



Abschlussprüfung Sommer 2016

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Entwicklung einer Statistik-App

Webbasierte App für zur statistischen Analyse von KPIs

Abgabetermin: Gera, den 30.11.2016

Prüfungsbewerber:

Steven Hergt
Georg-Büchner-Str. 9
07749 Jena



Ausbildungsbetrieb:

ePages GmbH
Heinrich-Heine-Str. 1
07749 Jena

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors un-

zulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Inhaltsverzeichnis

Inhaltsverzeichnis

Abbildungsverzeichnis	III
------------------------------	------------

Tabellenverzeichnis	IV
----------------------------	-----------

Listings	V
-----------------	----------

Abkürzungsverzeichnis	VI
------------------------------	-----------

1	Einleitung	1
1.1	Projektfeld	1
1.2	Projektziel	1
1.3	Projektbegründung	1
1.4	Projektschnittstellen	2
1.5	Projektabgrenzung	3
2	Projektplanung	4
2.1	Projektphasen	4
2.2	Abweichungen vom Projektantrag	4
2.3	Ressourcenplanung	5
2.4	Entwicklungsprozess	5
3	Analysephase	5
3.1	Ist-Analyse	5
3.2	Wirtschaftlichkeitsanalyse	7
3.2.1	„Make or Buy“-Entscheidung	7
3.2.2	Projektkosten	8
3.2.3	Amortisationsdauer	9
3.3	Anwendungsfälle	10
3.4	Qualitätsanforderungen	10
3.5	Lastenheft/Fachkonzept	11
4	Entwurfsphase	13
4.1	Zielplattform	13
4.2	Architekturdesign	14
4.3	Entwurf der Benutzeroberfläche	15
4.4	Datenmodell	15
4.5	Geschäftslogik	15
4.6	Maßnahmen zur Qualitätssicherung	17

Inhaltsverzeichnis

5	Implementierungsphase	17
5.1	Verwendete Bibliotheken und Tools	17
5.2	Implementierung der Datenstrukturen	17
5.3	Implementierung der Benutzeroberfläche	18
5.4	Implementierung der Geschäftslogik	20
5.5	Implementierung des Session-Managements	20
6	Abnahmephase	21
6.0.1	CodeReview	21
6.0.2	QA-Test	22
7	Dokumentation	23
8	Fazit	23
8.1	Soll-/Ist-Vergleich	23
8.2	Lessons Learned	24
8.3	Ausblick	24
	Literaturverzeichnis	25
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Vergleich zwischen geplanter und tatsächlich gebrauchter Zeit	ii
A.3	Ausschnitt aus der Tabellenstruktur in phpMyAdmin	iii
A.4	Datenbankmodell für die MySQL-Datenbanktabellen	iv
A.5	Sequenzdiagramm zum Synchronisationsprozess	v
A.6	React-Komponenten-Architektur und Abhängigkeiten	vi
A.7	Einstiegspunkt für die Single-Page-App	vii
A.8	Clientseitige Abhandlung der Synchronisation	vii
A.9	Serverseitige Abhandlung der Synchronisation	viii

Abbildungsverzeichnis

Abbildungsverzeichnis

1	Widgetauswahl im Dashboard eines Merchant-Backoffice eines ePages Onlineshops	6
2	Graphische Darstellung von Bestellstatistiken	6
3	Auflistung von Artikelstatistiken	7
4	Amortisationszeit pro monatlicher Miete	9
5	Anwendungsfalldiagramm für die Statistik-App	10
6	ER-Modell: Beziehungen und Kardinalitäten zwischen den Entitäten . .	15
7	Design und Aufbau des Loginbereichs	18
8	Design und Aufbau des Headers im Benutzerbereich	19
9	Beispiel eines Dropdown-Feldes nach einem Klickereignis	19
10	Diagramme für statistische Auswertungen	19
11	Top-Ten-Tabelle der umsatzstärksten Kunden	20
12	Cookiesetzung in der Login-Komponente	21

Tabellenverzeichnis

Tabellenverzeichnis

1	Zeitplanung	4
2	Kostenaufstellung	8

Listings

Listings

Abkürzungsverzeichnis

Abkürzungsverzeichnis

App	Applikation
DOM	Document Object Model
FTP	File Transfer Protocol
Git	Versionskontrollsystem
GitHub	Online-Dienst zur Verwaltung quelloffener Software
QA	Quality Assurance
REST	Representational state transfer
REST-API	REST-Schnittstelle
SCP	Secure Copy
SFTP	SSH File Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
UML	Unified Modeling Language

1 Einleitung

1.1 Projektumfeld

Die ePages GmbH ist ein deutsches Software- und Dienstleistungsunternehmen, das Produkte zur Ermöglichung eines elektronischen Handels (E-Commerce) bereitstellt, d.h. Kunden können mit der Produktsoftware, die über Hosting-Provider wie Strato AG, 1 & 1, T-Online etc. vertrieben wird, einen individualisierten Onlineshop aufsetzen und ihn gegen eine monatliche Gebühr betreiben.

Die Firma wurde 1983 als "Beeck & Dahms GbR" von dem jetzigen Geschäftsführer Wilfried Beeck in Kiel gegründet und war später Teil der Intershop AG bis zur Absplittierung im Jahr 2002. Momentan arbeiten in der Firma insgesamt rund 180 Mitarbeiter. Der Hauptsitz der Firma ist in Hamburg, danach kommt Jena als Firmensitz mit rund 40 Mitarbeitern. Der Auftrag zur Erstellung der App kommt vom Produktmanagement und ist aus Kundenrückmeldungen entstanden. Innerhalb der Firma gibt es verschiedene mehr oder minder unabhängige Entwicklungsteams. Ich bin dabei Teil des ePages6-Core-Teams als Frontend-/Javascriptentwickler, das wiederum Teil der R&D-Abteilung ist. Die Projekterstellung findet halbtags während der Sprints statt, d.h. ich stehe daneben noch dem Team halbtäglich zur Unterstützung zur Verfügung und gehe in den Dailys auch immer auf den Status meines Projektes ein.

1.2 Projektziel

Ziel des Projektes ist die Erstellung einer externen WebApp für Endkunden eines ePages Onlineshops. Mit dieser App soll es für den Kunden möglich sein spezielle KPIs für deren Onlineshop berichtsmäßig dokumentiert und deren zeitliche Entwicklung angezeigt zu bekommen. Daraus sollen Hinweise zum Anpreisen bestimmter Artikel resultieren. Auf alle relevanten Bestelldaten des Onlineshops soll per REST-API zugegriffen werden. Die wichtigsten KPIs sind hierbei der Umsatz und die meistverkauften Produkte. Die Berichte sollen anpassbar an frei wählbare Zeiträume und den Bezahlstatus sein.

1.3 Projektbegründung

Für Endkunden eines ePages-Onlineshops besteht standardmäßig die Möglichkeit über das Erstellungsmenü ihres Shops verschiedene Analysewidgets auszuwählen,

1 Einleitung

die jedoch nur die wesentlichen KPIs als Umsatz- und Artikelverkaufsstatistiken bereitstellen ohne daraus spezielleren Handlungsbedarf des Händlers abzuleiten. Durch eine Nutzerumfrage wurde festgestellt, dass von den Händlern genauere Analysen des Käuferverhaltens (Kaufabbruchrate, Herkunft, Bestellzeiten etc.) und mehr Anpassungsmöglichkeiten (KPIs pro Kunde) gewünscht werden. Die Kunden haben zwar die Möglichkeit sich bei externen Firmen wie etracker für umfangreiche Statistikauswertungen für ihren Onlineshop anzumelden, jedoch besitzt das ePages System auch einen eigenen App-Store, der sich als Verkaufsplattform für eine eigens dafür programmierte Statistik-App ebenfalls anbietet, welche auf die Nutzerwünsche zugeschnitten ist und damit höhere Nutzerbindung und Nutzerzufriedenheit als Zielsetzung hat. Das ist auch hinsichtlich der Vermarktung des neuesten Softwareproduktes von ePages sinnvoll, welches im nächsten Jahr ausgerollt wird und eine moderne Variante des Bestandsproduktes darstellt, wodurch neue Kunden gewonnen werden sollen und die Konkurrenzfähigkeit auf dem bestehenden Markt gewährleistet werden soll. Nicht zuletzt verdient ePages auch aktiv an der Vermarktung der App pro Nutzer.

1.4 Projektschnittstellen

Das Projekt wurde von meinem Ausbilder (Markus Höllein) und meinem direkten Disziplinarvorgesetzten (Mario Rieß) genehmigt, welcher stellvertretend für den Ausbildungsbetrieb spricht.

Die Projektmittel umfassen an Hardware einen leistungsstarken Laptop, zwei Monitore, ein externes Keyboard und eine Maus. An Software wird ein externer Zugang zu einem ePages-Developer-Shop benötigt, der exemplarisch als Anbindungsstelle für die zu entwickelnde App fungiert. Hard- und Software werden dabei vom Ausbildungsbetrieb bereitgestellt. Die App wird auf einem externen Server von "uberspace.de" gehostet, auf dem sich auch die MySQL-Datenbank der App befindet. Die monatlichen Kosten dafür übernehme ich selbst. Des Weiteren wird Git als Versionierungstool verwendet, wobei der firmeninterne GitHub-Zugang verwendet wird, um das Projekt auch extern auf dem GitHub-Server speichern zu können. Die Anbindung der App an die ePages-Software geschieht über die REST-API von ePages unter Verwendung einer speziellen PHP-Klasse¹. Dies geschieht in Zusammenarbeit mit erfahrenen Programmierern. Die finanziellen Mittel bzw. die Ausbildungsvergütung stellt die HR-Abteilung zur Verfügung.

Nutzer der App können alle ePages-Endkunden sein, die sich für die kostenbehaftete Nutzung der App für ihren Onlineshop entschließen. Das Ergebnis des Projekts muss zuallererst der teaminternen QA-Abteilung zum Testen präsentiert werden. Bestehen

¹<https://github.com/ePages-de/epages-rest-php>

1 Einleitung

hier keine Bedenken mehr, wird die App dem Produktmanagement vorgelegt, welches letztendlich darüber entscheidet die App in dem ePages-App-Store zur Nutzung anzubieten oder ob weitere Verbesserungen notwendig sind.

Für die Programmierung der App werden das PHP-Mikroframework Slim² und verschiedene Javascript-Frameworks und Bibliotheken verwendet wie JQuery, React.js³ und D3.js⁴ bzw. die davon abgeleitete C3.js-Bibliothek⁵, welche allesamt kostenlos sind.

- Mit welchen anderen Systemen interagiert die Anwendung (technische Schnittstellen)?
- Wer genehmigt das Projekt bzw. stellt Mittel zur Verfügung?
- Wer sind die Benutzer der Anwendung?
- Wem muss das Ergebnis präsentiert werden?

1.5 Projektabgrenzung

Das Projekt ist nicht Teil der ePages-Bestandssoftware oder irgendeines anderen ePages-Softwareproduktes, sondern stellt als externe App eine eigenständige Software dar, die mittels geeigneter API nicht nur an ePages, sondern auch an andere Onlineshopsoftware angebunden werden könnte. Vorrangig wird jedoch die Anbindung an ePages sein. Durch die Eigenständigkeit der App wird zudem gewährleistet, dass die Bestandssoftware bei Entwicklungsfehlern/Bugs in der App nicht in Mitleidenschaft gezogen wird.

Das Projekt wird nur soweit entwickelt, dass die Grundfunktionalitäten vorhanden sind, womit alle geplanten Anwendungsfälle realisiert werden können. Der Feinschliff durch das Feedback aus der QA-Abteilung wird aus Zeitgründen in seiner Gänze nicht mehr Teil dieses Projektes sein können genauso wie die Einbeziehung des Feedbacks vom Produktmanagement oder die Integration in den ePages-App-Store.

²<http://www.slimframework.com/>

³<https://facebook.github.io/react/>

⁴<https://d3js.org/>

⁵<http://c3js.org/>

2 Projektplanung

2.1 Projektphasen

Die Gesamtprojektbearbeitungszeit ist auf 70 Stunden festgelegt. Der Start des Projekts ist der 10.10.2016 und der Abschluss ist der 30.11.2016. Die 70 Stunden werden auf den Zeitraum relativ gleichmäßig verteilt, sodass mindestens halbtäglich noch den Tagesgeschäften nachgegangen und das Team unterstützt werden kann. Außerdem muss gewährleistet werden, dass ich trotz des Projektes an den wichtigsten Scrum- und Team-Meetings teilnehmen kann, die die Arbeitszeit regelmäßig und unregelmäßig unterbrechen.

