planungsphase des Projekts stärker zu berücksichtigen.

8.3 Ausblick

Es wird davon ausgegangen, dass das Projekt zukünftig in mehrere Online-Shops eingebunden wird. Die Flexibilität und Kompatibilität der Anwendung mit Isotope-Shops ermöglichen eine einfache Integration.

Dank der modularen Struktur und der Nutzung von Symfony-Standards kann das Programm relativ einfach an die spezifischen Bedürfnisse und Anforderungen unterschiedlicher Online-Shops angepasst werden.

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	5 h
1. Analyse des Ist-Zustands	1.5 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h
3. Erstellen eines „Use-Case“-Diagramms	1 h
4. Erstellen des Lastenhefts	1.5 h
Planungsphase	10 h
1. Prozessentwurf	5 h
2. Datenbankentwurf	1 h
2.1. ER-Modell erstellen	1 h
3. Erstellen des Pflichtenhefts	3 h
Implementierungsphase	37.5 h
1. Anlegen der Datenbank	2.5 h
2. Umsetzung der HTML-Oberflächen und Stylesheets	20 h
3. Programmierung der PHP-Module für die Funktionen	15 h
Kontrolle	27.5 h
1. Erstellen der Benutzerdokumentation	2.5 h
2. Erstellen der Projektdokumentation	22 h
3. Entwicklerdokumentation / Code-Dokumentation	1.5 h
4. Abnahme Fachabteilung	2.5 h
Gesamt	80 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)

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 Problembeseitigung 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

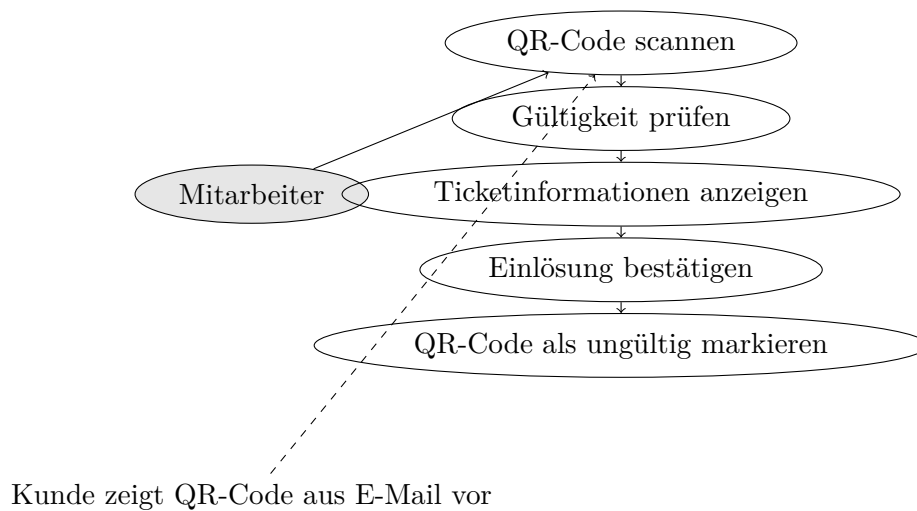


Abbildung 3: Use-Case-Diagramm: Einlösung des QR-Codes durch einen Mitarbeiter

A.5 Pflichtenheft (Auszug)

Zielbestimmung

1. Generierung und Verwaltung von QR-Codes

- 1.1. Das Programm wird eine Funktion zur automatischen Generierung eines einzigartigen QR-Codes für jede abgeschlossene Bestellung implementieren.
- 1.2. Die QR-Codes werden mithilfe einer Bibliothek namens `php-qrcode` erstellt. Diese Bibliothek ermöglicht es eine Zeichenkette in einen QR-Code zu verwandeln.
- 1.3. Jeder generierte QR-Code wird in der Datenbank gespeichert. Dazu wird eine Tabelle `tl_iso_order_hashes` verwendet, die die QR-Codes zusammen mit der zugehörigen Bestell-ID und weiteren Metadaten speichert.
- 1.4. Die generierten QR-Codes werden automatisch per E-Mail an die Kunden versendet. Dafür wird eine E-Mail-Versandfunktion integriert, die die QR-Codes als Anhang oder Bild in die E-Mail einfügt.

2. Einlösung und Verifizierung von QR-Codes

- 2.1. Das Programm wird eine Scanfunktion bereitstellen, die mithilfe der Bibliothek `html5-qrcode` die QR-Codes mit der Kamera eines mobilen Endgeräts scannt.
- 2.2. Die Gültigkeit der gescannten QR-Codes wird überprüft, indem der QR-Code mit den in der Datenbank gespeicherten Hashes abgeglichen wird. Es wird sichergestellt, dass der QR-Code gültig und nicht bereits eingelöst ist.

