

Abschlussprüfung Sommer 2016

Fachinformatiker Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

HBS Proxy

Entwickeln einer Serversoftware zur Reservierung von Hotelzimmern

Prüfungsbewerber:

Jan Erik Helmle

Maybachufer 6

12047 Berlin

jhelmle@gmx.de

tel.: 0163-8747711

Azubi-Ident-Nummer: 3600437

Kundennummer: 107/3600437

mobile-ad-media

Ausbildungsbetrieb:

mobile-ad-media GmbH

Scheringstraße 2

13355 Berlin



Prüfungsausschuss: FIAN 11

Abgabetermin 27.05.2016

Inhaltsverzeichnis

1.	Einleitung.....
1.1	Projektbeschreibung.....
1.2	Projektziel.....
1.3	Projektumfeld und projektbeteiligte Personen.....
1.4	Betriebliche Prozessschnittstellen.....
1.5	Projektbegründung.....
1.6	Ist-Analyse.....
2.	Projektplanung.....
2.1	Soll Konzept.....
2.2	Projektphasen und Zeitplanung.....
2.3	Eingesetzte Mittel.....
2.4	Projektkosten.....
3.	Analysephase.....
3.1	Analyse des grafischen Konzepts.....
3.2	Ermittlung der beteiligten Entitäten und deren Beziehungen.....
3.3	Analyse des Dialogs zwischen Proxy und Apps.....
4.	Entwurfsphase.....
4.1	Erstellung des UML – Sequenzdiagramms.....
4.2	Erstellen des UML – Klassendiagramms.....
4.3	Entwurf des Datenbankmodells.....
4.4	Definition der Qualitätssicherungsmaßnahmen.....
4.5	Definition der JSON Objekte.....
4.6	Definition der Endpunkte.....
5.	Realisierungsphase.....
5.1	Aufbau der Repositorystruktur und Installation Symfony3.....
5.2	Implementierung der Modelle.....
5.3	Implementierung der Steuerung und Abfragen.....
5.4	Implementierung der Testdaten.....
6.	Qualitätsmanagement.....
6.1	Fehlerbehebung.....
6.2	Codereview.....
7.	Einführungsphase.....
7.1	Übergabe der Software und Inbetriebnahme.....
8.	Projektergebnisse.....

8.1	Soll- Istvergleich.....
8.2	Fazit.....

Anhang

A.1	Detaillierte Zeitplanung.....
A.2	Kosten-Nutzen-Analyse.....
A.3	Sequenzdiagramm.....
A.4	Klassendiagramm (ohne Getter- und Settermethoden).....
A.5	Datenbankdiagramm.....
A.6	Ausschnitte des grafischen Konzepts.....

Abkürzungsverzeichnis

DBMS:	Database Management System - Datenbanksystem
DQL:	Doctrine Query Language – SQL ähnliche Abfragesprache
HBS:	Hotel-Buchungs-System – Softwaresystem der mobile-ad-media GmbH
HTTP:	Hypertext Transfer Protocol – Kommunikationsgrundlage des World-Wide-Web
IDE:	Integrated Development Environment – integrierte Entwicklungsumgebung
iOS:	Betriebssystem für mobile Endgeräte
JSON:	Java Script Object Notation – kompaktes Datenformat
MVC:	Model-View-Controller – Entwurfsmuster der strukturierten Softwareentwicklung
PHP:	PHP: Hypertext Processor – Skriptsprache
ORM:	Object Relational Mapper – Objektrelationale Abbildung
SQL:	Structured Query Language – Datenbanksprache
UML:	Unified Modeling Language – Vereinheitlichte Modellierungssprache
URL:	Uniform Resource Locator – einheitlicher Ressourcenzeiger

1 Einleitung

Im Folgenden wird der Verlauf der betrieblichen Projektarbeit des Autors im Rahmen seiner Ausbildung zum Fachinformatiker für Anwendungsentwicklung erläutert.

1.1 Projektbeschreibung

Die mobile-ad-media GmbH (MAM) entwickelt derzeit ein Softwaresystem zur Reservierung von Hotelzimmern. Das Gesamtsystem wird nach der Umsetzung aus einem zentralen Server (Proxy) und mehreren nativen und mobilen Anwendungen (Apps) bestehen, welche die derzeit gängigsten Plattformen für Mobiltelefone abdecken. Die Apps ermöglichen dem Nutzer eine flüssige Reservierung von Hotelzimmern, stellen in ansprechendem Design alle nötigen Informationen zur Verfügung und nehmen Eingaben der Nutzer entgegen.