Tabelle 1 zeigt die grobe Zeitplanung.

Projektphase	Geplante Zeit
Analysephase	8 h
Entwurfsphase	6 h
Implementierungsphase	50 h
Erstellen der Dokumentation	6 h
Gesamt	70 h

Tabelle 1: Zeitplanung

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

2.2 Abweichungen vom Projektantrag

Eine wesentliche Ziel der App sollte die Darstellung statistischer Analysen verkaufter Artikel des Onlineshops sein, der die App nutzt. Während der Projektdurchführung wurde jedoch festgestellt, dass die genutzte PHP-Order-Klasse für die REST-Calls noch keine Möglichkeit bietet Artikel bzw. Produkte aus den Shopbestellungen zu extrahieren, obwohl im Vorfeld fälschlicherweise davon ausgegangen wurde, dass diese Möglichkeit besteht⁶. Aus Zeitgründen war es bisher weder mir noch dem verantwortlichen Entwickler möglich die Klasse dahingehend zu erweitern, sodass auf Artikelstatistiken gänzlich verzichtet wurde. Des Weiteren wurde die Registrierungsmöglichkeit für Neukunden der App bisher nur soweit aufbereitet, dass alle Kunden automatisch

⁶<https://github.com/ePages-de/epages-rest-php/blob/master/src/shopobjects/order/Order.class.php>

3 Analysephase

den ePages-Developer-Test-Shop zugewiesen bekommen, da die Anbindung an echte Onlineshops noch gar nicht getestet werden kann. Dazu ist erst das Einverständnis des Produktmanagements nach Sichtung der App notwendig.

2.3 Ressourcenplanung

An Ressourcen ist ein PC-Arbeitsplatz (Schreibtisch, Rollcontainer, Schreibtischstuhl, Schreibtischlampe) in einem Firmenbüro vonnöten. Dazu kommt an Hardware ein LAN- und WLAN-fähiger COREi7-Laptop mit 16 GByte Arbeitsspeicher und mindestens 100 GByte großer Festplatte sowie zwei Monitore, eine Maus und ein Headset. An Software wird ein Windows 10 Betriebssystem, Sublime Text 3 als Text-Editor, ein Internetbrowser (Google Chrom und Firefox), Putty als [SSH-Client](#), WinSCP als [SFTP](#) und [FTP-Client-Software](#), ein externer Server, auf dem die App „gehostet“ wird bereitgestellt von „uberspace.de“, phpMyAdmin zur Verwaltung der Datenbank, HipChat als internes Chattool der Firma, JIRA von Atlassian als Verwaltungssoftware der agilen Entwicklungsprozesse unter Scrum sowie Confluence von Atlassian als interne Kollaborationssoftware für Teams. Dazu kommt die Verwendung von [Git](#) und [GitHub](#) zur Versionierungskontrolle. Darüberhinaus werden auch personelle Ressourcen beansprucht, die einen Backendentwickler zum Einrichten des epages Developer-Shops und des Slim-Frameworks sowie eine [QA-Kollegin](#) zum Testen der App-Funktionen und meinen Ausbilder zur generellen Überwachung meines Projektes umfassen.

2.4 Entwicklungsprozess

Für die Entwicklung wird das Wasserfallmodell verwendet, in dem bestimmte Meilensteine gesetzt und abgeschlossene Phasen definiert werden können. Dadurch werden insbesondere eine gut definierte und ausgereifte Planungs- und Entwurfsphase ermöglicht, die zur eigentlichen Implementierung notwendig sind. enü

3 Analysephase

3.1 Ist-Analyse

Derzeit haben Mieter eines ePages-Onlineshops die Möglichkeit im Backoffice ihres Onlineshop, in dem alle wichtigen Shopeinstellungen zu treffen sind, zwei Analysewidgets auszuwählen, siehe [Abbildung 1](#) Das erste zeigt eine Top-3-Liste der meistverkauften Artikel der letzten 30 Tage an und das zweite den Nettoumsatz sowie die

3 Analysephase

Anzahl der Bestellungen für heute, gestern, diese Woche, letzte Woche, diesen und letzten Monat an. Des Weiteren wurde bereits auf einem zweitägigen firmeninternen

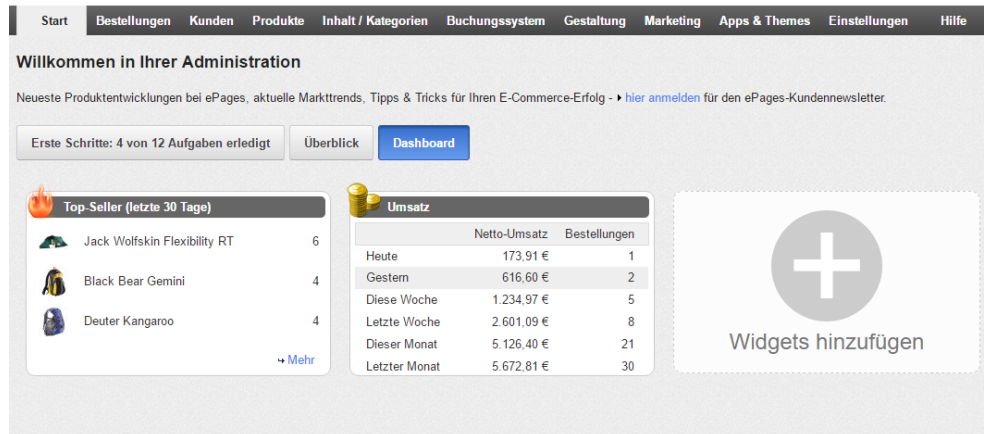


Abbildung 1: Widgetauswahl im Dashboard eines Merchant-Backoffice eines ePages Onlineshops

Hackathon im Frühjahr 2016 der Versuch der Programmierung einer Statistik-App unternommen, wobei dabei nur mit „Mock-Daten“ hantiert wurde und keine echte Shopanbindung realisiert wurde. Die Ergebnisse zu dieser App sind auf einer internen Wikiseite veröffentlicht und liefern einen ersten Eindruck wie so eine App aussehen könnte und welche Funktionalitäten und Use-Cases ePages vorrangig für solch eine Statistik-App vorsieht. Zu sehen ist ein Menü, über das Bestellungs- und Artikelstatistiken anwählbar sind. In [Abbildung 2](#) sieht man wie eine graphische Darstellung von Umsätzen pro Bestellung aussehen könnte.

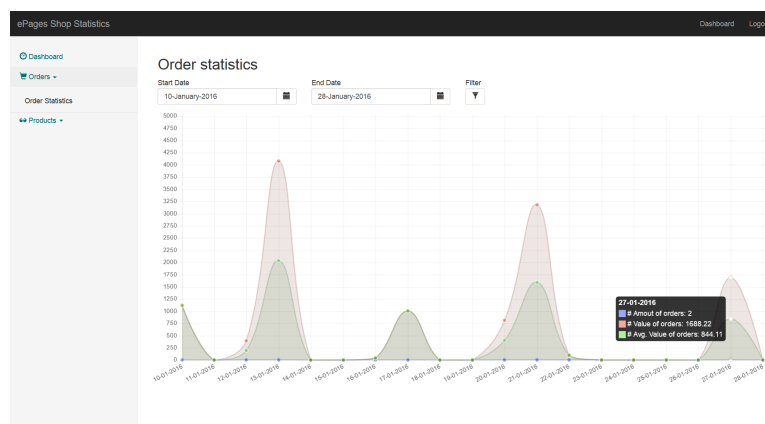
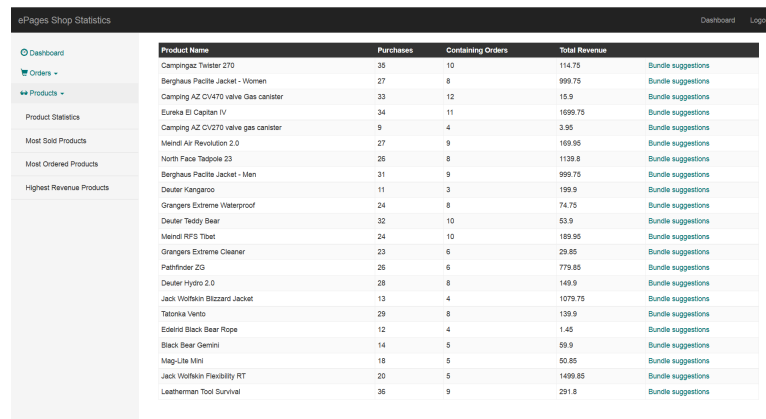


Abbildung 2: Graphische Darstellung von Bestellstatistiken

3 Analysephase

In [Abbildung 3](#) werden Artikel mit Gesamtumsatz und Bestellanzahl aufgelistet. Diese Beispiel-App verwendet für das Frontend das AngularJS-Framework und die Backendskripte sind in Ruby on Rails programmiert. Datenbankabfragen oder angelegte Datenbanken gibt es keine, genauso wenig eine Registrierungs- oder Loginfunktionalität.



The screenshot shows the 'ePages Shop Statistics' dashboard. On the left is a sidebar with navigation links: Dashboard (selected), Orders, Products, Product Statistics, Most Sold Products, Most Ordered Products, and Highest Revenue Products. The main content area displays a table with the following columns: Product Name, Purchases, Containing Orders, Total Revenue, and a link for Bundle suggestions. The table lists 20 different products with their respective statistics.

Product Name	Purchases	Containing Orders	Total Revenue	
Campingaz Twister 270	35	10	114.75	Bundle suggestions
Berghaus Pacille Jacket - Women	27	8	999.75	Bundle suggestions
Camping AZ CIV470 valve Gas canister	33	12	15.9	Bundle suggestions
Eureka El Captain IV	34	11	1699.75	Bundle suggestions
Camping AZ CIV270 valve gas canister	9	4	3.95	Bundle suggestions
Mendi AJ Revolution 2.0	27	9	189.95	Bundle suggestions
North Face Tardieu 23	26	8	1139.9	Bundle suggestions
Berghaus Pacille Jacket - Men	21	9	999.75	Bundle suggestions
Deuter Kangaroo	11	3	159.9	Bundle suggestions
Grangers Extreme Waterproof	24	8	74.75	Bundle suggestions
Deuter Teddy Bear	32	10	53.9	Bundle suggestions
Mendi RFS Tibet	24	10	189.95	Bundle suggestions
Grangers Extreme Cleaner	23	6	29.85	Bundle suggestions
Patfinder ZG	26	6	779.85	Bundle suggestions
Deuter Hytto 2.0	28	8	149.9	Bundle suggestions
Jack Wolfskin Blizzard Jacket	13	4	1079.75	Bundle suggestions
Tatonka Vento	29	8	139.9	Bundle suggestions
Edsind Black Bear Rope	12	4	1.45	Bundle suggestions
Black Bear Gemini	14	5	59.9	Bundle suggestions
Mag-Lite Mini	18	5	50.85	Bundle suggestions
Jack Wolfskin Flexibility RT	20	5	1499.85	Bundle suggestions
Leatherman Tool Survival	36	9	291.8	Bundle suggestions

Abbildung 3: Auflistung von Artikelstatistiken

3.2 Wirtschaftlichkeitsanalyse

3.2.1 „Make or Buy“-Entscheidung

Abgesehen von den im MBO hinzufügbaren Statistik-Wigdetts fehlen tiefergehende statistische Analysen und grafische Darstellungen der wichtigsten und interessantesten KPIs, die sich dynamisch über REST-Calls aktualisieren lassen. Darunter fallen die Möglichkeit der Berichterstattung von KPIs pro Kunde und genauere Analysen des Käuferverhaltens (Kaufabbruchrate, Herkunft, Bestellzeiten etc.). Der Wunsch nach einer auf das ePages-System zugeschnittenen App für den ePages-App-Store wurde direkt vom Produktmanagement geäußert und sollte bereits schon mal während eines zweitägigen Hackathons entwickelt werden, was jedoch aufgrund des zu hohem Zeitaufwands für die Entwicklung bei weitem nicht fertig gestellt werden konnte. Durch diese App verspricht sich das Produktmanagement eine höhere Kundenzufriedenheit und damit eine bessere Kundenbindung an die ePages Softwareprodukte sowie längerfristig eine Anwerbung neuer Kunden, die sich durch die Existenz dieser speziellen App vorzugsweise für ePages anstatt für eine andere Onlineshopsoftware entscheiden würden.

3 Analysephase

3.2.2 Projektkosten

Die Kosten für die Durchführung des Projektes setzen sich für die 70 Stunden Bearbeitungszeit sowie aus Personal- als auch Ressourcenkosten zusammen.