- 2.3. Nach dem Scannen eines QR-Codes zeigt das Programm die relevanten Bestellinformationen an, die aus der Datenbank abgerufen werden. Diese Informationen umfassen die Bestellnummer, den Namen des Kunden und die Details der Bestellung.
- 2.4. Die Einlösung der QR-Codes wird im System markiert, indem der Status des QR-Codes in der Datenbank auf "eingelöst" gesetzt wird. Dies verhindert die erneute Nutzung des gleichen QR-Codes.
- 2.5. Das Programm stellt sicher, dass ungültige oder bereits eingelöste QR-Codes nicht akzeptiert werden, indem es eine entsprechende Fehlermeldung anzeigt.

3. Benutzeroberfläche und Usability

- 3.1. Die Benutzeroberfläche wird mithilfe von HTML5-Templates und SCSS entwickelt. Diese Templates werden so gestaltet, dass sie intuitiv und benutzerfreundlich sind.
- 3.2. Die Navigation innerhalb des Programms wird klar und logisch strukturiert, sodass Benutzer leicht zwischen den verschiedenen Funktionen wechseln können.
- 3.3. Fehlermeldungen werden klar formuliert und hilfreich gestaltet, um den Benutzern die Lösung von Problemen zu erleichtern. Für das anzeigen dynamischer Fehlermeldungen wird JavaScript verwendet.

4. Sicherheit und Zugriffskontrolle

- 4.1. Das Programm wird Maßnahmen zum Schutz vor gefälschten und manipulierten QR-Codes integrieren.
- 4.2. Es wird sichergestellt, dass nur autorisierte Mitarbeiter Zugang zum Programm haben. Dies wird durch die Mitgliedsverwaltung in Contao realisiert, die den Zugriff auf bestimmte Funktionen und Daten einschränkt.
- 4.3. Die Zugriffskontrolle verhindert, dass unbefugte Personen sensible Bestellinformationen einsehen oder Tickets versehentlich entwerten können.

5. Integrationsfähigkeit

- 5.1. Das Programm wird als Symfony Bundle entwickelt, das sich nahtlos in die bestehende Contao-Umgebung integrieren lässt. Dies erleichtert die Installation und Wartung.
- 5.2. Die Struktur des Programms entspricht den Anforderungen eines Symfony Bundles, sodass es problemlos in andere Projekte eingebunden werden kann.

6. Zuverlässigkeit und Effizienz

- 6.1. Das Programm wird so entwickelt, dass es zuverlässig und korrekt arbeitet. Dies wird durch manuelle Frontend-Tests sichergestellt.

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 \LaTeX zeichnen, siehe z. B. <http://www.texample.net/tikz/examples/entity-relationship-diagram/>.

