

Bachelorarbeit

Design und Entwicklung einer REST-API für eine verteilte Gutscheinverwaltung

Betreuer: Prof. Dr. Alexandre de Spindler

Student: Raphael Prader
praderap@students.zhaw.ch, raphael.prader@gmail.com

Departement: ZHAW School of Management and Law

Studiengang: Wirtschaftsinformatik

Klasse: WIN-15-VZB

Immatrikulationsnummer: 15-536-709

Durchführungszeitraum: 19.02.2018 bis 24.05.2018

Abgabedatum: 24.05.2018

Management Summary

Verschiedene Geschäfte aus dem Grossraum Ostschweiz haben 2016 ein gemeinsames Gutscheineheft für ihre Kunden lanciert. Das Prinzip des Gutscheins funktioniert so, dass alle Einkäufe bei den teilnehmenden Geschäften von den Kunden gesammelt werden, wobei beim letzten Einkauf ein individueller Gutschein ausgestellt wird. Der gewährte Gutscheinbetrag ist dabei abhängig vom Gesamteinkaufsbetrag und wird beim letzten Einkauf abgezogen. An einem eingelösten Gutschein beteiligen sich jene Unternehmen, bei denen die jeweilige Kundschaft eingekauft hat, wobei das zuletzt besuchte Unternehmen zwangsläufig in Vorleistung geht und den Rabatt vom Einkauf in Abzug bringt. Bis anhin haben die Geschäfte in unregelmässigen Abständen untereinander abgerechnet, was mit grossem manuellem Aufwand verbunden war.

Diese Arbeit beantwortet die Frage, wie sich die manuellen Aufwände mit einer Softwarelösung maximal automatisieren und dadurch reduzieren lassen. Um sämtliche Vorgänge so effizient wie möglich abwickeln zu können, wurde die optimale Benutzerführung (Usability) für die Anwendung erarbeitet. Die am Gutscheineheft beteiligten Geschäfte können mit der neuen Lösung die gewährleisteten Gutscheinbeträge untereinander automatisch abrechnen lassen. Der manuelle Rechenaufwand entfällt dabei.

Der Aufbau der gesamten Arbeit folgt dem Wasserfallmodell mit dem Unterschied, dass jedes produzierte Artefakt bereits frühzeitig, anstatt erst zum Schluss, validiert und entsprechend überarbeitet wurde. Der Projektplan startet bei der Konzeption der Mockups und endet mit der Auslieferung der fertigen Software. Sämtliche Arbeitsschritte sind angemessen validiert worden, womit ein fehlerfreier Berechnungsalgorithmus sichergestellt werden kann.

Mit der Auslieferung der neu erstellten Gutscheinverwaltungssoftware können die identifizierten Probleme vollumfänglich gelöst werden. Die identifizierten Probleme sind die fehlende Transparenz unter den Geschäften, das Fehlerpotenzial bei der Abrechnung sowie der für die Abrechnung notwendige manuelle Aufwand. Die entwickelte REST-API ist unbeschränkt skalierbar und bietet die Möglichkeit, die Businesslogik der Gutscheinverwaltung anderen autorisierten Systemen zur Verfügung zu stellen.

Zum Schluss der Arbeit werden weiterführende Anwendungsmöglichkeiten eruiert, die erläutern, wie die erstellte Lösung in Zukunft erweitert werden kann. Dazu lässt sich festhalten, dass die Applikation auch Anwendung in anderen Branchen findet und zudem grosses Potenzial für Erweiterungen bietet. Zusätzlich gewährleistet die eingesetzte DevOps-Strategie mit den automatisierten Unit-Tests eine langfristige Lauffähigkeit der Applikation sowie ein rasches Ausliefern von Anpassungen und Erweiterungen.

Inhaltsverzeichnis

1	Einleitung.....	1
1.1	Aufbau der Arbeit.....	1
1.2	Zielsetzung.....	1
1.2.1	Hauptziele (erwartet).....	1
1.2.2	Übertroffene Ziele (nicht erwartet).....	2
2	Problem.....	3
2.1	Ausgangslage.....	3
2.2	Problemstellungen.....	4
2.2.1	Grosser Zeitaufwand.....	4
2.2.2	Fehlerpotenzial.....	4
2.2.3	Fehlende Transparenz.....	4
2.2.4	Weitere Probleme.....	5
3	Herausforderungen.....	6
3.1	Artefakte.....	6
3.1.1	REST-API (Backend).....	6
3.1.2	Mockups.....	7
3.1.3	Frontend.....	7
3.1.4	Server Deployment.....	7
3.2	Berechnungsgrundlage und Regeln.....	8
3.3	Risiken.....	9
3.3.1	Humanressourcen.....	9
3.3.2	Performance.....	9
3.3.3	Algorithmus.....	10
3.3.4	Komplexität.....	10
4	Nutzen.....	11

4.1	Kundennutzen	11
4.2	Geschäftsnutzen	12
5	Vorgehensweise	13
5.1	Arbeitspaket 1: Frontend Design	13
5.2	Arbeitspaket 2: Validierung Frontend Design	13
5.3	Arbeitspaket 3: Backend Design	14
5.4	Arbeitspaket 4: Backend Umsetzung	14
5.5	Arbeitspaket 5: Validierung Backend	14
5.6	Optionales Arbeitspaket 6: Frontend Umsetzung	15
5.7	Optionales Arbeitspaket 7: DevOps	15
5.8	Projektplan	15
6	Lösung	17
6.1	Frontend Design	17
6.1.1	User Stories	17
6.1.2	Mockups	18
6.1.2.1	Login	19
6.1.2.2	Einkauf verbuchen	20
6.1.2.3	Gutschein einlösen	22
6.1.2.4	Kundenstamm	24
6.1.2.5	Weitere Funktionen	25
6.1.2.6	Laufende Abrechnungsperiode	25
6.1.2.7	Vergangene Abrechnungen	26
6.1.2.8	Häufig gestellte Fragen	26
6.1.2.9	Offene Zahlungen	26
6.1.2.10	PDF-Bestätigungen	26
6.2	Systemarchitektur und Technologiewahl	26

6.2.1	Frontend.....	27
6.2.2	Backend	29
6.2.3	Webserver.....	30
6.2.4	Containervirtualisierung.....	30
6.2.5	Repository.....	31
6.2.5.1	Verzeichnisse im Repository-Root.....	31
6.2.5.2	Dateien im Repository-Root	32
6.3	REST-API.....	32
6.3.1	Was ist REST.....	33
6.3.2	HTTP Statuscodes.....	33
6.3.3	Datenbank Design	34
6.3.3.1	Customer	34
6.3.3.2	Customer Contact.....	34
6.3.3.3	Report Temporary Access.....	35
6.3.3.4	Purchase.....	35
6.3.3.5	Final Discount Rule.....	35
6.3.3.6	Company	35
6.3.3.7	Customer Closure	35
6.3.3.8	Transaction.....	36
6.3.4	API Design und Development.....	36
6.3.5	Authentifizierung mit OAuth2.....	42
6.4	Frontend Umsetzung.....	43
6.4.1	Komponenten.....	43
6.4.2	Interfaces	43
6.4.3	Services.....	44
6.4.4	Routing.....	44

6.4.5	Resultat	45
6.5	DevOps	45
6.5.1	Continuous Integration	46
6.5.1.1	gitlab-ci.yml	46
6.5.1.2	Runner	49
6.5.2	Continuous Delivery	49
6.5.3	Continuous Deployment	50
6.6	Alternative Lösungsmöglichkeiten	50
6.6.1	Gewöhnliche Webseite	50
6.6.2	ReactJS anstelle von Angular	51
7	Validierung	52
7.1	Validierung User Interface	52
7.1.1	Validierung durch den Verfasser	53
7.1.2	Validierung durch Labhart-Chronometrie	53
7.1.3	Validierung durch übrige Teilnehmerfirmen	55
7.2	Validierung REST-API	57
7.2.1	Smoke-Tests	57
7.2.2	Unit-Tests	57
7.2.3	Akzeptanztests	58
7.2.4	Frontend Umsetzung	58
8	Weitergehende Anwendungsmöglichkeiten der Lösung	59
8.1	Applikationsspezifische Hardware	59
8.1.1	Raspberry Pi	59
8.1.2	Hybrid Smartphone App	59
8.2	Anbindung an Drittsysteme	60
8.2.1	Kassensysteme	60

8.2.2	E-Commerce	60
8.3	Retargeting	60
8.4	Branchenunabhängige Lösung.....	61
9	Schlussfolgerungen.....	62
10	Literaturverzeichnis	63
11	Anhang.....	66
11.1	Projektplan	66
11.2	Mockups.....	67
11.3	UML Klassendiagramm.....	78
11.4	Backend Validierung	79
11.5	Frontend Umsetzung.....	80
11.5.1	iFrame	80
11.5.2	View Komponenten	80
11.5.3	Komponenten.....	84
11.6	Zugangsdaten	85
11.6.1	Repository	85
11.6.2	Applikation.....	86

Abbildungsverzeichnis

Abbildung 1: Gantt-Projektplan.....	16
Abbildung 2: Mockup von dem Screen «Login».....	19
Abbildung 3: Mockup von dem Screen «Einkauf verbuchen».....	20
Abbildung 4: Mockup von dem Screen «Einkauf verbucht».....	21
Abbildung 5: Mockup von dem Screen «Gutschein einlösen».....	22
Abbildung 6: Mockup von dem Screen «Abschliessender Einkauf erfassen».....	23
Abbildung 7: Mockup von dem Screen «Kunden bearbeiten».....	24
Abbildung 8: Mockup von dem Screen «Laufende Abrechnungsperiode».....	25
Abbildung 9: Verwendete Paketversionen für das Frontend.....	29
Abbildung 10: Ordnerstruktur des Repository.....	31
Abbildung 11: Grafische Darstellung aller API-Endpunkte.....	38
Abbildung 12: Programmablaufplan für den Authentifizierungsprozess.....	42
Abbildung 13: Definierte Routen für die Applikation.....	45
Abbildung 14: gitlab-ci.yml – CI Konfiguration für das Projekt.....	48
Abbildung 15: Auszug aus dem Kanban-Board auf GitLab zur Mockup-Validierung..	52
Abbildung 16: Detaillierter Projektplan.....	66
Abbildung 17: Mockup von dem Screen «Login».....	67
Abbildung 18: Mockup von dem Screen «Einkauf verbuchen».....	67
Abbildung 19: Mockup von dem Screen «Kunden suchen».....	68
Abbildung 20: Mockup von dem Screen «Kunden erfassen».....	69
Abbildung 21: Mockup von dem Screen «Einkaufsdaten erfasst».....	70
Abbildung 22: Mockup von dem Screen «Einkauf verbucht».....	70
Abbildung 23: Mockup von dem Screen «Gutschein einlösen».....	71
Abbildung 24: Mockup von dem Screen «Abschliessender Einkauf erfassen».....	71
Abbildung 25: Mockup von dem Screen «Letzter Einkauf bereits erfasst».....	72

Abbildung 26: Mockup von dem Screen «Gutschein eingelöst».....	72
Abbildung 27: Mockup von dem Screen «Kundenstamm».....	73
Abbildung 28: Mockup von dem Screen «Kunden bearbeiten».....	74
Abbildung 29: Mockup von dem Screen «Weitere Funktionen».....	75
Abbildung 30: Mockup von dem Screen «Laufende Abrechnungsperiode».....	75
Abbildung 31: Mockup von dem Screen «Offene Zahlungen».....	76
Abbildung 32: Mockup von dem Screen «Vergangene Abrechnungsperioden».....	76
Abbildung 33: Mockup von dem Screen «Häufig gestellte Fragen».....	77
Abbildung 34: Mockup der PDF-Berichte.....	77
Abbildung 35: UML Klassendiagramm von dem Backend.....	78
Abbildung 36: Coverage Report für die REST-API.....	79
Abbildung 37: Ansicht von dem iFrame für die Hochzeitspaare.....	80
Abbildung 38: Ansicht der Seite «Login».....	80
Abbildung 39: Ansicht der Seite «Einkauf verbuchen».....	81
Abbildung 40: Ansicht der Seite «Gutschein einlösen».....	81
Abbildung 41: Ansicht der Seite «Kundenstamm».....	82
Abbildung 42: Ansicht der Seite «Kunde Detail».....	82
Abbildung 43: Ansicht der Seite «Häufig gestellte Fragen».....	83
Abbildung 44: Ansicht der Seite «Laufende Abrechnungsperiode».....	83
Abbildung 45: Komponente «ConfirmTransactionPaymentComponent».....	84
Abbildung 46: Komponente «CreateCustomerComponent» und «EditCustomerComponent».....	84
Abbildung 47: Komponente «SelectCustomerComponent».....	85
Abbildung 48: Darstellung der Komponente «ShowCustomerComponent».....	85
Abbildung 49: Darstellung der Komponente «TablePurchaseRowComponent».....	85

Tabellenverzeichnis

Tabelle 1: User Stories	18
Tabelle 2: Interaktionsmöglichkeiten auf dem Screen «Einkauf verbuchen»	21
Tabelle 3: Interaktionsmöglichkeiten auf dem Screen «Einkauf verbucht».....	22
Tabelle 4: Interaktionsmöglichkeiten auf dem Screen «Abschliessender Einkauf erfassen»	23
Tabelle 5: Interaktionsmöglichkeiten auf dem Screen «Kunden bearbeiten»	25
Tabelle 6: Gewichtete Nutzwertanalyse zur Wahl der Softwareart	27
Tabelle 7: Issues nach Validierung durch den Erfasser.....	53
Tabelle 8: Issues nach Validierung durch Labhart-Chronometrie	55
Tabelle 9: Issues nach Validierung durch Liluca, nisago, lightplay und Blumeria Marbach	56

Abkürzungsverzeichnis

API	Application Programming Interface
APP	Applikation
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
PDF	Portable Document Format
REST	Representational State Transfer
UUID	Universally Unique Identifier
WWW	World Wide Web
YAML	Yet Another Multicolumn Layout

1 Einleitung

In der vorliegenden Arbeit geht es um die Konzeption, Entwicklung und Validierung einer verteilten Gutscheinverwaltungsapplikation. Im Rahmen dieser Arbeit wird die Frage behandelt, welcher Ansatz sowohl aus konzeptioneller als auch aus technischer Sicht der sinnvollste ist, um die identifizierten Probleme zu lösen.

1.1 Aufbau der Arbeit

Aus Gründen der Lesbarkeit wird in dieser Arbeit die männliche Form gewählt, nichtsdestoweniger beziehen sich die Angaben auf Angehörige beider Geschlechter.

Die vorliegende Arbeit folgt im Wesentlichen derselben Struktur wie das Dokument der vorgängig eingereichten Disposition. Im nachfolgenden zweiten Kapitel werden die gegenwärtigen Probleme aufgezeigt, welche die Entwicklung der neuen Lösung legitimieren. In Kapitel 3 werden die Herausforderungen der Problemlösung diskutiert. Kapitel 4 widmet sich dem Nutzen, welcher mit der Bereitstellung von der in Kapitel 6 beschriebenen Lösung resultiert. Vorgängig zur Lösungsbeschreibung zeigt das Kapitel 5 die Vorgehensweise auf, wie die Realisierung dieses Projekts geplant ist. Aufbauend auf den Resultaten aus Kapitel 6 erbringt die Validierung in Kapitel 7 den Nachweis, dass die entwickelte Lösung die gegenwärtigen Probleme tatsächlich löst.

Abschliessend werden in Kapitel 8 weitergehende Anwendungsmöglichkeiten aufgezeigt, bevor in Kapitel 9 die Schlussfolgerungen gezogen werden.

1.2 Zielsetzung

Dieser Abschnitt beschreibt die Ziele dieser Arbeit, wobei diese nachfolgend unterteilt werden in die Kategorien erwartete Hauptziele und übertroffene, also nicht im Rahmen dieser Arbeit erwartete, Ziele.

1.2.1 Hauptziele (erwartet)

Mit der Bearbeitung dieser Bachelorarbeit werden als messbarer Output die Mockups von dem Frontend erwartet, inklusive des Feedbacks aus der Validierung mit den zukünftigen Anwendern. Darauf aufbauend soll eine funktionsfähige REST-API mit entsprechender

Businesslogik erstellt werden. Zur Validierung von der REST-API und deren Berechnungen sollen Unit-Tests erstellt werden, die die korrekte Funktionsweise gewährleisten.

1.2.2 Übertroffene Ziele (nicht erwartet)

Die in Kapitel 6.4 beschriebene Umsetzung von dem kompletten Frontend stellt ein übertroffenes Ziel dar. Ebenso wird mit der Implementierung von DevOps in Kapitel 6.5 ein nicht erwartetes Ziel erreicht, das der Lösung als Ganzes eine deutliche Wertsteigerung einbringt.

2 Problem

Dieses Kapitel zeigt die Ist-Situation auf und beschreibt anschliessend die Nachteile der bestehenden Lösung. Dazu werden die drei Hauptprobleme identifiziert und erläutert.

2.1 Ausgangslage

Der Verein «Wedding Award Switzerland» zeichnet seit 2016 jährlich die besten Schweizer Dienstleister im Hochzeitsbereich aus, die mittels Publikumsvoting bestimmt werden (Wedding Award Switzerland, o. J.-a). Das Ziel der Veranstalter ist es, die Hochzeitsbranche zu fördern und deren Dienstleister zu innovativen und kreativen Leistungen anzuspornen. Gleichzeitig soll ein Qualitätssiegel etabliert werden, das Brautpaaren bei der Auswahl ihrer Dienstleistungspartner helfen soll (Wedding Award Switzerland, o. J.-b).

Fünf Dienstleister aus unterschiedlichen Bereichen haben sich nach ihrer Nominierung im Jahr 2016 zusammengeschlossen, um ein gemeinsames Gutscheineheft zu lancieren. Bei diesen Dienstleistern handelt es sich um folgende Geschäfte:

- Labhart-Chronometrie und Schmuckgalerie, Uhren- und Schmuckgeschäft
- Blumeria Marbach, Blumengeschäft
- Lightplay GmbH, Fotografie
- Liluca, Braut- und Festmode
- nisago, Herrenausstatter

Die fünf Dienstleister, allesamt aus dem Grossraum Ostschweiz, schenken Hochzeitspaaren einen Gutschein, sofern diese ihre Dienstleistungen und/oder Produkte bei mindestens zwei dieser Geschäfte beziehen. Die Höhe des gutgeschriebenen Betrages hängt vom Gesamtumsatz aller vorgängigen Einkäufe ab, wobei jeweils das Geschäft, das am Schluss besucht wird, den Gutschein auf den Einkauf gewährt. Der zuletzt getätigte Einkauf wird nicht mehr zu der vorherigen Einkaufssumme addiert.

In unregelmässigen Abständen werden die gewährten Rabatte unter den beteiligten Geschäften anteilmässig zum jeweiligen Ertrag ausgeglichen. Für die Aufteilung der entstandenen Rabattaufwände wird eine umfangreiche Berechnungsformel angewandt, welche verschiedene Bedingungen berücksichtigt. Die genauen Regeln für die Berechnung werden im Kapitel 3 erläutert.

2.2 Problemstellungen

In den nachfolgenden Punkten werden die drei Hauptprobleme beschrieben, mit denen die teilnehmenden Geschäfte momentan konfrontiert sind. Zudem finden sich zusammengefasst im Abschnitt 2.2.4 weitere Probleme, die vergleichsweise wenig ins Gewicht fallen.

2.2.1 Grosser Zeitaufwand

Aktuell wird die ausgleichende Abrechnung jeweils mit grossem manuellem Aufwand ausgewertet, wozu alle eingelösten Gutscheinhefte physisch eingesammelt und an einen zentralen Ort gebracht werden müssen. Die Auswertung benötigt jeweils einige Arbeitsstunden. Die knappen Personalressourcen verhindern eine regelmässige und häufige Abrechnung unter den Geschäften. Hinzu kommt, dass der manuelle Aufwand linear zur Anzahl Gutscheine ansteigt.

2.2.2 Fehlerpotenzial

Die bestehende Lösung birgt Fehlerpotenzial in den drei Bereichen nachfolgenden Bereichen:

Die Einkaufssummen in den Gutscheinheften werden allesamt von Hand eingetragen. Unleserliche Handschriften werden bei der Auswertung folgegleich zur potenziellen Fehlerquelle, da die Werte unter Umständen nicht oder nur schlecht entziffert werden können. Die Übertragung der handschriftlichen Werte in die Berechnungstabelle bringt weiteres Fehlerpotenzial mit sich. Problematisch ist hierbei vor allem, dass allfällige Berechnungsfehler nicht erkannt würden, da sich die Gutscheinwerte in 2'000 Fr.-Schritten ändern.

Die Ad-Hoc-Berechnung des Gutscheinbetrages in denjenigen Geschäften, wo der Gutschein eingelöst wird, birgt zudem die Gefahr von Flüchtigkeitsfehlern in der Berechnung, wobei diese entweder zu Ungunsten der Kundschaft, oder zu Ungunsten der beteiligten Geschäfte ausfällt.

2.2.3 Fehlende Transparenz

Das dritte Hauptproblem besteht in der fehlenden Transparenz zwischen den teilnehmenden Geschäften. Die Geschäfte sind zu keinem Zeitpunkt über aktuelle Zwischenstände

informiert, sondern erfahren erst bei der Endabrechnung aller Gutscheine, welche Zahlungen ihnen zustehen, oder in welcher Höhe sie sich an Gutscheinen beteiligen müssen. Zudem fehlen den Geschäften sämtliche Informationen über die sich im Umlauf befindenden Gutscheinefte.

Fehlende Transparenz führte in der Vergangenheit zu fehlender Motivation bei den Dienstleistern, wodurch das Gutscheinsystem gegenüber der Kundschaft gar nicht erst kommuniziert wurde. Zudem führt die fehlende Transparenz dazu, dass die teilnehmenden Geschäfte keine genauen Prognosen für ihre Finanzabschlüsse abgeben können.

2.2.4 Weitere Probleme

Nebst den drei Hauptproblemen bietet die bestehende Lösung keine effiziente Möglichkeit, Daten zu sammeln und Transaktionen zu analysieren. Dadurch entfällt die Möglichkeit einer gezielten Kontaktaufnahme mit Käuferpaaren, deren Gutschein kurz vor dem Verfall steht und bis dato noch nicht eingelöst wurde.

In den letzten zwei Jahren hat es sich bei den teilnehmenden Geschäften immer um dieselben Geschäfte gehandelt. Die Auszeichnung des Vereins «Wedding Award Switzerland» wird jedes Jahr verliehen, weshalb es denkbar ist, dass einst ein weiterer Dienstleister in das Gutscheinsystem aufgenommen wird. Dazu muss das Gutscheineft neu gedruckt werden und alle sich im Umlauf befindenden Hefte müssen aktualisiert werden.

Sollte ein Gutscheineft verloren gehen, müssen Kunden momentan auf die Kulanz der Geschäfte hoffen und sich ein neues Gutscheineft ausfüllen lassen.

3 Herausforderungen

Folgendes Kapitel umfasst die Herausforderungen, die zur Lösung des Problems gemeistert werden müssen. Dazu werden die Berechnungsregeln, die erwarteten Artefakte und die möglichen Risiken erläutert.

3.1 Artefakte

Die nachfolgend beschriebene Lösungsidee dient als Hypothese auf die Frage, welche Artefakte die in Kapitel 2 beschriebenen Probleme lösen.

Als ganzheitliche Lösung soll eine Software erstellt werden, die zur Erfassung und Verwaltung aller gutscheinrelevanten Vorgänge dient, womit die physischen Gutscheinhefte obsolet werden. Dadurch kann das bestehende Gutscheinheft abgeschafft werden, wodurch die in Kapitel 2.2.4 beschriebenen Probleme gelöst werden können.

Für die Erstellung dieser Software benötigt es die nachfolgend erwähnten Komponenten, die, entsprechend der Spezifikation im Abschnitt 1.2, teilweise zum erwarteten Umfang dieser Bachelorarbeit gehören.

3.1.1 REST-API (Backend)

Die Komponente REST-API verknüpft das visuelle Frontend mit der Datenbank. Über die in Kapitel 6.3.4 festgelegten Aktionen können über die REST-API verschiedene Transaktionen und Abfragen getätigt werden, sofern der authentifizierte Benutzer über die entsprechenden Berechtigungen verfügt.

Die REST-API greift auf eine zentrale Datenbank zu, und erlaubt nur Operationen, die den vorgegebenen Regeln des Gutscheinsystems nicht widersprechen. Der genaue Funktionsumfang der REST-API wird in Kapitel 6.3 beschrieben.

Mit der Bereitstellung und Nutzung der REST-API werden die manuellen Ausgleichsrechnungen zwischen den teilnehmenden Geschäften hinfällig, da diese in beliebigen Abständen automatisiert berechnet werden können. Damit wird das Problem des grossen Zeitaufwandes (Kapitel 2.2.1) gelöst, und die verschiedenen potenziellen Fehlerquellen (Kapitel 2.2.2) werden ebenfalls minimiert. Das Risiko, dass der Einkaufsbetrag im Geschäft falsch erfasst wird, bleibt bestehen. Gemäss Franz & Mattes (1991, S. 27) bleibt

die manuelle Datenerfassung als Fehlerquelle immer bestehen, kann jedoch mittels peripheren und internen Prüftechniken auf ein Minimum verringert werden. Die periphere Prüftechnik kommt bei dem entwickelten Frontend zum Einsatz und validiert die eingegebenen Daten schon vor deren Verarbeitung durch die REST-API (Franz & Mattes, 1991, S. 27). Im Falle einer fehlgeschlagenen Validierung werden die Fehler visuell kenntlich gemacht und das Übermitteln der Daten wird unterbunden (Franz & Mattes, 1991, S. 27). Interne Prüftechniken kommen innerhalb der REST-API zum Einsatz und werden angewendet, sobald die Daten an den Applikationsserver geschickt werden.

Mit der REST-API wird eine Single Source Of Truth geschaffen, die die vorhandenen Transparenzprobleme (Abschnitt 2.2.3) beseitigt, indem über die REST-API jederzeit die aktuellen Zwischenstände über sämtliche Gutscheine abgefragt werden können.

3.1.2 Mockups

Bei diesem Projekt besteht eine wichtige Herausforderung darin, die fehlende Transparenz in der visuellen Oberfläche verständlich darzustellen. Zugehörig dazu ist die optimale Benutzerführung (Usability) für die Anwendung zu finden, um sämtliche Vorgänge so effizient wie möglich abwickeln zu können. Um diese Anforderungen zu erfüllen, werden visuelle Mockups erstellt und mit der Zielgruppe auf deren Praxistauglichkeit validiert. Die validierten Mockups stellen sodann die Grundlage zur Strukturierung und Gestaltung von dem Frontend dar.

3.1.3 Frontend

Mit der Entwicklung und Bereitstellung von dem Frontend wird die Interaktionsschnittstelle zwischen der REST-API und dem Endbenutzer geschaffen. Folglich muss das Frontend zwingend die Möglichkeit bieten, alle gutscheinrelevanten Transaktionen zu tätigen. Die gesamte Businesslogik wird von der REST-API vollzogen, wodurch das Frontend keine Berechnungen durchführen muss. Ungeachtet dessen sollen sämtliche Benutzereingaben so weit als möglich bereits vor dem Datentransfer zur REST-API clientseitig überprüft und im Fehlerfall mit einer entsprechenden Meldung deklariert werden.

3.1.4 Server Deployment

Die REST-API bildet mit dem zusätzlich entwickelten Frontend eine lauffähige Applikation mit einer grafischen Benutzeroberfläche, die über alle modernen Webbrowser aufgerufen werden kann. Damit die Applikation standortunabhängig erreichbar ist, muss sie

auf einen mit dem Internet verbundenen Server deployt werden. Damit dieser Prozess möglichst automatisiert ausgeführt werden kann, bietet sich eine DevOps-Strategie an.

3.2 Berechnungsgrundlage und Regeln

Für die – in Zukunft regelmässige – Abrechnung zwischen den teilnehmenden Geschäften bedarf es eines komplexen Algorithmus, der verschiedene Vorschriften berücksichtigt und die Berechnung wie folgt durchführt:

- An einem eingelösten Gutschein beteiligen sich nur jene Unternehmen, bei denen das jeweilige Hochzeitspaar eingekauft hat.
- Welches Unternehmen sich wie stark am ausgestellten Gutschein beteiligt, hängt proportional von jenem Betrag ab, welcher das Hochzeitspaar im betreffenden Geschäft ausgegeben hat. Zudem darf der Anteil nicht höher sein als 10 Prozent des Betrags, welcher das Hochzeitspaar im jeweiligen Geschäft ausgegeben hat.
- Jenes Geschäft, welches das Hochzeitspaar vom Gutscheinsystem überzeugt hat, soll belohnt werden. Zum einen soll es sich nur am Betrag des Gutscheins beteiligen müssen, wenn die proportionalen Beiträge der restlichen Geschäfte unter Einhaltung der 10%-Regel dafür nicht ausreichen. Zum anderen soll es – wiederum unter Einhaltung der 10%-Regel – 50 Franken gutgeschrieben bekommen.
- Die 10%-Regel muss zwingend eingehalten werden. Dies kann bei bestimmten Konstellationen dazu führen, dass sich das erste Geschäft dennoch an den Kosten (bis zu 10 Prozent von dessen Umsatz) beteiligen muss.
- Die Gutscheinbeträge sind wie folgt definiert:

Ab Einkaufstotal von	Geschenkgutschein in Höhe von
2'000 Fr.	100 Fr.
4'000 Fr.	250 Fr.
6'000 Fr.	350 Fr.
8'000 Fr.	500 Fr.
10'000 Fr.	650 Fr.
12'000 Fr.	800 Fr.
14'000 Fr.	1'000 Fr.