Berechnung der Entwicklungskosten für ePages Laut Arbeitsvertrag liegt meine Ausbildungsvergütung im aktuellen Lehrjahr bei 680 € pro Monat. Damit verursache ich der Firma jährliche Kosten in Höhe von

$$680 \text{ €/Monat} \cdot 12 \text{ Monate/Jahr} = 8160 \text{ €/Jahr.} \quad (1)$$

Die Anzahl der Arbeitstage 2016 belaufen sich auf 252 Tage in Thüringen (Jena). Davon stehen mir 25 Urlaubstage zu. Es verbleiben $(252 - 25)$ Tage = 227 Tage vollwertige achtstündige Arbeitstage. Meine Stundenlohn berechnet sich damit zu

$$8 \text{ h/Tag} \cdot 227 \text{ Tage/Jahr} = 1816 \text{ h/Jahr.} \quad (2)$$

$$\frac{8160 \text{ €/Jahr}}{1816 \text{ h/Jahr}} \approx 4,49 \text{ €/h.} \quad (3)$$

Für die Nutzung der Ressourcen⁷ wird ein pauschaler Stundensatz von 15 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 25 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt 1844,30 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	70 h	4,49 € + 15 € = 19,49 €	1364,30 €
Unterstützung durch Mitarbeiter	10 h	25 € + 15 € = 40 €	400,00 €
Abnahmetest durch QA	2 h	25 € + 15 € = 40 €	80,00 €
			1844,30 €

Tabelle 2: Kostenaufstellung

⁷Laptop, Monitore, Servernutzung, Büro- und Firmenräumlichkeiten, Stromverbrauch, Heizkosten etc.

3 Analysephase

3.2.3 Amortisationsdauer

Geplant ist die fertige App den Nutzern der ePages-Software zur monatlichen Miete zur Verfügung zu stellen. Der Mietpreis wird in dem Bereich [5 €, 50 €] liegen. Die angenommene Zahl an Nutzern liegt in dem Bereich [1, 20000].

Berechnung der Amortisationszeit Für den Umsatz ($=U$), die diese App abhängig von der Zeit in Monaten ($=t_m$), den Nutzern ($=N$) und von dem Mietpreis pro Monat ($=p_m$) erwirtschaften soll wird die Formel $U = p_m \cdot N \cdot t_m$ zugrunde gelegt. Es wird angenommen, dass die App nach einem Monat 10 Nutzer hat. Dieser Wert wird modellhaft als linear ansteigend mit der Zeit t_m angenommen, d.h. $N = 10 \cdot t_m$. Daraus lässt eine Formel zur Berechnung der Amortisationszeit (t_m^A) ableiten:

$$t_m^A = \frac{U}{p_m \cdot N}, \quad \text{wobei } U = 1844,30 \text{ €}, \quad N = 10 \cdot t_m^A \quad (4)$$

$$\Rightarrow t_m^A = \sqrt{\frac{1844,30 \text{ €}}{10 \cdot p_m}} = 13,58 \cdot p_m^{-\frac{1}{2}}. \quad (5)$$

Aus dem Funktionsgraphen aus [Abbildung 4](#) lässt sich die Amortisationszeit t_m^A für jeden möglichen Mietpreis pro Monat ablesen, wobei die Unsicherheit über das Ergebnis bei kleinen Monatsmietpreisen am geringsten ist, so liegt beispielsweise die Amortisationszeit für 5 € Monatsmiete bei ca. 6 Monaten, was durchaus im akzeptablen Budget- bzw. Investitionsrahmen der Firma ist.

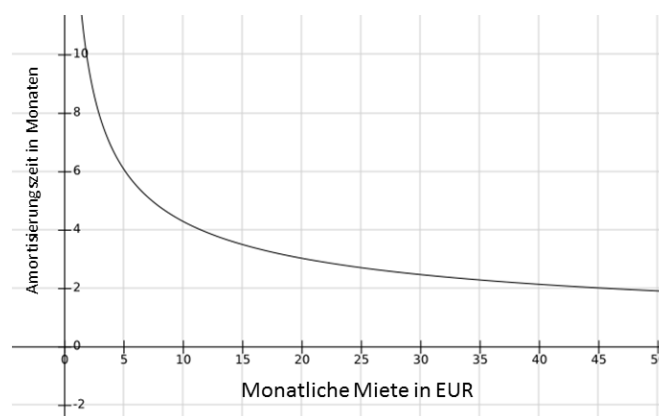


Abbildung 4: Amortisationszeit pro monatlicher Miete

3 Analysephase

3.3 Anwendungsfälle

Die zu entwickelnde App soll die folgenden Anwendungsfälle für den Nutzer/Merchant dargestellt in einem Anwendungsfalldiagramm mindestens realisiert haben.

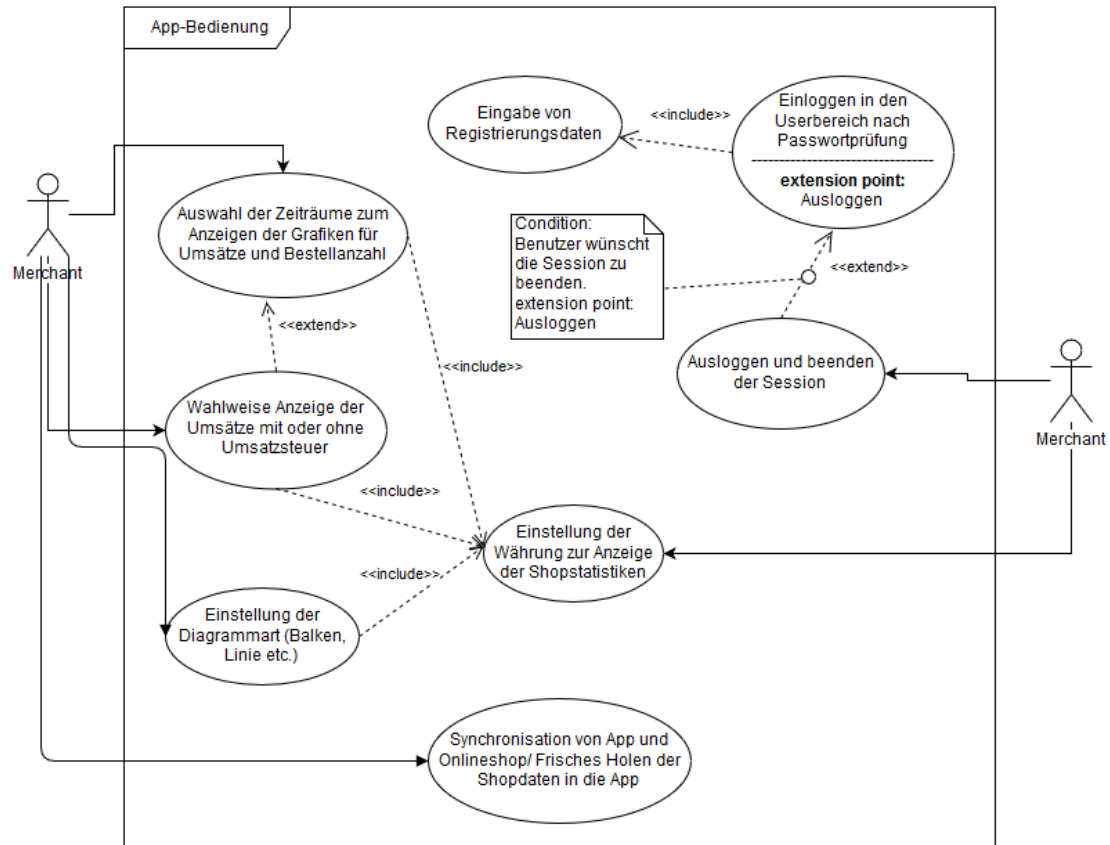


Abbildung 5: Anwendungsfalldiagramm für die Statistik-App

3.4 Qualitätsanforderungen

Die App soll intuitiv vom Benutzer bedien- bzw. navigierbar sein. Alle dargestellten Statistiken und Berichte sollen selbsterklärend sein. Die App wird als Single-Page-Anwendung programmiert, d.h. bei keiner Aktion des Benutzer soll ein Seitenneuladen ausgelöst werden. Bei Aktualisierung der Daten durch Datenbankabfragen, wenn beispielsweise die Währung oder der Zeitraum geändert wird, sollen alle betroffenen Diagramme und Berichte nahezu gleichzeitig mit den neuen Daten aktualisiert werden. Bei der Datensynchronisation soll der Nutzer eine Rückmeldung darüber erhalten, ob die Synchronisation noch im Gange ist oder nicht oder gar fehlgeschlagen ist. Zusätzlich soll für jeden Nutzer beim Login eine einstündig gültige Session gestartet werden,

3 Analysephase

sodass die URL für diesen Client einzigartig ist und bei keinem anderen Clientcomputer oder Browser gleichzeitig verwendet werden kann. Das Design sollte für den Nutzer ansprechend sein und die App sollte in Ansätzen „responsive“ sein, um auch auf mobilen Geräten benutzbar sein zu können. Die App soll frontend- und backendseitig modular aufgebaut sein, so dass Fehlerbeseitigung oder Modifizierungen schnell und übersichtlich möglich sind und sich konsistent in das Gesamtprogramm einfügen. Der Programmcode sollte strukturiert nach gängigen oder firminternen Code-Conventions sein und an allen wichtigen Stellen Kommentare für Entwickler enthalten.

3.5 Lastenheft/Fachkonzept

Die Anwendung muss folgende Anforderungen erfüllen:

1. Registrierungsmechanismus

- 1.1. Der Nutzer muss Vorname, Name, Email-Adresse, Passwort und seine ShopUrl zur Registrierung eingeben können.
- 1.2. Die Passworteingabe sollte blicksicher sein und verifiziert werden.
- 1.3. Nach der Registrierung bekommt man eine Erfolgsmeldung und wird auf die Loginseite weitergeleitet.

2. Login-Mechanismus

- 2.1. Die Logindaten sollen die Email-Adresse und das Passwort sein
- 2.2. Bei Falscheingabe von Email-Adresse oder Passwort soll eine entsprechende Rückmeldung erfolgen.

3. Logout-Mechanismus

- 3.1. Im Nutzerbereich der App soll eine Möglichkeit zum Beenden der Session/Logout geschaffen werden.
- 3.2. Beim Ausloggen soll die Session beendet und eine Weiterleitung auf den Loginbereich erfolgen.

4. Navigation im Nutzerbereich

- 4.1. Alle wesentlichen Benutzerinteraktionen der App sollen nur über ein einziges Navigationspanel entweder an der Seite oder oben als Leiste durchführbar sein

3 Analysephase

- 4.2. Die Navigation sollte einen logischen und intuitiven Aufbau haben und sollte vom Design her einheitlich sein
- 5. Graphische Darstellungen statistischer Auswertungen
 - 5.1. Die Umsätze und Bestellungen im Onlineshop sollen wahlweise als Balken- oder Liniendiagramme darstellbar sein.
 - 5.2. Für die Diagramme soll der Zeitraum auswählbar über die Navigationsleiste sein. Die auswählbaren Zeiträume sollen mindestens die einzelnen Kalenderwochen, Monate und Jahre umfassen.
 - 5.3. Es soll möglich sein, die Diagramme wahlweise aufgrund von Brutto - oder Nettoumsätzen anzuzeigen.
- 6. Währungsfilter
 - 6.1. Die Anzeige der Statistiken und Berichte sollte abhängig von der ausgewählten Währung sein, da im Onlineshop mehrere Währungen eingestellt sein können.
 - 6.2. Hierzu muss eine Auswahlmöglichkeit über einen Währungsfilter geschaffen werden.
- 7. Datensynchronisation
 - 7.1. Es soll die Möglichkeit für den Nutzer geschaffen werden über einen „Button“ die Datensynchronisation zwischen seinem Shop und der App auszulösen, falls der Nutzer eine neuerliche Synchronisation für notwendig hält.
 - 7.2. Der Nutzer soll eine Rückmeldung über den Prozess der Synchronisation bekommen und eine entsprechende Mitteilung falls die Synchronisation fehlgeschlagen ist.
 - 7.3. Die Daten sollen über REST-Calls unter Verwendung der ePages REST-API vom Onlineshop angefordert und in einer MySQL-Datenbank abgespeichert werden.
- 8. Datenbank
 - 8.1. Zum Aktualisieren der App-Statistiken durch Benutzerinteraktionen abseits der Datensynchronisation sollen keine REST-Calls mehr ausgelöst werden, um die ePages-REST-API nicht zu sehr zu belasten. Vielmehr sollen hier die benötigten Daten aus der Datenbank abgefragt werden.
 - 8.2. Es soll ein Datenbankmodell in der 3. Normalform entworfen werden.