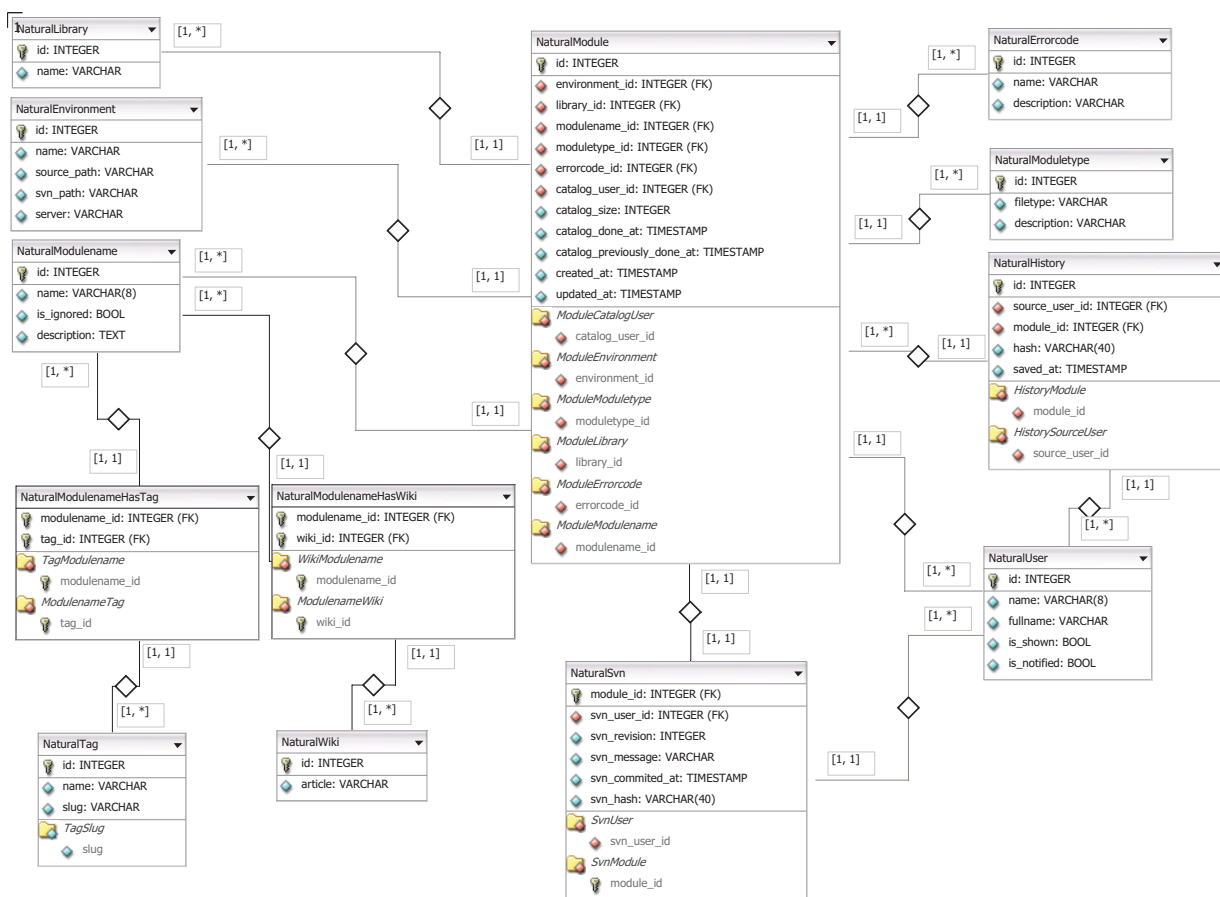


Abbildung 4: Datenbankmodell

A.7 Screenshots des Codes

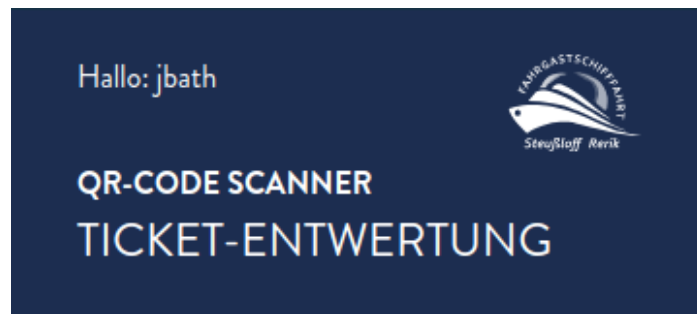
A.7.1 DCA

A screenshot of a code editor with a dark background and light-colored text. The code is PHP, defining a table structure for 'tl_iso_order_hashes'. It includes comments for configuration, palettes, and fields. The 'fields' section defines 'id' as a primary key, 'hash' as a varchar, 'iso_product_collection_id' as an integer, and 'valid' as a boolean.

```
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

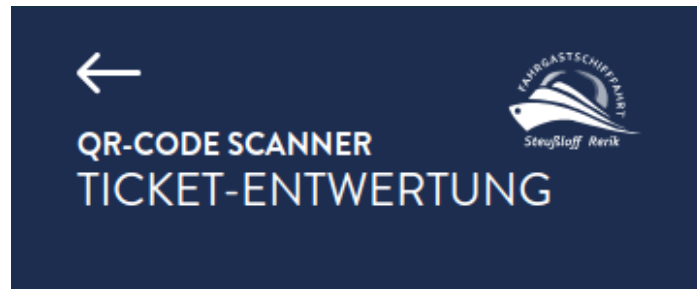
Haffrundfahrt
Jannick Bath
Ticketnummer: 0008

Tickets:

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

TICKET ENTWERTEN

Abbildung 7: Ansicht zum einlösen von 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

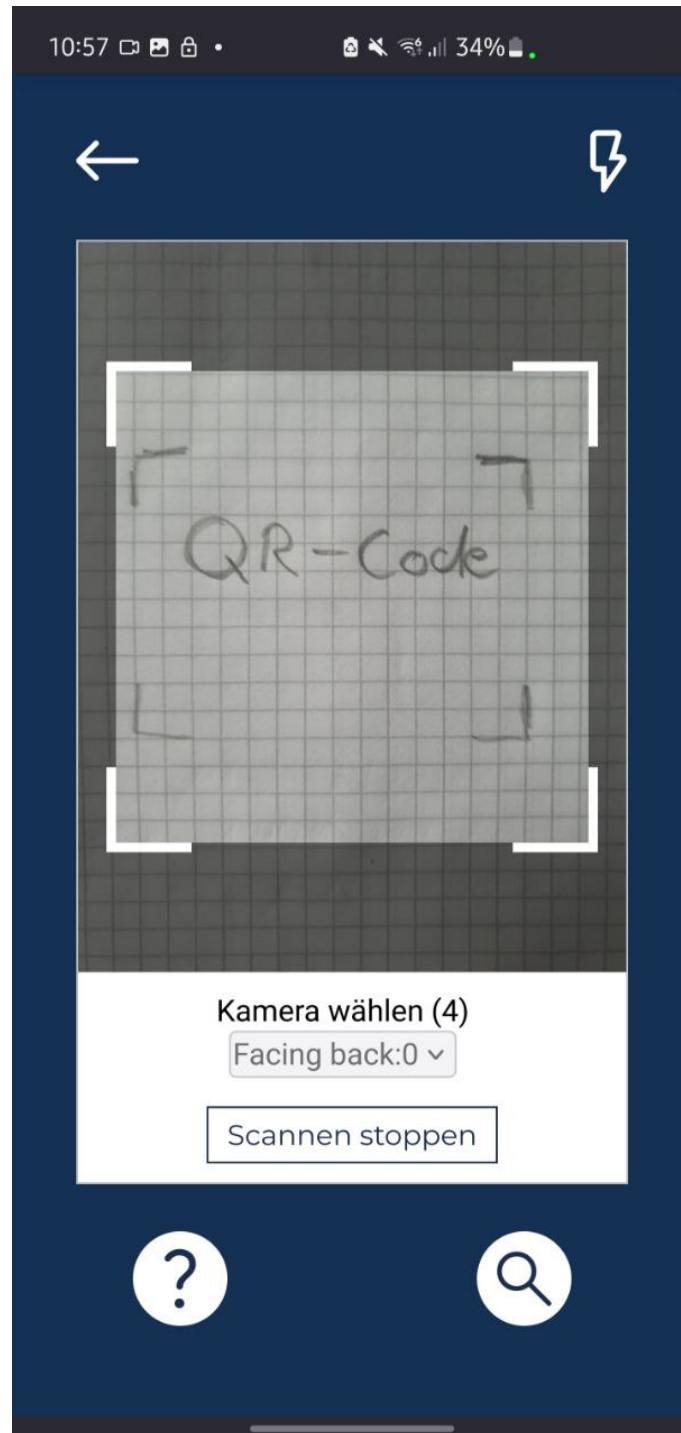


Abbildung 10: Ansicht für das einscannen eines QR-Codes

A.9 Entwicklerdokumentation

```
1  /**
2   * Retrieves detailed order information based on a valid hash.
3   *
4   * If the provided hash is non-empty and valid, this method performs a SQL query across multiple tables
5   * to fetch and return order details. Returns an empty array if the hash is invalid or empty.
6   *
7   * @param string $hash The hash used to lookup order details.
8   * @return array The array of order details; empty if no valid entry is found.
9   */
10 public function getOrderDetailsByHash(string $hash): array {...}
```

```
1  /**
2   * Generates and parses a template using provided data.
3   *
4   * This static method creates a BackendTemplate object with the specified template name,
5   * sets the provided data and returns the parsed content of the template.
6   *
7   * @param string $templateName The name of the template to be used.
8   * @param array $templateVars An associative array of variables to be used in the template.
9   * @return string The parsed content of the generated template.
10  */
11 public static function generateTemplate($templateName, $templateVars) {...}
```