Die REST-API muss die nachfolgenden Regeln befolgen und darf empfangene Daten nur dann verarbeiten und speichern, wenn alle Regeln eingehalten werden.

- Wenn ein gutscheinrelevanter Betrag verbucht wird, kann der Gesamtgutschein anschliessend nicht im selben Geschäft eingelöst werden. Das heisst, dass der Betrag, den die Kunden in jenem Geschäft, wo sie den Gutschein einlösen, ausgeben, nicht mehr an den gutscheinrelevanten Gesamtbetrag angerechnet werden kann.
- Die Hochzeitspaare können ihre Beträge nicht aufteilen, sondern erhalten gemeinsam nur einen Gutschein.
- Der Gutschein muss vor dem Hochzeitsdatum eingelöst werden.

3.3 Risiken

Die oben genannten Herausforderungen bergen verschiedene Risiken, die es bei der Planung und Umsetzung zu berücksichtigen gilt. Um welche Risiken es sich dabei handelt und mit welchen Massnahmen die Gefahren verringert werden, wird in den nachfolgenden Absätzen erläutert.

3.3.1 Humanressourcen

Die festgelegten Ziele für diese Arbeit (Abschnitt 1.2) sind für den vorgegebenen Zeitraum von knapp 14 Wochen sehr umfangreich. Dennoch wird die Erfüllung aller Hauptziele als realistisch eingestuft, da der Autor dies vorgängig mit dem Betreuer abgesprochen hat.

3.3.2 Performance

Mit der Auslieferung der Softwarelösung muss sichergestellt sein, dass die Operationen performant durchgeführt werden und für den Benutzer keine Wartezeiten entstehen. Wenn die bereitgestellte Software langsam reagiert, schafft dies Unzufriedenheit bei den Anwendern und es entsteht die Gefahr, dass die Nutzer von der Benutzung der Software mittel- bis langfristig absehen werden.

Um dieser Gefahr entgegenwirken zu können, werden während des gesamten Entwicklungsprozesses laufend Performancetests durchgeführt und wo notwendig entsprechende Optimierungen vorgenommen.

3.3.3 Algorithmus

Bei der Einführung der neuen Software besteht das Risiko, dass die implementierten Berechnungsalgorithmen entweder in jedem Fall fehlerhaft sind, oder dass nur in bestimmten Konstellationen falsche Berechnungen durchgeführt werden. Um diesen Gefahren entgegenzuwirken, werden verschiedene Arten von Tests, sowohl während der Entwicklungszeit, als auch nach Fertigstellung der Applikation, durchgeführt.

Zur Validierung und Sicherstellung eines fehlerfreien Algorithmus wird die gesamte Code-Basis mit Unit-Tests abgedeckt. Details zu den Unit-Tests sind dem Kapitel 7.2.2 zu entnehmen.

Nebst den Unit-Tests werden beim Entwickeln der Software laufend Smoke-Tests durchgeführt, womit die grundsätzliche Lauffähigkeit der Software sichergestellt wird. Nach Fertigstellung der Software werden mit jenem Geschäft, welches das Gutscheinsystem ins Leben gerufen hat, Akzeptanztests durchgeführt.

Mit der Kombination aus Unit-Tests, Smoke-Tests und Akzeptanztests wird sichergestellt, dass der entwickelte Algorithmus korrekt funktioniert.

3.3.4 Komplexität

Die bisherigen Prozesse des Gutscheinsystems sind den verantwortlichen Mitarbeitern bekannt und werden als leicht verständlich betrachtet. Mit der Einführung der neuen Software besteht die Gefahr, dass die daraus resultierenden Prozesse komplexer sind, oder dass das User Interface bei unregelmässiger Nutzung nicht verstanden wird. Um dieses Risiko so klein wie möglich zu halten, werden die zukünftigen Benutzer bereits in den Planungsprozess involviert. Die Umsetzung kann erst starten, nachdem sämtliche Abläufe und User Interfaces von der Zielgruppe als nützlich und benutzerfreundlich befunden wurden.

4 Nutzen

In diesem Kapitel wird der Nutzen beschrieben, der aus der geplanten Softwarelösung resultiert. Mit der Entwicklung und Inbetriebnahme der neuen Applikation zur Gutscheinverwaltung wird der Aufwand an Mitarbeiterressourcen minimiert, das Fehlerpotenzial verringert, sowie ein Maximum an Transparenz unter den teilnehmenden Geschäften geschaffen. Der neu geschaffene Produktnutzen wird nachfolgend in Kundennutzen und Geschäftsnutzen (teilnehmende Dienstleister) aufgeteilt.

4.1 Kundennutzen

In diesem Abschnitt werden die neu entstehenden Vorteile für die Hochzeitspaare diskutiert. Dazu wird zuerst der Status Quo aus Kundensicht erläutert und anschliessend werden die wichtigsten Neuerungen aufgeführt.

Die bisherige Lösung sieht vor, dass die Hochzeitspaare ihr Gutscheinheft bei jedem Einkauf in einem der teilnehmenden Geschäfte vorweisen. Dadurch entfällt einerseits die Möglichkeit, dass die Partner zeitgleich in unterschiedlichen Geschäften Einkäufe tätigen und diese jeweils anrechnen lassen können. Ebenso besteht das Risiko, dass die Paare ihr Gutscheinheft bei einem Spontaneinkauf nicht bei sich haben und dadurch nicht profitieren können.

Die neu geschaffene Softwarelösung erlaubt es den Hochzeitspaaren, dass sie ihr Gutscheinheft nicht mehr physisch mit sich tragen müssen, sondern sich mit Name und Hochzeitsdatum im Geschäft «ausweisen» können, um den jeweiligen Einkaufsbetrag auf ihr Gutscheinkonto gutschreiben zu lassen. Folglich erlaubt das neue System auch, dass die Partner zur selben Zeit in verschiedenen Geschäften einkaufen und dabei ihre Einkäufe anrechnen lassen können. Mit der Einführung der neuen Software entfällt die Pflicht, ein physisches Gutscheinheft bei Einkäufen mit sich zu führen, womit das Risiko, das Heft zu verlieren, wegfällt.

Das neue System verringert ferner das Risiko, dass Kunden unbeabsichtigt mehrere Gutscheinhefte eröffnen, da in jedem Geschäft in Erfahrung gebracht werden kann, ob bereits Einkäufe für die Kundschaft erfasst sind.

Mit dem bereitgestellten iFrame können erfasste Hochzeitspaare ihre bisherigen Einkäufe und den aktuellen Gutscheinwert abfragen. Das manuelle Berechnen entfällt dadurch.

4.2 Geschäftsnutzen

In diesem Abschnitt werden die neu entstehenden Vorteile für die teilnehmenden Hochzeitsdienstleister diskutiert. Dazu wird zuerst der Status Quo aus Firmensicht erläutert und anschliessend werden die wichtigsten Neuerungen aufgeführt.

Bei der aktuellen Lösung wird ein physisches Gutscheinheft benötigt, das in allen teilnehmenden Geschäften vorrätig vorhanden ist. Beim ersten Einkauf eines Hochzeitspaars wird das Gutscheinheft abgegeben und der Einkaufsbetrag wird darin vermerkt. Sobald ein Gutschein eingelöst wird, wird das zugehörige Gutscheinheft im einlösenden Geschäft zurückgegeben. Dort wird es für die spätere Verrechnung, unter den am Gutschein beteiligten Geschäften, aufbewahrt. Bis zum Zeitpunkt der Verrechnung wissen die einzelnen Geschäfte nicht, welche offenen Debitorpositionen ihnen zustehen oder welche Beträge sie zu begleichen haben.

Die Einführung der neuen Softwarelösung ermöglicht es den teilnehmenden Geschäften, zu jedem Zeitpunkt über den aktuellen Zwischenstand informiert zu sein. Die neu gewonnene Transparenz reduziert Unsicherheiten und Falscheinschätzungen. Zudem werden die physischen Gutscheinhefte in Zukunft dank der digitalen Lösung nicht mehr gebraucht, was zu Kosteneinsparungen führt.

Zusammenfassend werden mit der digitalen Gutscheinverwaltung die gesamten Abrechnungsprozesse unter den teilnehmenden Geschäften weitestgehend automatisiert, was einerseits zu Fehlervermeidung, und andererseits vor allem zu einer Effizienzsteigerung hinsichtlich der Accounting-Prozesse führen wird. Durch die einfachere Handhabung werden zudem die Hemmungen verringert, dem Kunden das bisher als umständlich betrachtete Gutscheinheft abzugeben.

5 Vorgehensweise

Die vorliegende Arbeit ist in verschiedene Arbeitspakete gegliedert, die in diesem Kapitel erläutert werden. Sodann ist im Abschnitt 5.8 der Projektplan aufgeführt.

Die nachfolgende Aufteilung in die verschiedenen Arbeitspakete gleicht im Vorgehen stark dem Wasserfallmodell. Dieses sieht vor, die Anforderungen in verschiedene Phasen zu gliedern und diese sequentiell abzuarbeiten (Arndt, Hermanns, Kuchen & Poldner, 2009, S. 7). Die Autoren Arndt et. al. bemängeln am Wasserfallmodell, dass sämtliche Test- und Integrationsaktivitäten erst nach Abschluss der Umsetzungsarbeiten durchgeführt werden (2009, S. 7). Dieser Schwachstelle kommt die angepasste Arbeitsweise dieser Arbeit entgegen, indem in allen Arbeitspaketen eine sinnvolle Validation vorgesehen ist.

5.1 Arbeitspaket 1: Frontend Design

Im ersten Schritt werden die Mockups für die Benutzeroberfläche der Gutscheilverwaltung erstellt. Diese sollen die Bedürfnisse der Anwender, in diesem Falle die der Hochzeitsdienstleister, mit bestmöglicher User Experience befriedigen. Als Grundlage dienen bilaterale Gespräche mit den beteiligten Unternehmen sowie die Spezifikation der Anforderungen. Um sicherzustellen, dass die Ergebnisse alle Bedürfnisse abdecken, werden im Arbeitspaket 2 sämtliche Mockups mit allen betroffenen Firmen validiert. Dies wird in Absatz 5.2 erläutert.

Die Resultate dieses Arbeitspakets sind im Anhang unter Punkt 11.2 zu finden.

5.2 Arbeitspaket 2: Validierung Frontend Design

In diesem Arbeitspaket werden die erstellten Mockups mit den realen Endbenutzern auf deren Bedürfnisse validiert. Entscheidend ist dabei, ob alle relevanten Informationen und Interaktionsmöglichkeiten vorhanden sind. Aufgrund des Feedbacks werden die Mockups iterativ überarbeitet und erneut validiert, bis sie vollumfänglich den Anforderungen der Benutzer entsprechen. Zur Kontrolle der Änderungen sind die Mockups mit dem Online-Tool InVision (www.invisionapp.com) online zur Verfügung gestellt und die beanstandeten Änderungen können über das Kanban-Board in Gitlab verwaltet werden.

Dadurch kann sichergestellt werden, dass alle Änderungswünsche vollumfänglich umgesetzt werden. Zur bestmöglichen Validierung durch die verantwortlichen Hochzeitsdienstleister sind die Mockups auf InVision so erweitert worden, dass es sich um klickbare Mockups handelt.

Die Resultate dieses Arbeitspakets sind im Abschnitt 7.1 zu finden.

5.3 Arbeitspaket 3: Backend Design

Mit der Fertigstellung der Mockups und der Freigabe aller Beteiligten startet die Umsetzung des dritten Arbeitspakets. Dabei werden die für die REST-API benötigten Endpunkte mit der OpenAPI-Spezifikation definiert. Dazu wird der Online-Editor Swagger verwendet, der von jener Firma stammt, die als ursprüngliche Initiatorin der aktuell existierenden OpenAPI-Spezifikation gilt (OpenAPI Initiative, o. J.).

Die spezifizierten Endpunkte werden laufend mit den Mockups verglichen um sicherzustellen, dass alle Informationen ressourcenschonend abgefragt werden können und die Endpunkte die notwendigen Aktionsmöglichkeiten bereitstellen.

Die Resultate dieses Arbeitspakets sind unter Punkt 6.3.4 zu finden.

5.4 Arbeitspaket 4: Backend Umsetzung

Die im Arbeitspaket 3 erstellte API-Spezifikation dient als Basis zur Umsetzung dieses Arbeitspakets, der Umsetzung der REST-API. Die REST-API wird in der Programmiersprache Python geschrieben mit den dazugehörigen Webframeworks Django und Django REST. Die REST-API muss die verschiedenen Berechnungen und Abrechnungen nach den in Abschnitt 3.2 beschriebenen Regeln befolgen. Damit die Richtigkeit dieser Berechnungen überprüft werden kann, werden sämtliche Code-Fragmente im Arbeitspaket 5 mit Unit-Tests validiert.

Die Resultate dieses Arbeitspakets sind im mitgelieferten Repository zu finden.

5.5 Arbeitspaket 5: Validierung Backend

In diesem Arbeitspaket werden alle Endpoints der API und insbesondere die dahinterliegende Logik auf ihre korrekte Funktionalität überprüft. Damit wird ein Richtigkeitserweis erbracht, indem die eingegebenen Testdaten mit ihren erwarteten Resultaten verglichen

werden. Um die Richtigkeit auch bei zukünftigen Weiterentwicklungen sicherzustellen, wird der gesamte Python-Code mit Unit-Tests versehen. Mit diesen Tests wird sichergestellt, dass die erstellten Komponenten so funktionieren wie erwartet. Weiter kann mit der erreichten Codeabdeckung von 100 Prozent sichergestellt werden, dass alle möglichen Codeausprägungen getestet wurden.

Die Resultate dieses Arbeitspakets sind im Anhang unter Punkt 11.4 zu finden.

5.6 Optionales Arbeitspaket 6: Frontend Umsetzung

(Nicht im erwarteten Rahmen dieser Arbeit)

Mit der Erfüllung dieses Arbeitspakets werden die im Arbeitspaket 1 erstellten Mockups als interaktives Frontend umgesetzt und mit der REST-API ans Backend angebunden. Damit steht eine funktionsfähige und einsatzbereite Webapplikation zur Verfügung.

Die Resultate dieses Arbeitspakets sind unter Punkt 6.4 sowie im Anhang unter Punkt 11.5 zu finden.

5.7 Optionales Arbeitspaket 7: DevOps

(Nicht im erwarteten Rahmen dieser Arbeit)

In diesem Arbeitspaket werden verschiedene Vorkehrungen zur Erreichung einer DevOps-Strategie getroffen. Mit der Implementierung von DevOps wird eine fehlerfreie und schnelle Auslieferung von neuen Komponenten ermöglicht, wodurch die Deployment- und Testingprozesse verbessert werden können. Nebst der Prozessverbesserung wird mit DevOps eine Effizienzsteigerung als auch eine Qualitätsverbesserung erzielt (Gartner, o. J.).

Die Resultate dieses Arbeitspakets sind im Abschnitt 6.5 aufgeführt.

5.8 Projektplan

Die vorgängig beschriebenen Arbeitspakete widerspiegeln die zu planenden Aufgaben dieser Bachelorarbeit. Zur visuellen Darstellung, Einteilung sowie Zeitplanung der Arbeiten, wird ein Gantt-Projektplan verwendet. Ein wichtiger Vorteil von Gantt-Projektplänen stellt deren einfache und übersichtliche Abbildungsform dar, wie die Autoren Biethahn, Mucksch und Ruf hervorheben (2004, S. 398). Biethan et. al. kritisieren jedoch,

dass eine logische Verbindung zwischen einzelnen Vorgängen im Gantt-Projektplan nicht deutlich zu erkennen sei (2004, S. 398). Dieser Kritik wird in dieser Arbeit entgegengehalten, indem in der eingesetzten Software Merlin Project die einzelnen Vorgänge miteinander verknüpft werden. Im detaillierten Projektplan sind die logischen Verbindungen gut ersichtlich. Die nachfolgende Abbildung 1 stellt den auf die Arbeitspakete reduzierten Projektplan dar. Im Anhang ist der detaillierte Plan in Abbildung 16 ersichtlich, worin die logischen Verknüpfungen gut zu erkennen sind.

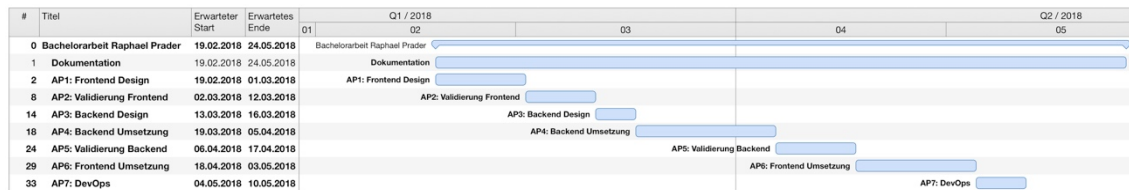


Abbildung 1: Gantt-Projektplan

6 Lösung

Das vorliegende Kapitel befasst sich mit der Lösungsentwicklung der gesamten Arbeit. In diesem Zusammenhang wird das zentrale Lösungsartefakt, die Software, vorgestellt, sowie sämtliche Artefakte die zur Lösungsentwicklung beigetragen haben. Abschliessend werden alternative Lösungsmöglichkeiten untersucht.

6.1 Frontend Design

In diesem Abschnitt werden zuerst die Grundlagen zur Erstellung der Mockups erläutert. Anschliessend werden die erstellten Mockups gezeigt und dazu jeweils die Informationen und Interaktionsmöglichkeiten erklärt.

6.1.1 User Stories

Als Grundlage zur Erstellung der Mockups dient zum einen die vorliegende Problemstellung sowie die in Tabelle 1 aufgeführten User Stories. Mit den User Stories werden die Bedürfnisse der Zielgruppen beschrieben, wobei eine User Story jeweils der Struktur «WER? WAS? WARUM?» folgt.

Die User Stories werden sowohl zur initialen Erstellung der Mockups verwendet, als auch zur anschliessenden Validierung durch den Verfasser dieser Arbeit (Abschnitt 7.1.1). Die Benutzer der Applikation sind in drei Hauptanwendergruppen unterteilt: transaktionale Mitarbeiter, Buchhalter und Geschäftsführer.

ID	Hauptanwendergruppe	User Story
US-1	Transaktionale Mitarbeiter	Als Mitarbeiter will ich einen gutscheinrelevanten Einkauf für einen Kunden verbuchen, sodass ihm der Einkaufsbetrag an den Gesamtgutschein angerechnet wird.
US-2	Transaktionale Mitarbeiter	Als Mitarbeiter will ich den Gutschein eines Kunden einlösen, sodass ihm beim letzten Einkauf der Gutscheinbetrag abgezogen wird.
US-3	Transaktionale Mitarbeiter	Als Mitarbeiter will ich einen neuen Kunden erfassen können, sodass dieser vom Gutscheinsystem profitieren kann.
US-4	Transaktionale Mitarbeiter	Als Mitarbeiter will ich einen Kunden über seinen aktuellen Einkaufszwischenstand informieren können, sodass er stets den Überblick behält.

US-5	Transaktionale Mitarbeiter	Als Mitarbeiter will ich einem Kunden jederzeit eine Quittung mit seinem Einkaufs-Zwischenstand ausstellen können, sodass er stets den Überblick behält.
US-6	Buchhalter	Als Verantwortlicher der Buchhaltung will ich jederzeit die ausstehenden Kreditoren/Debitoren sehen, sodass ich die Finanzen planen kann.
US-7	Buchhalter	Als Verantwortlicher der Buchhaltung will ich vergangene Abrechnungen aufrufen können, sodass ich allfällige Unklarheiten überprüfen kann.
US-8	Geschäftsführer	Als Geschäftsführer will ich aktive Kunden mit nicht-eingelösten Gutscheinen erkennen können, sodass ich potenzielle Käufer abschätzen kann.
US-9	Geschäftsführer	Als Geschäftsführer will ich eine Übersicht über die aktuelle Abrechnungsperiode sehen, sodass ich weiss, welche Umsätze mein Unternehmen mit dem Gutscheinsystem erzielt hat.

Tabelle 1: User Stories

6.1.2 Mockups

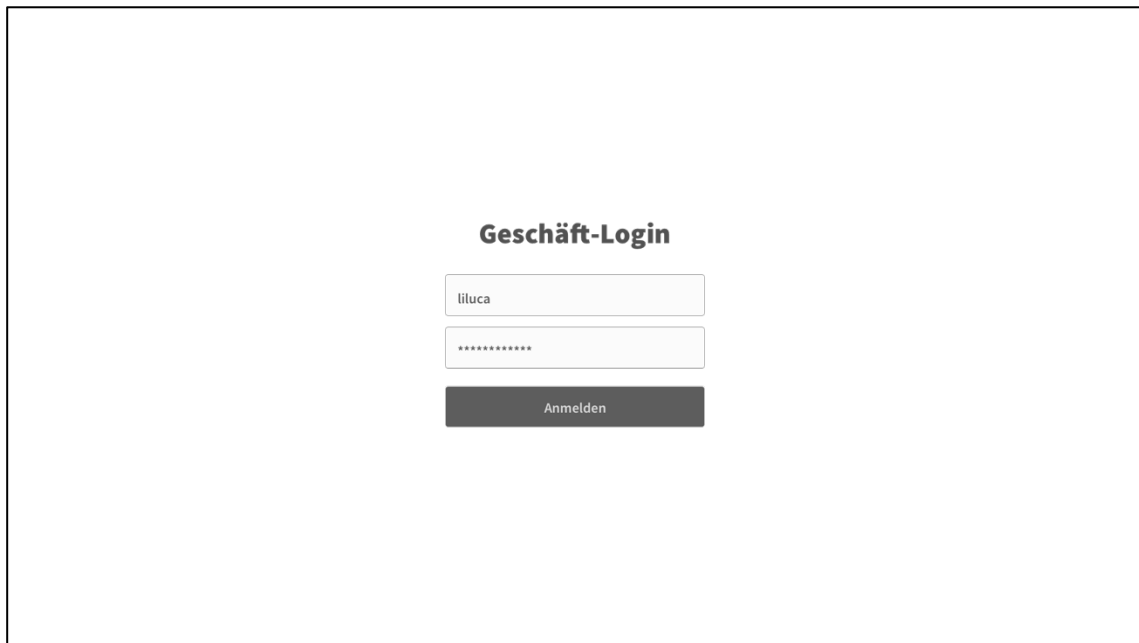
In diesem Kapitel werden die Mockups dargestellt, die den geplanten Aufbau der Applikation widerspiegeln. Mockups sind bestens dazu geeignet, die Anforderungen mit der Zielgruppe zu validieren, da Mockups die Software nicht nur textlich sondern auch visuell beschreiben (Berenbrink, Purucker & Bahlinger, 2013, S. 30). Mit der sorgfältigen Konzeption, Entwicklung und der dazugehörigen Validierung von Mockups werden Änderungen im Verlauf der Programmierphase weitestgehend vermieden (Berenbrink et al., 2013, S. 27).

Da gewisse Mockups nur geringfügige Abweichungen oder selbsterklärende Elemente aufweisen, werden der Übersichtlichkeit halber nicht alle Mockups in diesem Abschnitt dargestellt. Folglich werden der Vollständigkeit halber sämtliche Mockups im Anhang im Abschnitt 11.2 aufgeführt.

Zur Erstellung der Mockups wird das digitale Design-Tool Sketch verwendet.

Die Verwaltungsapplikation richtet sich an die drei identifizierten Zielgruppen transaktionale Mitarbeiter, Buchhalter und Geschäftsführer. Allesamt sind sie Mitarbeiter der am Gutscheinsystem beteiligten Firmen, womit ihnen das grundsätzliche Prinzip der Gutscheine vertraut ist.

6.1.2.1 Login



The image shows a mockup of a login screen titled "Geschäft-Login". It features a central form with three elements: a text input field containing the username "lituca", a password input field with masked characters "*****", and a dark grey button labeled "Anmelden".

Abbildung 2: Mockup von dem Screen «Login»

Beim erstmaligen Aufruf der Applikation im Webbrowser wird dem Benutzer ein Formular mit der Eingabeaufforderung von Benutzername und Passwort angezeigt. In Abbildung 2 ist das Mockup ersichtlich, welches diesen Zustand darstellt. Es werden bewusst keine Informationen zur Anwendung aufgeführt, da die Login-Seite öffentlich einsehbar ist.

6.1.2.2 Einkauf verbuchen

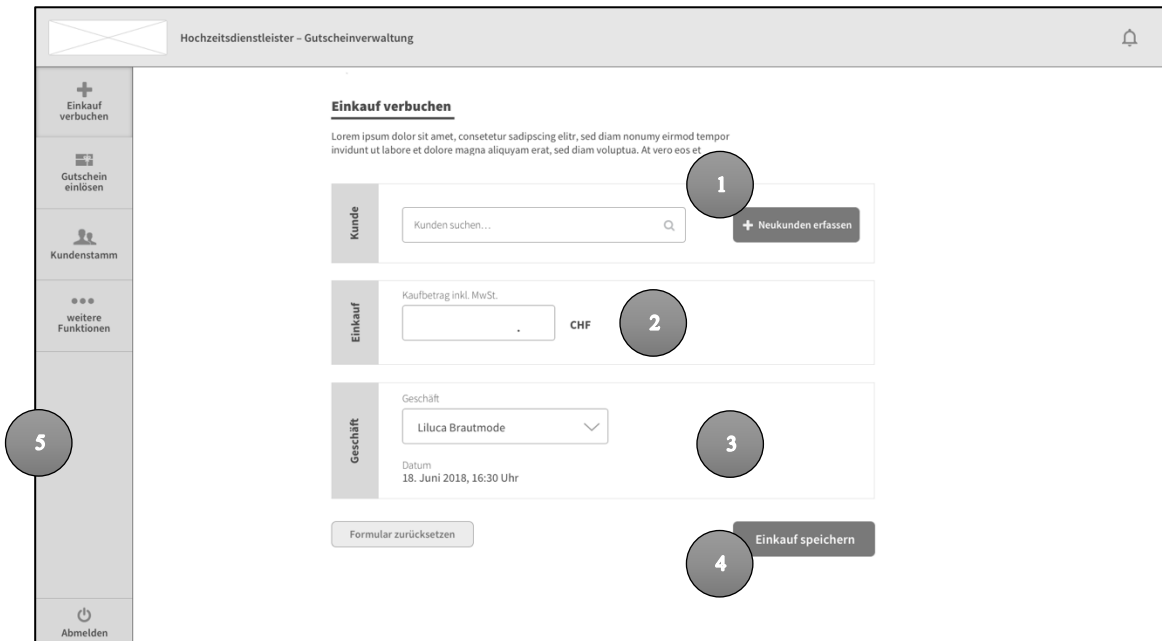


Abbildung 3: Mockup von dem Screen «Einkauf verbuchen»

Das in Abbildung 3 gezeigte Mockup fungiert als Einstiegsseite nach erfolgreichem Login. In der Annahme, dass das Verbuchen von Einkäufen der häufigste Use Case ist, wird auf eine einleitende Einstiegsseite verzichtet. Die wichtigsten Interaktionsmöglichkeiten von diesem Screen sind in Tabelle 2 aufgeführt.

Nummer	Beschreibung
1	Das Eingabefeld dient zur Suche eines bereits erfassten Kundenpaares. Sobald der Benutzer mit der Eingabe beginnt, werden passende Suchvorschläge in einer Auswahlliste dargestellt. Diese Darstellung ist in Abbildung 19 ersichtlich. Alternativ kann der Button «Neukunden erfassen» benutzt werden, der das Formular zur Erstellung eines neuen Hochzeitspaares öffnet. Dieses Formular ist in Abbildung 20 ersichtlich.
2	Über dieses Eingabefeld wird der Gesamtbetrag des getätigten Einkaufs erfasst.
3	Im Falle eines angemeldeten Benutzers mit mehreren Firmen wird hier ein Auswahlfeld dargestellt, bei dem das tangierte Geschäft ausgewählt werden muss. Wenn der angemeldete Benutzer nur einer Firma zugeordnet ist, erscheint anstelle des Eingabefeldes der Geschäftsname.
4	Dieser Button dient zum Abschicken der eingetragenen Formulare Daten und somit zur Speicherung des Einkaufs. Der Button wird erst aktiv, sobald alle Daten gültig eingetragen sind.

5	In der linken Bildschirmhälfte ist die Navigation der Applikation dargestellt. Benutzer können darüber andere Funktionen als die aktuell ausgewählte Funktion «Einkauf verbuchen» aufrufen.
---	---

Tabelle 2: Interaktionsmöglichkeiten auf dem Screen «Einkauf verbuchen»

Abbildung 4 zeigt die Darstellung, nachdem ein Einkauf erfolgreich gespeichert und somit verbucht wurde. In dieser Darstellung werden zum einen die Details vorheriger Einkäufe des Kunden dargestellt, sowie zum anderen der entsprechende Zwischenstand angezeigt. Zwei Buttons am unteren Bildschirmrand ermöglichen das automatische Versenden einer Bestätigungsmail an den Kunden, sowie das Drucken einer Bestätigung in Papierform.