4 Entwurfsphase

9. Umsatzstatistik nach Bundesland

9.1. Es soll über ein Tortendiagramm möglich sein, die Umsatzverteilung pro Bundesland angezeigt zu bekommen.

9.2. Außerdem sollen Shopkunden dabei berücksichtigt werden, denen kein Bundesland zugewiesen ist oder die im Ausland wohnen.

10. Darstellung umsatzstärkster Kunden

10.1. Die App soll in tabellarischen Stil die Kunden geordnet nach Umsatz auflisten mit Namen, Email-Adresse, Gesamtumsatz, Gesamtzahl an Bestellungen und letztem Bestelldatum.

10.2. Die Erstellung der Tabelle sollte dynamisch erfolgen.

11. Design

11.1. Die gesamte App sollte zumindest ein grundlegendes responsives Verhalten aufgrund des 12er-Grid-Systems zeigen.

4 Entwurfsphase

4.1 Zielplattform

Die Statistik-App wird als Web-Applikation entwickelt und sollte damit lauffähig in allen gängigen Internetbrowsern eines jeden Heim-PCs oder Laptops sein, mindestens jedoch im Internet-Explorer, Firefox und Google-Chrome. Die App ist über eine feste URL auf dem „uberspace.de“-Server erreichbar. Die Wahl auf „uberspace“ als Host fiel aufgrund der geringen monatlichen Mietkosten für das Hosting und die Servernutzung und der Unterstützung von MySQL. Die verwendeten Programmiersprachen umfassen Javascript und PHP. Dies ist der Tatsache geschuldet, dass viele meiner Kollegen und mein Ausbilder ein großes Know-How in diesen Sprachen haben und mich so effektiv unterstützen können. Des Weiteren ist für mich als Auszubildender in der Frontendentwicklung Javascript die Programmiersprache, in der ich mich am besten auskenne. Die Wahl auf PHP als serverseitige Sprache fiel aufgrund der großen Community im Vergleich zu NodeJS und der einfachen Datenbankanbindung an eine MySQL-Datenbank. NodeJS wird zwar für Single-Page-Anwendungen empfohlen, jedoch sollte man für rechenintensive Anwendungen, wie die Statistik-App eine ist, dann doch eher auf PHP zurückgreifen. Zudem gibt es von ePages auf der App-Entwicklungs-Webseite Tutorials zur App-Entwicklung in PHP und auch bereits eine

4 Entwurfsphase

eigene PHP-Klasse zur Anforderung von REST-Daten. Prinzipiell hätte man für diese Anwendung auch MongoDB als Datenbanksystem nehmen können. Das Vorhaben wurde aber mangels Erfahrung im Umgang damit verworfen.

4.2 Architekturdesign

Die Frontendarchitektur der App wird unter Verwendung des Javascript-Frameworks ReactJS realisiert. ReactJS hat Ähnlichkeiten zu traditionellen **MVC-Frameworks**!, fokussiert sich jedoch besonders auf die Idee eine Website aus einzelnen interaktiven Komponenten zusammengebaut zu betrachten, die voneinander abhängig sein können (über Eltern-Kind-Beziehungen) und immer einen definierten (Daten-) Zustand haben, der sich dynamisch in Echtzeit ohne erneutes Laden der gesamten Website ändern kann. Die Statistik-App soll genau diesen Ansprüchen entsprechen, da die einzelnen Diagramme, Tabellen und Interaktionsmöglichkeiten jeweils als eigenständige Komponenten bzw. Module aufgefasst werden können, die alle einen vordefinierten Zustand haben, der sich durch Interaktionen des Nutzers ändern kann. Dadurch fiel die Wahl schnell auf ReactJS als modernes Framework mit groß gewachsener Community und dessen Fokus auf zustandsbehaftete Anwendungen. Zudem wird es bereits innerhalb ePages für das neue Softwareprodukt verwendet, sodass auch hier schon ein hilfreiches Know-How vorhanden ist. Nicht zuletzt erleichtert ReactJS die modularisierte Programmierung mit Javascript, durch Einführung von Komponenten die alle von einer Basiskomponente abgeleitet sind. Außerdem unterstützt ReactJS die Verwendung von ECMAScript 6, wodurch nun auch mit Javascript eine vorgegaukelte objektorientierte Programmierung möglich ist. Für die graphischen Darstellungen wird die Javascript-Bibliothek C3.js verwendet, welche kostenlos ist und von der extrem mächtigen D3.js-Bibliothek abgeleitet ist. C3.js bietet gerade den nötigen Umfang an Einstellungen, Darstellungsmöglichkeiten und Beispielen für die Statistik-App. Es gibt zwar wesentlich bessere und umfangreichere Bibliotheken für graphische Darstellungen, die meisten davon sind jedoch mindestens ab der kommerziellen Nutzung der App kostenpflichtig. Serverseitig fiel die Entscheidung auf das PHP-Slim-Framework auf Anraten von Kollegen. Slim ist extra für Webanwendungen entwickelt worden, die mit vielen REST-Requests hantieren müssen und bietet einen schnellen und einfachen Einstieg in diese Grundfunktionalitäten, auf die es hier serverseitig für die App ankommt. Die Kommunikation zwischen Frontend- und Backendskripten findet mittels AJAX statt. Um die App ansatzweise „responsive“ zu gestalten, wird das CSS-Framework Materialize ⁸ verwendet. Es bietet darüber hinaus vorgefertigte Designs

⁸<http://materializecss.com/>

4 Entwurfsphase

und Javascript-Interaktionen für Navigationsleisten, Menüs, Buttons etc., welche in der App dann auch verwendet werden.

4.3 Entwurf der Benutzeroberfläche

4.4 Datenmodell

Das verwendete Datenmodell ist in [Abbildung 6](#) dargestellt. Dort sind insbesondere die Beziehungen und Kardinalitäten zwischen den Entitäten Shop, Kunde, Bestellung, Artikel, Kundenadresse usw. dargestellt.

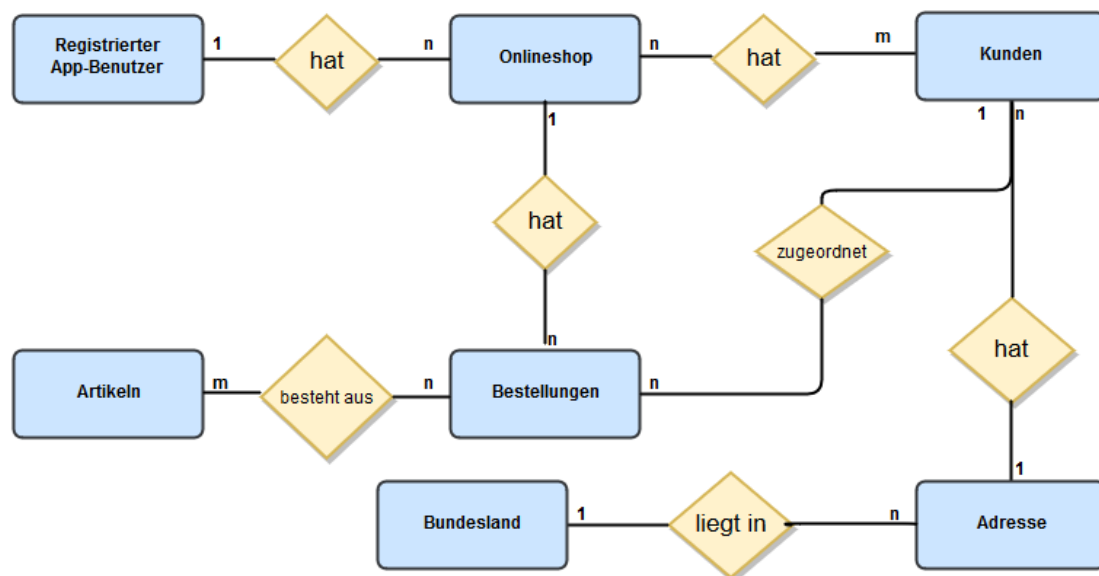


Abbildung 6: ER-Modell: Beziehungen und Kardinalitäten zwischen den Entitäten

4.5 Geschäftslogik

Der Kern der Geschäftslogik liegt in den Programmabläufen der Frontendskripte, die abhängig von den Benutzerinteraktionen mit der App beim Client in den einzelnen React-Komponenten ablaufen und gegebenenfalls AJAX-POST-Request zur (Daten-) Zustandsaktualisierung an den Server senden, bei dem Skripte für entsprechende Datenbankabfragen ablaufen. Exemplarisch sei hier die geplante Logik des Synchronisationsprozesses zwischen dem Onlineshop und der App anhand eines Sequenzdiagramms in Anhang [A.5: Sequenzdiagramm zum Synchronisationsprozess](#) auf Seite [v](#) dargelegt. Der Prozess soll durch ein Klickereignis auf den SYNC-Button in der Menüleiste ausgelöst werden, infolgedessen wird ein AJAX-Request an den Server gestellt,

4 Entwurfsphase

der sich wiederum per REST-calls die aktuellen Shopdaten holt und in die Datenbank einträgt.

Alle weiteren Klickereignisse, die der Nutzer über das Menü auslösen kann, sollen dann wenn notwendig nur noch Datenbankabfragen auslösen und keine weiteren REST-calls, um die REST-Schnittstelle zu schonen und die Abfrage- und Verarbeitungsprozesse zu beschleunigen.

Logik der Frontendarchitektur Die Statistik-App wird als Single-Page-Anwendung entwickelt, d.h. an keiner Stelle der App wird ein Neuladen der Seite ausgelöst. Dies wird durch React-Komponenten realisiert, die über Eltern-Kind-Beziehungen miteinander verknüpft sind. Dabei besitzt jede Komponente einen (Daten-)Zustand, der konstant oder veränderlich sein kann je nachdem, ob Daten aktualisiert werden oder nicht. Die Aktualisierung des Datenzustands findet ausschließlich asynchron statt, um während dieses Vorgangs die Benutzbarkeit der App nicht einzuschränken. Zusätzlich zum Zustand können jeder Kindkomponente von der Elternkomponente sog. „props“ (properties/Eigenschaften) übergeben werden ähnlich zu dem Vererbungsprozess in der objektorientierten Programmierung bei abgeleiteten Klassen mit dem Unterschied, dass die Kindkomponente nicht automatisch alle Zustandsattribute und Methoden der Elternkomponente erbt, sondern nur jene, die ihr direkt zur Verwendung zugewiesen bzw. weitergereicht wurden, d.h. die übergebenen props können sowohl Zustandsattribute wie auch Methoden der Elternkomponente sein.

Der architektonische Frontendaufbau über die React-Komponenten und deren Abhängigkeiten zueinander ist in Anhang [A.6: React-Komponenten-Architektur und Abhängigkeiten](#) auf Seite [vi](#) dargestellt. Ähnlich zu einem UML-Klassendiagramm zeigen die Pfeile in Richtung der Elternkomponente, von der „geerbt“ wird. Die Basiskomponente, von der alle anderen Komponenten aufgerufen werden, ist die **RoutedContent**-Komponente, in der über die React-Router-Komponente/Bibliothek die URL-Routen zu den anderen Basis-App-Komponenten definiert werden. Jede Komponente in der Abbildung besitzt einen Eintrag **state attributes**, der alle Zustandsdaten, falls vorhanden, auflistet. Diese Daten definieren den Zustand der Komponente. Zusätzlich werden alle in den Komponenten implementierten Methoden aufgelistet, die als props auch an Kindkomponenten weitergereicht werden können. Falls eine Komponente auch props übergeben bekommt, so werden diese zusätzlich auch noch aufgelistet.

5 Implementierungsphase

4.6 Maßnahmen zur Qualitätssicherung

Um die Umsetzung aller Qualitätsanforderungen aus 3.4 abzusichern, werden ständig händische Entwicklertests in verschiedenen Browsern durchgeführt, die darauf ausgerichtet sind, die Gesamtfunktionalität der App bei jeder neuen Implementierung oder Code-Änderung nicht zu gefährden. Zuletzt wird ein erfahrener teaminterner Mitarbeiter aus der QA-Abteilung damit beauftragt alle genannten Qualitätsanforderungen durchzutesten. Automatische Tests sind für dieses Projekt nicht vorgesehen.