Die Apps kommunizieren ausschließlich mit dem Proxy, welcher andererseits mit entsprechenden Datendiensten, wie etwa Channelmanagern verbunden wird. Aufgabe des Proxy ist die Bereitstellung einer einfachen und einheitlichen Schnittstelle für die Apps sowie die Pufferung von Daten um dem Nutzer eine zügige Reservierung zu ermöglichen.

Die Apps werden an die grafischen Wünsche der jeweiligen Hotels individuell angepasst (Corporate Identity) und richten sich vornehmlich an Stammkunden des Gästehauses.

1.2 Projektziel

Im Rahmen des hier beschriebenen Teilprojekts sollte das Kommunikationsprotokoll zwischen Apps und Proxy als Webservice entwickelt, dokumentiert und Proxyseitig implementiert werden. Grundlage der Arbeit ist ein grafisches Konzept für die mobilen Endgeräte. Eine beispielhafte Reservierung mit Testdaten sollte mit ausgehender E-Mail erfolgreich durchführbar sein. Der Webservice soll extern beherbergt werden um zuverlässig jederzeit verfügbar zu sein.

1.3 Projektumfeld und projektbeteiligte Personen

Die Arbeit wurde bei der mobile-ad-media GmbH, einer Agentur zur Entwicklung und Realisierung von Kommunikationskonzepten in Berlin durchgeführt. Die Agentur hat bereits zahlreiche Webprojekte für namhafte Kunden umgesetzt. Des Weiteren betreibt das Unternehmen eine Sportpartnersuche und sammelt derzeit Erfahrungen in der Entwicklung von Apps für mobile Endgeräte. Das Unternehmen pflegt Beziehungen zu selbstständigen Entwicklern und Grafikern, die je nach Auftragslage in die Projekte eingebunden werden können.

Die Geschäftsleitung wird von Herrn Schmitz unternommen, der Diplom-Informatiker Herr Thierfelder ist festangestellter Mitarbeiter von MAM und betreut das Praktikum des Autors. Das

grafische Konzept wurde von Herrn Mischka Bandur im Rahmen eines Praktikums angefertigt. Herr Jens Korth wird anschließend im Rahmen des Gesamtprojekts die iOS-Version der Apps entwickeln und stand als wertvoller Berater in Bezug auf das Kommunikationsprotokoll zur Verfügung. Herr Friedemar Blohm unternahm als externer Mitarbeiter einen nicht minder wertvollen sowie umfassenden Codereview. Dadurch konnte die Qualität des Quelltextes erheblich gesteigert werden.

Der Autor wurde durch Herrn Schmitz mit der Durchführung des Projekts formlos beauftragt.

1.4 Betriebliche Prozessschnittstellen

Das gesamte Reservierungssystem wird von Grund auf neu entwickelt, daher existieren noch keine Schnittstellen nach außen. Die Datenanbindung an externe Systeme ist Bestandteil des Gesamtprojekts und findet zeitlich hinter diesem Teilprojekt statt. Für Testzwecke werden manuell fiktive Daten, welche sich an kleineren Hotels orientieren, in den Quelltext mittels eigener Fixture-Klassen eingetragen.

1.5 Projektbegründung

Die MAM möchte künftig Hotels als Kunden gewinnen und eine individualisierte Buchungsapp zur Verfügung stellen. Die Geschäftsleitung der MAM verspricht sich von dem Gesamtprojekt die Gewinnung zahlreicher Hoteliere als Kunden und einen Einstieg in den Bereich der Reisetechologien. Der im Rahmen dieses Teilprojekts entwickelte Proxy wird zwischen den Apps und den externen Datendiensten geschaltet, von welchen der Proxy sich häufig ändernde Daten wie etwa Preise und Verfügbarkeiten von Hotelzimmern erhält und an welche der Proxy Reservierungsdaten weitergeben kann. Die Apps dienen lediglich der Präsentation der Daten und nehmen Nutzereingaben entgegen. Der Proxy versorgt die Apps mit Daten und nimmt die Eingaben der Nutzer entgegen. Die Entwicklung der Apps wird dadurch wesentlich vereinfacht.