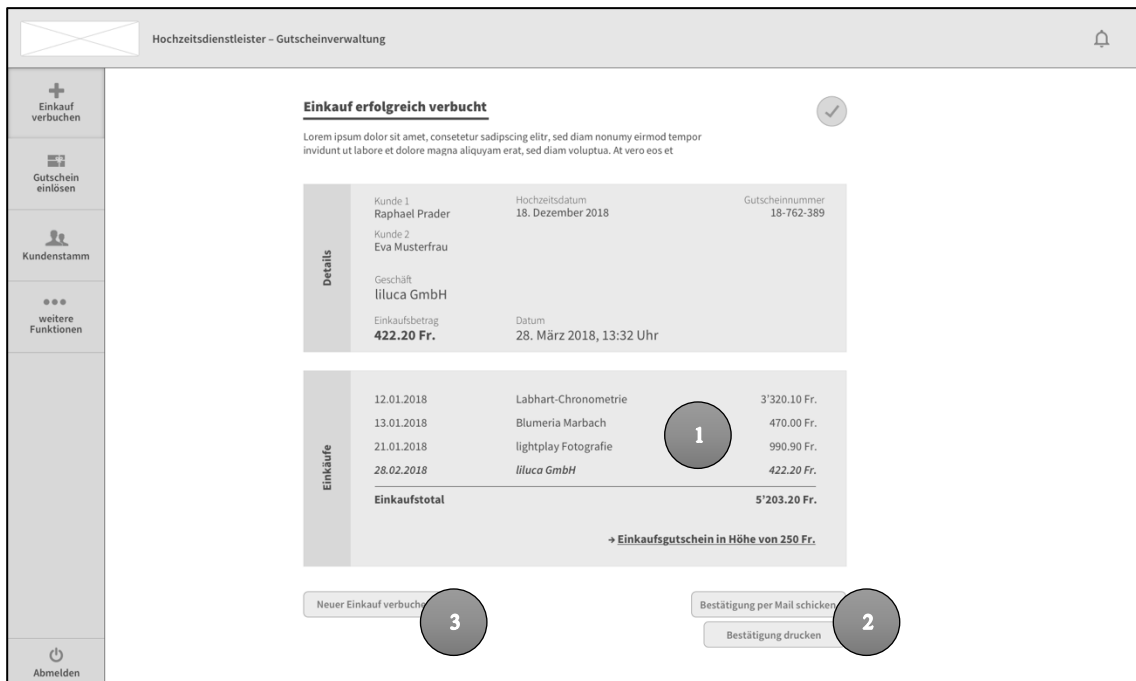


Abbildung 4: Mockup von dem Screen «Einkauf verbucht»

Die wichtigsten Interaktionsmöglichkeiten von diesem Screen sind in Tabelle 3 aufgeführt.

Nummer	Beschreibung
1	Diese Tabelle zeigt die bisherigen Einkäufe des eben bedienten Kunden, sodass diesem sein aktueller Zwischenstand mündlich mitgeteilt werden kann.
2	Diese beiden Buttons erlauben einerseits den Versand einer Bestätigungsmail an den Kunden sowie andererseits ein direkter Ausdruck dieser Bestätigung.
3	Mit diesem Button gelangt der Benutzer zurück zum Formular, womit er Einkäufe verbuchen kann.

Tabelle 3: Interaktionsmöglichkeiten auf dem Screen «Einkauf verbucht»

6.1.2.3 Gutschein einlösen

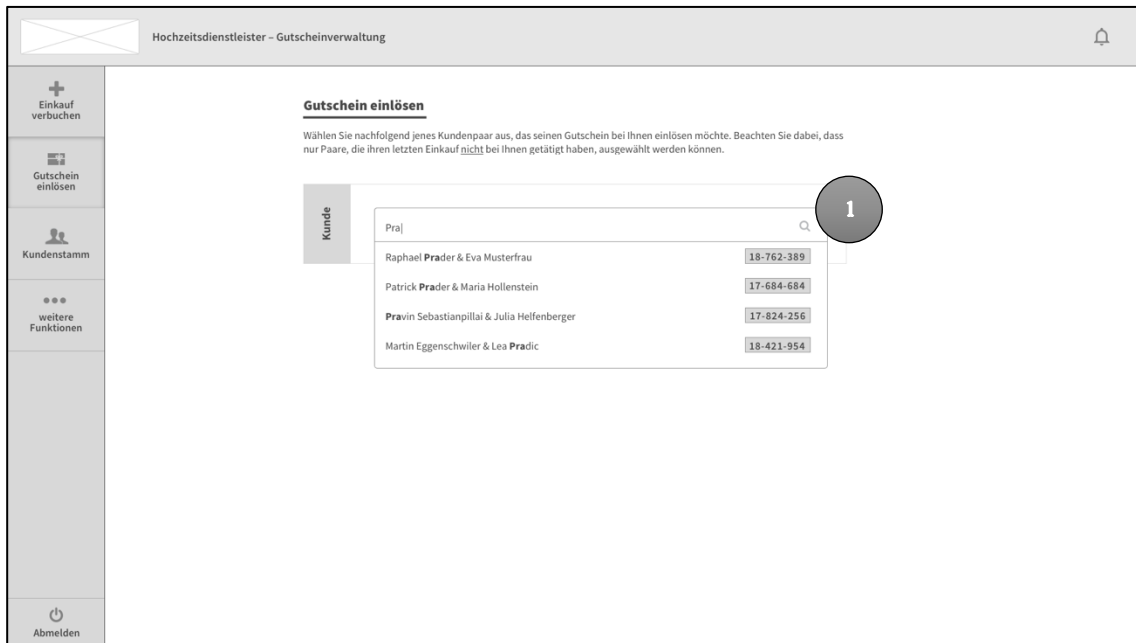


Abbildung 5: Mockup von dem Screen «Gutschein einlösen»

Mit Klick auf den Navigationspunkt «Gutschein einlösen» öffnet sich die in Abbildung 5 dargestellte Eingabemaske. Einkäufe von Kunden, die im Geschäft ihren letzten Hochzeitseinkauf tätigen, können so abgerechnet werden. Nachdem die Auswahl des entsprechenden Hochzeitspaares erfolgt ist, öffnet sich entweder die Darstellung von Abbildung 6, oder die Darstellung von Abbildung 25. Die Darstellung von Abbildung 25 öffnet sich, wenn bereits ein Einkauf im aktuellen Geschäft vorliegt und der letzte Einkauf entsprechend nicht mehr erfasst werden muss. Im Falle eines noch nicht vorliegenden letzten Einkaufs präsentiert sich Abbildung 6.

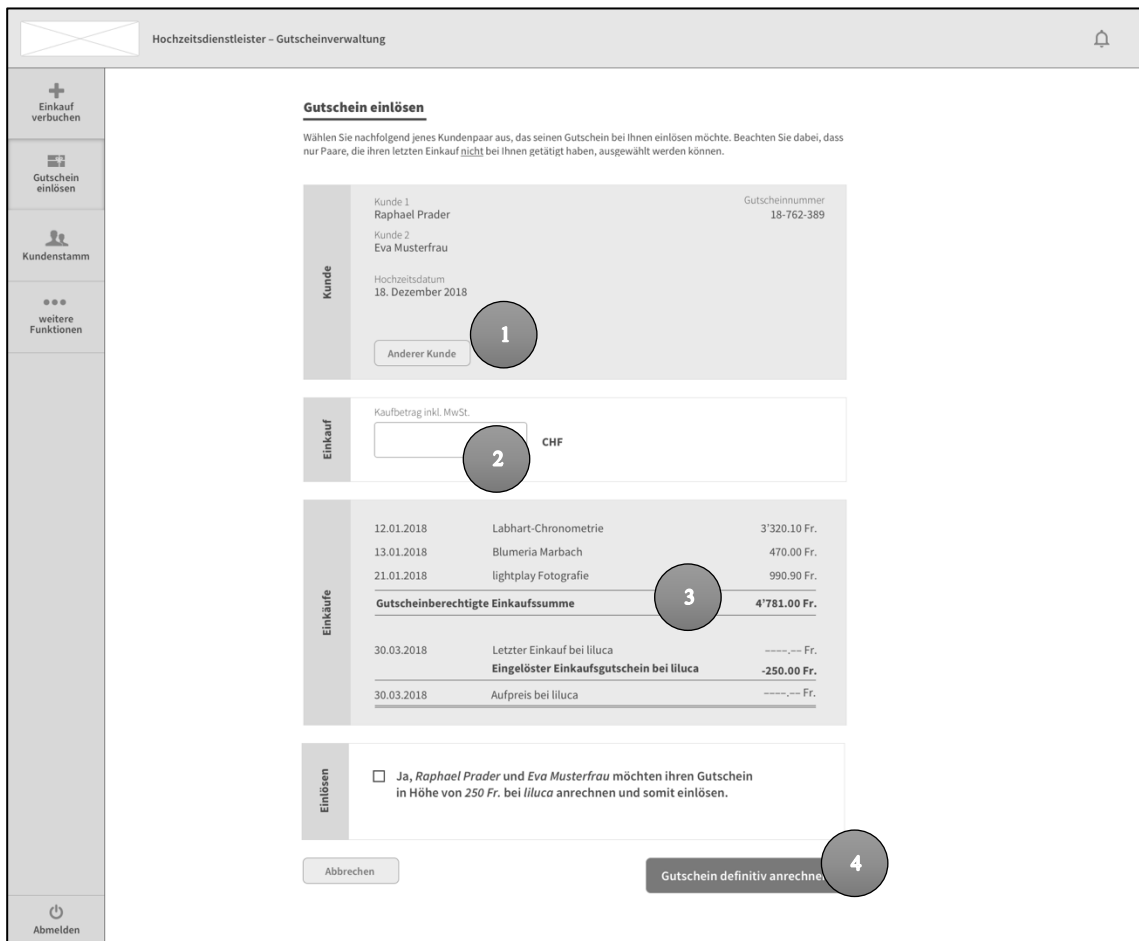


Abbildung 6: Mockup von dem Screen «Abschliessender Einkauf erfassen»

Die wichtigsten Interaktionsmöglichkeiten von diesem Screen sind in Tabelle 4 aufgeführt.

Nummer	Beschreibung
1	Das ausgewählte Hochzeitspaar wird im Bereich ganz oben dargestellt. Im Falle einer Fehlauswahl kann mit Klick auf diesen Button ein anderes Hochzeitspaar ausgewählt werden.
2	In dieses Feld wird der Betrag des gutscheineinlösenden Einkaufs eingetragen. Anhand der Eingabe berechnet sich direkt darunter die Schlussabrechnung.
3	Die Schlussabrechnung zeigt die vorgängigen Einkäufe, die daraus resultierende Summe und den Aufpreis im letzten Geschäft.
4	Sobald alle notwendigen Informationen eingetragen sind, kann der Gutschein mit diesem Button definitiv abgerechnet werden.

Tabelle 4: Interaktionsmöglichkeiten auf dem Screen «Abschliessender Einkauf erfassen»

6.1.2.4 Kundenstamm

Mit Klick auf den Navigationspunkt «Kundenstamm» öffnet sich der in Abbildung 27 dargestellte Kundenstamm. Es wird eine Liste aller aktiven Kunden dargestellt, die mit dem darüber liegenden Eingabefeld durchsucht werden kann. Aktive Kunden bedeutet in diesem Fall, dass es sich um Kunden handelt, die ihren letzten Einkauf noch nicht getätigt haben.

Weitere Details zu den entsprechenden Einträgen können mit Klick auf die Kundennummer erlangt werden. Folglich präsentieren sich die Informationen und Interaktionsmöglichkeiten aus der Abbildung 7.

Hochzeitsdienstleister - Gutscheinverwaltung

Raphael Prader + Eva Musterfrau, #18-762-389

← Zurück zur Übersicht

Einkäufe	
12.01.2018	Labhart-Chronometrie 3'320.10 Fr.
13.01.2018	Blumeria Marbach 470.00 Fr.
21.01.2018	lightplay Fotografie 990.90 Fr.
28.02.2018	liluca GmbH 422.20 Fr.
Einkaufstotal 5'203.20 Fr.	
→ Einkaufsgutschein in Höhe von 250 Fr.	

Kontaktangaben

Herr Frau
Herr Frau
Gemeinsam
Fr. Vogel Hr. Schmidt
Gemeinsam
Fr. Vogel Hr. Schmidt

Vorname Nachname
Katrin Vogel
Jonas Schmidt
Gemeinsamer Nachname
Schmidt-Vogel

Adresse
Pflanzschulstrasse 37
PLZ 8400 Ort Winterthur
+ Weitere Adresse hinzufügen

E-Mail-Adresse
hoj...
Handy
+41...
+ Weitere Kontaktdaten hinzufügen

Wahlrechtstag
18.12.2018

Änderungen speichern

← Zurück zur Übersicht Bestätigung drucken

Abmelden

Abbildung 7: Mockup von dem Screen «Kunden bearbeiten»

Die wichtigsten Interaktionsmöglichkeiten von diesem Screen sind in Tabelle 5 aufgeführt.

Nummer	Beschreibung
1	Im obersten Abschnitt werden die bisherigen Einkäufe der ausgewählten Kundschaft dargestellt. Sofern der Gutschein noch nicht eingelöst wurde, können die Einkäufe, die im angemeldeten Geschäft getätigt wurden, angepasst werden.
2	Die Detailansicht zeigt die Kontaktinformationen der ausgewählten Kundschaft und bietet die Möglichkeit, diese zu bearbeiten. Wenn die ausgewählte Kundschaft bisher noch keine Einkäufe im angemeldeten Geschäft getätigt hat, sind keine detaillierten Kontaktinformationen zu sehen.
3	Eine Zwischenbestätigung der ausgewählten Kundschaft kann mit diesem Button ausgedruckt werden.

Tabelle 5: Interaktionsmöglichkeiten auf dem Screen «Kunden bearbeiten»

6.1.2.5 Weitere Funktionen

In Abbildung 29 wird die Übersicht der weiteren Funktionen dargestellt. Es lassen sich die laufende Abrechnungsperiode, die vergangenen Abrechnungen, häufig gestellte Fragen sowie offene Zahlungen anzeigen.

6.1.2.6 Laufende Abrechnungsperiode

#	Kunden	Einlösedatum	Eigener Umsatz	Gutscheinbeteiligung
18-723-685	Raphael Prader + Eva Musterfrau	18.11.2017	3'320.10 Fr.	-111.69 Fr.
18-385-664	Claudio Beeli + Yvonne Gschwend	12.12.2017	125.20 Fr.	-7.14 Fr.
18-686-666	Marlon Gelpke + Patricia Rosas	14.01.2018	420.20 Fr.	-24.10 Fr.
18-122-650	Samuel Goldberger + Dalia Goldberger	27.02.2018	2'120.00 Fr.	-94.28 Fr.

Abbildung 8: Mockup von dem Screen «Laufende Abrechnungsperiode»

Abbildung 8 stellt die laufende Abrechnungsperiode dar. Darin werden alle Gutscheine, die eingelöst, jedoch unter den beteiligten Geschäften noch nicht abgerechnet wurden,

dargestellt. Die Darstellung gibt zu erkennen, welche Auslagen oder Einkünfte das angemeldete Geschäft je Gutschein erwartet. Übergeordnet ist eine Zusammenfassung vorgesehen, welche die kumulierten Auslagen oder Einkünfte darstellt.

6.1.2.7 Vergangene Abrechnungen

Die in Abbildung 32 ersichtliche Darstellung beruht auf demselben Aufbau wie die laufende Abrechnungsperiode, die in Abbildung 8 ersichtlich ist. Der wesentliche Unterschied besteht darin, dass es sich bei den vergangenen Abrechnungsperioden um eine Liste von abgeschlossenen Abrechnungen handelt.

6.1.2.8 Häufig gestellte Fragen

Abbildung 33 gruppiert häufig gestellte Fragen in verschiedene Kategorien. Die Fragen dienen den Benutzern der Applikation als erste Anlaufstelle bei allfälligen Unklarheiten.

6.1.2.9 Offene Zahlungen

Mit der Gutscheinverwaltungssoftware werden Abrechnungen in regelmässigen Abständen durchgeführt. Aus diesen Abrechnungen entstehen Ausgleichstransaktionen, die zwischen den teilnehmenden Geschäften getätigt werden müssen. Abbildung 31 stellt offene Zahlungsausgänge und offene Zahlungseingänge dar. Sobald eine Zahlung bei einem Gläubigergeschäft eingegangen ist, kann diese Zahlung in der Ansicht von Abbildung 31 bestätigt werden. Entsprechend verschwindet der offene Zahlungsausgang beim Schuldnergeschäft.

6.1.2.10 PDF-Bestätigungen

Die Applikation sieht das Generieren von PDF-Berichten an verschiedenen Stellen vor. Dabei wird jeweils eine Zwischenbestätigung oder bei eingelösten Gutscheinen eine Abschlussbestätigung generiert. Der strukturelle Aufbau dieser Berichte ist der Abbildung 34 zu entnehmen.

6.2 Systemarchitektur und Technologiewahl

In diesem Abschnitt wird einleitend aufgezeigt, welche Gründe für die Umsetzung einer webbasierten Applikation sprechen. Anschliessend werden die eingesetzten Technologien beleuchtet. In Abschnitt 6.6 werden weitere mögliche Umsetzungsvarianten aufgezeigt und dabei deren Unterschiede zur gewählten Variante erläutert.

In einem ersten Schritt wird anhand einer gewichteten Nutzwertanalyse evaluiert, welche Art von Software sich für dieses Vorhaben am besten eignet. Zur Auswahl stehen eine native Desktop-Applikation, eine browserbasierte Web-Applikation oder eine native Mobile-Applikation.

Kriterium	Gewicht	Native Desktop-Applikation		Browserbasierte Web-Applikation		Native Mobile-Applikation	
		Punkte	gewichtet	Punkte	gewichtet	Punkte	gewichtet
Performance	35%	10	3.5	8	2.8	7	2.45
Wartbarkeit	20%	3	0.6	10	2	5	1
Nützlichkeit	45%	6	2.7	10	4.5	4	1.8
Summe	100%		6.8		9.3		5.25

Tabelle 6: Gewichtete Nutzwertanalyse zur Wahl der Softwareart

Die Punktevergabe in Tabelle 6 beruht auf persönlichen Erfahrungswerten und Einschätzungen. Das Kriterium Performance definiert sich durch die durchschnittlichen Lade- und Wartezeiten der Applikation. Wie gut sich die Codebasis zu einem späteren Zeitpunkt warten und aktualisieren lässt, wird im Kriterium «Wartbarkeit» bewertet. Das Kriterium «Nützlichkeit» bewertet die Eignung der Softwareart hinsichtlich deren Verwendungsart.

Die Evaluation der gewichteten Nutzwertanalyse (Tabelle 6) zeigt, dass es vor allem aus Gründen der Nützlichkeit und Wartbarkeit am sinnvollsten ist, eine browserbasierte Web-Applikation zu erstellen.

Web-Applikationen, die im Browser angewendet werden, bringen den Vorteil mit sich, dass sie unterschiedlichen Infrastruktur-Voraussetzungen gerecht werden und somit plattformunabhängig funktionieren. Für die Gutscheinverwaltung ist das ideal, da die Geschäfte verschiedene Geräte mit verschiedenen Betriebssystemen nutzen.

6.2.1 Frontend

Für die Entwicklung von dem Frontend wird die, zum Zeitpunkt der Umsetzung, aktuellste Version 6.0.0 des in TypeScript geschriebenen Frameworks Angular verwendet. Angular ist eine Open-Source-Lösung, deren Entwicklung massgeblich von Google vorangetrieben wird (Angular, o. J.-a).

Mit Angular kann die Applikation als Single Page Application (SPA) umgesetzt werden, womit signifikante Performancesteigerungen im Vergleich zu klassischen Webseiten erreicht werden. Bei einer Single Page Application wird nur beim initialen Aufruf eine HTML-Seite vom Server gerendert (Suresh, 2016). Anschliessend wird mit der mitgeschickten Application Engine zwischen 90 und 95 Prozent des gesamten Codes im Browser des Benutzers ausgeführt (Suresh, 2016). Das bedeutet, dass der Webserver, der die initiale Seite ausliefert, bei den anschliessenden Interaktionen der Applikation nahezu nicht mehr belastet wird und somit eine extrem hohe Anzahl User handhaben kann (Suresh, 2016). Angular besticht zudem durch die Verwendung von TypeScript, einem Superset von JavaScript, womit das Typisieren von Variablen und Objekten ermöglicht wird.

Bei der verwendeten Version 6 von Angular handelt es sich um eine Long Term Support (LTS) Version, womit die Weiterentwicklung und alle notwendigen Security Updates für mindestens eineinhalb Jahre von Google gewährleistet sind (JAX Editorial Team, 2018).

Die verwendeten Versionen der verschiedenen Pakete für das Frontend sind der Abbildung 9 zu entnehmen. Der gesamte Frontend-Code ist dem mitgelieferten ZIP-File zu entnehmen.


```

{
  "name": "frontend",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve --host 0.0.0.0 --proxy-config proxy.conf.json",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "6.0.0",
    "@angular/cdk": "^6.0.1",
    "@angular/common": "6.0.0",
    "@angular/compiler": "6.0.0",
    "@angular/core": "6.0.0",
    "@angular/forms": "6.0.0",
    "@angular/http": "6.0.0",
    "@angular/material": "^6.0.1",
    "@angular/material-moment-adapter": "^6.0.1",
    "@angular/platform-browser": "6.0.0",
    "@angular/platform-browser-dynamic": "6.0.0",
    "@angular/router": "6.0.0",
    "core-js": "^2.4.1",
    "moment": "^2.22.1",
    "rxjs": "^6.1.0",
    "zone.js": "^0.8.26"
  },
  "devDependencies": {
    "@angular/cli": "6.0.0",
    "@angular/compiler-cli": "6.0.0",
    "@angular/language-service": "6.0.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codecyzer": "^4.0.1",
    "jasmine-core": "~2.8.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "2.0.0",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "^1.2.1",
    "karma-jasmine": "~1.1.0",
    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.1.2",
    "ts-node": "~4.1.0",
    "tslint": "~5.9.1",
    "typescript": "2.7.2",
    "@angular-devkit/build-angular": "~0.6.0"
  }
}

```

Abbildung 9: Verwendete Paketversionen für das Frontend

6.2.2 Backend

Für die Entwicklung von dem Backend, konkret von der REST-API, wird das in Python geschriebene Open-Source Webframework Django mit der dazugehörigen Erweiterung DjangoREST verwendet. Die Programmiersprache Python überzeugt mit ihrer klaren und gut verständlichen Syntax sowie der Möglichkeit, mit vergleichsweise kleinem Zeitaufwand viel bewerkstelligen zu können (Hansen, 2017).

Zwei der wichtigsten Vorteile von dem Webframework Django sind gemäss Hansen die einfache Skalierbarkeit sowie die eingebauten Sicherheitsmassnahmen in verschiedenen Belangen (2017). Als wichtigster Nachteil wird die Voraussetzung, das gesamte Framework gut zu kennen, genannt (Hansen, 2017).

6.2.3 Webserver

Für das spätere Deployment und den produktiven Betrieb auf einem Server wird ein Webserver benötigt, der einerseits die statischen Files von dem Frontend ausliefert und andererseits die in Python geschriebene REST-API zur Verfügung stellt.

Für die entwickelte Applikation wird der plattformunabhängige Webserver nginx verwendet. Gemäss einer Statistik von W3 Techs werden, zum Zeitpunkt vom 19. Mai 2018, 64.0% aller Top 10'000 Webseiten mit einem nginx-Webserver betrieben (W3 Techs, 2018). Gemäss Herstellerangaben eignet sich nginx nebst der Auslieferung von statischen Dateien besonders gut als Load Balancer sowie als Proxy-Server (NGINX, 2018). Die Gutscheinverwaltungsapplikation macht von den Funktionalitäten des Proxy-Servers Gebrauch, denkbar ist auch der Einsatz eines Load Balancer zu einem späteren Zeitpunkt.

6.2.4 Containervirtualisierung

Die Gutscheinapplikation wird auf einem macOS-System entwickelt und anschliessend auf einen Cloud-Server mit Linux CentOS als Betriebssystem deployt. Das heisst, es bestehen zwei verschiedene Environments, auf denen die Applikation und all ihre Komponenten lauffähig sein müssen. Verschiedene Environments bringen verschiedene Eigenschaften mit sich, die verschiedene, gewollte oder ungewollte, Einflüsse auf die Applikation haben können.

Um Fehler in einer Applikation so effizient wie möglich zu beheben, müssen diese zeitnah und identisch zur Fehlerquelle reproduziert werden können.

Docker ist eine Open-Source Lösung, die es ermöglicht, Applikationen mit all ihren Abhängigkeiten zu virtualisieren und anschliessend in einem Container auf einem beliebigen Linux-basierten Betriebssystem isoliert bereitzustellen (Noyes, 2013). Man spricht hier von der sogenannten Containervirtualisierung. Die bereitgestellten Container können als eine Art leichtgewichtige, virtuelle Maschinen betrachtet werden, wobei der wichtigste Unterschied zu virtuellen Maschinen darin besteht, dass Docker-Container ihren Kernel mit dem Host-Betriebssystem teilen (Cito & Gall, 2016, S. 906). Das Teilen des Kernels ermöglicht es den Docker-Containern, sehr klein in ihrer Grösse zu sein und dabei eine hohe Performance zu erreichen (Cito & Gall, 2016, S. 906).

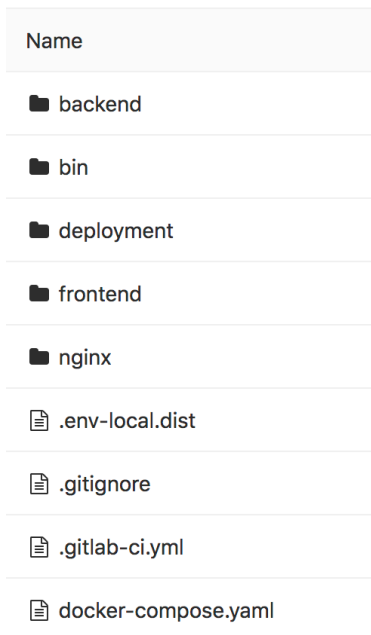
Mit der Verwendung von Docker-Containern muss die Applikation auf unterschiedlichen Environments (Lokale Entwicklungsumgebung, Staging-Umgebung, Produktiv-Umgebung) nicht individuell konfiguriert werden, sondern läuft in allen Instanzen isoliert und

unter denselben Bedingungen. Datenbank-Engine (PostgreSQL), Webserver (nginx), Host-OS (Linux Alpine), sowie alle verwendeten Libraries sind somit überall dieselben, womit eine sehr effiziente Reproduzierbarkeit erreicht wird.

6.2.5 Repository

Der Code der Gutscheinverwaltungs-Applikation wird mit Git versioniert und auf einer selber gehosteten Instanz der GitLab Community Edition gespeichert. Mit den entsprechenden Berechtigungen kann das Repository unter folgender URL aufgerufen werden: <https://git.lab360.cloud/raphael.prader/bachelorarbeit-gutscheinverwaltung/>.

Der Grundaufbau von dem Repository ist der Abbildung 10 zu entnehmen. In den beiden nachfolgenden Unterkapiteln werden die abgebildeten Verzeichnisse und Dateien erläutert.



Name
📁 backend
📁 bin
📁 deployment
📁 frontend
📁 nginx
📄 .env-local.dist
📄 .gitignore
📄 .gitlab-ci.yml
📄 docker-compose.yaml

Abbildung 10: Ordnerstruktur des Repository

6.2.5.1 Verzeichnisse im Repository-Root

Nachfolgend werden die Verzeichnisse aus dem Repository-Root aufgeführt.

`/backend/` Die REST-API beruht auf der aktuellsten LTS-Version 1.11.12 des Python-basierten Webframeworks Django und wird aus diesem Verzeichnis bereitgestellt. Alle relevanten Dateien um die API zur Verfügung zu stellen, befinden sich in diesem Verzeichnis.

<code>/bin/</code>	In diesem Verzeichnis befinden sich Dateien, die die Entwicklungsarbeit erleichtern, indem sie längere Code-Befehle zusammenfassen.
<code>/deployment/</code>	Verschiedene Abläufe, die für das Deployment von Relevanz sind, befinden sich in Konfigurations- und Beschreibungsdateien in diesem Verzeichnis.
<code>/frontend/</code>	Das Frontend beruht auf der aktuellsten Version 6.0.0 des TypeScript-basierten Webframeworks Angular. Alle relevanten Files um das Frontend zur Verfügung zu stellen, befinden sich in diesem Verzeichnis.
<code>/nginx/</code>	In diesem Verzeichnis befindet sich der Container zur Bereitstellung des Webservers nginx.

6.2.5.2 Dateien im Repository-Root

Nachfolgend werden die Dateien aus dem Repository-Root erläutert.

<code>.env-local.dist</code>	Dieses File dient als Template-File, welches als <code>.env-local</code> erstellt wird. Darin befinden sich sämtliche Environment-Variablen, die nur für die lokale Entwicklung relevant sind. Aus diesem Grund wird das File <code>.env-local</code> nicht in die Versionierung aufgenommen, sondern mittels <code>.gitignore</code> ausgeschlossen.
<code>.gitignore</code>	Dieses File regelt, welche Files und welche Verzeichnisse von der Git-Versionierung ignoriert werden sollen.
<code>.gitlab-ci.yml</code>	In diesem File werden die Jobs, die GitLab im Rahmen der Continuous Integration durchführen soll, definiert.
<code>docker-compose.yml</code>	Dieses File definiert die verschiedenen Services, die beim Start des Docker-Services im Root-Verzeichnis gestartet werden sollen. Im Falle der lokalen Entwicklungsumgebung sind es nebst der Postgres-Datenbank und dem Front- und Backend auch noch ein SMTP-Server, der die verschickten Mails abfängt.