5 Implementierungsphase

5.1 Verwendete Bibliotheken und Tools

Der Einstiegspunkt für die gesamte App ist die *index.php*-Datei zu sehen im Anhang A.7: [Einstiegspunkt für die Single-Page-App](#) auf Seite vii. Die verwendeten Javascript-Dateien für die App umfassen die JQuery-Bibliothek, die darauf aufbauende Materialize-Bibliothek als CSS-Framework und die Haupt-Javascript-Datei *main.js*, welche im *build*-Ordner liegt. Diese Datei enthält die gesamte ReactJS-Logik und die gebauten Komponenten. Erzeugt wurde sie durch einen Gulp-Task. Gulp ⁹ ist ein Task Runner, der häufig in Verbindung mit ReactJS eingesetzt wird. Alle benötigten Bibliotheken liegen lokal vor und sind damit unabhängig von einer vorhandenen Netzwerkverbindung.

5.2 Implementierung der Datenstrukturen

Ausgangspunkt hierfür ist das Datenmodell aus 4.4, welches Anlass zur Erstellung der in Anhang A.4: [Datenbankmodell für die MySQL-Datenbanktabellen](#) auf Seite iv aufgeführten miteinander in Verbindung stehenden Tabellen hat. Primär- und Fremdschlüssel sind entsprechend gekennzeichnet. Man beachte, dass zwischen dem Onlineshop und dem Kunden eine m:n Beziehung besteht, da ein Kunde in verschiedenen Onlineshops einkaufen kann, sodass hier eine Zwischentabelle vonnöten ist. Die Tabelle für die Artikel/Produkte wird nur pro forma angelegt, da diese während der Dauer des Projekts noch nicht mit Daten gefüllt werden kann, sodass auch keine Beziehungen zu den anderen Tabellen hergestellt werden können. Die Tabelle *tbl_states* gibt Auskunft darüber, ob und in welchem Bundesland der Kunde wohnhaft ist. Die Anbindung an

⁹<http://gulpjs.com>

5 Implementierungsphase

den Shop geschieht über das Attribut `shop_url` und dem besonders geheimzuhaltenden `access_token` aus der Tabelle `tbl_shop`, der während des Registrierungsprozesses von `epages` erzeugt und automatisch in der App-Datenbank abgelegt wird.

Die Erstellung der Tabellen wird unter Verwendung des Datenbank-Management-Systems *phpMyAdmin* ¹⁰ durchgeführt, was sehr einfach und intuitiv zu bedienen ist und daher auch für die Verwaltung der Tabellen und deren Daten verwendet wird. Eine Screenshot von den in *phpMyAdmin* angelegten Tabellen ist im Anhang [A.3: Ausschnitt aus der Tabellenstruktur in phpMyAdmin](#) auf Seite [iii](#) zu sehen.

5.3 Implementierung der Benutzeroberfläche

Loginbereich Für die Benutzeroberfläche des Login- und Registrierungsbereichs wird sich an diverser anderer Apps orientiert. Dabei fiel auch die Wahl auf die zu verwendende Schriftart „Dosis:700“, die in vielen modernen Apps zur Anwendung kommt. Wie in [Abbildung 7](#) zu erkennen sind alle Texte in Deutsch verfasst. Dies wird innerhalb

Abbildung 7: Design und Aufbau des Loginbereichs

dieses Projekt für die gesamte App gelten. Über eine Lokalisierung und Unterstützung mehrerer Sprachen kann man sich abseits dieses Projektes später Gedanken machen.

¹⁰<https://www.phpmyadmin.net/>

5 Implementierungsphase

Header im Nutzerbereich Die Implementierung des Headers im Nutzerbereich nach erfolgreichem Einloggen in die App ist in [Abbildung 8](#) zu sehen. Alle Interaktionen des

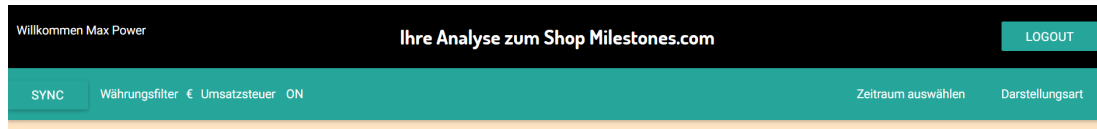


Abbildung 8: Design und Aufbau des Headers im Benutzerbereich

Benutzers mit der App sollen über die Menüleiste über davon zugängliche Dropdown-Felder oder Buttons erfolgen wie in [Abbildung 9](#) zu erkennen. Mittig oben erscheint der Shopname des Nutzers, links oben eine Willkommensnachricht und rechts oben die Möglichkeit zum Ausloggen.

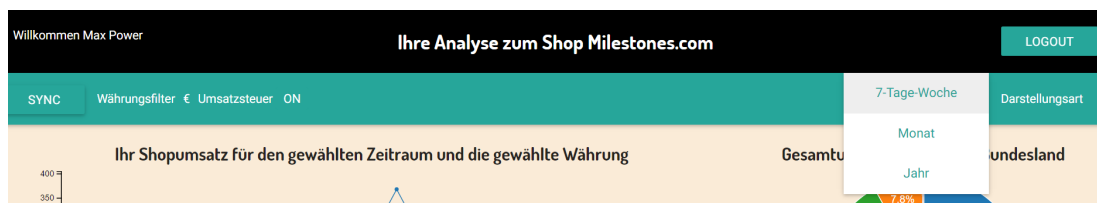


Abbildung 9: Beispiel eines Dropdown-Feldes nach einem Klickereignis

Anzeige der Statistiken Im Nutzerbereich werden dem Shopbesitzer insgesamt drei Diagramme angezeigt, zwei sind in [Abbildung 11](#) zu sehen. Die Balkendiagramme

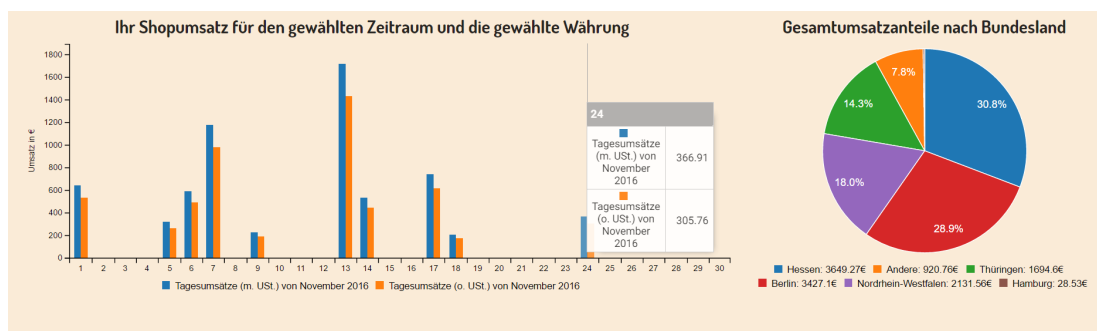


Abbildung 10: Diagramme für statistische Auswertungen

stellen den Umsatz für die ausgewählte Währung pro Tag, Woche, Monat oder Jahr dar je nach Auswahl des Nutzers. Das Tortendiagramm schlüsselt den Gesamtumsatz nach Bundesländer auf, wobei alle ausländischen oder nicht zugeordneten Umsätze unter die Kategorie „Andere“ fallen. Ein drittes hier nicht sichtbares Diagramm stellt die Anzahl der Bestellungen pro gewähltem Zeitraum dar und zeigt zusätzlich an, ob diese schon bezahlt sind oder nicht.

5 Implementierungsphase

Des Weiteren wird weiter unten eine Tabelle angezeigt, die die zehn umsatzstärksten Kunden sortiert nach Umsatz auflistet.

Top Ten der umsatzstärksten Kunden für die ausgewählte Währung				
Platz	Kunde (Vorname, Name, Email)	Gesamtumsatz in €	Bestellungen	Datum letzter Bestellung
1.	Jayrod Jason: j-rod@gmx.de	3649.27	18	24.11.2016
2.	Rayjay Jayray: rayjay@herr-der-mails.de	3427.1	15	24.11.2016
3.	Moritz Meyer: h.ditze@epages.com	2131.56	10	01.11.2016
4.	Max Power: shergt@gmx.de	1520.69	7	17.11.2016
5.	Max Mustermann: m.mustermann@epages.de	665.42	2	28.06.2009
6.	Olga Olga: olgaolga@epages.com	173.91	1	20.10.2016
7.	Nikola Tesla: nikola@tesla.de	173.91	1	18.11.2016
8.	John Doe: j.doe@epages.de	81.43	1	29.06.2009
9.	Johann von Goethe: johann@goethe.com	28.53	1	24.10.2016

Abbildung 11: Top-Ten-Tabelle der umsatzstärksten Kunden

5.4 Implementierung der Geschäftslogik

Exemplarisch wird hier die Implementierung des Synchronisationsprozesses aus Anhang [A.5: Sequenzdiagramm zum Synchronisationsprozess](#) auf Seite [v](#) aufgeführt. Der Code dazu ist im Anhang zu finden. Nach einem Klick auf den SYNC-Button wird zunächst die Funktion `loadSalesData()` aufgerufen, siehe Anhang [A.8: Clientseitige Abhandlung der Synchronisation](#) auf Seite [vii](#). Diese löst einen AJAX-Request an den Server aus. Dort wird die Funktion `handleLoadShopData()` aufgerufen, siehe Anhang [A.9: Serverseitige Abhandlung der Synchronisation](#) auf Seite [viii](#), welche REST-calls zum Shop des App-Nutzers abgesetzt, um alle relevanten Shop- und Bestelldaten zu holen, die dann in die Datenbank eingefügt werden. Anschließend wird bei erfolgreicher Ausführung des AJAX-Requests in der `done()`-Methode die Funktion `loadCurrentSales(event, month, year, tax, paidOrders, currency, currencySymbol)` aufgerufen, welche sich über mehrere AJAX-Requests alle aktuellen Daten für die Appstatistiken aus der Datenbank beschafft und diese an die Komponenten zur Aktualisierung ihrer Datenzustände und zwecks Neuladens weiterreicht.

5.5 Implementierung des Session-Managements

Bei erfolgreichem Einloggen des Nutzers wird eine Weiterleitungs-URL aus einem Cookie und einem Serversessiontoken zusammengebaut, wie in [Abbildung 12](#) zu sehen. Der Cookie ist acht Stunden lang gültig. Auf der App-Komponente angelangt,

6 Abnahmephase

wird der Sessiontoken auf dem Server nochmal gegengeprüft. Erst nach erfolgreicher Prüfung wird die App-Komponente geladen. Beim Ausloggen wird der Sessiontoken

```
30 $.ajax({
31   url: "../server_files/public/index.php/login",
32   type: "POST",
33   data: login
34 }).done((data) => {
35   console.log('rawlogindata: ', data);
36   var parsedData = JSON.parse(data),
37       d = new Date();
38
39   d.setTime(d.getTime() + 480*60*1000);
40
41   console.log('login: ', parsedData);
42
43   if (Number(parsedData[0])) {
44     const firstname = parsedData[1][0].firstname,
45         lastname = parsedData[1][0].lastname,
46         token = parsedData[2],
47         fulltoken = firstname + ':' + lastname + ':' + token;
48
49     document.cookie = token + ';path=/;expires=' + d.toGMTString() + ';max-age=' + 480*60 + ';';
50     console.log(fulltoken);
51     this.context.router.push('/userarea/' + fulltoken);
52   } else {
53     $('#idLoginError').css("display", "block");
54   }
55 });
```

Abbildung 12: Cookiesetzung in der Login-Komponente

auf dem Server gelöscht und der Nutzer wird auf die Login-Komponente umgeleitet. Damit wird die Session-URL ungültig gemacht.

6 Abnahmephase

6.0.1 CodeReview

Das Code-Review hat mein Ausbilder durchgeführt. Dabei wurden folgende Verbesserungsvorschläge gemacht, die aus Zeitgründen noch nicht umgesetzt werden konnten:

- Verbessertes Session-Handling:
Die Cookie-Ablaufzeit sollte sich bei Interaktion mit der App erneuern und synchron zur Server-Session sein. Nach Ablauf des Cookies sollte der Nutzer automatisch eine Meldung erhalten, dass er ausgeloggt wurde.
- Verbesserte Modularisierung:
Die AJAX-Requests zur Aktualisierung des Zustands der Kindkomponenten sollten nicht Teil der App/Eltern-Komponente sein, sondern Teil der einzelnen Komponenten, welche über definierte Routen zum Aktualisieren verleitet werden,

6 Abnahmephase

falls das entsprechende Ereignis dazu in der Eltern-Komponente ausgelöst wurde.

- **Verbesserte Performance:**
Solange keine Daten in der Datenbank geändert wurden, reicht es die Daten einmalig vom Server zu holen und auf dem Client zwischenzuspeichern. Alle Abfragen passieren dann clientseitig, entlasten den Server und sollten die Performance verbessern.