```
1  /**
2   * Hashes a given string using the SHA-256 algorithm.
3   *
4   * @param string $str The string to hash.
5   * @return string The SHA-256 hash of the input string.
6   */
7  private function hash($str) {...}
```


A.10 Testprotokoll

Testprotokoll für das QR-Code-basierte Einlösungsprogramm

Tester: Kristin Rietentiet

Datum: 23.05.2024

Version: 0.0.14

Einleitung

Dieses Testprotokoll dokumentiert die Ergebnisse der Frontend-Tests des QR-Code-basierten Einlösungsprogramms. Der Zweck des Tests ist es, sicherzustellen, dass die Benutzeroberfläche des Programms alle funktionalen und nicht-funktionalen Anforderungen erfüllt und frei von Fehlern ist.

Testumgebung

- Gerät: Samsung Galaxy A41 (Android)
- Browser: Firefox

Testfälle

1. Starten der Anwendung

Testfall-ID: TF001

Beschreibung: Öffnen der Anwendung über die gespeicherte Verknüpfung auf dem Home-Bildschirm. Via Website geöffnet. **Erwartetes Ergebnis:** Die Anwendung startet ohne Fehler und zeigt das Hauptmenü fehlerfrei an.

Ergebnis: Bestanden **Bemerkungen:** Anmeldung erfolgte ohne Probleme. Auch bei Wiederholung.

2. QR-Code scannen

Testfall-ID: TF002

Beschreibung: Öffnen des QR-Code-Scanners und Scannen eines gültigen QR-Codes.

Erwartetes Ergebnis: Der QR-Code wird erfolgreich gescannt, und die Ticketinformationen werden angezeigt.

Ergebnis: Bestanden **Bemerkungen:** Die Kamera wird nicht direkt beim Klick auf die Schaltfläche geöffnet, man muss die Kamera (Front/Back) jedes Mal auswählen

3. Ticket entwerten

Testfall-ID: TF003

Beschreibung: Entwerten eines gescannten Tickets.

Erwartetes Ergebnis: Das Ticket wird erfolgreich entwertet, und der QR-Code wird als eingelöst markiert und so auch in der Ticketliste angezeigt.

Ergebnis: Bestanden **Bemerkungen:** Das Entwerten des QR-Codes geschieht schnell und auch bei mehreren Versuchen ohne Probleme.

4. Fehlerhafte QR-Codes scannen

Testfall-ID: TF004

Beschreibung: Scannen eines ungültigen oder bereits eingelösten QR-Codes.

Erwartetes Ergebnis: Eine Fehlermeldung wird angezeigt, die den Benutzer darauf hinweist, dass der QR-Code ungültig oder bereits eingelöst ist.

Ergebnis: Bestanden **Bemerkungen:** QR-Codes werden fehlerfrei erkannt.

5. Manuelle Suche eines Tickets

Testfall-ID: TF005

Beschreibung: Manuelle Suche nach einem Ticket anhand der Ticketnummer.

Erwartetes Ergebnis: Das Ticket wird korrekt in der Ticketliste gefunden und kann entwertet werden.

Ergebnis: Bestanden **Bemerkungen:** Die Suche nach einem Ticket ist einfach und kann via Datum, Nummer oder Namen erfolgen.

6. Benutzeroberfläche und Usability

Testfall-ID: TF006

Beschreibung: Überprüfung der Benutzeroberfläche auf Intuitivität und Benutzerfreundlichkeit.

Erwartetes Ergebnis: Die Benutzeroberfläche ist intuitiv und einfach zu navigieren, alle Elemente sind klar beschriftet und gut sichtbar.

Ergebnis: Bestanden **Bemerkungen:** Die Benutzeroberfläche ist so gebaut, dass es keinen Raum für Verwirrung gibt. Der Scan passiert nicht sofort, man muss die Kamera zuerst starten, manchmal zweimal weil man zuerst eine Zustimmung zur Kameranutzung geben muss.

7. Fehlermeldungen

Testfall-ID: TF007

Beschreibung: Überprüfung der angezeigten Fehlermeldungen bei verschiedenen Fehlerzuständen.

Erwartetes Ergebnis: Fehlermeldungen sind klar formuliert und enthalten hilfreiche Informationen zur Behebung des Problems.

Ergebnis: Bestanden **Bemerkungen:** Fehlermeldung sind kurz und deutlich formuliert.

8. Sicherheitsüberprüfung

Testfall-ID: TF008

Beschreibung: Überprüfung der Zugriffskontrollen und Sicherheit beim Zugriff auf sensible Daten.

Erwartetes Ergebnis: Nur autorisierte Benutzer können auf die Anwendung zugreifen und Tickets entwerten.

Ergebnis: Bestanden **Bemerkungen:** Der Nutzer muss einen Account besitzen um auf die Anwendung zugreifen zu können.

Zusammenfassung der Testergebnisse

- **Bestandene Tests:** 8/8
- **Fehlgeschlagene Tests:** 0/8
- **Gesamtbewertung:** Die Anwendung wurde nach den bestehenden Vorgaben umgesetzt.

Empfehlungen

Es gibt minimalen Verbesserungsbedarf in der Nutzerfreundlichkeit. Die Anwendung ist einfach zu verstehen und einfach in der Handhabung. Nach dem Klick auf die "Ticket-Scannen" Schaltfläche soll die Kamera direkt starten.

Abschlussbemerkungen

Das QR-Code-basierte Einlösungsprogramm wurde umfassend getestet. Alle notwendigen und angelegten Funktionen funktionieren und erfüllen so Ihren Zweck wie gewünscht. Nutzerfreundlichkeit kann minimal verbessert werden, wie in der Empfehlung zur Verbesserung bereits erwähnt.

Kristin Rietentiet 23.05.2024

A.11 Klasse: TicketRoutes

Verkürzte Variante. Kommentare und Imports werden nicht angezeigt.