6.3 REST-API

Das Hauptartefakt der entwickelten Lösung bildet das Application Programming Interface, kurz API. In den nachfolgenden Unterpunkten ist erläutert, welche Art von API

umgesetzt wurde, wie sie funktioniert, wie die Authentifizierung funktioniert und welche Endpunkte nach erfolgreicher Authentifizierung vorliegen.

Ein Application Programming Interface (API) stellt in seiner Hauptfunktion eine Schnittstelle zum Informationsaustausch zwischen verschiedenen Geräten dar, die sich in einem Netzwerk befinden. Für die vorliegende Lösung wurde eine REST-API entwickelt, die über das World Wide Web (WWW) aufrufbar ist.

6.3.1 Was ist REST

REST ist die englische Abkürzung für Representational State Transfer und stellt eine API auf dem HTTP/S-Protokoll dar. Eine API erlaubt es Maschinen und Benutzern, über vordefinierte Strukturen an bestimmte Ressourcen zu kommen. Abhängig von den Berechtigungen, die gegenüber der API vorgewiesen werden, können unterschiedliche Benutzer an unterschiedliche Ressourcen gelangen.

Die vier wichtigsten Eigenschaften einer Ressource sind gemäss Helmich (2013):

- Adressierbarkeit
- Zustandslosigkeit
- Einheitliche Schnittstelle
- Entkoppelung von Ressourcen und Repräsentation

Konkret bedeutet das, dass jede Ressource über eine eindeutige Adresse erreichbar sein muss. Weiter existieren keine Benutzersitzungen, sondern es werden bei jeder Anfrage jeweils alle Informationen erneut mitgeschickt. Überdies sollen alle Ressourcen über dieselben Schnittstellen zu erreichen sein, wobei die Ressourcen nicht an eine bestimmte Art der Repräsentation gebunden sind.

6.3.2 HTTP Statuscodes

Mit der Verwendung des HTTP-Protokolls können verschiedene Statuscodes als Antwort von dem Webserver an den Anfrager geschickt werden. Dabei handelt es sich jeweils um eine dreistellige Zahl, wobei die erste Ziffer die Kategorie des jeweiligen Status beschreibt. Hunter II listet die verschiedenen Kategorien auf und beschreibt sie wie folgt (2017, S. 32f.):

1xx Informationen

Antworten in diesem Code-Range sind rein informativer Natur deuten darauf hin, dass die Bearbeitung von Anfragen noch andauern.

2xx Erfolgreiche Operation

Antworten in diesem Code-Range bestätigen die erfolgreiche Verarbeitung einer Anfrage.

3xx Umleitung / Weiterleitung

Antworten in diesem Code-Range stellen eine Weiterleitung auf eine andere Ressource dar, wobei die meisten APIs keine Antworten mit diesen Status-Codes schicken.

4xx Client Fehler

Antworten in diesem Code-Range signalisieren einen Fehler von dem Anfragesteller. Die Anfragen verändern keine Daten auf dem Server, sondern weisen den Absender darauf hin, dass seine Eingabe fehlerhaft ist.

5xx Server Fehler

Antworten in diesem Code-Range deuten auf einen serverseitigen Fehler in der Bearbeitung der Anfrage. Dem Anfragesteller ist dabei nicht bekannt, ob seine Daten gespeichert werden konnten oder nicht.

6.3.3 Datenbank Design

In Abbildung 35 ist die erstellte Datenstruktur anhand eines UML-Klassendiagrammes ersichtlich. Die Tabellen mit dem Prefix auth, django, oauth2, filer und easy_thumbnails stammen von dem eingesetzten Framework. In den nachfolgenden Punkten werden die für dieses Projekt spezifisch erstellten Tabellen erläutert.

6.3.3.1 Customer

Die Django Model Klasse «Customer» wird in der Datenbank als Tabelle mit der Bezeichnung «customers_customer» repräsentiert. In dieser Tabelle werden die Kontaktinformationen der Kunden gespeichert, sowie die Information, wann und von welchem Benutzer der Datensatz erstellt wurde.

6.3.3.2 Customer Contact

Die Django Model Klasse «CustomerContact» wird in der Datenbank als Tabelle mit der Bezeichnung «customers_customercontact» repräsentiert und stellt eine Verbindung zur Tabelle «customers_customer» her. In der Tabelle wird die Telefonnummer und die E-

Mailadresse gespeichert, wobei jeweils deklariert wird, zu welchem der beiden Kunden die Information gehört.

6.3.3.3 Report Temporary Access

Die Django Model Klasse «ReportTemporaryAccess» wird in der Datenbank als Tabelle mit der Bezeichnung «customers_reporttemporaryaccess» repräsentiert. Die Tabelle verantwortet temporäre Zugriffe auf die PDF-Berichte ohne Authentifizierung. Das Erstellen einer solchen Zugriffsberechtigung kann nur über eine autorisierte API-Anfrage geschehen, worauf dann eine URL mit einer UUID generiert wird. Die Zugriffsberechtigung ist während 15 Minuten gültig.

6.3.3.4 Purchase

Die Django Model Klasse «Purchase» wird in der Datenbank als Tabelle mit der Bezeichnung «purchases_purchase» repräsentiert. Die Tabelle beinhaltet sämtliche Einkäufe aller Kunden, wobei diese jeweils mit dem entsprechenden Kaufdatum und dem Betrag versehen sind.

6.3.3.5 Final Discount Rule

Die Django Model Klasse «FinalDiscountRule» wird in der Datenbank als Tabelle mit der Bezeichnung «purchases_finaldiscountrule» repräsentiert. Sie beinhaltet die Rabattregeln die besagen, ab welcher Einkaufssumme welche Rabattsumme resultiert.

6.3.3.6 Company

Die Django Model Klasse «Company» wird in der Datenbank als Tabelle mit der Bezeichnung «companies_company» repräsentiert. Jeder Eintrag in dieser Tabelle entspricht einer der teilnehmenden Firmen. Die Verbindung zum Systembenutzer, welcher für die Authentifizierung verantwortlich ist, ist so angelegt, dass einem Benutzer mehrere Firmen zugewiesen werden können.

6.3.3.7 Customer Closure

Die Django Model Klasse «CustomerClosure» wird in der Datenbank als Tabelle mit der Bezeichnung «accounting_customerclosure» repräsentiert. Direkt nach dem erfolgreichen Einlösen eines Gutscheins wird ausgerechnet, welches Geschäft welche Forderungen zu begleichen hat, sowie welchem Geschäft welche Guthaben zustehen.

6.3.3.8 Transaction

Die Django Model Klasse «Transaction» wird in der Datenbank als Tabelle mit der Bezeichnung «accounting_transaction» repräsentiert. Einträge in der Transaktionstabelle werden dann erstellt, wenn ein Finanzausgleich zwischen den Geschäften durchgeführt wird. Dazu werden alle noch nicht verrechneten Einträge aus der Tabelle «accounting_customerclosure» so miteinander summiert, dass zwischen jeweils zwei Geschäften höchstens eine Zahlung entsteht. Die zugehörige Verbindungstabelle «accounting_transaction_consulted_closures» hält fest, welche Einträge aus der Tabelle «accounting_customerclosure» für die jeweilige Zahlung verwendet werden.

6.3.4 API Design und Development

Zur Spezifikation und Beschreibung der API wird die OpenAPI Spezifikation genutzt. Die OpenAPI Spezifikation wird von der OpenAPI Initiative, einem OpenSource Projekt aus der Linux Foundation, erstellt und gefördert und als herstellerneutrales Beschreibungsformat betrieben (OpenAPI Initiative, o. J.). Eine grafische Übersicht aller Endpunkte ist der Abbildung 11 zu entnehmen.

Nachfolgend sind die Verwendungszwecke der verschiedenen Endpunkte mit den verschiedenen HTTP-Methoden beschrieben, wobei gleich auch die dahinterliegende Businesslogik erläutert wird. Beim Design einer Applikation mit einer API ist es wichtig, dass die Businesslogik innerhalb der API geschieht, so dass die Benutzer keinen Einfluss auf diese haben können (Hunter II, 2017, S. 14).

Gutscheinverwaltungs API ^{1.0.0}

[Base URL: app.hochzeits-dienstleister.ch]

The API for interacting with the application which handles the voucher management of «Die besten Hochzeitsdienstleister».

Schemes

HTTPS ▾

Authorize 🔒

api ▾

GET	/api/companies/
GET	/api/companies/{id}/
GET	/api/customers/
POST	/api/customers/
GET	/api/customers/{customer_pk}/contacts/
POST	/api/customers/{customer_pk}/contacts/
GET	/api/customers/{customer_pk}/contacts/{id}/
PUT	/api/customers/{customer_pk}/contacts/{id}/
PATCH	/api/customers/{customer_pk}/contacts/{id}/
DELETE	/api/customers/{customer_pk}/contacts/{id}/
GET	/api/customers/{customer_pk}/purchases/
POST	/api/customers/{customer_pk}/purchases/
GET	/api/customers/{customer_pk}/purchases/{id}/
PUT	/api/customers/{customer_pk}/purchases/{id}/
PATCH	/api/customers/{customer_pk}/purchases/{id}/
GET	/api/customers/{id}/
PUT	/api/customers/{id}/
PATCH	/api/customers/{id}/
GET	/api/customers/{id}/generate_pdf_report/
GET	/api/customers/{id}/get_purchases_details/
POST	/api/customers/{id}/send_pdf_report/
GET	/api/discount_rules/
GET	/api/discount_rules/{id}/
GET	/api/faq/
GET	/api/faq/{id}/
POST	/api/logout/
GET	/api/open_transactions/
GET	/api/open_transactions/{id}/
PUT	/api/open_transactions/{id}/
PATCH	/api/open_transactions/{id}/
GET	/api/previous_closures/
GET	/api/previous_closures/{id}/
GET	/api/purchases/
GET	/api/purchases/{id}/
GET	/api/upcoming_closures/
GET	/api/upcoming_closures/{id}/
GET	/api/user/

Abbildung 11: Grafische Darstellung aller API-Endpunkte

Pfad	Methode	Rückgabewert / Verwendungszweck
/api/companies/	GET	Liefert Informationen über die dem angemeldeten Benutzer zugeordneten Geschäfte.
/api/companies/{id}/	GET	Liefert Informationen über das mit {id} spezifizierte Geschäft. Dabei muss es sich um ein Geschäft handeln, das dem angemeldeten Benutzer zugeordnet ist.
/api/customers/	GET	Liefert die Kontaktinformationen aller registrierten Kunden. Zudem werden API-Pfade, die weiterführende Informationen zu diesem Kunden liefern, abgebildet.
/api/customers/	POST	Erstellt einen neuen Eintrag für ein Kundenpaar, wobei der angemeldete Benutzer als Erfasser dieser Kontaktdaten eingetragen wird.
/api/customers/{customer_pk}/contacts/	GET	Liefert eine Liste aller zusätzlichen Kontaktinformationen zu den mit {id} bestimmten Kunden.
/api/customers/{customer_pk}/contacts/	POST	Erstellt einen neuen Eintrag mit zusätzlichen Kontaktinformationen zu den mit {id} bestimmten Kunden.
/api/customers/{customer_pk}/contacts/{id}/	GET	Liefert den einzelnen Eintrag mit zusätzlichen Kontaktinformationen eines Kundenpaares, wobei die Kundennummer dem Wert {customer_pk} und die ID zur Kontaktinformation dem Wert {id} entsprechen muss.
/api/customers/{customer_pk}/contacts/{id}/	PUT	Aktualisiert den einzelnen Eintrag mit zusätzlichen Kontaktinformationen eines Kundenpaares, wobei die Kundennummer dem Wert {customer_pk} und die ID zur Kontaktinformation dem Wert {id} entsprechen muss.
/api/customers/{customer_pk}/contacts/{id}/	PATCH	<i>Gleiche Funktion wie die Zeile oberhalb.</i>
/api/customers/{customer_pk}/contacts/{id}/	DELETE	Löscht den einzelnen Eintrag mit zusätzlichen Kontaktinformationen eines Kundenpaares, wobei die Kundennummer dem Wert {customer_pk} und

		die ID zur Kontaktinformation dem Wert {id} entsprechen muss.
/api/customers/{customer_pk}/purchases/	GET	Gibt die Liste aller getätigten Einkäufe von dem mit {customer_pk} bestimmten Kundenpaar zurück.
/api/customers/{customer_pk}/purchases/	POST	Erstellt einen Einkauf für das mit {customer_pk} spezifizierte Kundenpaar. Die Bedingungen dazu sind, dass das Paar einerseits seinen letzten Einkauf noch nicht getätigt hat und andererseits noch keinen Einkauf in dem angemeldeten Geschäft getätigt hat. Zusätzlich darf das Hochzeitsdatum nicht in der Vergangenheit liegen.
/api/customers/{customer_pk}/purchases/{id}/	GET	Liefert Informationen zu einem einzelnen Einkauf, wobei die Kundennummer dem Wert {customer_pk} und die ID zu dem Einkauf dem Wert {id} entsprechen muss.
/api/customers/{customer_pk}/purchases/{id}/	PUT	Aktualisiert einen einzelnen Einkauf, wobei die Kundennummer dem Wert {customer_pk} und die ID zu dem Einkauf dem Wert {id} entsprechen muss. Ein Einkauf kann nur von jenem Benutzer angepasst werden, der ihn initial erfasst hat.
/api/customers/{customer_pk}/purchases/{id}/	PATCH	<i>Gleiche Funktion wie die Zeile oberhalb.</i>
/api/customers/{id}/	GET	Liefert Informationen zu einem einzelnen Kundenpaar, wobei die Kundennummer dem Wert {id} entsprechen muss.
/api/customers/{id}/	PUT	Aktualisiert die Informationen zu einem einzelnen Kundenpaar, wobei die Kundennummer dem Wert {id} entsprechen muss.
/api/customers/{id}/	PATCH	<i>Gleiche Funktion wie die Zeile oberhalb.</i>
/api/customers/{id}/generate_pdf_report/	GET	Liefert die URL zum Download des PDF-Berichts.
/api/customers/{id}/get_purchases_details/	GET	Liefert die aktuellen Zwischenstände zu einem Kundenpaar, wobei die Kundennummer dem Wert {id} entsprechen muss.

/api/customers/{id}/ send_pdf_report/	POST	Verschiebt den PDF-Bericht per E-Mail an das Kundenpaar, das mit der Kundennummer in {id} bestimmt ist.
/api/discount_rules/	GET	Liefert eine Liste aller geltenden Rabattregeln.
/api/discount_rules/{id}/	GET	Liefert die Informationen einer geltenden Rabattregel, wobei die gewünschte Regel mit dem Wert {id} bestimmt ist.
/api/faq/	GET	Liefert eine Liste aller FAQ-Kategorien mit den zugehörigen Einträgen.
/api/faq/{id}/	GET	Liefert die Informationen einer FAQ-Kategorie mit den dazugehörigen Einträgen, wobei die Kategorie mit dem Wert {id} bestimmt ist.
/api/logout/	POST	Meldet den authentifizierten Benutzer ab.
/api/open_transactions/	GET	Gibt die Liste aller offenen Transaktionen zurück, die den angemeldeten Benutzer betreffen.
/api/open_transactions/{id}/	GET	Liefert die Informationen zu einer spezifischen Transaktion, wobei diese mit dem Wert {id} bestimmt ist.
/api/open_transactions/{id}/	PUT	Aktualisiert eine offene Transaktion, wobei diese mit dem Wert {id} bestimmt ist. Es kann nur das Bezahldatum angepasst werden, sofern der angemeldete Benutzer als Gläubiger eingetragen ist.
/api/open_transactions/{id}/	PATCH	<i>Gleiche Funktion wie die Zeile oberhalb.</i>
/api/previous_closures/	GET	Gibt die Liste aller Kundenabschlüsse zurück, an denen der angemeldete Benutzer beteiligt ist und die bereits unter den Geschäften verrechnet wurden.
/api/previous_closures/{id}/	GET	Gibt Informationen zu einem einzelnen, bereits verrechneten Kundenabschluss zurück, welcher mit dem Wert {id} bestimmt ist.
/api/purchases/	GET	Gibt die Liste aller getätigten Einkäufe zurück.
/api/purchases/{id}/	GET	Gibt Informationen zu einem einzelnen Einkauf zurück, wobei dieser mit dem Wert {id} bestimmt ist.
/api/upcoming_closures/	GET	Gibt die Liste aller Kundenabschlüsse zurück, an denen der angemeldete Benutzer beteiligt ist und die noch nicht unter den Geschäften verrechnet wurden.

/api/upcoming_closures/{id}/	GET	Gibt Informationen zu einem einzelnen, noch nicht verrechneten Kundenabschluss zurück, welcher mit dem Wert {id} bestimmt ist.
/api/user/	GET	Gibt Informationen zu dem angemeldeten Systembenutzer zurück.

6.3.5 Authentifizierung mit OAuth2

Die Applikation speichert persönliche Daten der Kunden. Diese gilt es vor unerlaubten Zugriffen zu schützen. Die Benutzung der API ist daher authentifizierten Benutzern vorbehalten. Dieser Abschnitt handelt davon, um welche Art von Authentifizierung es sich handelt und wie sie funktioniert.

Für die Authentifizierung wird OAuth2 verwendet, ein Verfahren das unter anderem von Hunter II als Standard für die Authentifizierung bei modernen HTTP APIs beschrieben wird (2017, S. 126). Wie schon in Kapitel 6.3.1 beschrieben, sind Anfragen an eine REST-API statuslos, womit bei jeder Anfrage eine Authentifizierung benötigt wird. Genau dazu ist das Verfahren von OAuth2 ausgelegt, denn es wird bei jeder Anfrage ein Access Token mitgeschickt, der den jeweiligen Nutzer eindeutig identifiziert und dessen Berechtigungen kennt. Der konkrete Ablauf, wie die Authentifizierung am Beispiel der Gutscheinapplikation abläuft, ist in Abbildung 12 in Form eines Programmablaufplans ersichtlich.

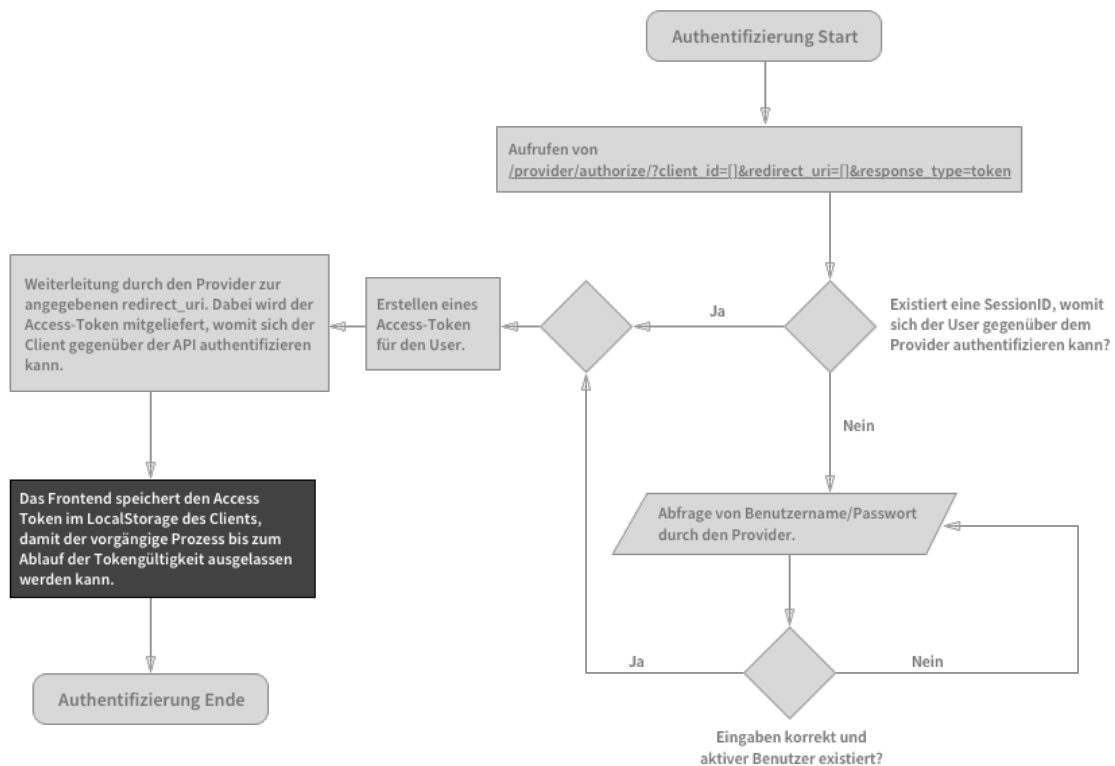


Abbildung 12: Programmablaufplan für den Authentifizierungsprozess

6.4 Frontend Umsetzung

Dieser Abschnitt beschreibt die Umsetzung von dem Frontend mit dem Framework Angular. Die Grundlagen zur Technologie sind im Punkt 6.2.1 erläutert, folglich wird hier auf die Umsetzungsdetails eingegangen. Dazu wird die Anwendung von Komponenten, Interfaces, Services, und Routing beschrieben, bevor zum Schluss das Resultat von der Frontend Umsetzung aufgezeigt wird.

6.4.1 Komponenten

Komponenten stellen das wichtigste Konzept bei der Entwicklung von User Interfaces mit Angular dar (Angular, o. J.-b). Eine mit Angular erstellte Applikation besteht aus verschiedenen Verschachtelungen von unterschiedlichen Komponenten, wobei jede Komponente einem HTML-Template zugewiesen ist, das die Komponente rendert (Angular, o. J.-b). Der grosse Vorteil von Komponenten, der auch in dieser Applikation Anwendung findet, ist, dass sie an unterschiedlichen Stellen eingesetzt und somit mehrfach verwendet werden können.

In der Gutscheinapplikation sind Komponenten in gewöhnliche Komponente und View-Komponente aufgeteilt. Eine View-Komponente stellt das Grundgerüst einer einzelnen Applikationsansicht dar, worin dann wiederum verschiedene andere Komponenten eingebettet sind. In den Komponenten finden verschiedene Datenaufbereitungsoperationen statt, nicht aber ist Businesslogik in Angular integriert. Diese wird vollständig durch die REST-API behandelt. Im Sinne der peripheren Prüfmethode nach Franz & Mattes wird bei Formularen bereits eine clientseitige Datenvalidierung vorgenommen, bevor die Daten an die REST-API geschickt werden (1991, S. 27). Das Resultat der umgesetzten Komponenten ist im Anhang im Abschnitt 11.5 ersichtlich.

6.4.2 Interfaces

Interfaces sind ein wichtiger Bestandteil einer fortschrittlichen Applikation, die mit TypeScript umgesetzt ist (Microsoft, o. J.-a). Es handelt sich hierbei um ein Konzept, das direkt von TypeScript stammt und nicht von dem Framework Angular. Mit Interfaces werden verschiedene Objekte definiert und dabei festgelegt, welche Attribute sie beinhalten, von welchem Datentyp diese sind, sowie ob sie zwingend oder optional sind. Interfacedefinitionen können vererbt und somit erweitert werden. Interfaces tragen einen

wichtigen Teil dazu bei, sauberen Code zu schreiben und sind daher auch ein Bestandteil der Gutscheinverwaltungsapplikation.

6.4.3 Services

Damit die Komponenten möglichst schlank und effizient gestaltet werden können, bietet Angular die Möglichkeit, Services zu schreiben. Komponenten sollen im Wesentlichen für den Datenaustausch mit dem Template verantwortlich sein, nicht jedoch für den Verbindungsaufbau und das Datenhandling mit der REST-API (Angular, o. J.-c). Diesem Grundsatz wird in der Gutscheinverwaltungsapplikation Rechnung getragen, indem die gesamte Logik zum Datenaustausch mit der API in verschiedene Services ausgelagert wird.

6.4.4 Routing

Um zwischen den erstellten View-Komponenten navigieren zu können, wird das Routing-Modul von Angular eingesetzt. Das Routing-Modul erkennt die URL aus dem Browser und ersetzt diese, sobald der User zu einer anderen Seite navigiert (Angular, o. J.-d). Für die Konfiguration des Routing-Moduls werden zuerst alle Ansichten mit dem zugehörigen Pfad definiert, gefolgt von einem Wildcard-Selector, der alle nicht erkannten URLs zu einer «Seite nicht gefunden»-Darstellung weiterleitet (Angular, o. J.-d). Die Konfiguration für die Gutscheinverwaltungsapplikation ist der Abbildung 13 zu entnehmen.


```

18 const routes: Routes = [{
19   path: '', component: BaseProtectedViewComponent, canActivate: [AuthGuard],
20   children: [{
21     path: '', redirectTo: '/purchase', pathMatch: 'full'
22   }, {
23     path: 'purchase', component: MakePurchaseViewComponent
24   }, {
25     path: 'purchase_success/:id', component: PurchaseSuccessViewComponent
26   }, {
27     path: 'redeem', component: RedeemVoucherViewComponent
28   }, {
29     path: 'customer-list', component: CustomerListViewComponent
30   }, {
31     path: 'customer/:id', component: CustomerDetailViewComponent
32   }, {
33     path: 'other-functions', component: OtherFunctionsViewComponent
34   }, {
35     path: 'other-functions/upcoming-closure', component: UpcomingClosureViewComponent
36   }, {
37     path: 'other-functions/open-transactions', component: OpenTransactionsViewComponent
38   }, {
39     path: 'other-functions/previous-closures', component: PreviousClosuresViewComponent
40   }, {
41     path: 'other-functions/faq', component: FaqViewComponent
42   }
43   ], {
44     path: 'auth', component: AuthViewComponent
45   }, {
46     path: '**', component: PageNotFoundViewComponent, canActivate: [AuthGuard]
47   }
48 ];

```

Abbildung 13: Definierte Routen für die Applikation

6.4.5 Resultat

Das mit Angular entwickelte Frontend ist vollumfänglich funktionsfähig und interagiert fehlerfrei mit der entwickelten REST-API. Im Anhang im Abschnitt 11.6.2 finden sich Zugangsdaten, womit die Applikation während dem Bewertungszeitraum dieser Arbeit getestet werden kann. Weiter befinden sich Screenshots im Abschnitt 11.5, die die wichtigsten Ansichten der Applikation darstellen.

Sobald bekannt ist, wo das erstellte iFrame zukünftig eingebunden wird, müssen die «X-Frame-Options» der nginx-Konfiguration so angepasst werden, dass das iFrame auf dem entsprechenden Host dargestellt wird.

6.5 DevOps

Dieses Kapitel erläutert den Begriff DevOps einleitend und beleuchtet anschliessend die Techniken Continuous Integration, Continuous Delivery und Continuous Deployment. Dazu wird beschrieben, wie und in welchem Umfang die jeweiligen Techniken Anwendung bei der Entwicklung der Gutscheinapplikation finden.

Gemäss Definition von Gartner repräsentiert der Begriff DevOps eine Veränderung in der IT-Kultur, ermöglicht Prozessverbesserungen und führt dadurch zu schnelleren und zuverlässigeren Auslieferungen von IT-Produkten und IT-Dienstleistungen (Gartner,

o. J.). Der Begriff an sich ist eine Kombination aus den beiden englischen Wörtern Development und Operations.

In den nachfolgenden Unterkapiteln werden drei Techniken, die bei der Softwareentwicklung bestens Anwendung finden, beleuchtet.

6.5.1 Continuous Integration

Continuous Integration verlangt von den Programmierern, dass sie ihre Änderungen am Code so häufig wie möglich in den Hauptzweig (Main Branch) der Code-Basis integrieren, wodurch eine ständige Integration entsteht (Pittet, o. J.). Dabei wird bei jeder Integration überprüft, ob sich die gemachten Änderungen in die bestehende Codebasis fehlerlos integrieren lassen, so dass die Applikation nach wie vor lauffähig ist. Wenn dies der Fall ist, spricht man von einem sogenannten «clean build».

Continuous Integration ermöglicht durch die ständige Überprüfung eine möglichst reibungslose Zusammenarbeit in Teams mit mehreren Entwicklern und informiert die entsprechenden Verantwortlichen bei nicht erfolgreichem Zusammenführen von verschiedenen Code-Änderungen. Continuous Integration bringt aber auch Sicherheit in kleinen Teams oder bei einzelnen Entwicklern, denn auch für sie wird sichergestellt, dass der Code fehlerfrei ist.

Das Git-Repository für die Gutscheinverwaltungs-Applikation befindet sich, wie in Kapitel 6.2.5 erwähnt, auf GitLab. GitLab bietet die Möglichkeit, mit der entsprechenden Konfiguration, Continuous Integration einzurichten. Dazu wird im Root-Verzeichnis vom Repository das File `.gitlab-ci.yml` benötigt und es muss ein Runner, entweder projektspezifisch oder für alle GitLab-Projekte, eingerichtet werden, der die definierten Jobs ausführt (GitLab, o. J.-a). Diese beiden Anforderungen werden in den nachfolgenden Unterpunkten behandelt.