6.0.2 QA-Test

Um die implementierten Funktionalitäten zu testen, hat sich eine teaminterne Kollegin aus dem QA-Bereich dazu bereit erklärt die App auf Nutzertauglichkeit/Usability zu testen. Aus Termingründen war der Test erst recht spät gegen Ende des Projekts möglich, sodass die gefundenen Mängel nur aufgenommen, aber während der Anfertigung und Fertigstellung der Projektdokumentation nicht behoben werden konnten. Der Test hat folgende Mängel zu Tage gefördert:

1. **Wechsel der Währung:**
Es gibt ein Aktualisierungsproblem - Die alte Top-Ten-Liste ist nach dem Wechsel noch angezeigt.
2. **Shopumsatz und Umsatzsteuer ON/OFF:**
Da die Umsatzsteuer ein durchlaufender Posten ist, ist die Anzeige mit Steuer nicht sehr sinnvoll. Der Schalter kann dann auch entfallen.
3. **Bestellung nach Eingangsdatum - Skalierung der Anzahl der Bestellungen:**
Nur ganze Zahlen sollten angezeigt werden. Ist die Menge sehr klein, werden Zehntel abgebildet.
4. **Zeitraum auswählen: -> Monat:**
Durch die unterschiedliche Länge der Monatsnamen hüpfte beim Betätigen des Rückwärtspfeiles das Monats- und das Jahresfeld - Besser ist hier mit einer festen Breite für das Monatsfeld zu arbeiten.
5. **Darstellungsart wechseln:**
Die Einstellung „davon bezahlte Rechnungen anzeigen“ ist Off, d.h. wird beim Wechsel der Darstellungsart nicht berücksichtigt. Diese werden mit angezeigt.
6. **DropDown Felder für Währungsauswahl, Zeitraum und Darstellungsart:**
Die Auswahlfelder sollten unterhalb der Menüleiste angezeigt werden.

7 Dokumentation

7. TopTen-Liste:

In der Umsatzspalte bitte alle Beträge mit zwei Stellen nach dem Komma anzeigen und die Tausenderstelle kennzeichnen.

8. Gesamtumsatzanteile nach Bundesland:

Gruppen, die einen Mindestprozentsatz nicht überschreiten (z.B. <0,5%), sollten in die Gruppe „Andere“ aufgenommen werden. (Hamburg mit 0,2% ist separiert dargestellt)

Eine Einführungs- oder Deploymentphase entfällt bei diesem Projekt, da noch viele Nachbesserungen und Ergänzungen durchgeführt werden müssen bevor das Produkt-Management der Integration der App im ePages-App-Store zustimmt.

7 Dokumentation

Neben der Projektdokumentation wurde auch eine Entwicklerdokumentation in Form von ausführlicher Kommentierung der Implementierung der Programme/Module verfasst. Eine Benutzerdokumentation oder Benutzerhandbuch kann aufgrund des unfertigen Zustands der App noch nicht angefertigt werden.

8 Fazit

8.1 Soll-/Ist-Vergleich

Abgesehen von den in [2.2](#) erwähnten Abweichungen, die fehlende Artikelstatistiken und multiple Shopregistrierungen betreffen wurden alle anderen Sollvorgaben erfüllt und die App läuft in allen angestrebten Anforderungen stabil und korrekt.

In dem jetzigen Zustand ist die App noch nicht „deploymentfähig“, weshalb sie auch dem Produkt-Management zur Sichtung noch nicht vorgelegt wurde. Erst nach dem Beheben der Mängel, die sich aus dem Code-Review und dem QA-Testprotokoll ergaben, kann das auch das Feedback vom Produkt-Management gesucht werden.

8 Fazit

Vergleich mit der Zeitplanung Summa summarum konnte der geplante Zeitrahmen für das Projekt eingehalten werden. Ein Vergleich ist im Anhang [A.2: Vergleich zwischen geplanter und tatsächlich gebrauchter Zeit](#) auf Seite [ii](#) zu finden. Die einzigen größeren Abweichungen betreffen einen wesentlichen Mehraufwand bei der Erstellung aller notwendigen php-Skripte und einem reduzierten Aufwand bei der Erstellungen der notwendigen Javascript-UI-Interaktionen. Dies ist der geringeren Erfahrung im Umgang mit PHP geschuldet.

8.2 Lessons Learned

Der Umgang mit neuen Frameworks (hier ReactJS) erfordert im Vorfeld ein hohes Maß an Recherche und Einarbeitung genauso wie die Verwendung eher unbekannter Programmiersprachen (hier PHP). Dies muss in der Zeitplanung eines Projekts unbedingt berücksichtigt werden, da der Mehraufwand, der daraus erwuchs, unterschätzt wurde. Außerdem muss mehr Augenmerk auf die „Definition of Ready“-Anforderungen gelegt werden bevor mit dem Projekt überhaupt begonnen werden kann, sodass Abweichungen wie in [2.2](#) nicht erst überraschend während des Projektdurchführung auftreten, sondern im Vorfeld als bekannte Hinderlichkeiten in die Projektplanung miteinfließen.

Des Weiteren wäre es im Nachhinein schlauer gewesen, das Feedback aus der QA-Abteilung schon viel früher noch während der Entwicklung zu suchen genauso wie das Code-Review. Das legt den Schluss nahe, dass das Wasserfallmodell als Vorgangsmodell für solch ein umfangreiches Projekt eher unpassend und entweder durch das Spiralmodell oder agile Entwicklungszyklen ersetzt werden sollte.

8.3 Ausblick

Die Entwicklung der App ist noch lange nicht abgeschlossen. Zunächst müssen die angesprochenen Mängel aus dem Code-Review und dem Testprotokoll der QA-Abteilung beseitigt werden. Danach müssen die fehlenden Anforderungen aus Abschnitt [2.2](#) noch implementiert werden. Daraus ergibt sich auch eine Änderung der Menüführung bei der Benutzung der App für den Nutzer, da so viele Statistiken nicht mehr auf einer Seite angezeigt werden können, sondern über das Menü dynamisch auf die Seite geladen werden sollten. Dabei sollte dann auch über Responsivität und die Darstellung der App auf mobilen Geräten nachgedacht und Nachbesserungen betrieben werden, da ePages mobile Geräte generell unterstützt. Schließlich müssen Bezahlmodalitäten

8 Fazit

bestimmt und eine Bezahlbindung (Kreditkarte, Überweisung, PayPal etc.) geschaffen werden. Parallel dazu müssen AGBs formuliert und das Impressum erstellt werden. Zum Schluss sollten auch Sicherkonzepte überprüft und angepasst werden.

Literaturverzeichnis

ISO/IEC 9126-1 2001

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

phpdoc.org 2010

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

Sensio Labs 2010

SENSIO LABS: *Symfony - Open-Source PHP Web Framework*. Version: 2010. <http://www.symfony-project.org/>, Abruf: 20.04.2010

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase	8 h
1. Analyse des Ist-Zustands	1 h
1.1. Studium des Blogeintrags zur gewünschten App und Ver- schriftlichung	1 h
2. Wirtschaftlichkeitsprüfung und Amortisationsrechnung	2 h
3. Erstellen eines „Use-Case“-Diagramms	2 h
4. Erstellung eines Lastenheftes	3 h
Entwurfsphase	6 h
1. Erstellung eines Pflichtenheftes	3 h
2. Auswahl eines geeigneten Designs	2 h
3. Erstellung eines UML -Klassendiagramms	1 h
Implementierungsphase	50 h
1. Einrichtung eines ePages-Developer-Shops für die App - Entwicklung	0,5 h
2. Einrichtung des Slim-Frameworks zur Anbindung an die ePa- ges REST-API	1,5 h
3. Implementierung des DOM -Gerüsts mit React.js	16 h
3.1. Routing der REST -Calls von Slim zu React	1 h
4. Datenbankerstellung	4 h
4.1. Erstellung der MySQL -Datenbank	1 h
4.2. Datenbankmodellerstellung zur Speicherung von Kunden- und Shopdaten	2 h
4.3. Umsetzung des Datenbankmodells	1 h
5. Implementierung serverseitiger php-Skripte	6 h
6. Implementierung der Javascript UI-Interaktionen	18 h
7. Tests der App-Funktionalitäten	4 h
7.1. Anlegen von Kunden- und Shopdaten im epages Devel- opershop	2 h
7.1. Durchführung der Test	2 h
Erstellen der Dokumentation	6 h
1. Erstellen der Projektdokumentation	4 h
2. Erstellen der Entwicklerdokumentation	2 h
Gesamt	70 h

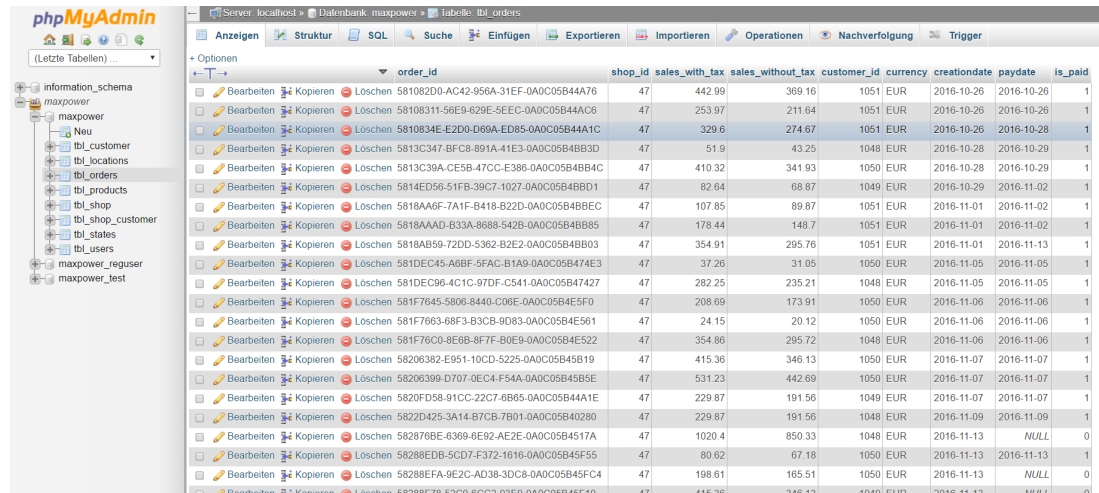
A Anhang

A.2 Vergleich zwischen geplanter und tatsächlich gebrauchter Zeit

Vorgang	Geplant	Tatsächlich	Differenz
Analysephase	8 h	7 h	-1 h
1. Analyse des Ist-Zustands	1 h	1 h	0 h
1.1. Studium des Blogeintrags zur gewünschten App und Verschriftlichung	1 h	1 h	1 h
2. Wirtschaftlichkeitsprüfung und Amortisationsrechnung	2 h	3 h	+ 1 h
3. Erstellen eines „Use-Case“-Diagramms	2 h	2 h	0 h
4. Erstellung eines Lastenheftes	3 h	1 h	-2 h
Entwurfsphase	6 h	6 h	0 h
1. Erstellung eines Pflichtenheftes	3 h	-	-
2. Auswahl eines geeigneten Designs	2 h	2 h	0 h
3. Erstellung eines UML -Klassendiagramms	1 h	4 h	+3 h
Implementierungsphase	50 h	50 h	0 h
1. Einrichtung eines ePages-Developer-Shops für die App -Entwicklung	0,5 h	0,5 h	0 h
2. Einrichtung des Slim-Frameworks zur Anbindung an die ePages REST-API	1,5 h	1,5 h	0 h
3. Implementierung des DOM -Gerüsts mit React.js	16 h	16 h	0 h
3.1. Routing der REST -Calls von Slim zu React	2 h	2 h	0 h
4. Datenbankerstellung	4 h	4 h	0 h
4.1. Erstellung der MySQL -Datenbank	1 h	1 h	0 h
4.2. Datenbankmodellerstellung zur Speicherung von Kunden- und Shopdaten	2 h	2 h	0 h
4.3. Umsetzung des Datenbankmodells	1 h	1 h	0 h
5. Implementierung serverseitiger php-Skripte	6 h	10 h	+4 h
6. Implementierung der Javascript UI-Interaktionen	18 h	12 h	-4 h
7. Tests der App-Funktionalitäten	4 h	5 h	+1 h
7.1. Anlegen von Kunden- und Shopdaten im epages Developershop	2 h	2 h	0 h
7.2. Durchführung der Tests	2 h	2 h	0 h
Erstellen der Dokumentation	6 h	6 h	0 h
1. Erstellen der Projektdokumentation	4 h	5 h	+1 h
2. Erstellen der Entwicklerdokumentation	2 h	2 h	0 h
Gesamt	70	70	0 h