```

1 <?php
2
3 namespace Lupcom\IsotopeQrCodeBundle\Controller;
4
5 class TicketRoutes
6 {
7     private $security;
8     private $templateArgs = [];
9
10    public function __construct(Security $security) {
11        $this->security = $security;
12    }
13
14    #[Route('/einloesen', name: "einloesen", defaults: ["_scope" => "frontend"])]
15    public function einloesen(Request $request): Response
16    {
17        if ($this->isAuthenticated()) {
18            return $this->accessGranted($request);
19        }
20
21        return $this->accessDenied($request);
22    }
23
24    #[Route('/einloesen/scanner', name: "einloesen_scanner", defaults: ["_scope" => "frontend"])]
25    public function scanner(Request $request) {
26        if ($this->isAuthenticated()) {
27            return $this->generateResponse("fe_tool_scanner");
28        }
29
30        return $this->accessDenied($request);
31    }
32
33    #[Route('/einloesen/suche', name: "einloesen_suche", defaults: ["_scope" => "frontend"])]
34    public function suche(Request $request) {
35        if ($this->isAuthenticated()) {
36            return $this->generateResponse("fe_tool_suche");
37        }
38
39        return $this->accessDenied($request);
40    }
41
42    private function accessDenied(Request $request): Response {
43        $container = \Contao\System::getContainer();
44        $legacyRouting = true;
45        $urlSuffix = "";
46        $loginPage = PageModel::findByIdOrAlias("login", ["having" => "published='1'"]);
47

```

```

48     if ($container->hasParameter("contao.legacy_routing")) {
49         $legacyRouting = $container->getParameter('contao.legacy_routing');
50     }
51
52     if ($legacyRouting) {
53         $urlSuffix = \Contao\Config::get("urlSuffix");
54     } else if (!empty($loginPage)) {
55         $urlSuffix = $loginPage->urlSuffix;
56     }
57
58     if (!empty($loginPage)) {
59         return new RedirectResponse("/" . $loginPage->alias . $urlSuffix);
60     }
61
62     return $this->generateResponse("fe_access_denied");
63 }
64
65 private function isAuthenticated(): bool {
66     if ($this->security->isGranted('IS_AUTHENTICATED_FULLY')) {
67         $this->addTemplateArgument("username", $this->security->getUser()->username);
68         return true;
69     }
70
71     return false ;
72 }
73
74 private function generateResponse($template, $args = []) {
75     return new Response(helper::generateTemplate($template, [...$args, ... $this->templateArgs]));
76 }
77
78 private function addTemplateArgument($key, $value) {
79     $this->templateArgs = [...$this->templateArgs, $key => $value];
80 }
81 }

```

Listing 1: Klasse: TicketRoutes

A.12 Klasse: TLOrderHashes

Verkürzte Variante. Kommentare und Imports werden nicht angezeigt.