6.5.1.1 gitlab-ci.yml

Das Konfigurationsfile `gitlab-ci.yml` befindet sich im Projekt-Root und definiert, welche Jobs in welcher Reihenfolge unter welchen Bedingungen vom Runner ausgeführt werden sollen. GitLab überprüft bei jedem Hochladen von Änderungen, ob sich ein solches Konfigurationsfile im Repository befindet und übergibt dann dem Runner die entsprechenden Aufgaben (GitLab, o. J.-b).

Für die Gutscheilverwaltungs-Applikation sind drei Stages definiert, die nacheinander ausgeführt werden. Im ersten Schritt (build) wird das Docker-Image für das Backend gebildet und anschliessend in die angegebene Docker-Registry gepusht. Zeitgleich werden die statischen Files, die das Backend benötigt, generiert und für den nächsten Stage (post_build) zur Verfügung gestellt. Letztlich wird in diesem Schritt das gesamte Angular-Frontend für den produktiven Einsatz gebildet.

Das Docker-Image für den nginx-Webserver wird im Stage post_build, erst nach Abschluss des build-Stage, gebildet und danach ebenfalls in die Docker-Registry gepusht.

Im Stage deploy wird, abhängig vom Git-Branch, entweder die Staging-Instanz aktualisiert, oder es wird eine neue Instanz zum Testen der Änderungen erstellt.

Das komplette Konfigurationsfile, das den Runner anweist, ist in Abbildung 14 ersichtlich und ebenfalls im Root-Verzeichnis vom mitgelieferten Repository zu finden.

```

1 stages:
2   - build
3   - post_build
4   - test
5   - deploy
6
7 variables:
8   PYTHON_IMAGE: $CI_REGISTRY/lab369/dev-images/python:latest
9   ANGULAR_IMAGE: $CI_REGISTRY/lab369/dev-images/angular:latest
10
11 build:frontend:
12   stage: build
13   image: $ANGULAR_IMAGE
14   script:
15     - cd frontend
16     - npm install
17     - npm run build
18   artifacts:
19     paths:
20     - frontend/dist/
21
22 build:backend:
23   stage: build
24   image: docker:latest
25   script:
26     - cd backend
27     - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $CI_REGISTRY
28     - docker build --pull -t $CI_REGISTRY_IMAGE/backend:$CI_COMMIT_REF_NAME .
29     - docker push $CI_REGISTRY_IMAGE/backend:$CI_COMMIT_REF_NAME
30
31 build:backend_static:
32   stage: build
33   image: $PYTHON_IMAGE
34   script:
35     - cd backend
36     - pip install -r requirements.txt
37     - python manage.py collectstatic --no-input
38   artifacts:
39     paths:
40     - backend/public_html/static/
41     expire_in: 1 day
42
43 post_build:nginx:
44   stage: post_build
45   image: docker:latest
46   script:
47     - cd nginx
48     - cp -R ../frontend/dist/* public_html/
49     - cp -R ../backend/public_html/* public_html/
50     - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $CI_REGISTRY
51     - docker build -t $CI_REGISTRY_IMAGE/nginx:$CI_COMMIT_REF_NAME .
52     - docker push $CI_REGISTRY_IMAGE/nginx:$CI_COMMIT_REF_NAME
53   dependencies:
54     - build:frontend
55     - build:backend_static
56
57 test:backend:
58   stage: test
59   image: $PYTHON_IMAGE
60   coverage: '/TOTAL.*([0-9]{1,3})%/'
61   script:
62     - cd backend
63     - pip install -r requirements.txt
64     - pip install -r requirements-testing.txt
65     - ./test.sh
66   artifacts:
67     paths:
68     - backend/reports/coverage/
69     expire_in: 1 day
70   dependencies:
71     - build:backend
72
73 deploy:review:
74   stage: deploy
75   image: alpine:3.7
76   script:
77     - 'which ssh-agent || ( apk update && apk add openssh )'
78     - mkdir -p ~/.ssh
79     - eval $(ssh-agent -s)
80     - '[ -f ~/.dockerenv ] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
81     - echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
82     - chmod 600 ~/.ssh/id_rsa
83     - cd deployment
84     - ./create_update.sh
85   environment:
86     name: review/$CI_COMMIT_REF_NAME
87     url: https://$CI_ENVIRONMENT_SLUG.app.hochzeits-dienstleister.ch
88   on_stop: stop_review
89   dependencies:
90     - build:frontend
91     - build:backend_static
92     - post_build:nginx
93   only:
94     - branches
95     except:
96     - master
97
98 stop:review:
99   stage: deploy
100  image: alpine:3.7
101  script:
102    - 'which ssh-agent || ( apk update && apk add openssh )'
103    - mkdir -p ~/.ssh
104    - eval $(ssh-agent -s)
105    - '[ -f ~/.dockerenv ] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
106    - echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
107    - chmod 600 ~/.ssh/id_rsa
108    - cd deployment
109    - ./destroy.sh
110  when: manual
111  environment:
112    name: review/$CI_COMMIT_REF_NAME
113  action: stop
114  only:
115    - branches
116  except:
117    - master
118
119 deploy:staging:
120   stage: deploy
121   image: alpine:3.7
122   script:
123     - 'which ssh-agent || ( apk update && apk add openssh )'
124     - mkdir -p ~/.ssh
125     - eval $(ssh-agent -s)
126     - '[ -f ~/.dockerenv ] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
127     - echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
128     - chmod 600 ~/.ssh/id_rsa
129     - cd deployment
130     - ./create_update.sh
131   environment:
132     name: staging
133     url: https://staging.app.hochzeits-dienstleister.ch
134   dependencies:
135     - build:frontend
136     - build:backend_static
137     - post_build:nginx
138   only:
139     - master
140
141 deploy:prod:
142   stage: deploy
143   image: alpine:3.7
144   script:
145     - 'which ssh-agent || ( apk update && apk add openssh )'
146     - mkdir -p ~/.ssh
147     - eval $(ssh-agent -s)
148     - '[ -f ~/.dockerenv ] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
149     - echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
150     - chmod 600 ~/.ssh/id_rsa
151     - export LOCAL_PORT=3000
152     - cd deployment
153     - ./create_update.sh
154   environment:
155     name: production
156     url: https://app.hochzeits-dienstleister.ch
157   dependencies:
158     - build:frontend
159     - build:backend_static
160     - post_build:nginx
161   when: manual
162   only:
163     - master

```

Abbildung 14: gitlab-ci.yml – CI Konfiguration für das Projekt

6.5.1.2 Runner

GitLab ermöglicht die Konfiguration von geteilten Runners (shared runners) und die Konfiguration von spezifischen Runners (specific runners). Die beiden Runners unterscheiden sich darin, dass ein spezifischer Runner nur für das definierte Repository verantwortlich ist und ihm dadurch spezifische Zugangsdaten oder Konfigurationen mitgegeben werden können (GitLab, o. J.-c).

Für die Gutscheinverwaltungs-Applikation ist ein geteilter Runner gewählt. Dieser Runner ist mit Docker so konfiguriert, dass sämtliche Aufgaben, die dem Runner übergeben werden, isoliert ausgeführt werden.

6.5.2 Continuous Delivery

Continuous Delivery wird als eine Erweiterung von Continuous Integration umschrieben, wobei das automatische Testen und Integrieren der Änderungen genau gleich wie auch schon bei der Continuous Integration verläuft. Im Zuge der Continuous Delivery wird zusätzlich der Release-Prozess weitestgehend automatisiert, so dass die Veröffentlichung auf Knopfdruck geschehen kann (Pittet, o. J.).

Die Praktik von Continuous Delivery bedingt, dass sämtliche Prozessschritte zur Veröffentlichung einer neuen Version automatisiert durchgeführt werden können, ohne dass sie Interaktionen des Verantwortlichen bedingen.

Continuous Delivery reduziert die Komplexität des Softwaredeployments und ermöglicht das Veröffentlichten einer neuen Version innert sehr kurzer Zeit.

Für die Gutscheinverwaltungsapplikation wird Continuous Delivery vollumfänglich genutzt. Das Deployen der produktiven Applikation geschieht im Sinne von Continuous Delivery erst auf manuellen Befehl. Im Konfigurationsfile (Abschnitt 6.5.1.1) ist dazu eine Stage definiert, die über GitLab gestartet werden kann:

Zuerst wird ein Docker-Container, basierend auf Linux Alpine gestartet, um anschließend die Aufgaben von dem Deployment zu übernehmen. Dazu werden die benötigten Dependencies installiert und dem Container wird der Private Key eines SSH-Schlüssel-Paares hinterlegt, wovon sich der PublicKey auf dem Cloud-Server befindet. Anschließend wird im Docker-Container das Shell-Script `create-update.sh` aufgerufen, welches die neusten Docker-Images aus der Registry herunterlädt und alle Services aktualisiert.

6.5.3 Continuous Deployment

Das Konzept von Continuous Deployment hängt bei Continuous Delivery noch einen Schritt an, wobei nicht nur die Möglichkeit des Deployens auf Knopfdruck gegeben wird, sondern das Deployment jeweils automatisch geschieht. Codeänderungen durchlaufen sämtliche Funktions- und Integrationstests, wie sie es bei Continuous Integration bereits tun, und werden nach erfolgreichem Abschluss aller Tests automatisch auf die Produktionsumgebung deployt (Pittet, o. J.).

Im Rahmen dieser Arbeit wird Continuous Deployment für das automatische Aktualisieren der Staging-Umgebung verwendet. Bei jeder Aktualisierung von dem Git-Branch master wird der Prozess durch den GitLab-Runner gestartet und endet damit, dass die neueste Version der Applikation in der Staging-Instanz zum Testen bereitsteht.

6.6 Alternative Lösungsmöglichkeiten

Die Möglichkeiten, die zur Entwicklung einer Webapplikation zur Verfügung stehen, scheinen grenzenlos. Nebst der gewählten Vorgehensweise, die in den vorgängigen Kapiteln beschrieben ist, gibt es noch andere mögliche Lösungswege, die in diesem Abschnitt eruiert werden.

6.6.1 Gewöhnliche Webseite

Grundsätzlich lässt sich die Applikation der Gutscheinverwaltung auch als gewöhnliche Webseite umsetzen, wobei die eingesetzte Backendtechnologie die Daten direkt an das Frontend liefert anstelle der Auslieferung über eine REST-API. Bei der Evaluation der Umsetzungsvariante haben sich zwei entscheidende Nachteile herausgestellt, weshalb von dieser Umsetzungsart abgesehen wurde:

1. Bei jeder Anfrage an die Applikation schickt der Webserver den gesamten Inhalt der Webseite als Antwort zurück, was einerseits zu einer höheren Belastung von dem Server führt und andererseits zu längeren Wartezeiten beim Benutzer. Die Einschränkung in der Benutzerfreundlichkeit durch längere Wartezeiten gilt es zu vermeiden.
2. Eine Umsetzung ohne REST-API verhindert die einfache Anbindung weiterer Dienste an die Applikationslogik zu einem späteren Zeitpunkt. Diese Einschränkung gilt es zu vermeiden.

6.6.2 ReactJS anstelle von Angular

Das verwendete Frontend-Framework Angular ist grundsätzlich eines von vielen. Eine populäre Alternative ist die von Facebook entwickelte JavaScript Library ReactJS. ReactJS funktioniert im Grundsatz ähnlich wie Angular, und setzt ebenfalls stark auf die Entwicklung und Verwendung von Komponenten (React, o. J.). Die Verwendung der REST-API passt bei ReactJS genauso gut in das Konzept wie bei Angular. Ein entscheidender Unterschied ist, dass ReactJS kein in sich vollständiges Framework darstellt, sondern nur eine sehr leichtgewichtige Basis (Eugeniya, o. J.). Das hat zur Folge, dass verschiedene Module von Dritt-Entwicklern hinzugezogen werden müssen und ein optimales Zusammenspiel somit nur schwierig gewährleistet werden kann.

Der Einsatz von Angular hingegen bringt grosse Sicherheit bezüglich Stabilität und einwandfreiem Zusammenspiel, da alle neuen Versionen von Angular zuerst in den mit Angular umgesetzten Produkten von Google getestet werden (Fluin, 2017). Dazu gehören unter anderem Google AdWords, Google Store, Google Analytics und Google Cloud Platform.

Die Entscheidung zur Verwendung von Angular anstelle von ReactJS wurde zudem massgeblich von den persönlichen Präferenzen des Entwicklers beeinflusst.

7 Validierung

Mit der Validierung wird sichergestellt, dass die entwickelte Lösung den Bedürfnissen der Benutzer entspricht und ihnen dabei in allen Prozessschritten die notwendigen Interaktionsmöglichkeiten zur Verfügung stehen. Somit dient die Validierung der Sicherstellung, dass die bereitgestellte Lösung nach Abschluss aller Arbeiten praxistauglich ist.

Die Validierung der entwickelten Lösung wird für das Frontend und für das Backend gemacht.

In den nachfolgenden Paragraphen ist die Validierung von dem User Interface sowie die Validierung der REST-API aufgeführt.

7.1 Validierung User Interface

Die Usability der Gutscheinverwaltungsapplikation wird anhand der erstellten Mockups auf die Anforderungen der zukünftigen Benutzer getestet. Die Validierung erfolgt in einem iterativen Prozess und ist in drei Validierungsschritte unterteilt: Validierung durch den Verfasser, Validierung durch Labhart-Chronometrie, Validierung durch übrige Teilnehmerfirmen.

Anhand der in Kapitel 6.1.1 definierten User Stories wird überprüft, ob alle relevanten Informationen dargestellt werden und das System die notwendigen Interaktionsmöglichkeiten bietet.

Sämtliches Feedback wurde in Form von Issues auf dem GitLab-Projekt für die Mockups erfasst und anschliessend bearbeitet. Die Issues sind mit Labels in die Kategorien *Bug Report*, *Change Request* und *Feature Request* eingeteilt. Ein zwischenzeitlicher Ausschnitt aus dem Kanban-Board ist in Abbildung 15 zu sehen.

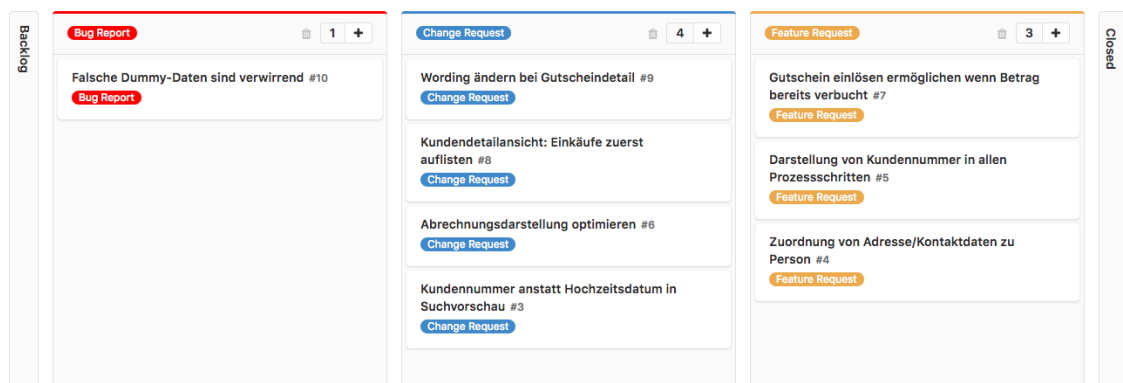


Abbildung 15: Auszug aus dem Kanban-Board auf GitLab zur Mockup-Validierung

7.1.1 Validierung durch den Verfasser

Zur erstmaligen Validierung durch den Verfasser werden die definierten User Stories mit den erstellten Mockups verglichen. Dabei wird überprüft, ob die darin spezifizierten Erwartungen durch das System ermöglicht werden. Aus dieser ersten Validierung resultieren drei Issues, die in Tabelle 7 ersichtlich sind.

ID	Erfasser	Kategorie	Beschreibung	Lösung
#1	@rprader	Change Request	PDF Berichte für Zwischenstände und Abschlüsse fehlen. Diese werden für die User Story US-5 benötigt.	Mockup für PDF Berichte wurde hinzugefügt.
#2	@rprader	Change Request	Home-Screen braucht es nicht, da er keinen Mehrwert bietet.	Home-Screen wurde aus den Mockups entfernt. Nach erfolgreichem Login gelangt der User somit direkt zur Erfassungsmaske, in der Einkäufe verbucht werden.
#10	@rprader	Bug Report	Die zufällig gewählten Zahlen sind in den Berechnungen jeweils nicht korrekt, was zu Verwirrungen führen kann.	Die Zahlenbeispiele wurden so angepasst, dass sie korrekt sind und reale Daten sein könnten.

Tabelle 7: Issues nach Validierung durch den Erfasser

7.1.2 Validierung durch Labhart-Chronometrie

Die Mockups werden, nach der Validierung durch den Verfasser, zuerst von der Firma Labhart-Chronometrie validiert, da der Firma vertiefte Kenntnisse der Anforderungen zugeschrieben werden. Labhart-Chronometrie ist die ursprüngliche Initiatorin des Gutscheinsystems und zeichnete sich bisher für die Abrechnung unter den Geschäften verantwortlich. Die resultierenden Issues aus der Validierung mit Labhart-Chronometrie sind in Tabelle 8 ersichtlich.

ID	Erfasser	Kategorie	Beschreibung	Lösung
#3	@labhart	Change Request	Kundennummer anstatt Hochzeitsdatum in Suchvorschau verwenden.	Das Hochzeitsdatum in der Suchvorschau wurde mit der Kundennummer ersetzt. Dies dient der rascheren Identifikation von Kunden anhand derer Kundennummer. Des weiteren wurde die Kundennummer in allen relevanten Prozessschritten hinzugefügt.
#4	@labhart	Feature Request	Bei der Erfassung der Kundendaten muss die angegebene E-Mail-Adresse und Telefonnummer einem der beiden Kunden zugewiesen werden können.	Die Mockups wurden so angepasst, dass bei der Erfassung von E-Mail-Adresse und Telefonnummer jeweils angegeben werden muss, zu wem die Information gehört.
#5	@labhart	Feature Request	Darstellung der Kundennummer in allen Prozessschritten.	Mit Bearbeitung von #3 wurde dieser Anfrage bereits nachgekommen.
#6	@labhart	Change Request	Abrechnungsdarstellung optimieren, damit klarer wird, welche Beträge gutscheinberechtigt sind.	Die Einkaufssumme wurde unterteilt in «Gutscheinberechtigte Einkaufssumme» und in den «gutscheineinlösenden Einkauf». Die erneute Validierung hat gezeigt, dass die neue Lösung sehr gut verständlich ist.
#7	@labhart	Feature Request	Wenn (fälschlicherweise) der abschliessende Einkauf nicht als solcher verbucht wird, sollte anschliessend daraus dennoch der gutscheineinlösende Einkauf gemacht werden können.	Wenn die Maske «Gutschein einlösen» geöffnet wird, nachdem bereits ein Einkauf in diesem Geschäft verbucht wurde, erscheint in dieser Maske der entsprechende Einkauf bereits vorausgefüllt. Somit bleibt die Einlösefunktionalität gewährleistet, ohne dass Anforderungen verletzt werden können.
#8	@labhart	Change Request	In der Detailansicht eines Kunden sollen die Einkäufe vor den	Wurde wie gewünscht angepasst und ist aus Usability Sicht in

			persönlichen Daten aufgelistet werden.	Ordnung, da der Name der Kunden noch immer als erstes im Titel ersichtlich bleibt.
#9	@labhart	Change Request	Bei der Detailansicht eines Gutscheins soll «Gutscheinberechtigtes Einkaufstotal» stehen anstelle von «Einkaufstotal».	Wurde wie gewünscht angepasst.

Tabelle 8: Issues nach Validierung durch Labhart-Chronometrie

Im Sinne einer iterativen Vorgehensweise wurden die aufgenommenen Änderungs- und Featurewünsche solange überarbeitet, bis sie von der Firma Labhart-Chronometrie als vollumfänglich geeignet abgenommen wurden.

7.1.3 Validierung durch übrige Teilnehmerfirmen

Die überarbeiteten Mockups werden mit den restlichen teilnehmenden Geschäften ebenfalls auf deren Anforderungen überprüft. Dabei wird sichergestellt, dass bei allen Beteiligten in jedem Prozessschritt sämtliche benötigten Informationen und Interaktionsmöglichkeiten vorhanden sind und die Elemente für die Benutzer intuitiv verständlich sind. Die resultierenden Issues aus der Validation mit den Teilnehmerfirmen Liluca, nisago, LightPlay und Blumeria Marbach sind in Tabelle 9 ersichtlich.

ID	Erfasser	Kategorie	Beschreibung	Lösung
#11	@light-play	Feature Request	Gemeinsamer zukünftiger Nachname sollte erfasst werden können.	In der Maske zur Erfassung der Kontaktangaben wird ein optionales Feld hinzugefügt, worin der gemeinsame Nachname nach der Hochzeit erfasst werden kann.
#12	@light-play	Change Request	Wording «Kundennummer» ändern, da dieser Begriff in der Regel mit der firmeninternen Kundennummer assoziiert wird.	Das Wort «Kundennummer» wird an sämtlichen Stellen ersetzt durch «Gutscheinnummer».
#13	@light-play	Change Request	In der Abrechnung sollen die Beteiligungskosten auch als Prozentzahl ersichtlich sein.	In der aktuellen Abrechnungsperiode wird der zu zahlende Betrag zusätzlich prozentual als

				Gutscheinabgaben gekennzeichnet.
#14	@light-play	Feature Request	Im Sinne eines papierlosen Prozesses soll es möglich sein, die Zwischen- und Abschlussbestätigungen per Mail an die Kunden zu schicken.	An den entsprechenden Stellen wird ein Button integriert, der auf Klick die jeweilige Bestätigung an die hinterlegte(n) E-Mail-Adresse(n) verschickt. Die Funktion PDF-Download wird nach wie vor angeboten.
#15	@nisago	Feature Request	Der Herrenausstatter nisago bedient ausschliesslich männliche Kundschaft. Entsprechend ist es für das Unternehmen wichtig, die Kontaktdaten des Herrn erfassen zu können, auch wenn bereits jene seiner Partnerin hinterlegt sind.	Das System bietet die Möglichkeit, optional zusätzliche Kontaktangaben zu erfassen.
#16	@liluca	Feature Request	Liluca hat sowohl eine Damen- als auch Herrenabteilung. Entsprechend sollte es ihnen erlaubt sein, zwei Einträge pro Hochzeitspaar zu buchen.	Beim Verbuchen des Einkaufs steht Liluca ein Dropdown zur Verfügung, bei dem sie die entsprechende Abteilung auswählen können.
#17	@blumeria	Feature Request	Die internen Prozesse der Blumeria Marbach verlangen die Möglichkeit einer nachträglichen Anpassung der Beträge, da sich diese in der Regel im Verlauf der Geschäftsbeziehung noch ändern.	So lange der Gutschein noch nicht eingelöst wurde haben die Geschäfte die Möglichkeit, die eigenen Umsätze nachträglich zu bearbeiten.

Tabelle 9: Issues nach Validierung durch Liluca, nisago, lightplay und Blumeria Marbach

Die aufgenommenen Änderungs- und Featurewünsche wurden bearbeitet und anschliessend erneut mit den beteiligten Geschäften validiert. In der zweiten Validierungsrunde wurden die überarbeiteten Mockups als vollumfänglich geeignet abgenommen.

7.2 Validierung REST-API

Dieses Kapitel erläutert die Validierung der REST-API. Dazu werden bei der Entwicklung laufend Smoke-Tests durchgeführt und der gesamte Code ist mit Unit-Tests abgedeckt. Dadurch wird sichergestellt, dass die bereitgestellte Lösung die Anforderungen erfüllt und das identifizierte Problem beheben kann.

7.2.1 Smoke-Tests

Bei der Entwicklung der REST-API werden sämtliche Funktionalitäten durch den Entwickler laufend manuell getestet. Dabei wird überprüft, ob die entwickelten Programmteile so funktionieren wie erwartet. Im Falle eines Fehlers, respektive eines unerwarteten Verhaltens wird erst mit der Entwicklung fortgefahren, wenn die festgestellte Unregelmässigkeit behoben werden konnte. Microsoft unterstreicht, dass Smoke-Tests eine der effizientesten und im Verhältnis kostengünstigsten Methoden sind, um Fehler in einer Software frühzeitig zu erkennen und zu beheben (o. J.-b).

7.2.2 Unit-Tests

Mit Unit-Tests werden einzelne Teile aus der Applikationslogik isoliert voneinander getestet, damit deren korrekte Funktionalität gewährleistet werden kann. Weiter werden mit Unit-Tests Fehlerquellen frühzeitig erkannt und verhindert. Aufgrund der Tatsache, dass alle geschriebenen Funktionen mit Tests validiert werden, wird bereits bei der initialen Umsetzung Wert auf eine besonders saubere Struktur gelegt, da damit das anschliessende Testing vereinfacht werden kann.

Das Durchlaufen der Tests wird mit einem Shell-Script aufgerufen, welches bei der Integration von Continuous Development fortan ebenfalls ausgeführt wird. So wird eine langfristig hohe Code-Qualität gewährleistet, da bei jeder Änderung sämtliche Unit-Tests durchlaufen werden.

Die REST-API deckt 100 Prozent von dem gesamten Code ab, womit jede mögliche Ausprägung im Code getestet ist. Insgesamt werden 1754 Statements vollständig getestet.

Der Coverage-Bericht ist der Abbildung 36 zu entnehmen.

7.2.3 Akzeptanztests

Zur Sicherstellung, dass die erstellte Applikation auch aus Sicht der zukünftigen Benutzer einwandfrei funktioniert, sind Akzeptanztests durchgeführt worden. Mit Hilfe von Akzeptanztests wird sichergestellt, dass die Software so funktioniert, wie es von der Anwendergruppe erwartet wird (Wirtz, o. J.). Der Akzeptanztest mit der Lead-Firma Labhart-Chronometrie hat bewiesen, dass die erstellte Software einwandfrei funktioniert und die Bedürfnisse der Anwender befriedigt.

7.2.4 Frontend Umsetzung

Durch die Realisierung von dem kompletten Frontend für die Applikation wird eine weitere erfolgreiche Validierung gewährleistet. Mit der Umsetzung wird bewiesen, dass die bereitgestellte API praxistauglich anwendbar ist und technisch einwandfrei funktioniert.

8 Weitergehende Anwendungsmöglichkeiten der Lösung

In diesem Kapitel werden weitergehende Anwendungsmöglichkeiten der umgesetzten Lösung diskutiert.

8.1 Applikationsspezifische Hardware

Der gewählte Ansatz mit der unabhängigen REST-API erlaubt es, beliebigen anderen Clients – über das Internet – Zugriff auf die Kernfunktionalität der Software zu geben, womit sich die Möglichkeit aufdrängt, weitere Clients mit der Businesslogik zu verbinden.

8.1.1 Raspberry Pi

Die Raspberry Pi Foundation stellt kostengünstige Einplatinencomputer im Kleinformat her, die in Relation zu ihrer Dimension performante Leistungen erbringen (Raspberry Pi Foundation, o. J.).

Die Entwicklung einer, für Touchscreen optimierten, nativen Applikation für ein Raspberry Pi wäre eine spannende Möglichkeit, um die wichtigsten Funktionen der Software ohne einen gewöhnlichen Computer bedienen zu können. Dadurch entfällt die Notwendigkeit, dass die teilnehmenden Geschäfte ein gewöhnliches Notebook an der Verkaufstheke griffbereit haben. Nebst einer Kamera wird ein kleiner Touchscreen an die Raspberry Pi Platine angehängt und somit die Möglichkeit geboten, Informationen visuell einlesen zu können (z.B. mittels QR- oder Barcode) oder mittels Fingertipp eingegeben zu können. Beim Verbuchen eines Einkaufs kann der Mitarbeiter entweder den QR- oder Barcode der Kunden einscannen, oder die 8-stellige Zahlennummer über den Touchscreen eintragen. Anschliessend muss nur noch der Einkaufsbetrag eingegeben werden und der Prozess ist abgeschlossen.

8.1.2 Hybrid Smartphone App

Die im Abschnitt 8.1.1 vorgestellte Möglichkeit lässt sich auch auf eine Smartphone-Applikation adaptieren. Dank der bestehenden Frontendumsetzung mit Angular kann mit einem verhältnismässig geringen Aufwand eine Smartphone-App mit dem Framework Ionic erstellt werden. Das Ionic Framework ermöglicht es, mit Webtechnologien sogenannte Hybrid-Apps zu erstellen (Ionicframework, o. J.). Mit Ionic kann die Codebasis

von dem bestehenden Frontend zur Entwicklung der Mobile-App verwendet werden, wobei Android und iOS gleichzeitig und standardmässig unterstützt werden. Für den Anwendungsfall der Hochzeitdienstleister eignet sich eine Smartphone-App gleich gut wie die Variante mit dem Raspberry Pi, ist jedoch hinsichtlich der Hardwareanschaffungen kostenintensiver.

8.2 Anbindung an Drittsysteme

Anstelle der in Abschnitt 8.1 beschriebenen Anwendungsmöglichkeiten lässt sich die erstellte REST-API auch mit bereits bestehenden Umsystemen verknüpfen. Diese Möglichkeiten werden in den beiden folgenden Unterpunkten diskutiert.