1.6 Ist-Analyse

Grundlage des Teilprojekts ist ein grafisches Konzept der Apps in welcher der Ablauf der Reservierung als Bildfolge dargestellt wird. Das Konzept wurde mittels einer Bildbearbeitungssoftware erstellt und stellt die einzelnen Bildschirme der Apps dar. Der gesamte Dialog mit dem Nutzer wird hier im Ablauf dargestellt. Bisher bestehen noch keine Verbindungen zu den externen Datendiensten. Diese Schnittstellen werden in einem späteren Verlauf des Gesamtprojekts erarbeitet. Auch stehen noch keine Apps zur Verfügung, sie werden ebenfalls in einem späteren Zeitraum entwickelt.

2. Projektplanung

2.1 Soll Konzept

Nach Abschluss des Teilprojekts ist das Übertragungsprotokoll zwischen Proxy und App definiert worden und steht als Webservice seitens des Proxy zur Verfügung. Grundlage des Dialogs zwischen App und Proxy ist das HTTP-Protokoll über welches die Daten mittels JSON-Objekten übertragen werden. Es wurde festgelegt, daß die Menge an Dialogdaten möglichst gering ausfallen soll. Die Definition der JSON-Objekte für den Datenaustausch geschieht mittels JSON-Schema, einer Schemadefinitionssprache für JSON-Objekte.

Da in der derzeitigen frühen Phase des Gesamtprojekts noch keine Datenverbindung zu den externen Datendiensten ausgearbeitet wurde, ist darauf Wert gelegt worden die Datenbestände auch händisch einpflegen zu können. Für kleinere Hotels, welche Ihre Buchungen noch nicht vollständig rechnergestützt durchführen und deren Daten sich selten ändern, kann dies Anfangs nützlich sein. Allgemeine Datenbankwerkzeuge wie z. B. PHPMysqlAdmin oder MySQL Workbench können hierfür gut genutzt werden.

Die Serversoftware wurde mittels des Frameworks Symfony3, einer quelloffenen Software für MVC-Anwendungen ausgeführt, welche die Skriptsprache PHP nutzt. Das verwendete DBMS ist MySQL. Die Revisionierung der Software sollte mittels Git, einer verteilten Versionsverwaltungssoftware erfolgen. Die Proxysoftware wurde lokal auf einem handelsüblichen Desktop-PC mit Windows 10 als Betriebssystem entwickelt und für den Betrieb auf einen externen, in einem kommerziellen Rechenzentrum betriebenen Linux-Server überspielt.

2.2 Projektphasen und Zeitplanung

Die Projektphasen wurden zeitlich wie folgt gegliedert:

Projektphase:	Dauer:
I) Analysephase	8h
II) Entwurfsphase	11h
III) Realisierungsphase	30h
IV) Qualitätsmanagement	9h
V) Einführungsphase	2h
VI) Dokumentation	10h
Gesamt:	70h

Eine detailliertere Einteilung ist im Anhang zu finden.

2.3 Eingesetzte Mittel

Es wurde darauf geachtet, schon vorhandene Mittel sowie quelloffene Softwareprodukte einzusetzen.

Ein Büroarbeitsplatz mit einem PC für Microsoft Windows 10 als Betriebssystem stand mit entsprechender Peripherie wie Bildschirm etc. schon zur Verfügung. Hier wurden sämtliche Entwicklungsarbeiten und die Erstellung der Dokumentation durchgeführt. Die eingesetzten Softwareprodukte fielen in die Kategorie der quelloffenen Software, dadurch entstanden keine zusätzlichen Kosten.

Als IDE kam Netbeans v8.1 zum Einsatz, als Revisionierungssoftware Git v2.6. PHP als Skriptsprache ist mit Version 5.6 in das Teilprojekt eingebunden, MySQL mit Version 5.7, MySQL Workbench mit Version 6.3.

Die Dokumentation wurde mittels LibreOffice 5, die UML Diagramme mit Umbrello v2.18 erstellt. Kleinere Werkzeuge wie etwa PuTTY als Terminalemulator, FileZilla zum Datentransfer oder bash als Shell wurden ebenfalls eingesetzt.

Zum Testen der Webservices eignet sich das Werkzeug Postman, einer grafischen Anwendung die den Browser Chrome im Hintergrund nutzt.

Zur Validierung der JSON-Objekte kam ein webgestützter Onlinedienst (jsonlint.com) zum Einsatz. Bezüglich des Personals waren der Autor sowie Herr Blohm an dem Teilprojekt wirksam.

2.4 Projektkosten

Tabellarische Darstellung der Teilprojektkosten:

Mitarbeiter	Zeit	Stundensatz	Gesamt
Praktikant	67h	10.-€	670.-€
Externer Mitarbeiter	3h	50.-€	150.-€
			820.-€

Die Kosten des Teilprojekts sind ausschließlich Personalkosten, welche sich auf den Autor sowie den externen Sachverständigen Herrn Blohm, welcher den Codereview vornahm, verteilen.