```

1 <?php
2
3 namespace Lupcom\IsotopeQrCodeBundle\Classes;
4
5 class TLOrderHashes
6 {
7     private $db;
8     private $tableName = "tl_iso_order_hashes";
9
10    public function __construct() {
11        $this->db = Database::getInstance();
12    }
13
14    public function addEntry($hash, $order_id, $valid = true) {
15        $this->db->prepare("INSERT INTO {$this->tableName} (hash, iso_product_collection_id, valid)
16            VALUES (?, ?, ?);")->execute([$hash, $order_id, $valid]);
17    }
18
19    public function getOrdersBySearch($search): array {
20        $sql = "SELECT pc.id, oh.hash, oh.valid, pc.document_number, DATE_FORMAT(FROM_UNIXTIME(pc.
21            tstamp), '%d.%m.%Y') AS date, ia.firstname, ia.lastname, ia.email
22        FROM {$this->tableName} AS oh
23        JOIN tl_iso_product_collection AS pc ON oh.iso_product_collection_id=pc.id
24        JOIN tl_iso_product_collection_item AS pci ON pc.id=pci.pid
25        JOIN tl_iso_address AS ia ON pc.id=ia.pid
26        WHERE pc.document_number LIKE CONCAT('%', ?, '%')
27            OR ia.firstname LIKE CONCAT('%', ?, '%')
28            OR ia.lastname LIKE CONCAT('%', ?, '%')
29            OR DATE_FORMAT(FROM_UNIXTIME(pc.tstamp), '%d.%m.%Y') LIKE CONCAT('%', ?, '%')
30        GROUP BY pc.id, oh.hash, oh.valid, pc.document_number, date, ia.firstname, ia.lastname, ia.email
31        ORDER BY date DESC;";
32        $result = $this->db->prepare($sql)->execute([$search, $search, $search, $search])->fetchAllAssoc();
33
34        return $result;
35    }
36
37    public function checkIfHashExists($hash) {
38        if (empty($hash)) return false;
39
40        $result = $this->db->prepare("SELECT id FROM {$this->tableName} WHERE hash=?")->execute([
41            $hash])->fetchAllAssoc();
42
43        if (!empty($result)) {
44            return true;
45        }
46
47        return false;
48    }
49

```