8.2.1 Kassensysteme

Mit der neu umgesetzten Softwarelösung sieht der interne Prozessablauf vor, dass nebst der Einkaufverbuchung im firmeninternen Kassensystem auch noch eine Buchung im Gutscheinsystem vorgenommen wird. Diese Prozessschritte beanspruchen bereits wenig Zeit, liessen sich aber mit einer direkten Implementierung der Gutscheinverwaltung in das jeweilige Kassensystem noch einmal verkürzen.

Die Voraussetzungen dazu sind lediglich, dass die verschiedenen Kassensysteme die Möglichkeit bieten, nach dem Speichern eines Einkaufs HTTP-Requests auf die API machen zu können und dabei die Einkaufsdetails mitschicken.

8.2.2 E-Commerce

Die Einsatzmöglichkeiten der REST-API sind nahezu unbegrenzt und lassen sich daher auch optimal in bestehende E-Commerce-Lösungen einbauen. Im Gegensatz zu den oben beschriebenen Kassensystemen kann bei Online-Shops davon ausgegangen werden, dass sie mit HTTP-Requests auf die REST-API der Gutscheinverwaltung keine Schwierigkeiten haben. Bei einer entsprechenden Implementierung profitieren Hochzeitspaare von der Gutscheinaktion auch bei Online-Einkäufen.

8.3 Retargeting

Die geschaffene Softwarelösung bildet für sämtliche gutscheinbezogenen Daten die sogenannte Single Source Of Truth, was bedeutet, dass sich alle Daten an einem zentralen

Ort befinden. Diese Informationen bilden eine hervorragende Möglichkeit, um zielgruppenspezifische Newsletter-Kampagnen zu erstellen, so wie vor allem auch gezielte Retargeting-Massnahmen einzurichten. Denkbar sind automatisierte E-Mail-Benachrichtigungen kurz vor dem Verfall eines noch nicht eingelösten Gutscheins. Ebenso naheliegend ist die Gratulation zum ersten Hochzeitstag, wobei im Optimalfall ein personalisierter Gutschein für Hochzeitstag-Geschenke mitgeschickt wird.

8.4 Branchenunabhängige Lösung

Das Konzept der verteilten Gutscheinverwaltung funktioniert nicht ausschliesslich für Hochzeitsdienstleister, sondern lässt sich auch auf verschiedene andere Branchen übertragen. Der gesamte Programmcode ist so aufgebaut, dass keine Businesslogik angepasst werden muss, sondern lediglich einige Änderungen in den Beschreibungstexten gemacht werden müssten, sowie das Logo ausgetauscht werden sollte. Anschliessend kann mit geringem Aufwand eine neue Instanz des vorliegenden Softwarestacks auf einen Server mit Docker-Host deployt werden.

9 Schlussfolgerungen

Das Ziel der vorliegenden Arbeit war es, die mit dem Gutscheineheft zusammenhängenden manuellen Prozesse so weit als möglich zu reduzieren und dabei zeitgleich die notwendige Transparenz und Sicherheit zu schaffen, damit unter den teilnehmenden Geschäften Klarheit und Vertrauen herrscht.

Die zu Beginn dieser Arbeit identifizierten Probleme können mit den produzierten Artefakten vollumfänglich gelöst werden. Als Output dieser Arbeit resultieren:

- Frontend Design in Form von Mockups
- Validierung Frontend Design
- Design einer REST-API
- Validierung der REST-API
- Frontend Umsetzung
- DevOps / Server Deployment

Zusammenfassend lässt sich festhalten, dass das eingesetzte Konzept bestens geeignet ist. Die Umsetzung als Single Page Application mit der REST-API als Datenquelle besticht durch sehr schnelle Ladezeiten und ein ansprechendes User Interface. Weiter besteht mit der REST-API die Möglichkeit, zu einem späteren Zeitpunkt weitere Applikationen mit der bestehenden Businesslogik zu verbinden.

Die Frage, wie sich die Applikation im Langzeittest bewähren wird, kann zum jetzigen Zeitpunkt noch nicht beantwortet werden, da der produktive Einsatz erst nach der abgeschlossenen Bewertung startet. Die Resultate der gemachten Validierungen lassen aber darauf schliessen, dass die Applikation die Bedürfnisse aller Benutzer befriedigt und sich die Applikation langfristig bewähren wird.

10 Literaturverzeichnis

- Angular. (o. J.-a). Architecture Overview. *Architecture Overview*. Verfügbar unter: <https://angular.io/guide/architecture>
- Angular. (o. J.-b). Component. Zugriff am 20.5.2018. Verfügbar unter: <https://angular.io/api/core/Component>
- Angular. (o. J.-c). Introduction to services and dependency injection. Zugriff am 20.5.2018. Verfügbar unter: <https://angular.io/guide/architecture-services>
- Angular. (o. J.-d). Routing & Navigation. Zugriff am 20.5.2018. Verfügbar unter: <https://angular.io/guide/router>
- Arndt, C., Hermanns, C., Kuchen, H. & Poldner, M. (2009). *Best Practices in der Softwareentwicklung*. Westfälische Willhelms-Universität Münster.
- Berenbrink, V., Purucker, J. & Bahlinger, T. (2013). Die Bedeutung von Wireframes in der agilen Softwareentwicklung. *HMD Praxis der Wirtschaftsinformatik*, 50 (2), 27–34. doi:10.1007/BF03340793
- Biethahn, J., Mucksch, H. & Ruf, W. (2004). *Grundlagen (Ganzheitliches Informationsmanagement)* (6., vollst. überarb. und neu gefasste Aufl.). München: Oldenbourg.
- Cito, J. & Gall, H. C. (2016). Using docker containers to improve reproducibility in software engineering research (S. 906–907). ACM Press. doi:10.1145/2889160.2891057
- Eugeniya, K. (o. J.). ReactJS vs Angular Comparison: Which is Better? Verfügbar unter: <https://hackernoon.com/reactjs-vs-angular-comparison-which-is-better-805c0b8091b1>
- Fluin, S. (2017, Dezember 25). Why Developers and Companies Choose Angular. Verfügbar unter: <https://medium.com/angular-japan-user-group/why-developers-and-companies-choose-angular-4c9ba6098e1c>
- Franz, D. & Mattes, R. (1991). Datenerfassung. *Elektronische Datenverarbeitung* (S. 26–32). Wiesbaden: Gabler Verlag. doi:10.1007/978-3-322-82836-1_4
- Gartner. (o. J.). DevOps. Zugriff am 16.5.2018. Verfügbar unter: <https://www.gartner.com/it-glossary/devops>

- GitLab. (o. J.-a). GitLab Continuous Integration (GitLab CI/CD). *GitLab Continuous Integration (GitLab CI/CD)*. Zugriff am 10.4.2018. Verfügbar unter: <https://docs.gitlab.com/ee/ci/>
- GitLab. (o. J.-b). Getting started with GitLab CI/CD. Zugriff am 10.4.2018. Verfügbar unter: https://docs.gitlab.com/ee/ci/quick_start/README.html
- GitLab. (o. J.-c). Configuring GitLab Runners. Verfügbar unter: <https://docs.gitlab.com/ee/ci/runners/README.html>
- Hansen, S. (2017, Mai 23). Advantages and Disadvantages of Django. *Hackernoon*. Verfügbar unter: <https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5>
- Helmich, M. (2013). RESTful Webservices (1): Was ist das überhaupt? Verfügbar unter: <https://www.mittwald.de/blog/webentwicklung-design/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt>
- Hunter II, T. (2017). HTTP API Design. *Advanced Microservices* (S. 13–54). Berkeley, CA: Apress. doi:10.1007/978-1-4842-2887-6_2
- Ionicframework. (o. J.). Hybrid vs. Native. Verfügbar unter: <https://ionicframework.com/books/hybrid-vs-native/preview?submissionGuid=9c4c89d1-1187-41e1-b7e9-c9fc863119f3>
- JAX Editorial Team. (2018, Mai 4). On the road to Angular v6: Pop the champagne, it's here! *JAX Enter*. Verfügbar unter: <https://jaxenter.com/road-to-angular-6-139479.html>
- Microsoft. (o. J.-a). Interfaces. *TypeScript*. Zugriff am 20.5.2018. Verfügbar unter: <https://www.typescriptlang.org/docs/handbook/interfaces.html>
- Microsoft. (o. J.-b). Guidelines for Smoke Testing. Verfügbar unter: [https://msdn.microsoft.com/en-us/library/ms182613\(VS.80\).aspx](https://msdn.microsoft.com/en-us/library/ms182613(VS.80).aspx)
- NGINX. (2018, Mai 19). Frequently Asked Questions. Verfügbar unter: <https://www.nginx.com/faqs/>
- Noyes, K. (2013, August 1). Docker: A «Shipping Container» for Linux Code. Zugriff am 9.4.2018. Verfügbar unter: <https://www.linux.com/news/docker-shipping-container-linux-code>

- OpenAPI Initiative. (o. J.). About OpenAPI Initiative. *About OpenAPI Initiative*. Verfügbar unter: <https://www.openapis.org/about>
- Pittet, S. (o. J.). Continuous integration vs. continuous delivery vs. continuous deployment. Zugriff am 10.4.2018. Verfügbar unter: <https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd>
- Raspberry Pi Foundation. (o. J.). About Us. Zugriff am 15.5.2018. Verfügbar unter: <https://www.raspberrypi.org/about/>
- React. (o. J.). ReactJS. Verfügbar unter: <https://reactjs.org/>
- Suresh, B. (2016, Dezember 14). Implement a single-page application with Angular 2. Verfügbar unter: <https://www.ibm.com/developerworks/library/wa-implement-a-single-page-application-with-angular2/index.html>
- W3 Techs. (2018, Mai 19). Usage of web servers broken down by ranking. Verfügbar unter: https://w3techs.com/technologies/cross/web_server/ranking
- Wedding Award Switzerland. (o. J.-a). Fakten. *Wedding Award Switzerland – Fakten*. Zugriff am 25.2.2018. Verfügbar unter: <http://weddingaward.org/fakten/kategorien/>
- Wedding Award Switzerland. (o. J.-b). Über den Award. *Wedding Award Switzerland – Über den Award*. Zugriff am 25.2.2018. Verfügbar unter: <http://weddingaward.org/home/wedding-award/>
- Wirtz, G. (o. J.). Enzyklopädie der Wirtschaftsinformatik Online-Lexikon. Zugriff am 21.5.2018. Verfügbar unter: <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Software-Implementierung/Testen-von-Software/Akzeptanztest>

11 Anhang

11.1 Projektplan

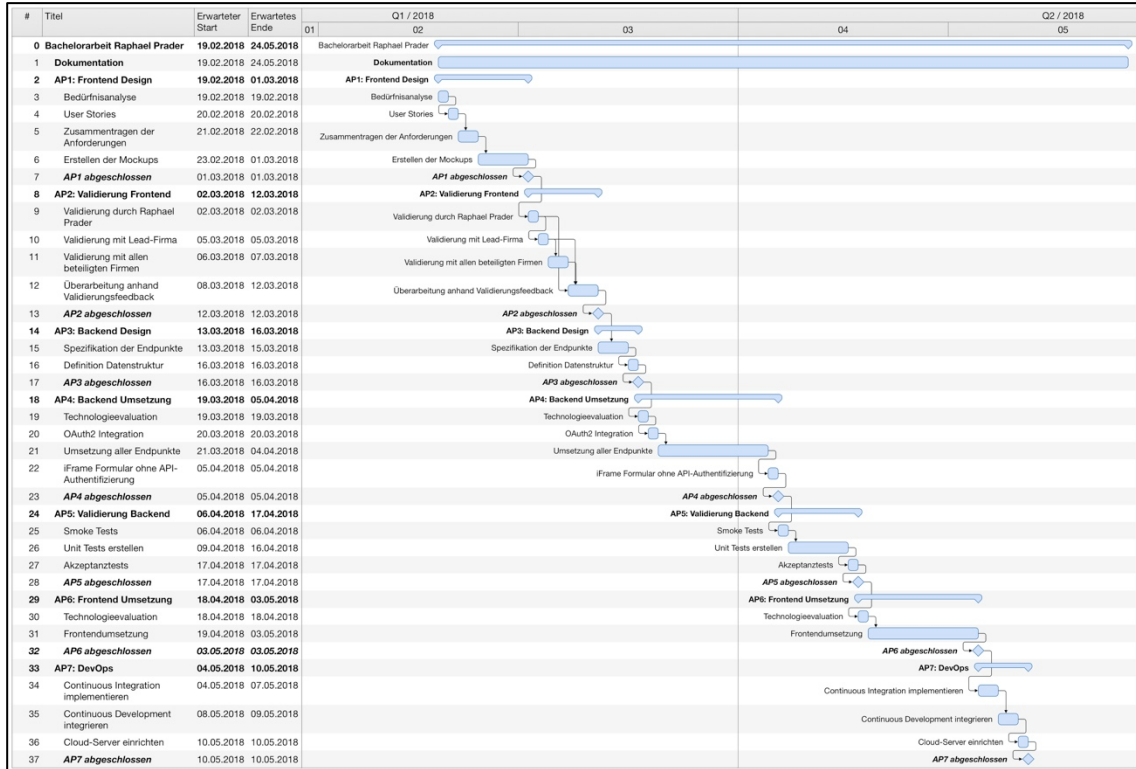


Abbildung 16: Detaillierter Projektplan

Aufgrund der begrenzten Platzverhältnisse ist der obige Projektplan zusätzlich im eingereichten ZIP-File angehängt.

11.2 Mockups

Geschäft-Login

liluca

Anmelden

Abbildung 17: Mockup von dem Screen «Login»

Hochzeitsdienstleister - Gutscheinverwaltung

Einkauf verbuchen

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et

Kunde Kunden suchen... **+ Neukunden erfassen**

Einkauf Kaufbetrag inkl. MwSt. CHF

Geschäft Geschäft: Liluca Brautmode

Datum: 18. Juni 2018, 16:30 Uhr

Formular zurücksetzen **Einkauf speichern**

Abmelden

Abbildung 18: Mockup von dem Screen «Einkauf verbuchen»

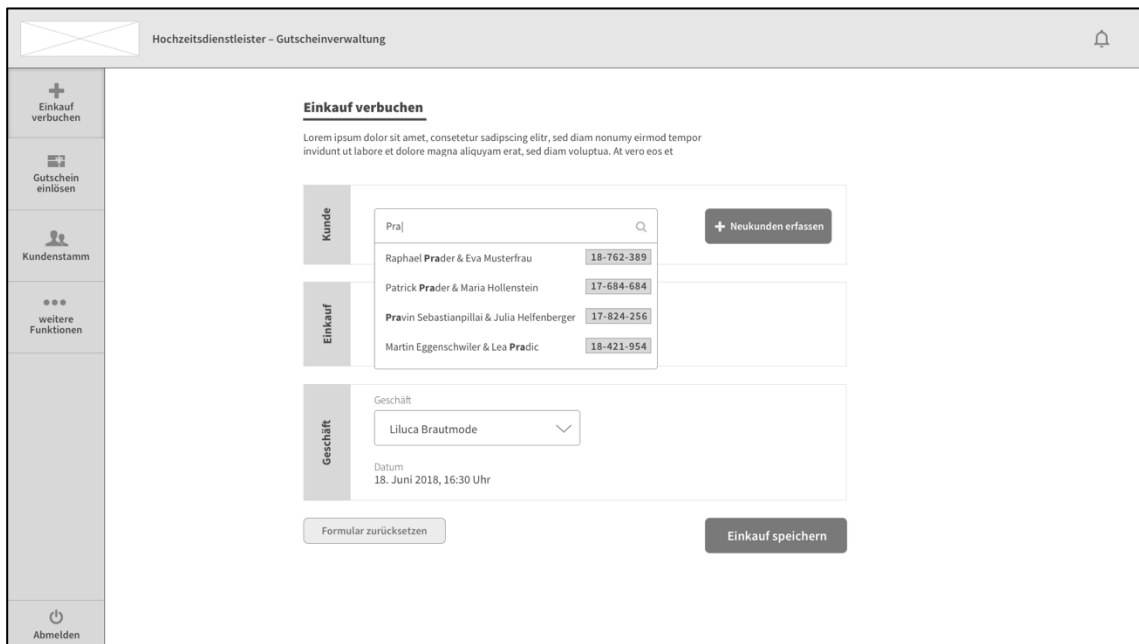


Abbildung 19: Mockup von dem Screen «Kunden suchen»

X

Hochzeitsdienstleister - Gutscheinverwaltung

🔔

+
Einkauf
verbuchen

👤
Gutschein
einlösen

👥
Kundenstamm

⋮
weitere
Funktionen

🔌
Abmelden

Einkauf verbuchen

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et

Kunde erfassen

Herr

Herr

Strasse, Hausnr.

Gemeinsam

Fr. Splichog
Hr. Wassermann

+ Weitere Kontaktdaten hinzufügen

Kunde suchen

Kunde speichern

Einkauf

Kaufbetrag inkl. MwSt.

CHF

Geschäft

Geschäft
Labhart Chronometrie

Datum
18. Juni 2018, 16:30 Uhr

Formular zurücksetzen

Einkauf speichern

Abbildung 20: Mockup von dem Screen «Kunden erfassen»

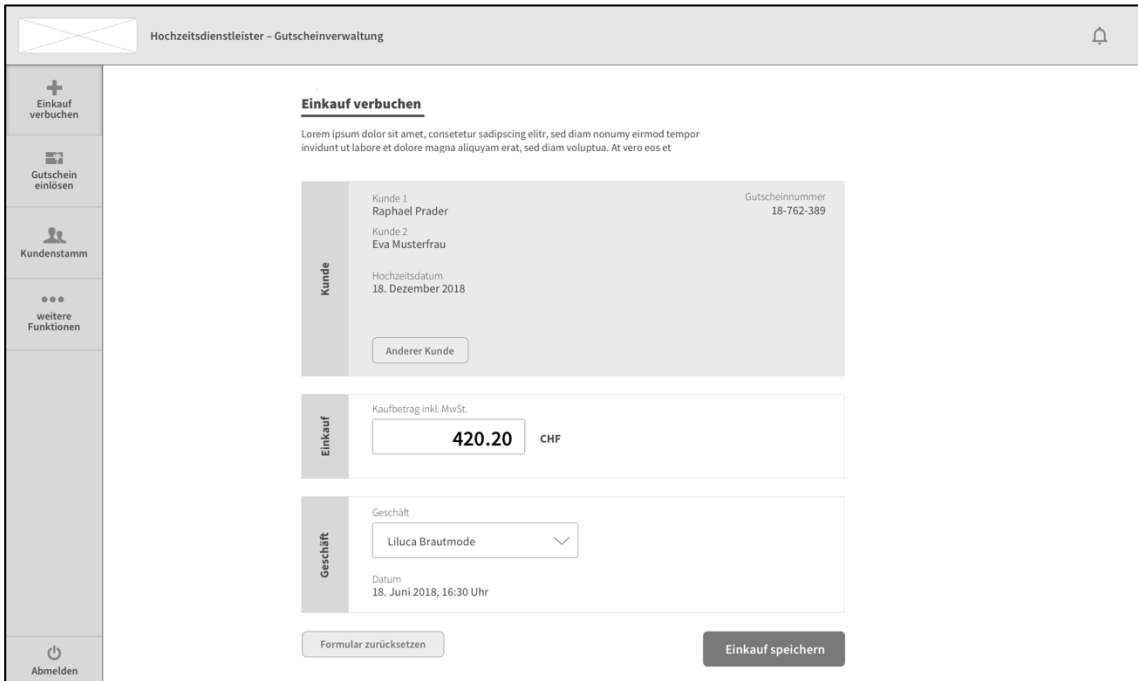


Abbildung 21: Mockup von dem Screen «Einkaufsdaten erfasst»

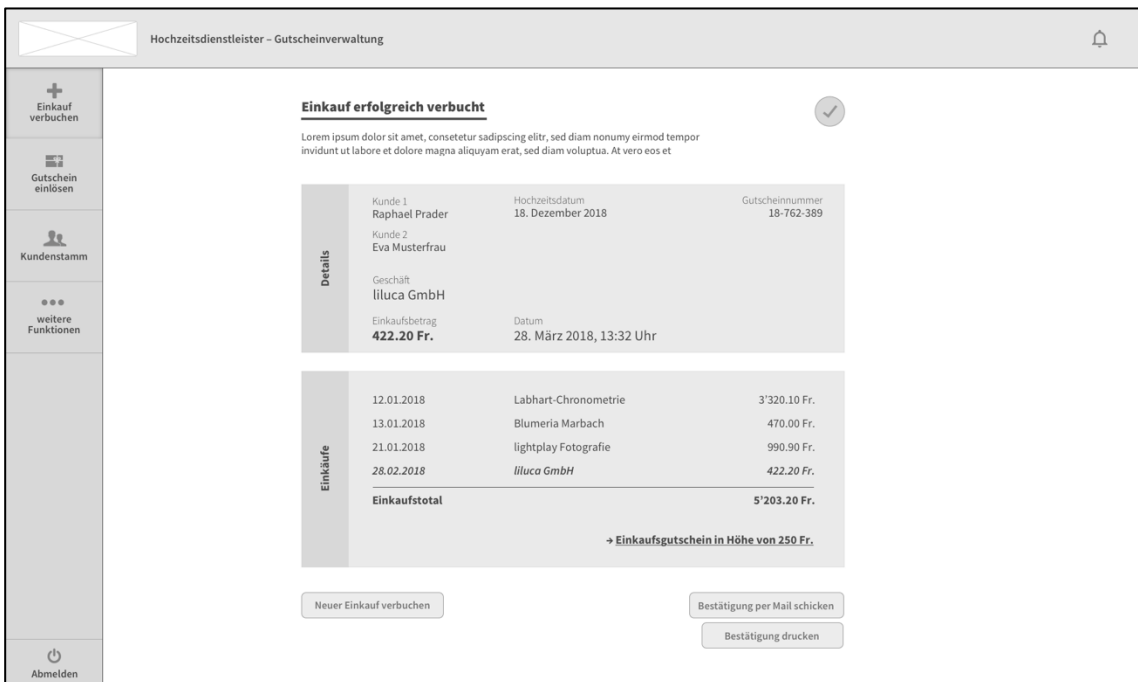


Abbildung 22: Mockup von dem Screen «Einkauf verbucht»

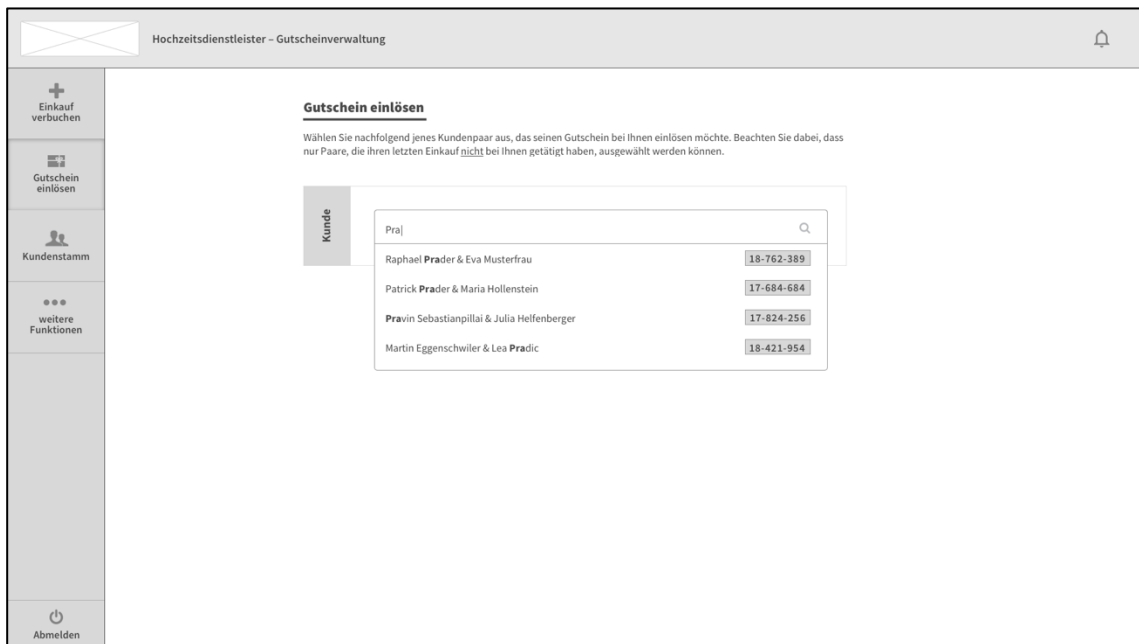


Abbildung 23: Mockup von dem Screen «Gutschein einlösen»

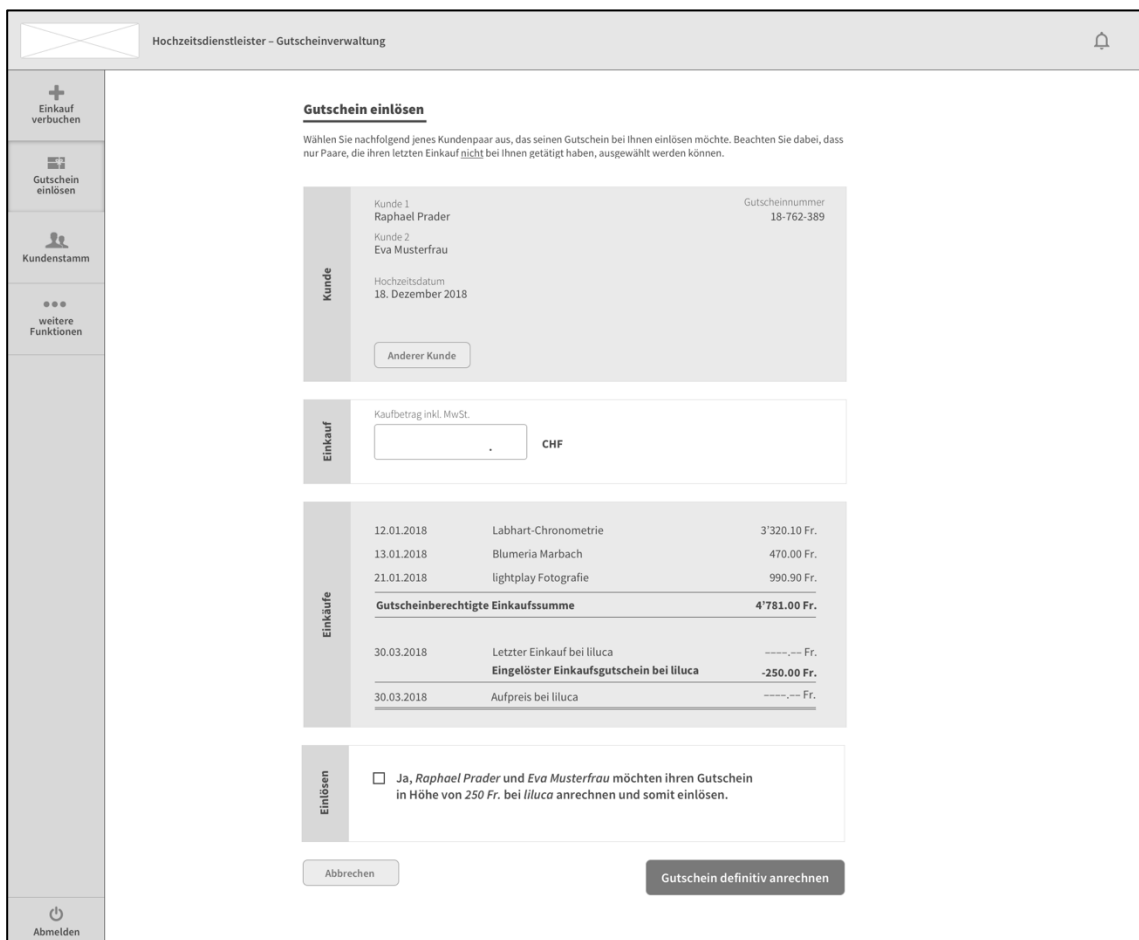


Abbildung 24: Mockup von dem Screen «Abschliessender Einkauf erfassen»