A Anhang

A.3 Ausschnitt aus der Tabellenstruktur in phpMyAdmin

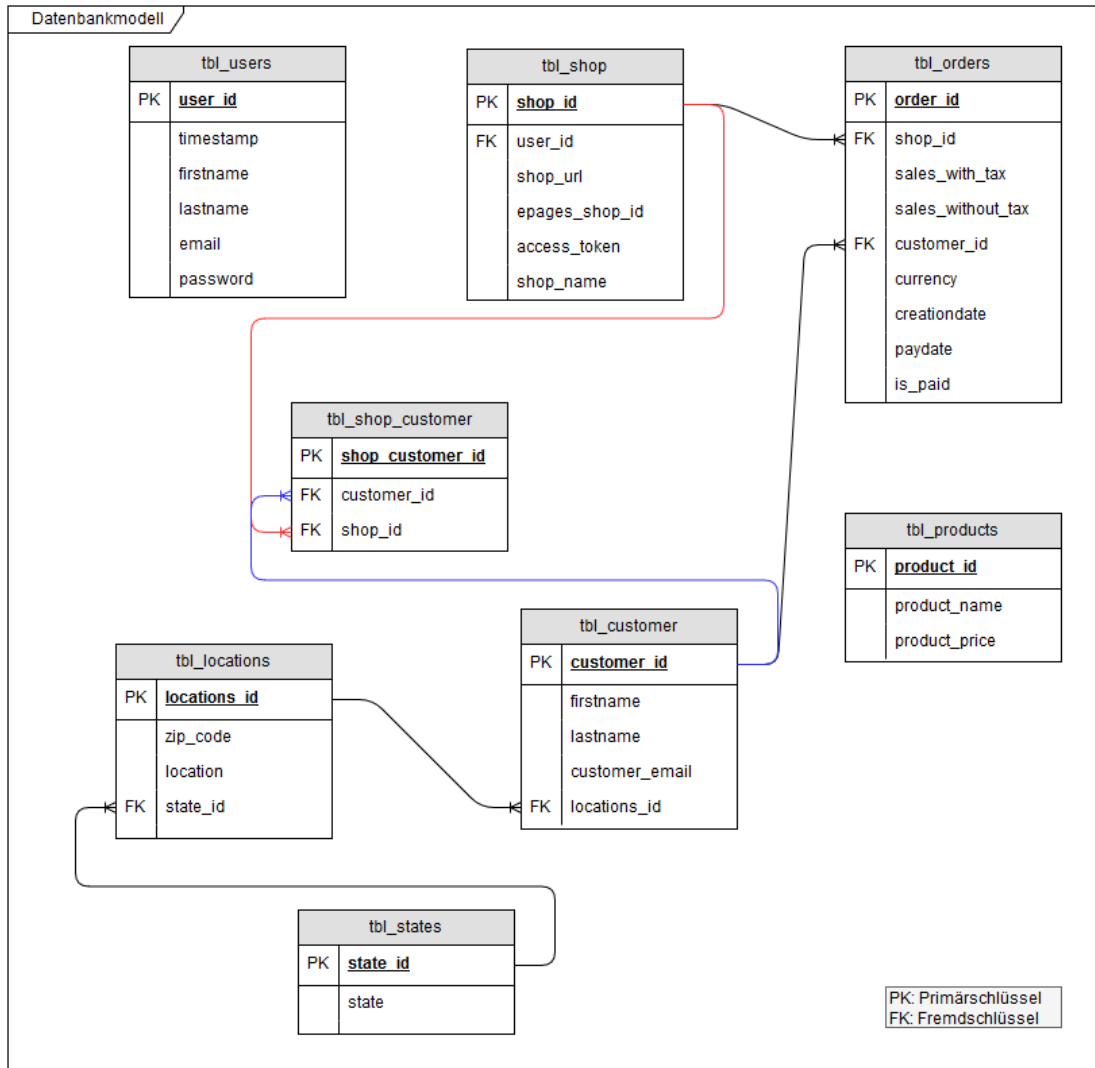


The screenshot shows the phpMyAdmin interface for a database named 'maxpower'. The left sidebar displays the database structure, including tables like 'tbl_customer', 'tbl_locations', 'tbl_products', 'tbl_shop_customer', 'tbl_states', 'tbl_users', 'tbl_orders', 'tbl_reguser', and 'tbl_test'. The main area shows the table 'tbl_orders' with its structure. The table has the following columns: 'order_id', 'shop_id', 'sales_with_tax', 'sales_without_tax', 'customer_id', 'currency', 'creationdate', 'paydate', and 'is_paid'. The table contains 20 rows of data, with the first row highlighted in blue. The 'order_id' column is the primary key.

order_id	shop_id	sales_with_tax	sales_without_tax	customer_id	currency	creationdate	paydate	is_paid
581082D0-AC42-956A-31EF-0A0C05B44A76	47	442.99	369.16	1051	EUR	2016-10-26	2016-10-26	1
58108311-56E9-629E-5EEC-0A0C05B44AC6	47	253.97	211.64	1051	EUR	2016-10-26	2016-10-26	1
5810834E-E2D0-D69A-ED85-0A0C05B44A1C	47	329.6	274.67	1051	EUR	2016-10-26	2016-10-28	1
5813C347-BFCB-891A-41E3-0A0C05B4BB3D	47	51.9	43.25	1048	EUR	2016-10-28	2016-10-29	1
5813C39A-CE5B-47CC-E386-0A0C05B4BB4C	47	410.32	341.93	1050	EUR	2016-10-28	2016-10-29	1
5814ED56-51FB-39C7-1027-0A0C05B4BBD1	47	82.64	68.87	1049	EUR	2016-10-29	2016-11-02	1
5818AA6F-7A1F-B418-B22D-0A0C05B4BBEC	47	107.85	89.87	1051	EUR	2016-11-01	2016-11-02	1
5818AAAD-B33A-8688-542B-0A0C05B4BB85	47	178.44	148.7	1051	EUR	2016-11-01	2016-11-02	1
5818AB59-72DD-5362-B2E2-0A0C05B4BB03	47	354.91	295.76	1051	EUR	2016-11-01	2016-11-13	1
581DEC45-A6BF-5FAC-B1A9-0A0C05B474E3	47	37.26	31.05	1050	EUR	2016-11-05	2016-11-05	1
581DEC96-4C1C-97DF-C541-0A0C05B47427	47	282.25	235.21	1048	EUR	2016-11-05	2016-11-05	1
581F7645-5806-8440-C06E-0A0C05B4E5F0	47	208.69	173.91	1050	EUR	2016-11-06	2016-11-06	1
581F7663-68F3-B3CB-9D83-0A0C05B4E561	47	24.15	20.12	1050	EUR	2016-11-06	2016-11-06	1
581F76C0-8E6B-8F7F-B0E9-0A0C05B4E522	47	354.86	295.72	1048	EUR	2016-11-06	2016-11-06	1
58206382-E951-10CD-5225-0A0C05B45B19	47	415.36	346.13	1050	EUR	2016-11-07	2016-11-07	1
58206399-D707-0EC4-F54A-0A0C05B45B5E	47	531.23	442.69	1050	EUR	2016-11-07	2016-11-07	1
5820FD58-91CC-22C7-6B85-0A0C05B44A1E	47	229.87	191.56	1049	EUR	2016-11-07	2016-11-07	1
5822D425-3A14-B7CB-7B01-0A0C05B40280	47	229.87	191.56	1048	EUR	2016-11-09	2016-11-09	1
582876BE-6309-6E92-AE2E-0A0C05B4517A	47	1020.4	850.33	1048	EUR	2016-11-13	NULL	0
58288EDB-5CD7-F372-1616-0A0C05B45F55	47	80.62	67.18	1050	EUR	2016-11-13	2016-11-13	1
58288EFA-9E2C-AD38-3DC8-0A0C05B45FC4	47	198.61	165.51	1050	EUR	2016-11-13	NULL	0
58388E78-82D0-6D63-63ED-0A0C05B45F4D	47	415.36	346.13	1048	EUR	2016-11-13	NULL	0