```
45     }
46
47     public function checkIfHashIsValid($hash) {
48         if (empty($hash)) return false;
49
50         $result = $this->db->prepare("SELECT valid FROM {$this->tableName} WHERE hash=?")->execute([
51             $hash])->fetchAllAssoc();
52
53         if (($result[0]["valid"] ?? 0) == 1) {
54             return true;
55         }
56
57         return false ;
58     }
59
60     public function redeemTicket($hash) {
61         if (empty($hash)) return false;
62
63         $result = $this->db->prepare("UPDATE {$this->tableName} SET valid=0 WHERE hash=?")->execute(
64             [$hash]);
65
66         if (!empty($result)) {
67             return true;
68         }
69
70         return false ;
71     }
72 }
```

Listing 2: Klasse: TLOrderHashes

A.13 Klassendiagramm

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

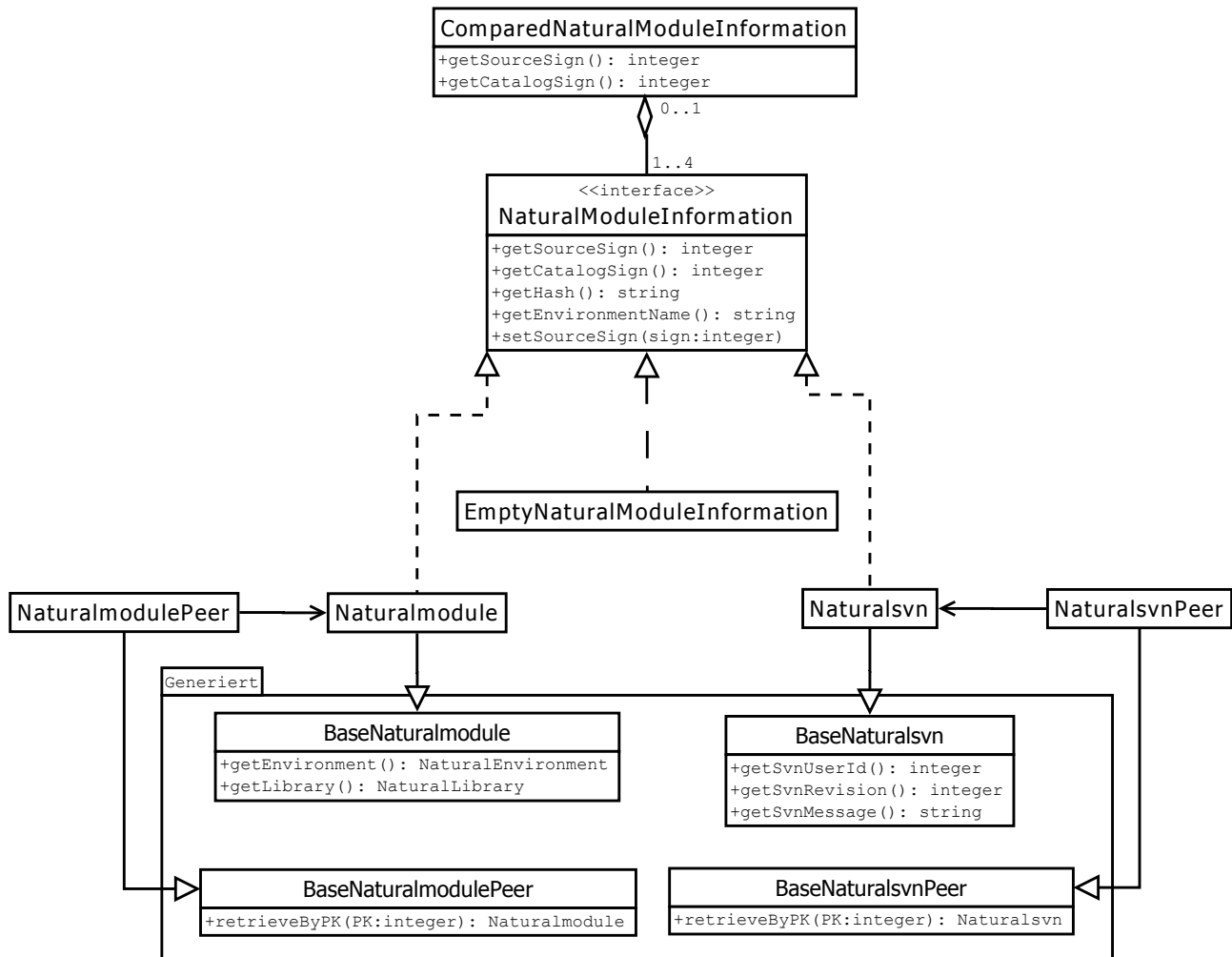


Abbildung 11: Klassendiagramm

A.14 Benutzerdokumentation

A.14.1 Starten des Programms

Öffnen Sie den Webbrowser Ihrer Wahl und navigieren Sie zu [https://\[domain\]/einloesen](https://[domain]/einloesen). Wenn Sie nicht eingeloggt sind, dann werden Sie nun zur Login-Seite weitergeleitet. Melden Sie sich hier mit Ihrem Mitgliedszugang von Contao an.

Nachdem Sie sich erfolgreich eingeloggt haben, werden Sie zur Startseite weitergeleitet.

Stellen Sie sicher, dass Ihre Kamera und Internetverbindung aktiviert sind, um alle Funktionen der Anwendung nutzen zu können.

A.14.2 Scannen eines QR-Codes

QR-Code-Scanner öffnen: Klicken sie auf der Startseite auf das QR-Code-Icon um den Scan-Prozess zu starten.

Kamera ausrichten: Richten Sie die Kamera Ihres Geräts auf den QR-Code des Tickets. Achten Sie darauf, dass der QR-Code vollständig im Sichtfeld der Kamera ist.

QR-Code scannen: Das Programm scannt den QR-Code automatisch und leitet Sie zur Ticketinformation weiter. Hier sehen Sie alle relevanten Details zur Bestellung.

A.14.3 Entwertung des Tickets

Nachdem der QR-Code erfolgreich gescannt wurde, können Sie das Ticket entwerten:

Ticketinformationen überprüfen: Stellen Sie sicher, dass die angezeigten Informationen mit den Daten des Gastes übereinstimmen.

Ticket entwerten: Klicken Sie auf die Schaltfläche "Ticket entwerten", um den QR-Code im System als eingelöst zu markieren.

A.14.4 Fehlerbehebung beim Scannen

Sollte der Scan fehlschlagen, gibt es einige Ansätze, um das Problem zu beheben:

Lichtverhältnisse anpassen: Starkes Sonnenlicht kann den Bildschirm eines Handys schwer erkennbar machen. Stellen Sie sich, wenn nötig, in den Schatten, um den QR-Code besser sichtbar zu machen.

QR-Code vergrößern: Wenn der QR-Code zu klein dargestellt wird, können Sie den Bildschirm vergrößern, um den QR-Code besser erkennbar zu machen. Nutzen Sie die Zoom-Funktion Ihres Geräts.

A.14.5 Manuelle Suche für Tickets

Falls der Scan-Prozess gerade nicht funktioniert, haben sie die Möglichkeit manuell nach Tickets zu suchen. Hierbei kann nach Datum, Bestellnummer oder Name gesucht werden.

Zur Ticketliste wechseln: Klicken Sie auf der Hauptseite oder im QR-Code-Scanner auf den Button mit der Lupe, um zur Ticketliste zu gelangen.

Ticketnummer eingeben: Geben Sie die Ticketnummer, Datum oder den Namen des Gastes in die Suchleiste ein.

Ticket entwerten: Wählen Sie das entsprechende Ticket aus der Liste aus und klicken Sie auf "Ticket entwerten", um es manuell zu entwerten.