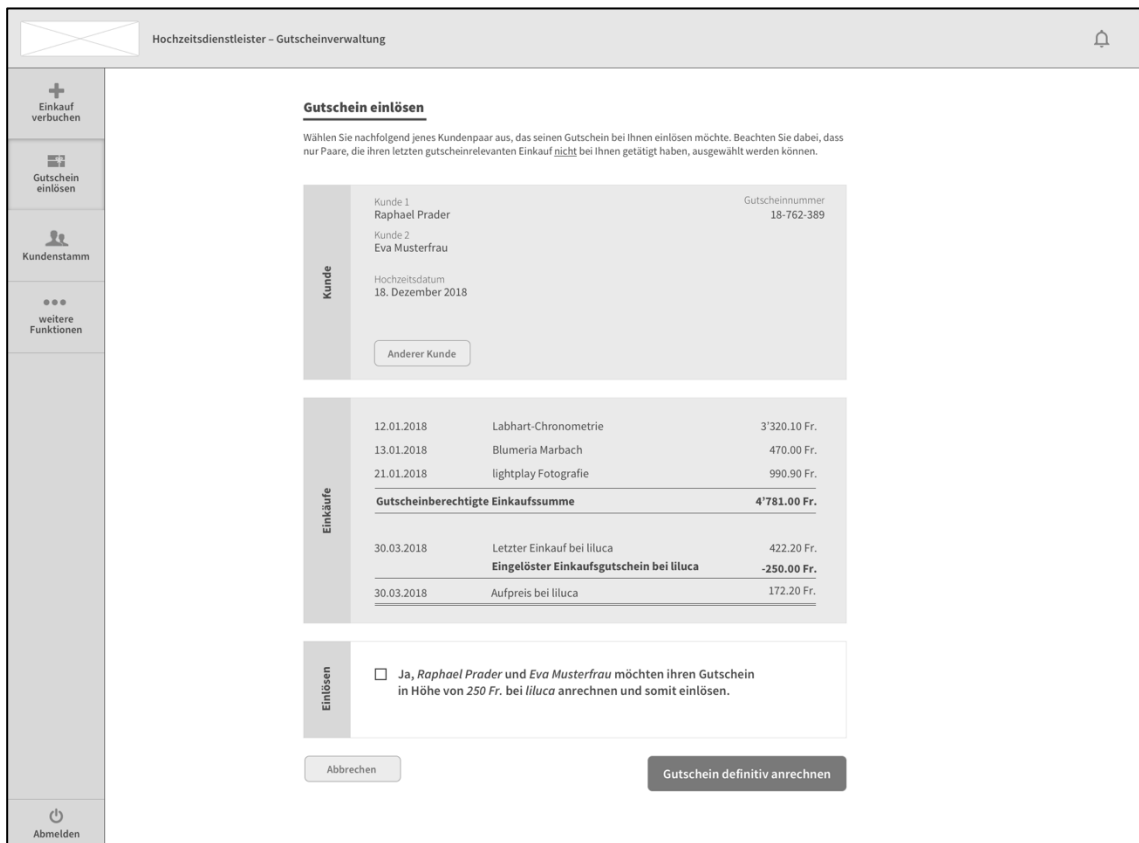


Abbildung 25: Mockup von dem Screen «Letzter Einkauf bereits erfasst»

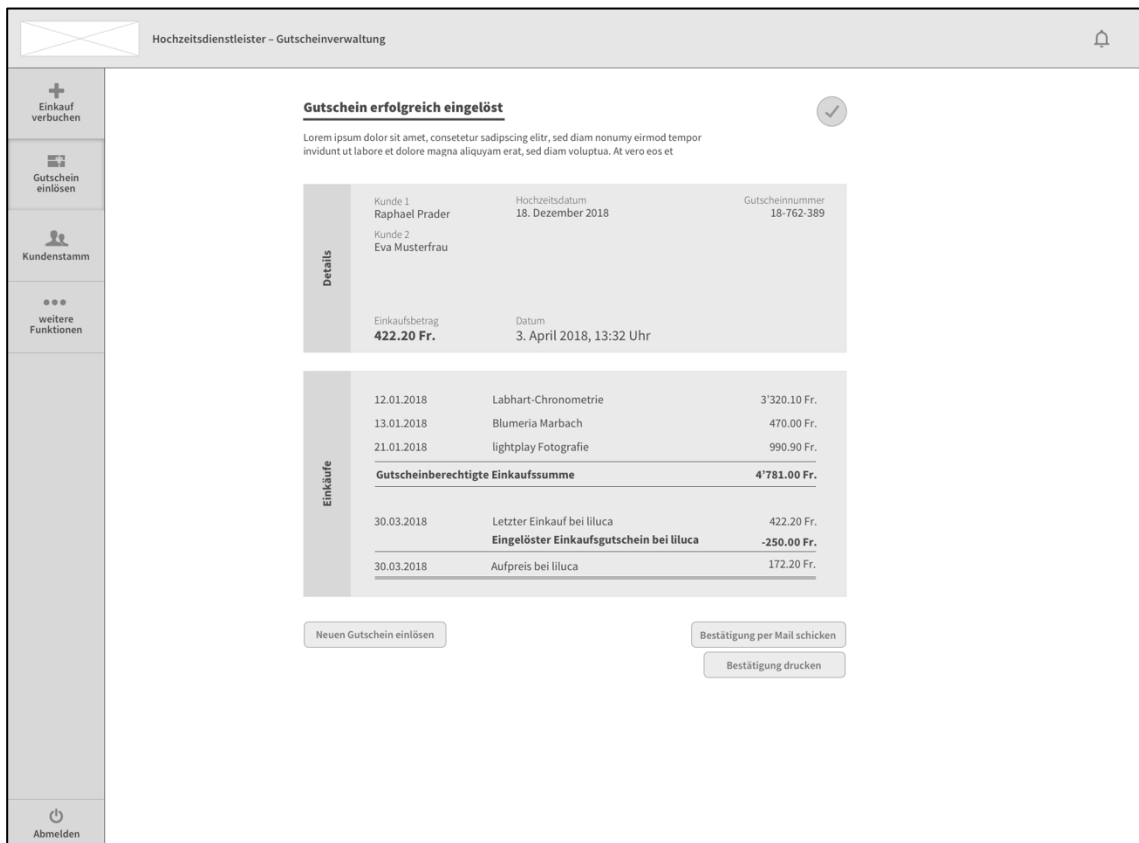


Abbildung 26: Mockup von dem Screen «Gutschein eingelöst»

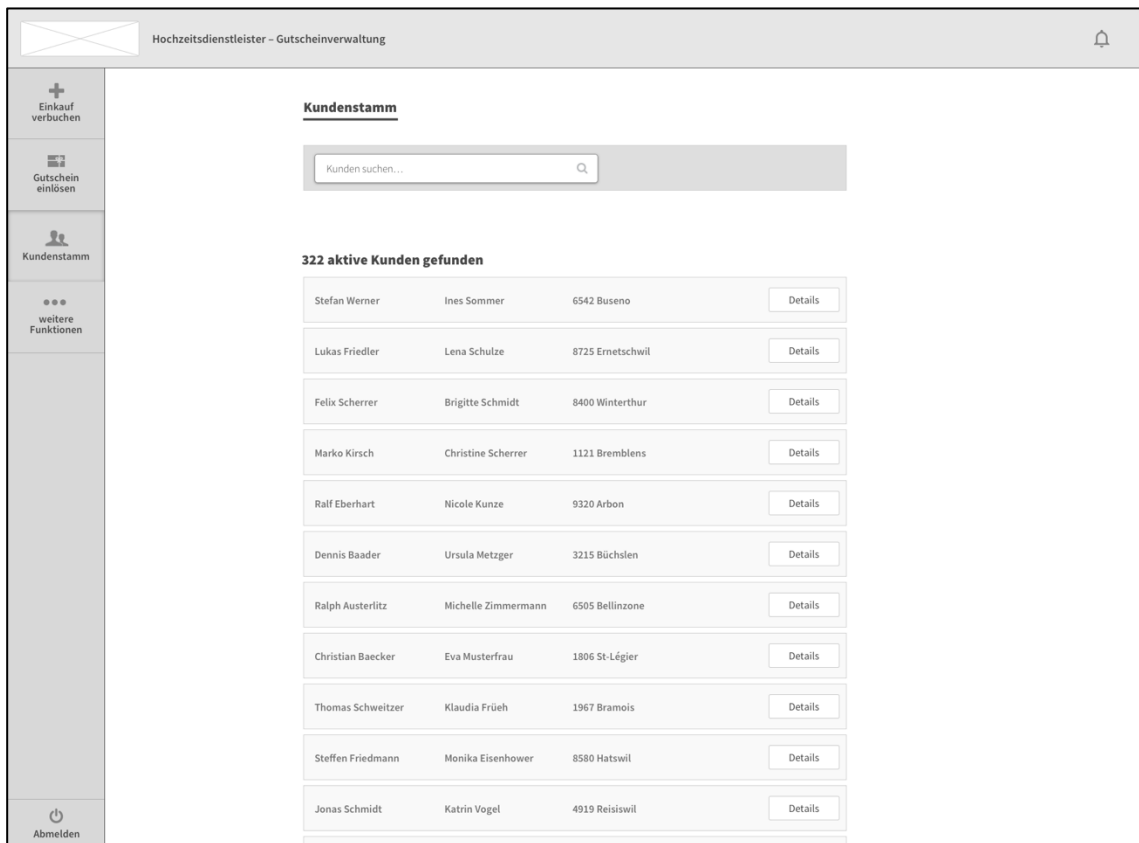


Abbildung 27: Mockup von dem Screen «Kundenstamm»

Hochzeitsdienstleister - Gutscheinverwaltung

Einkauf
verbuchen

Gutschein
einlösen

Kundenstamm

weitere
Funktionen

Abmelden

Raphael Prader + Eva Musterfrau, #18-762-389

- Zurück zur Übersicht

Einkäufe	12.01.2018	Labhart-Chronometrie	3'320.10 Fr.
	13.01.2018	Blumeria Marbach	470.00 Fr.
	21.01.2018	lightplay Fotografie	990.90 Fr.
	28.02.2018	liluca GmbH	422.20 Fr.
	Einkaufstotal		5'203.20 Fr.

→ Einkaufsgutschein in Höhe von 250 Fr.

Kontaktangaben

Herr
 Frau

Vorname
Karin

Nachname
Vogel

Herr
 Frau

Vorname
Jonas

Nachname
Schmidt

Gemeinsame Nachname
Schmidt-Vogel

Gemeinsam

Strasse
Pflanzschulstrasse 37

Fr. Vogel

PLZ
8400

Ort
Winterthur

[+ Weitere Adresse hinzufügen](#)

Gemeinsam

E-Mail-Adresse
ka

Fr. Vogel

Telefon
+41

[+ Weitere Kontaktdaten hinzufügen](#)

Wahltermin
18.12.2018

Änderungen speichern

- Zurück zur Übersicht

Bestätigung drucken

Abbildung 28: Mockup von dem Screen «Kunden bearbeiten»

74

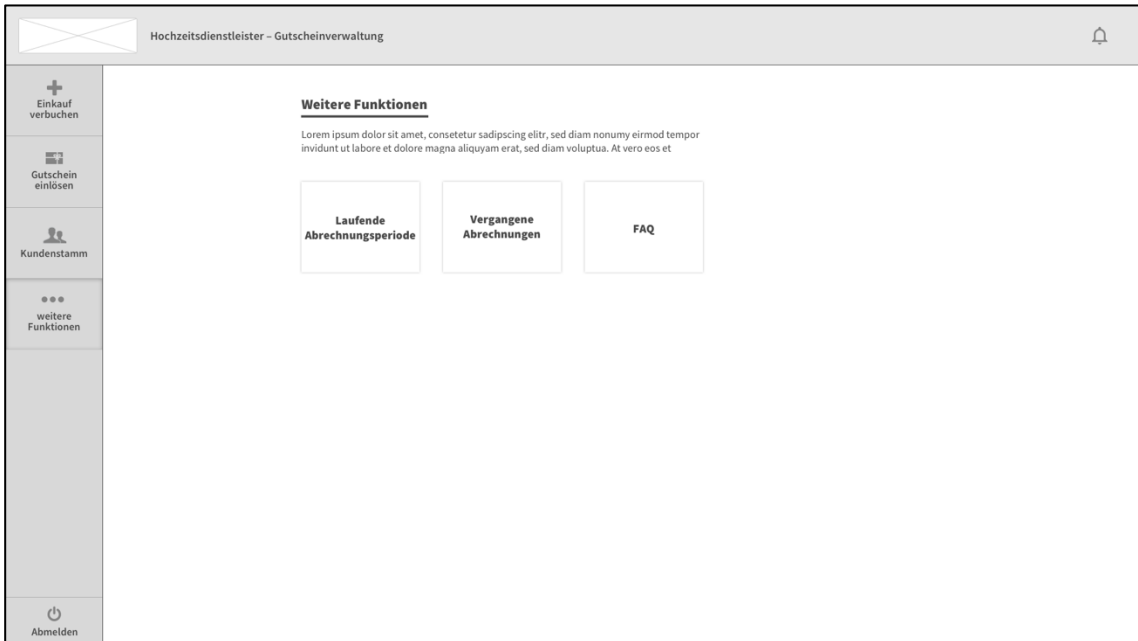


Abbildung 29: Mockup von dem Screen «Weitere Funktionen»

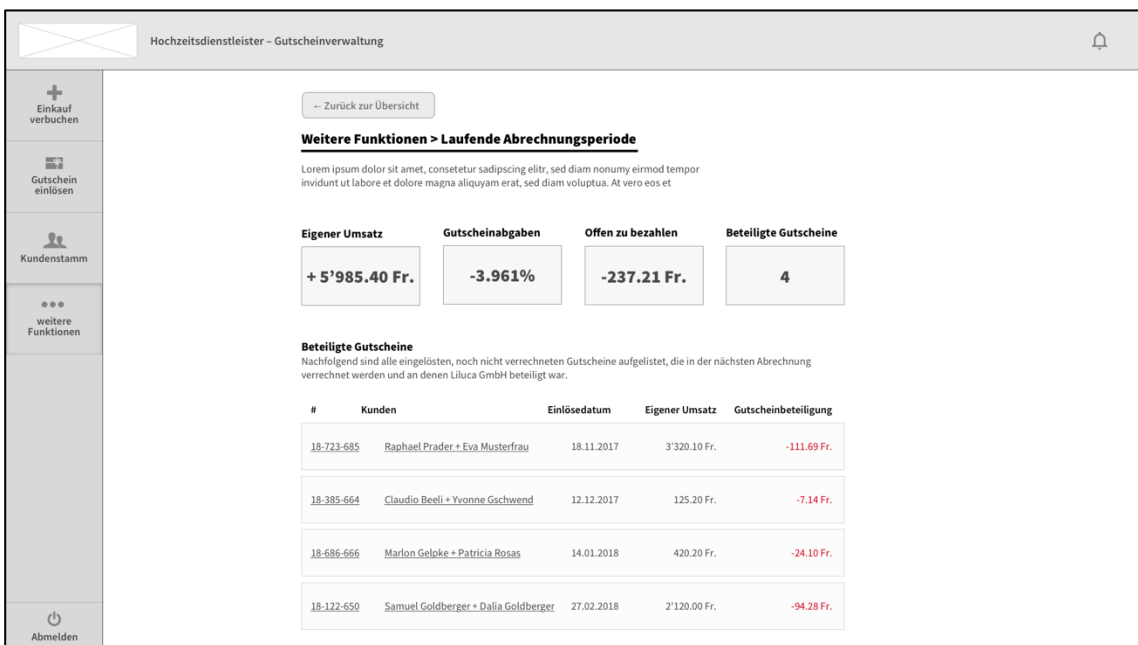


Abbildung 30: Mockup von dem Screen «Laufende Abrechnungsperiode»

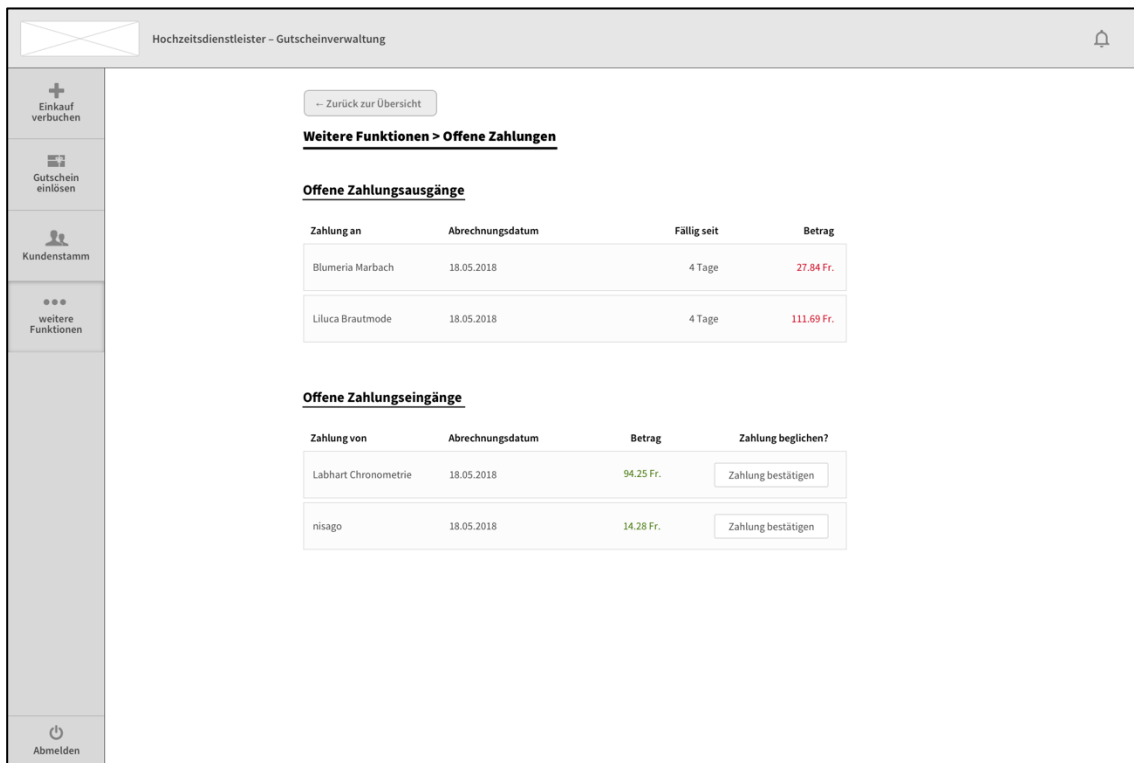


Abbildung 31: Mockup von dem Screen «Offene Zahlungen»

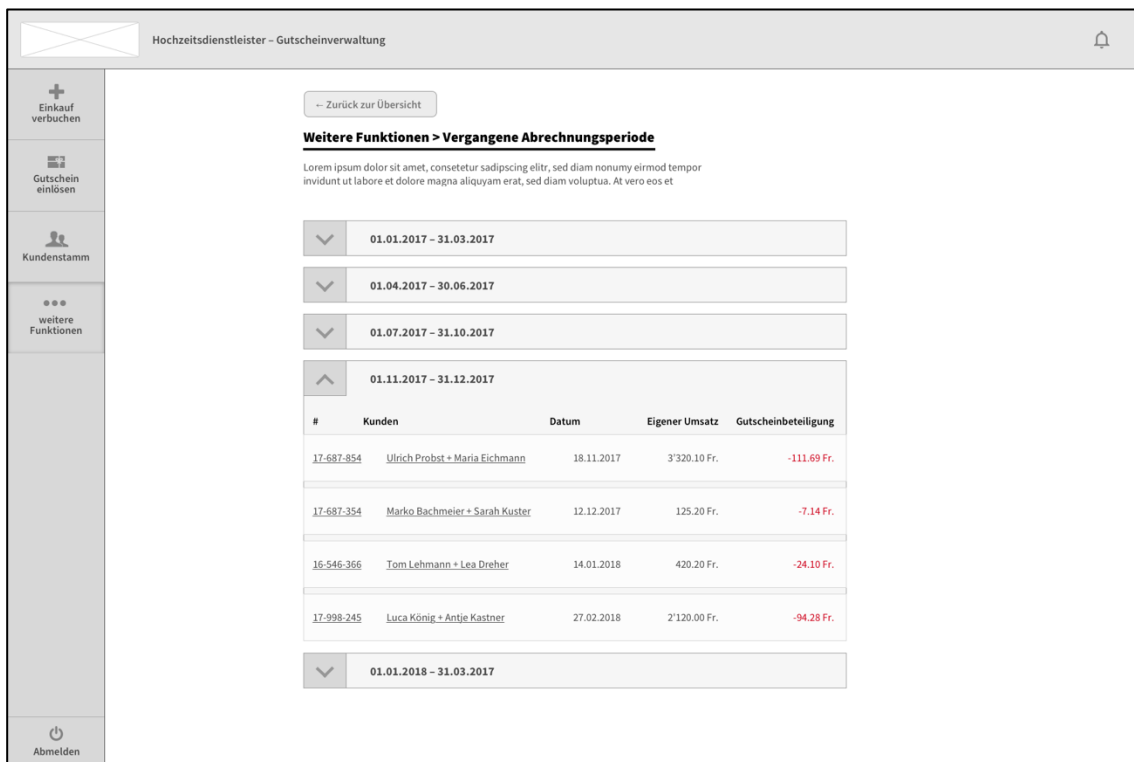


Abbildung 32: Mockup von dem Screen «Vergangene Abrechnungsperioden»

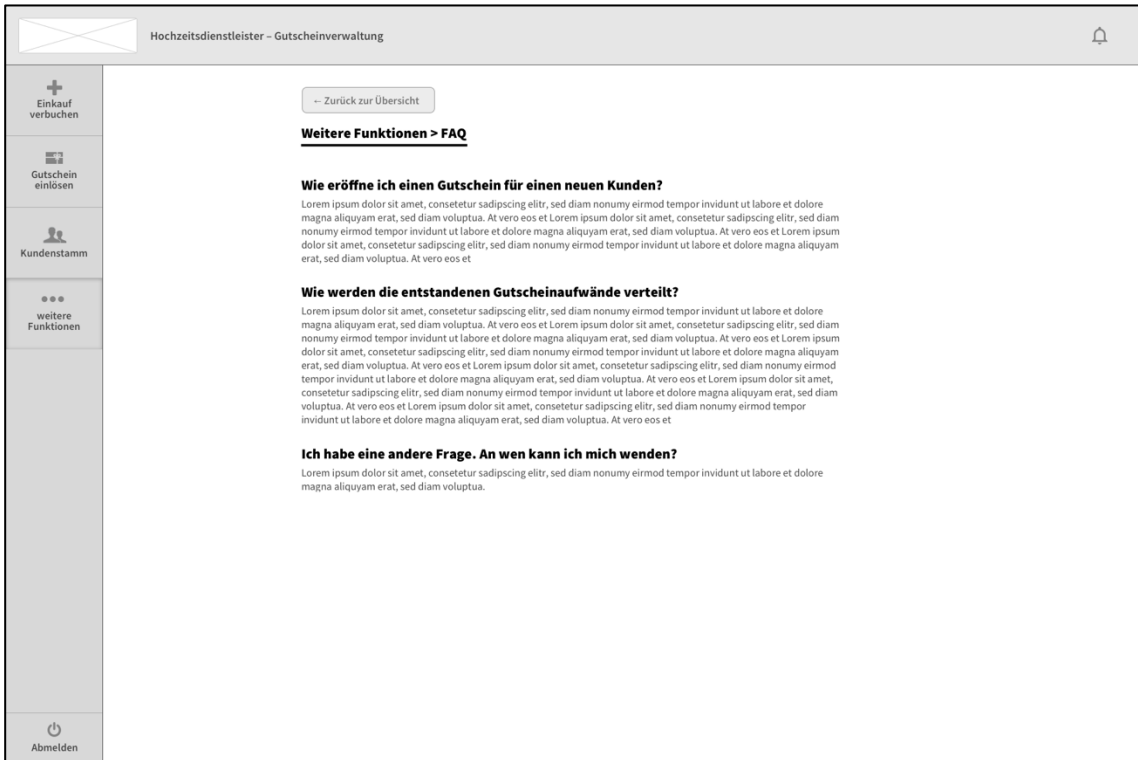


Abbildung 33: Mockup von dem Screen «Häufig gestellte Fragen»

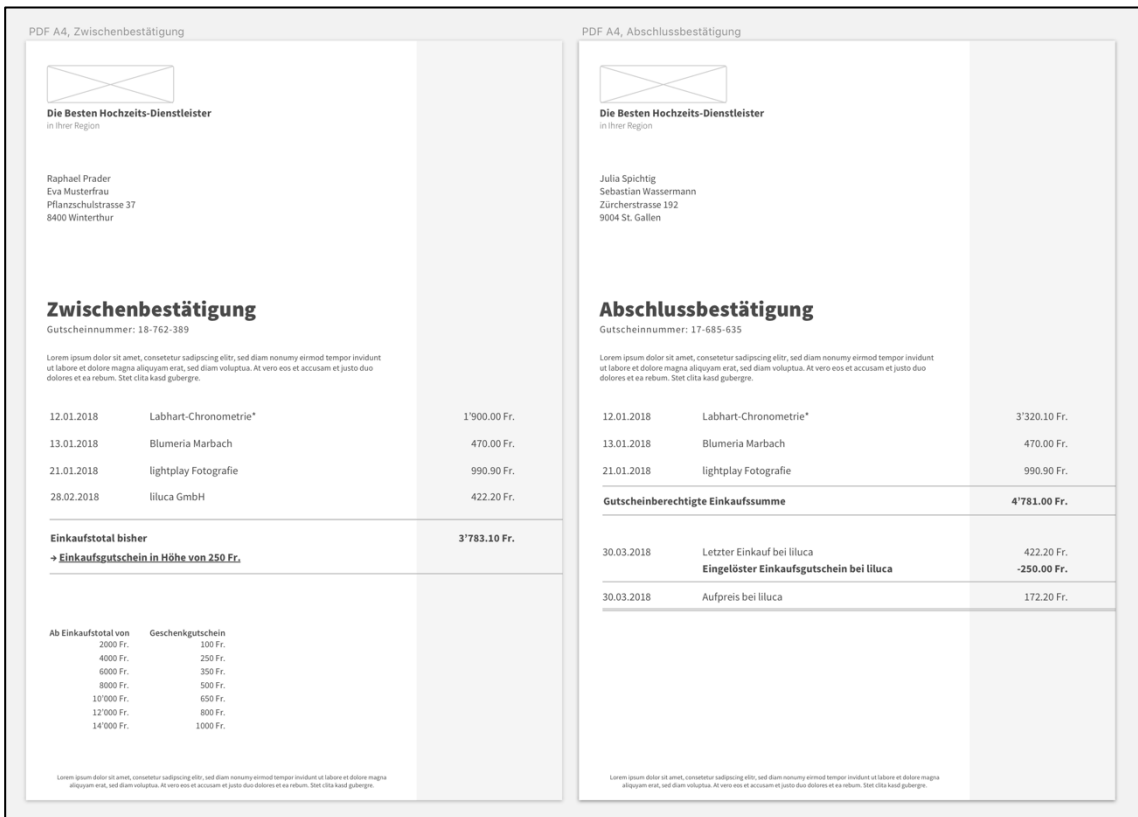


Abbildung 34: Mockup der PDF-Berichte

11.4 Backend Validierung

Coverage report: 100%						
Module ↓	statements	missing	excluded	branches	partial	coverage
accounting/admin.py	18	0	0	4	0	100%
accounting/apps.py	4	0	0	0	0	100%
accounting/management/commands/create_transactions.py	5	0	0	0	0	100%
accounting/models.py	26	0	0	0	0	100%
accounting/permissions.py	6	0	0	2	0	100%
accounting/serializers.py	30	0	0	2	0	100%
accounting/tests/test_admin.py	30	0	0	0	0	100%
accounting/tests/test_models.py	31	0	0	0	0	100%
accounting/tests/test_permissions.py	37	0	0	0	0	100%
accounting/tests/test_serializers.py	27	0	0	0	0	100%
accounting/tests/test_views.py	58	0	0	8	0	100%
accounting/tests/utils/test_closure_utils.py	99	0	0	0	0	100%
accounting/tests/utils/test_redeem_utils.py	36	0	0	0	0	100%
accounting/tests/utils/test_transaction_utils.py	28	0	0	2	0	100%
accounting/utils/closure_utils.py	43	0	0	16	0	100%
accounting/utils/redeem_utils.py	26	0	0	12	0	100%
accounting/utils/transaction_utils.py	29	0	0	10	0	100%
accounting/views.py	22	0	0	0	0	100%
common/admin.py	8	0	0	4	0	100%
common/exceptions.py	2	0	0	0	0	100%
common/fields.py	4	0	0	0	0	100%
common/forms.py	12	0	0	2	0	100%
common/models.py	21	0	0	0	0	100%
common/serializers.py	48	0	0	10	0	100%
common/tests/test_forms.py	15	0	0	0	0	100%
common/tests/test_models.py	11	0	0	0	0	100%
common/tests/test_serializers.py	70	0	0	0	0	100%
common/tests/test_views.py	8	0	0	0	0	100%
common/tests/test_viewsets.py	33	0	0	0	0	100%
common/views.py	14	0	0	0	0	100%
common/viewsets.py	15	0	0	0	0	100%
companies/admin.py	5	0	0	2	0	100%
companies/apps.py	4	0	0	0	0	100%
companies/management/commands/create_companies.py	20	0	0	8	0	100%
companies/models.py	14	0	0	0	0	100%
companies/serializers.py	9	0	0	0	0	100%
companies/tests/management/commands/test_create_companies.py	12	0	0	0	0	100%
companies/tests/test_models.py	9	0	0	0	0	100%
companies/tests/test_serializers.py	12	0	0	0	0	100%
companies/views.py	6	0	0	0	0	100%
customers/admin.py	37	0	0	4	0	100%
customers/apps.py	4	0	0	0	0	100%
customers/management/commands/create_fake_customers.py	28	0	0	8	0	100%
customers/models.py	118	0	0	18	0	100%
customers/serializers.py	39	0	0	0	0	100%
customers/tests/management/commands/test_create_fake_customers.py	10	0	0	0	0	100%
customers/tests/test_admin.py	36	0	0	0	0	100%
customers/tests/test_models.py	115	0	0	4	0	100%
customers/tests/test_serializers.py	27	0	0	0	0	100%
customers/tests/test_views.py	93	0	0	0	0	100%
customers/views.py	67	0	0	2	0	100%
purchases/admin.py	19	0	0	6	0	100%
purchases/apps.py	4	0	0	0	0	100%
purchases/models.py	26	0	0	0	0	100%
purchases/serializers.py	67	0	0	16	0	100%
purchases/tests/test_admin.py	25	0	0	0	0	100%
purchases/tests/test_models.py	15	0	0	0	0	100%
purchases/tests/test_serializers.py	92	0	0	0	0	100%
purchases/views.py	25	0	0	0	0	100%
Total	1754	0	0	140	0	100%