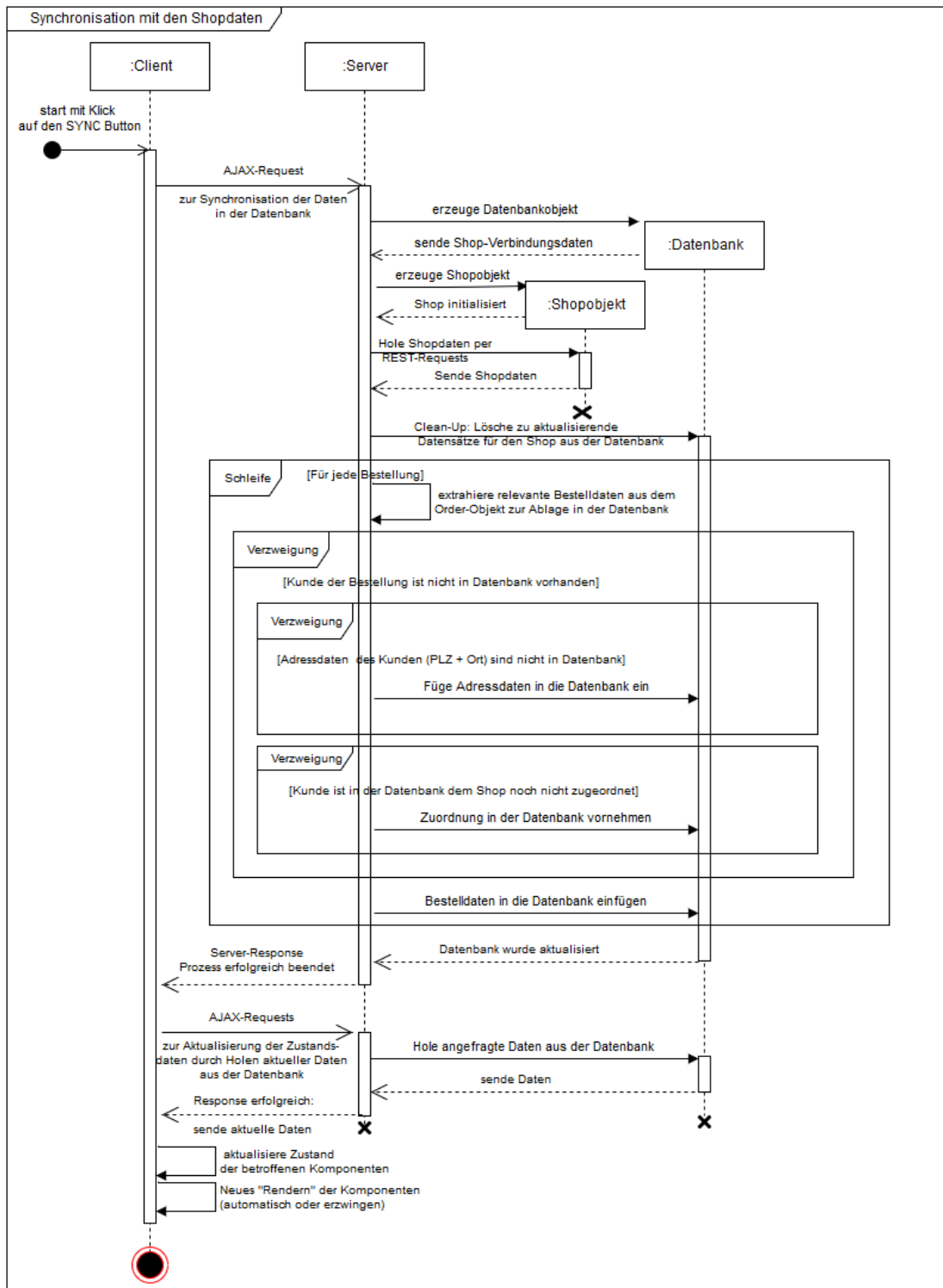
A Anhang

A.4 Datenbankmodell für die MySQL-Datenbanktabellen



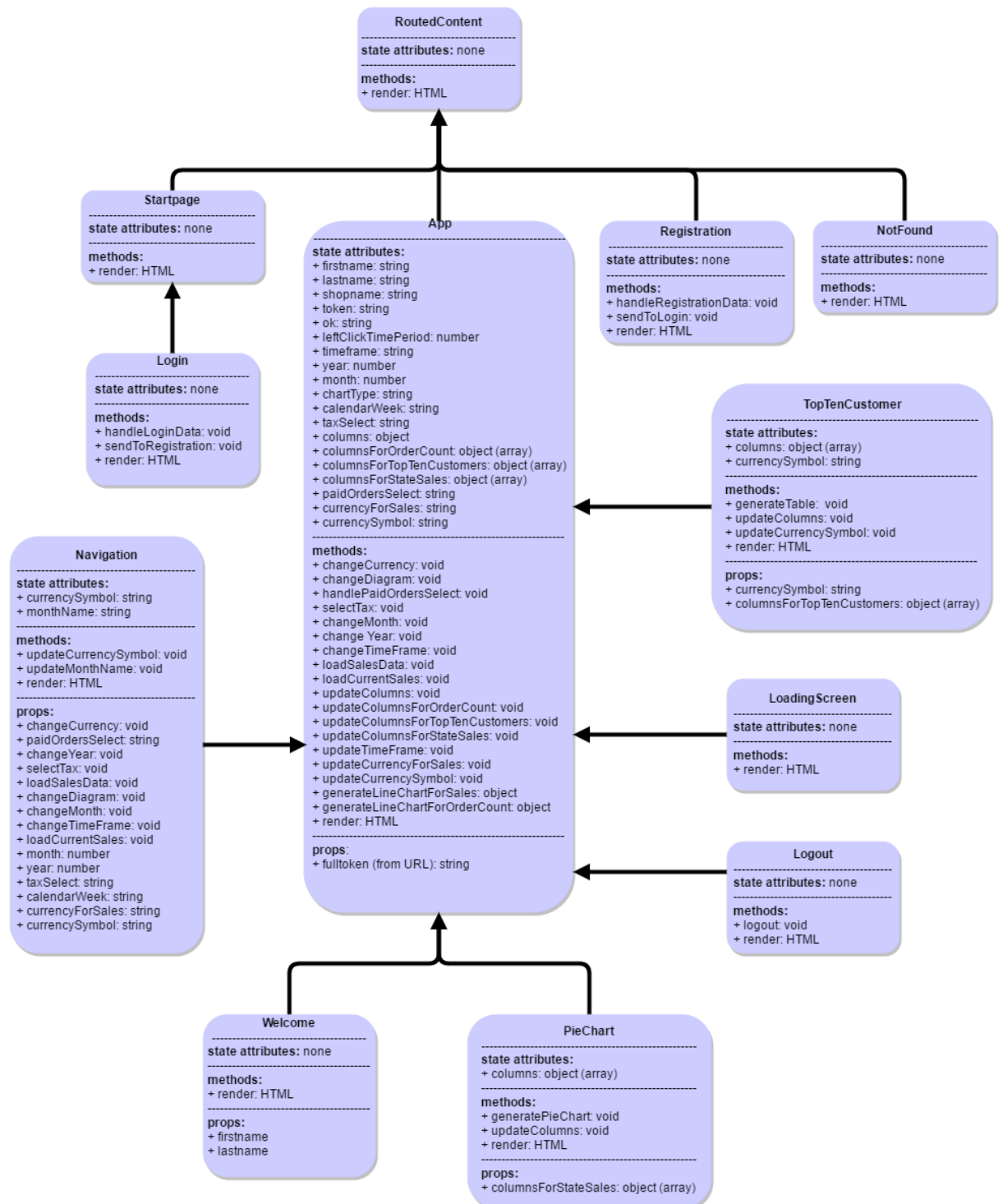
A Anhang

A.5 Sequenzdiagramm zum Synchronisationsprozess



A Anhang

A.6 React-Komponenten-Architektur und Abhängigkeiten



A Anhang

A.7 Einstiegspunkt für die Single-Page-App

```
1 <html>
2   <head>
3     <meta charset="UTF-8">
4     <title>StoreAnalyst</title>
5     <!--link to materialize library as the main css framework for the app-->
6     <link rel="stylesheet" href="../stat_app/bower_components/materialize/dist/css/materialize.css">
7     <link rel="stylesheet" href="build/css/c3.css">
8     <link rel="stylesheet" href="build/css/style.css">
9     <!--link to the primarily used font in the app-->
10    <link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Dosis:700">
11    <!--link to the font-awesome folder-->
12    <link rel="stylesheet" href="lib/font-awesome/css/Font-awesome.min.css">
13  </head>
14  <body>
15    <div id="main" class="non-footer">
16      <!-- This is where the main React app will go -->
17    </div>
18  </body>
19  <!--link to the jquery library folder-->
20  <script type="text/javascript" src="../stat_app/bower_components/jquery/dist/jquery.js"></script>
21  <!--link to the Javascript files needed for materialize-->
22  <script src="../stat_app/bower_components/materialize/dist/js/materialize.js"></script>
23  <!--the whole big Javascript file that the app uses to render out the react components and
24  their interactions build with gulp-->
25  <script src="./build/main.js"></script>
26 </html>
27
```

A.8 Clientseitige Abhandlung der Synchronisation

```
284
285 // synchronize data with the user's shop, fetch the new data from the server and trigger update of the app's state
286 loadSalesData() {
287   $.ajax({
288     url: "../server_files/public/index.php/allorders",
289     type: "POST",
290     data: ''
291   }).done((data) => {
292     console.log(data);
293
294     // define default/dummy state for the update function
295     var event = new CustomEvent(
296       "defaultEvent2",
297       {
298         detail: {},
299         bubbles: true,
300         cancelable: true
301       }
302     );
303
304     // trigger update of the app and of all child components
305     this.loadCurrentSales(event, this.state.month, this.state.year, this.state.taxSelect,
306     this.state.paidOrdersSelect, this.state.currencyForSales, this.state.currencySymbol);
307   });
308 }
309
```

A.9 Serverseitige Abhandlung der Synchronisation

```

1 <?php
2
3 function handleLoadShopData() {
4     // create database object
5     $db = new Database();
6     // get shop access data from database
7     $getShopDataRows = $db->getRows("SELECT tbl_shop.shop_url, tbl_shop.epages_shop_id, tbl_shop.access_token FROM (tbl_shop, tbl_users)
8                                     WHERE tbl_shop.user_id = tbl_users.user_id AND tbl_users.email = ?", [$SESSION['email']]);
9     // assign access data to variables
10    $devshopurl = $getShopDataRows[0]['shop_url'];
11    $epagesshopid = $getShopDataRows[0]['epages_shop_id'];
12    $access_token = $getShopDataRows[0]['access_token'];
13    // create shop object with access variables (REST-call)
14    $shop = new ep6\Shop($devshopurl, $epagesshopid, $access_token, true);
15    // create currency object for the shop
16    $shopCurrencies = new ep6\Currencies();
17    $shopCurrencyArray = $shopCurrencies->getItems();
18    // get shop data that fit to the access token
19    $getShopRow = $db->getRow("SELECT * FROM tbl_shop WHERE access_token = ?", [$access_token]);
20    // check if access token exist
21    if ($getShopRow) {
22        // delete all orders belonging to the shop and all shop-customer related info (not the customers!)
23        $db->deleteRow("DELETE tbl_orders FROM tbl_orders JOIN tbl_shop ON tbl_orders.shop_id = tbl_shop.shop_id
24                    WHERE tbl_shop.access_token = ?", [$access_token]);
25        $db->deleteRow("DELETE tbl_shop_customer FROM tbl_shop_customer JOIN tbl_customer
26                    ON tbl_shop_customer.customer_id = tbl_customer.customer_id
27                    JOIN tbl_shop ON tbl_shop_customer.shop_id = tbl_shop.shop_id WHERE tbl_shop.access_token = ?", [$access_token]);
28    }
29
30    // loop over all available currencies in the shop
31    foreach ($shopCurrencyArray as $shopCurrency) {
32        // set currency in the shop object
33        $shopCurrencies->setCurrency($shopCurrency);
34        // create order filter object
35        $filter = new ep6\OrderFilter();
36        // set displayed orders in the MBO (and in the shop object) to 100
37        $filter->setResultsPerPage(100);
38        // get all orders (max. 100) as an array
39        $fullOrders = $filter->getOrders();
40
41        $page = 1;
42        // now get ALL orders by joining all orders per page (while loop over the pages in the MBO where orders are present)
43        while (TRUE) {
44            $filter->setPage($page + 1);
45            $ordersPerPage = $filter->getOrders();
46            if (count($ordersPerPage)) {
47                $fullOrders = array_merge($fullOrders, $ordersPerPage);
48                $page++;
49            } else {
50                break;
51            }
52        }
53        // loop over all orders and extract valuable info for statistical analysis and insert them into the database
54        foreach ($fullOrders as $order) {
55            $isPaid = $order->isPaid();
56            $customer_email = $order->getBillingAddress()->getEmail();
57            $firstname = $order->getBillingAddress()->getFirstName();
58            $lastname = $order->getBillingAddress()->getLastName();
59            $customer_zip = $order->getBillingAddress()->getZipCode();
60            $customer_city = $order->getBillingAddress()->getCity();
61            $order_id = $order->getID();
62            $sales_with_tax = $order->getTotalPrice()->getAmount();
63            $sales_without_tax = $order->getTotalPriceWithoutTax()->getAmount();
64            $order_currency = $order->getTotalPrice()->getCurrency();
65            // get all customers from the database with the email address of the order
66            $getCustomerRow = $db->getRow("SELECT * FROM tbl_customer WHERE customer_email = ?", [$customer_email]);
67            // get all locations from database that have the same zip code and city as the customer of the order
68            $getAddressCustomerRow = $db->getRow("SELECT * FROM tbl_locations WHERE zip_code = ?
69                                            AND location = ?", [$customer_zip, $customer_city]);
70            // get all shop-customer relations from database with the email address from the shop's order
71            $getShopCustomerRow = $db->getRow("SELECT * FROM tbl_shop_customer JOIN tbl_customer
72                                            ON tbl_shop_customer.customer_id = tbl_customer.customer_id
73                                            JOIN tbl_shop ON tbl_shop_customer.shop_id = tbl_shop.shop_id
74                                            WHERE tbl_customer.customer_email = ? AND tbl_shop.access_token = ?", [$customer_email, $access_token]);
75
76            // check if customer exists in database
77            if (!$getCustomerRow) {
78                // check if address exists in database
79                if (!$getAddressCustomerRow) {
80                    // insert new address info into database
81                    $db->insertRow("INSERT INTO tbl_locations (zip_code, location) VALUES (?, ?)", [$customer_zip, $customer_city]);
82                }
83                // insert new customer into database
84                $db->insertRow("INSERT INTO tbl_customer (firstname, lastname, customer_email, locations_id) VALUES
85                            (?, ?, ?, (SELECT locations_id FROM tbl_locations WHERE zip_code = ? AND location = ?))",
86                            [$firstname, $lastname, $customer_email, $customer_zip, $customer_city]);
87            }
88            // check if shop-customer relation exists in database
89            if (!$getShopCustomerRow) {
90                // insert new customer data into database
91                $db->insertRow("INSERT INTO tbl_shop_customer (customer_id, shop_id) VALUES ((SELECT tbl_customer.customer_id
92                                                FROM tbl_customer WHERE customer_email = ?),
93                                                (SELECT tbl_shop.shop_id FROM tbl_shop WHERE tbl_shop.access_token = ?))", [$customer_email, $access_token]);
94            }
95            // assign state_id 17 to all locations that are not linked to a state
96            $db->updateRow("UPDATE tbl_locations SET state_id = ? WHERE state_id IS NULL", [17]);
97
98            // check if order is paid
99            if ($isPaid) {

```

A Anhang

```
96
97 // check if order is paid
98 if($isPaid) {
99     // extract pay date
100     $timestamp = $order->getPayDate()->getTimestamp();
101     $payDate = date("Y-m-d", $timestamp);
102 } else {
103     // set pay date to NULL
104     $payDate = NULL;
105 }
106
107 // extract the date the order was created in the shop
108 $cTimestamp = $order->getCreationDate()->getTimestamp();
109 $creationDate = date("Y-m-d", $cTimestamp);
110
111 // insert all relevant order info into the database
112 $db->insertRow("INSERT INTO tbl_orders (order_id, is_paid, shop_id, sales_with_tax, sales_without_tax, customer_id, currency,
113     paydate, creationdate)
114     VALUES (?, ?, (SELECT shop_id FROM tbl_shop WHERE access_token = ?), ?, ?, (SELECT customer_id FROM tbl_customer
115     WHERE customer_email = ?), ?, ?, ?)",
116     [$order_id, $isPaid, $access_token, $sales_with_tax, $sales_without_tax, $customer_email, $order_currency, $payDate,
117     $creationDate]);
118 }
119 }
```