3. Analysephase

In dieser Phase ging es im wesentlichen darum, das gegebene grafische Konzept genau unter die Lupe zu nehmen.

3.1 Analyse des grafischen Konzepts

Die Apps haben vom Aufbau her im oberen Bildschirmviertel das Hauptmenü, im zweiten Bildschirmviertel das Untermenü und in der unteren Hälfte ein Formular in welches der Nutzer seine Wünsche auswählen kann. Das Hauptmenü besteht aus den vier Menüpunkten Kalender (Auswahl von An- und Abreisedatum), Zimmerkategorien (Hauptprodukte), Verpflegung und Specials (zusätzliche Produkte) und Übersicht (Reservierungsdetails). Der Nutzer wandert während des Reservierungsvorgangs einmal von links nach rechts durch die vier Hauptmenüpunkte.

Es wurde zuerst eine Kategorisierung der Daten nach Änderungshäufigkeit vorgenommen. Hierbei wurde festgestellt, daß es Daten gibt welche sich nur sehr selten Ändern und somit als statisch angenommen werden.

Diese Daten sind unter anderem die angebotenen Produkte, also Zimmerkategorien sowie die Anzahl der jeweiligen Zimmer. Zu den optionalen Produkten zählen die Verpflegungen wie etwa Vollpension und Halbpension sowie die in einem eigenen Untermenüpunkt aufgeführten Specials wie zum Beispiel Sektf Frühstück oder eine Raftingtour. Diese Daten können beim Start der Apps vom Server abgefragt und auf den Apps gepuffert werden.

Zu den sich häufig ändernden Daten zählen die Preise und die Verfügbarkeiten der Produkte, wobei hier nur die Zimmer in der Verfügbarkeit begrenzt sind. Diese Daten werden erst nachdem der Nutzer im ersten Schritt seinen Reservierungszeitraum eingegeben hat, an die Apps übertragen um der Forderung nach möglichst wenig Datenverkehr zu entsprechen.

Nun verfügen die Apps über alle Daten die zum Befüllen des Warenkorbs durch den Nutzer erforderlich sind.

Nachdem der Nutzer seinen Warenkorb zusammengestellt hat, wird dieser zum Proxy gesendet und der Gesamtpreis berechnet. Dieser wird anschließend zu den Apps übertragen, woraufhin der Nutzer seinen Reservierungswunsch bestätigen und damit abschließen kann.

3.2 Ermittlung der beteiligten Entitäten und deren Beziehungen

Aus dem grafischen Konzept geht hervor, daß Produkte mehrere Preise haben können, die sich über die Zeit ändern. Daher wurde die Klasse Price implementiert welche mit der Klasse Product in einer n-zu-1 Beziehung steht. Somit können die Produkte mehrere Price-Objekte beinhalten, zu jedem Zeitpunkt genau einen.

Produkte gliedern sich auf in Zimmerkategorie (RoomType) und Zusatzprodukt (AdditionalProduct). Beides sind Spezialisierungen der abstrakten Oberklasse Product und gehen daher eine Vererbungsbeziehung ein. Hier fiel die Entscheidung auf die Verwendung des Begriffs der Zimmerkategorie und nicht des Zimmers selbst, da der Kunde nicht ein bestimmtes Zimmer

reservieren möchte, sondern eines aus einer gegebenen Kategorie auswählt.

Zusatzprodukte werden nochmals kategorisiert in Boarding (Verpflegung) sowie Specials. Dies war wichtig, da die Apps diese Informationen zur Darstellung auch der Untermenüs benötigen.

Es wurde weiterhin die Klasse Availability (Verfügbarkeit) erzeugt, um die Anzahl der zu einem Zeitpunkt erhältlichen Zimmer modellieren zu können. Der Nutzer kann aus einer Liste die Anzahl der Zimmer auswählen die er Reservieren möchte. In dem betreffenden Zeitraum nicht verfügbare, also schon reservierte Zimmer, werden entweder ausgegraut oder nicht angezeigt.

Es wurde außerdem die Klasse Hotel mit den relevanten Hoteldaten wie z. B. E-Mail-Adresse eingeführt.