coverage.py v4.5.1, created at 2018-05-22 11:32

Abbildung 36: Coverage Report für die REST-API

11.5 Frontend Umsetzung

11.5.1 iFrame

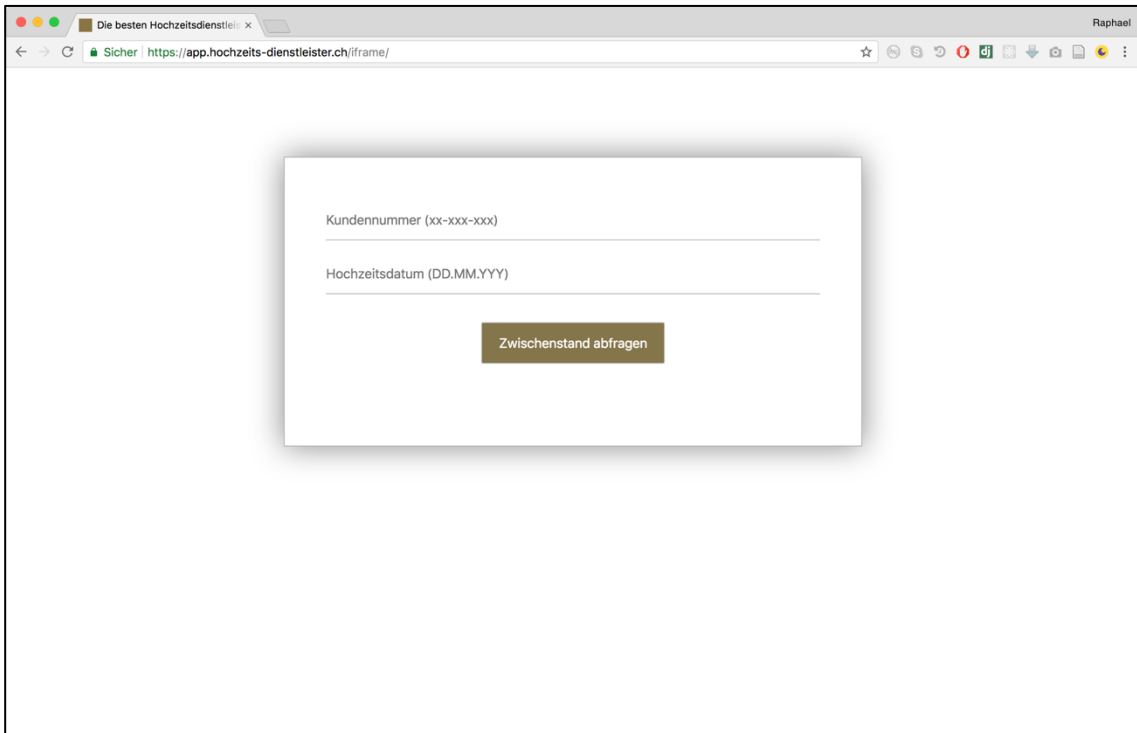


Abbildung 37: Ansicht von dem iFrame für die Hochzeitspaare

11.5.2 View Komponenten

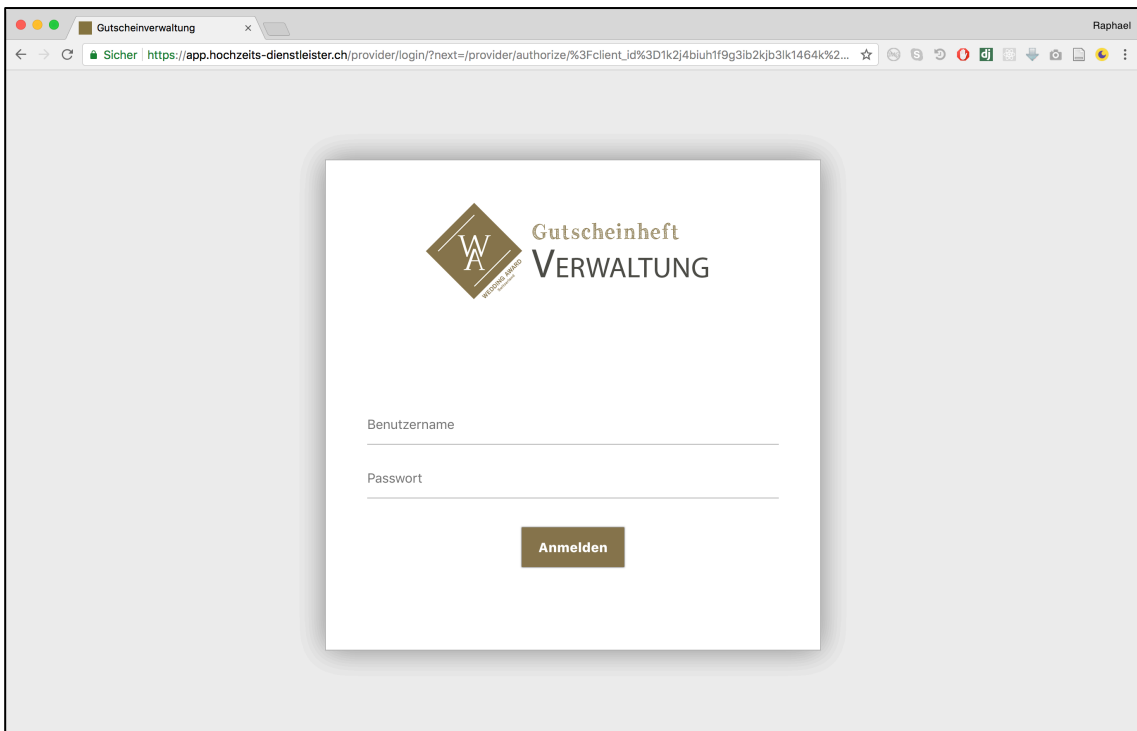


Abbildung 38: Ansicht der Seite «Login»

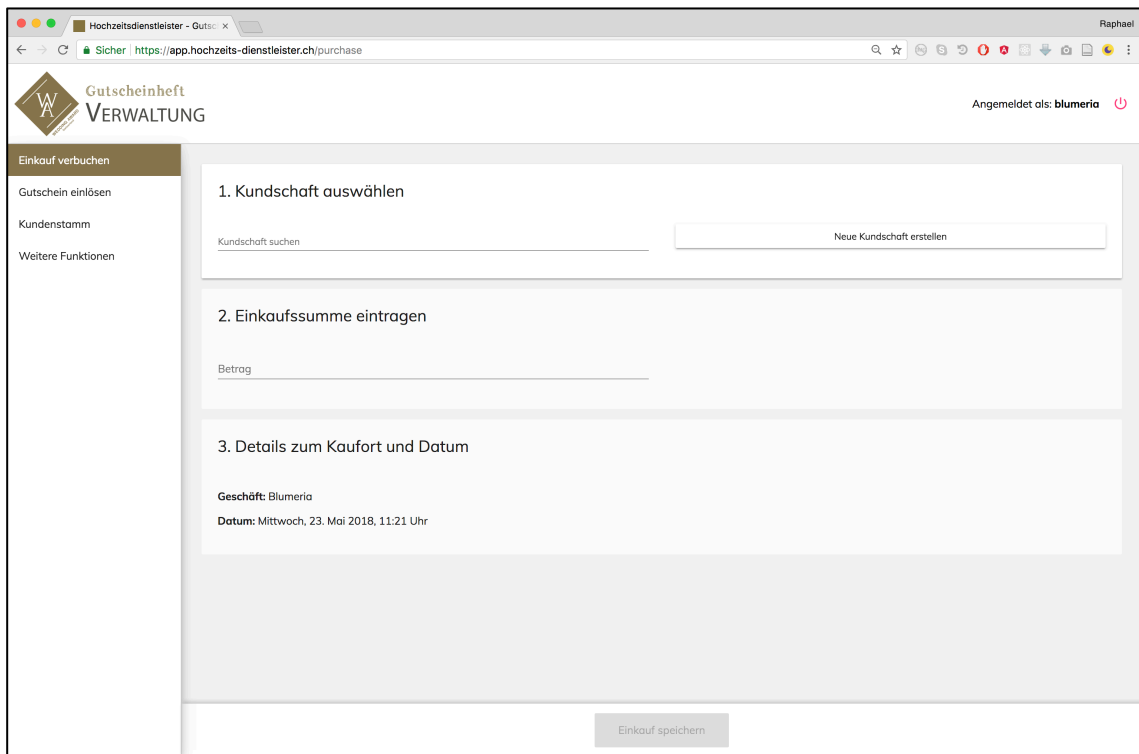


Abbildung 39: Ansicht der Seite «Einkauf verbuchen»

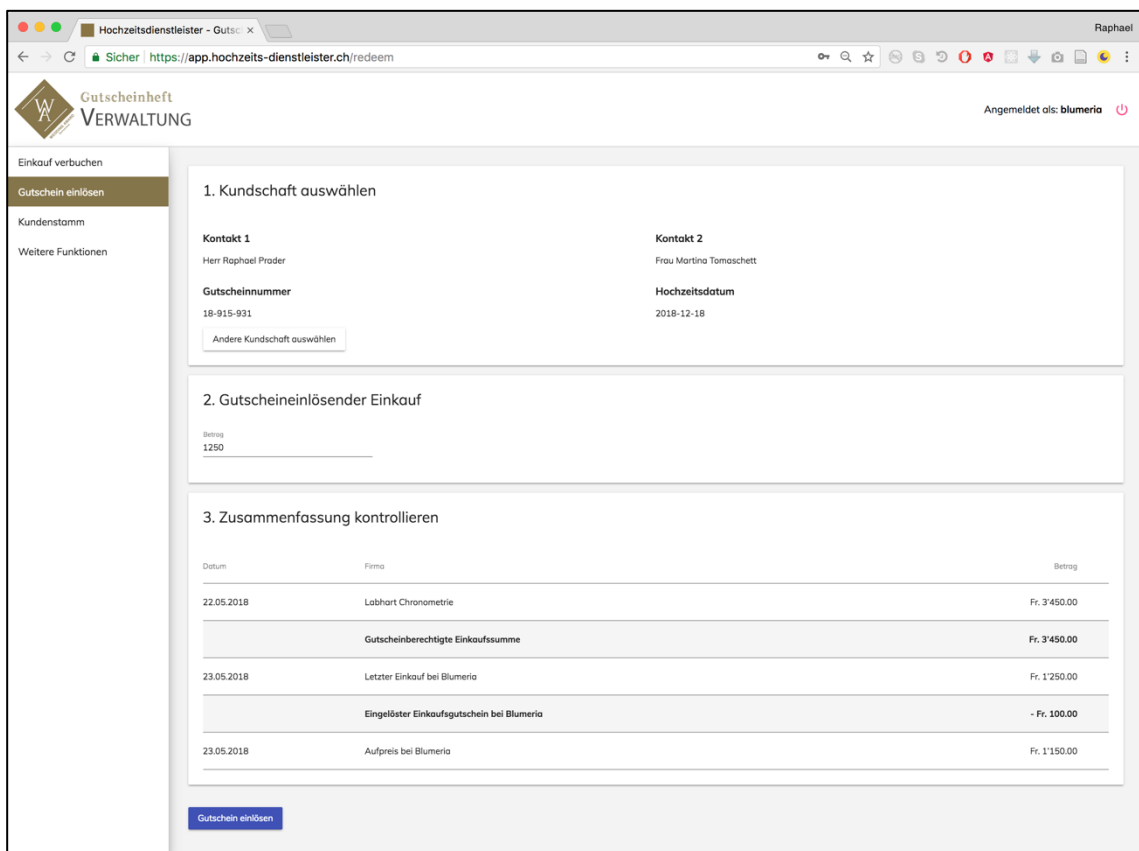


Abbildung 40: Ansicht der Seite «Gutschein einlösen»

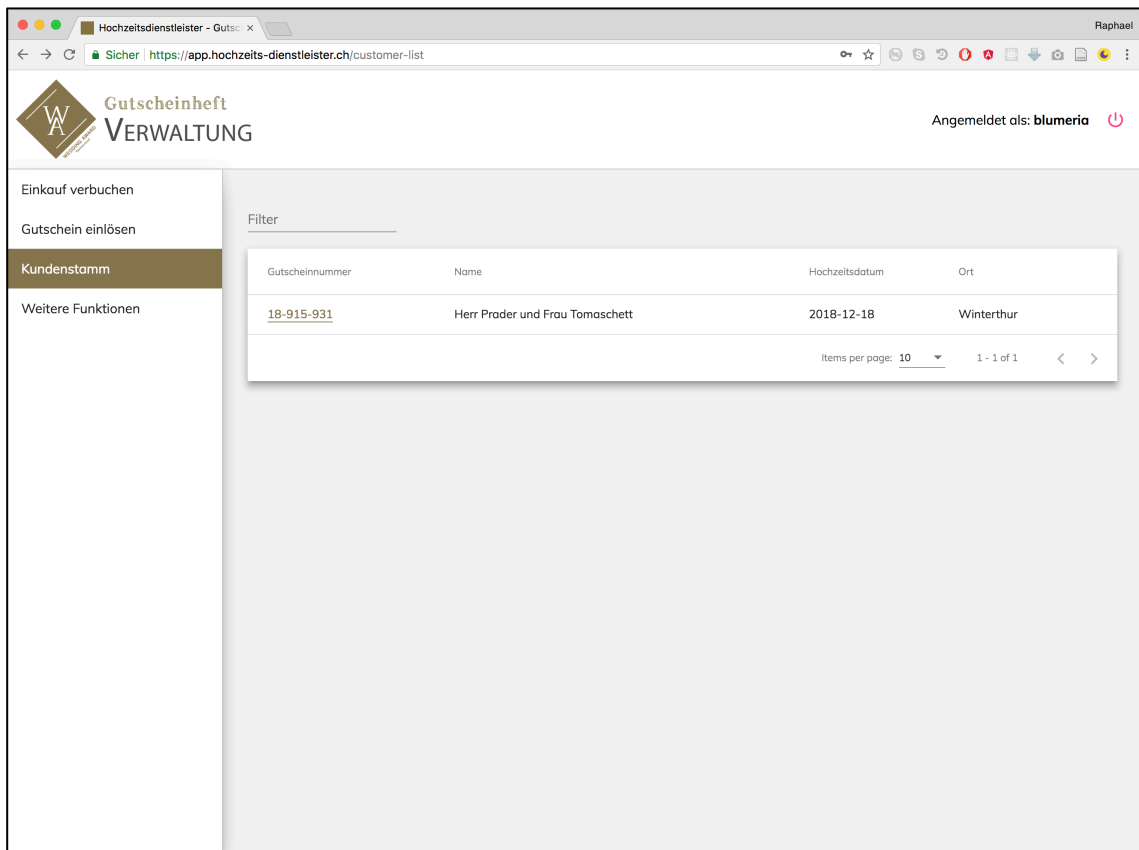


Abbildung 41: Ansicht der Seite «Kundenstamm»

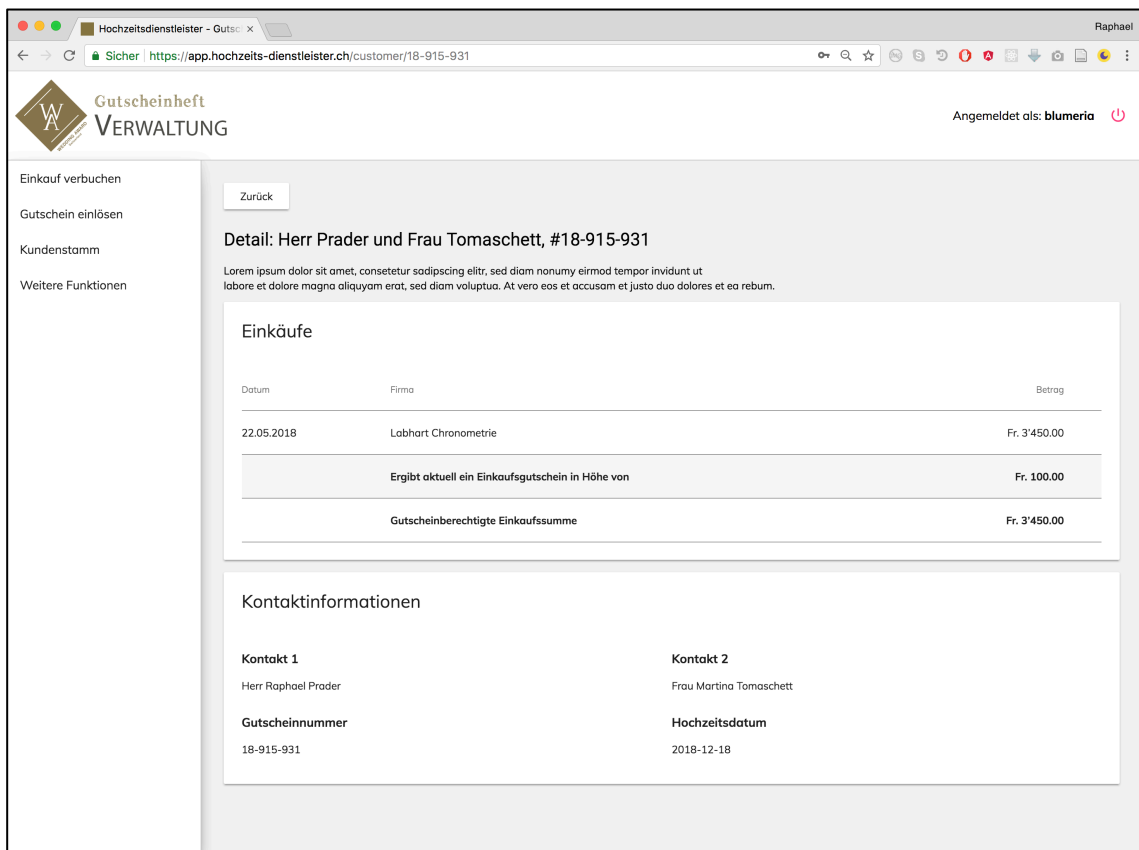


Abbildung 42: Ansicht der Seite «Kunde Detail»

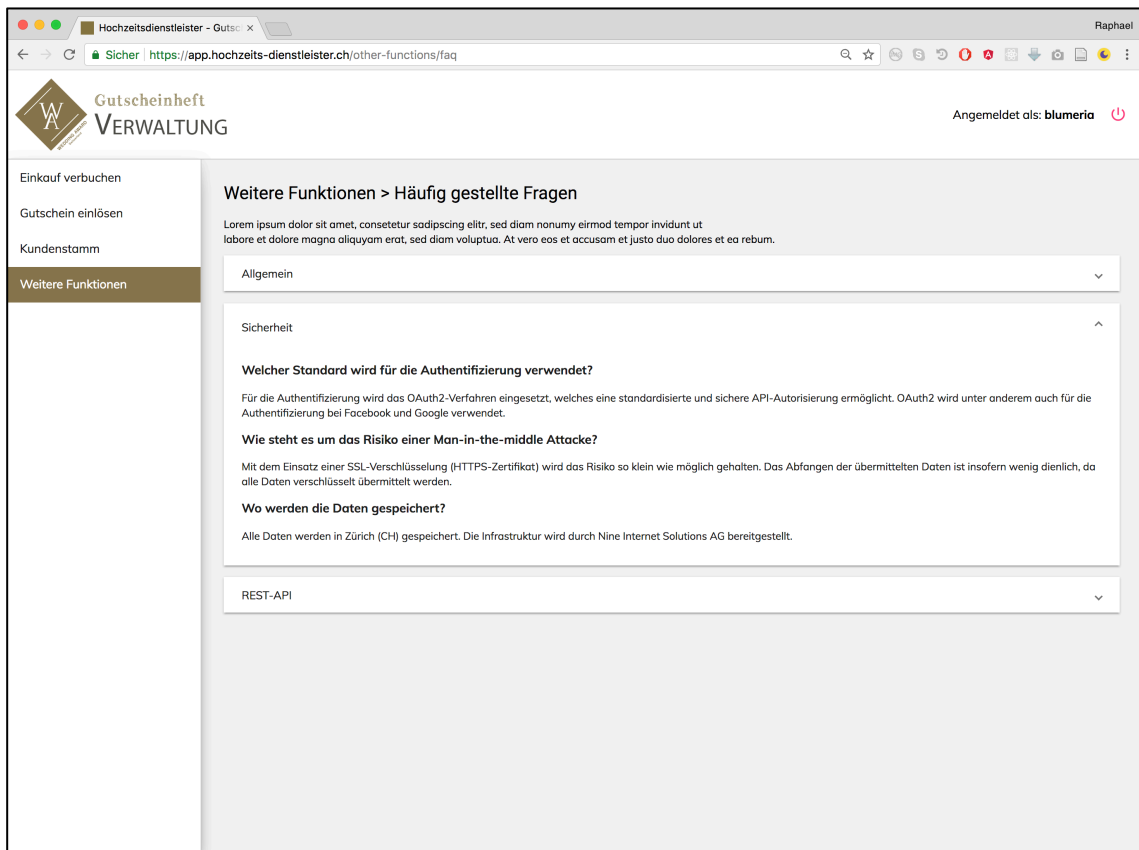


Abbildung 43: Ansicht der Seite «Häufig gestellte Fragen»

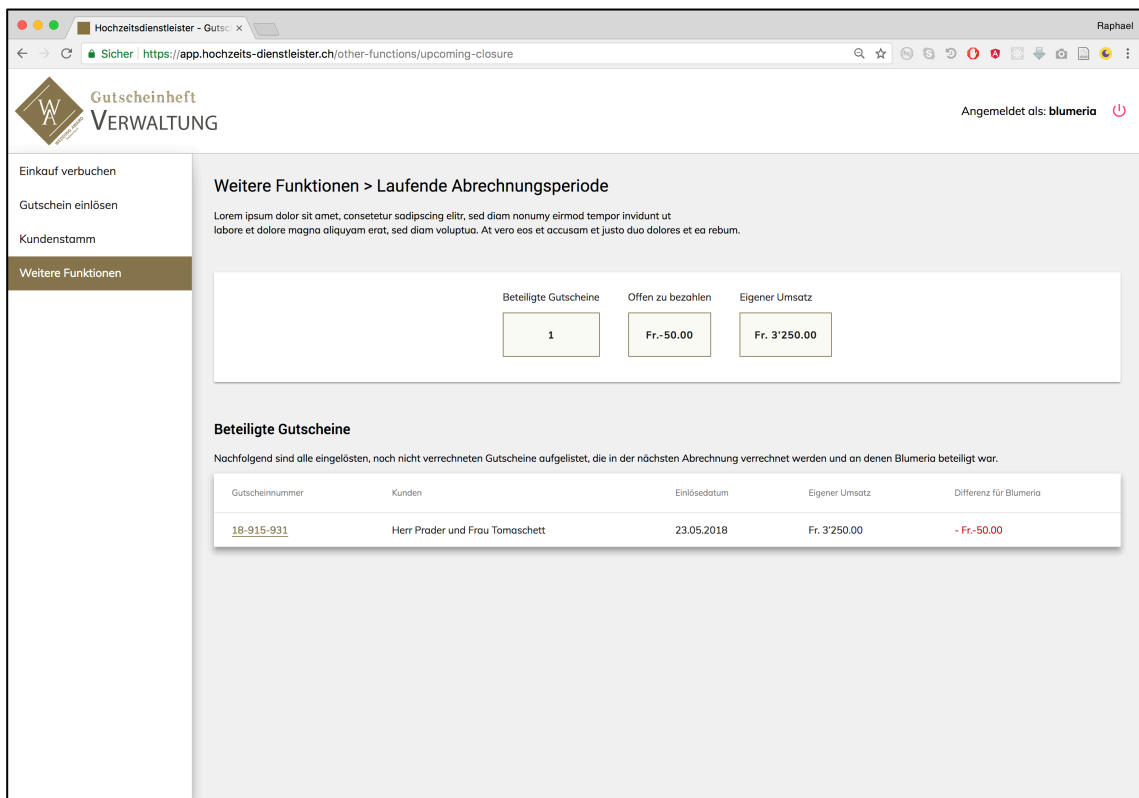


Abbildung 44: Ansicht der Seite «Laufende Abrechnungsperiode»

11.5.3 Komponenten

Offene Zahlungseingänge			
Zahlung von	Abrechnungsdatum	Betrag	Bezahlt?
Lightplay Fotografen	12.05.2018	Fr. 91.94	Als bezahlt markieren
Labhart Chronometrie & Schmuckgalerie	12.05.2018	Fr. 727.97	Als bezahlt markieren

Abbildung 45: Komponente «ConfirmTransactionPaymentComponent»

1. Neue Kundschaft erstellen

Anrede	Vorname 1	Nachname 1
Herr	▼ Max	Muster
Anrede	Vorname 2	Nachname 2
Frau	▼ Eva	Meier

Strasse
Teslastrasse 11

PLZ
8400

Ort
Winterthur

Gemeinsamer Nachname (optional)
Muster-Meier

Hochzeitsdatum
18.5.2018

Kontaktdaten von Max Muster Eva Meier

E-Mail
max.muster@example.com

Telefon
+41 44 123 45

MAI 2018

Mo	Di	Mi	Do	Fr	Sa	So
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Abbildung 46: Komponente «CreateCustomerComponent» und «EditCustomerComponent»

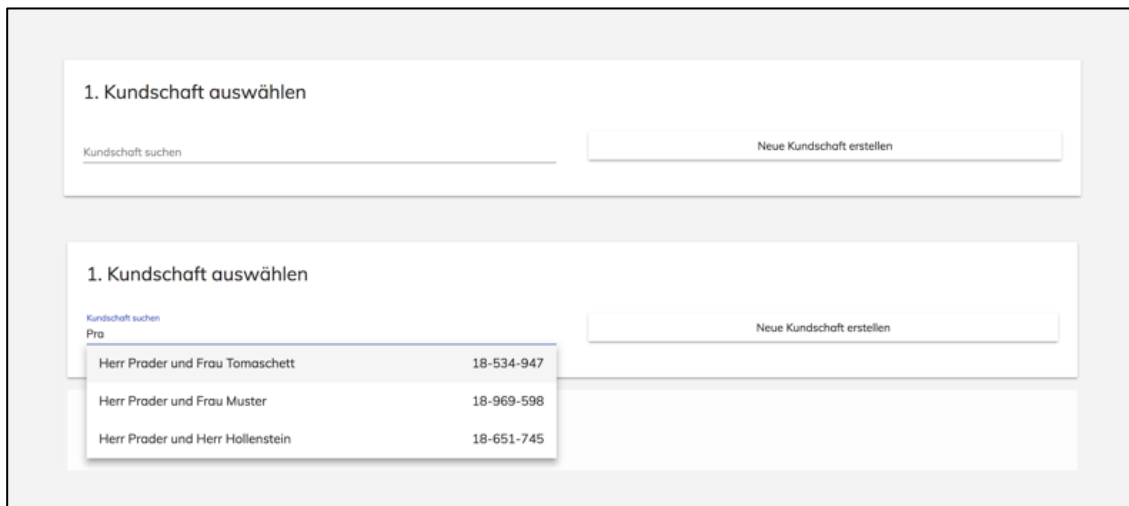


Abbildung 47: Komponente «SelectCustomerComponent»

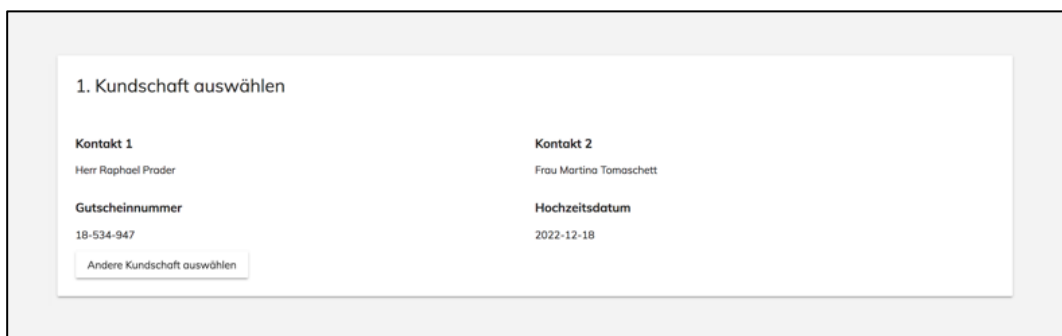


Abbildung 48: Darstellung der Komponente «ShowCustomerComponent»



Abbildung 49: Darstellung der Komponente «TablePurchaseRowComponent»

11.6 Zugangsdaten

11.6.1 Repository

Dem Betreuer dieser Arbeit wird während dem Bewertungszeitraum darauf Zugriff gewährt. Das Git-Repository ist unter folgender URL aufrufbar:

<https://git.lab360.cloud/raphael.prader/bachelorarbeit-gutscheinverwaltung> .

11.6.2 Applikation

Die Gutscheilverwaltungsapplikation kann während dem Bewertungszeitraum dieser Arbeit getestet werden. Die URL zum Aufruf der Applikation lautet:

<https://app.hochzeits-dienstleister.ch/>

Die Zugangsdaten lauten:

Benutzername	Passwort	Geschäft
blumeria	password	Blumeria Marbach
labhart	password	Labhart-Chronometrie
lightplay	password	Lightplay GmbH
liluca	password	Liluca Pronuptia Suisse AG
nisago	password	nisago GmbH