Da der Nutzer während des Reservierungsvorgangs einen Warenkorb befüllt, wurde die Klasse Cart ermittelt. Ein Produkt, welches in den Warenkorb gelegt wird, wird zu einem Item. Im Warenkorb können mehrere Items liegen, es besteht eine 1-zu-n Beziehung zwischen den Entitäten. In den Warenkorb gelegte Zusatzprodukte werden jeweils einem Hauptprodukt zugeordnet.

3.3 Analyse des Dialogs zwischen Proxy und Apps

Durch die Trennung von Darstellung und Geschäftslogik durch Apps und Proxy, sowie der Forderung nach möglichst wenig Datenverkehr, konnte eine Dialogsequenz ermittelt werden, welche aus vier aufeinanderfolgenden Schritten besteht. Hierbei stellen die Apps anfragen an den Proxy, der in einem definierten Format antwortet.

Im ersten Schritt stellen die Apps dem Proxy die Frage nach den statischen Daten. Es werden von Proxy Produktdaten zurückgeliefert, welche unter Anderem die Produktkategorie, Produktbezeichnung, Anzeigetexte und Kapazitäten (wie viele Zimmer sind pro Kategorie vorhanden) beinhalten.

Nun wählt der Nutzer das gewünschte An- und Abreisedatum aus, worauf der Proxy nach Preisen und Verfügbarkeiten für diesen Zeitraum angefragt wird. Der Proxy antwortet auf diese Frage in einem definierten Format, die Produktbezeichnungen sind den Apps ja schon bekannt.

Im Folgeschritt stellt der Nutzer auf den Apps seinen individuellen Warenkorb zusammen. Dieser Warenkorb wird nun zum Proxy gesendet, welcher den Gesamtpreis ermittelt und das Ergebnis an die Apps überträgt. Dieser Schritt ist nötig, da eine Trennung von Darstellung und Geschäftslogik als Voraussetzung vereinbart wurde.

Der Nutzer bestätigt daraufhin den Gesamtpreis und übermittelt den Warenkorb nochmals an den Proxy, ergänzt um persönliche Daten des Nutzers.

Hiermit ist die Reservierung abgeschlossen, der Dialog zwischen Apps und Proxy beendet.

4. Entwurfsphase

In dieser Phase ging es um die Erstellung von Diagrammen, die als Kommunikationsgrundlage dienen, sowie um die Definition der JSON-Objektstruktur.

4.1 Erstellung des UML – Sequenzdiagramms

Um den Ablauf des Dialogs zwischen Apps und Proxy übersichtlich darzustellen, wurde mittels Umbrello ein Sequenzdiagramm erstellt.

4.2 Erstellen des UML – Klassendiagramms

Anschließend wurde das Klassendiagramm des Proxy erstellt. Hierbei wurde auf die Darstellung der Getter- und Settermethoden aus Gründen der Übersichtlichkeit verzichtet.

4.3 Entwurf des Datenbankmodells

Da die Anwendung mittels ORM (Doctrine v2.5) auf die Datenbank zugreift, war eine manuelle Erstellung der Datenbank nicht nötig. Die Struktur der Datenbank wurde mittels Annotationen im Quelltext der Modelle beschrieben und automatisch von Doctrine per Kommandozeilenschnittstelle erzeugt. Die Datenbank dient somit der Speicherung von Objekten.

4.4 Definition der Qualitätssicherungsmaßnahmen

Um die definierten JSON-Objekte auf Korrektheit zu prüfen, wurde festgelegt einen Online-Validator einzusetzen. Die erzeugten JSON-Objekte sollten hierzu manuell auf JSON-Konformität geprüft und gegen die definierten JSON-Schemata geprüft werden.

4.5 Definition der JSON Objekte

Folgende JSON-Objekte wurden definiert:

Dialogschritt	Richtung	Inhalt	JSON-Objekt Nr.
1	Apps ← Server	Statische Produktdaten	1
2	Apps ← Server	Preise und Verfügbarkeiten der Produkte	2
3	Apps → Server	Warenkorb	3
3	Apps ← Server	Gesamtpreis	4
4	Apps → Server	Warenkorb + Nutzerdaten	5

Hierbei war es wichtig die Datenstruktur möglichst einfach zu halten und valides JSON zu erzeugen. Es wurde eine JSON-Schema-Definition der fünf Objekte vorgenommen.

4.6 Definition der Endpunkte

Aus der Analyse des Dialogs zwischen Apps und Proxy gehen vier Schritte hervor, die jeweils über eine eigene URL ansprechbar sind. In den ersten beiden Schritten möchten die Apps Informationen vom Proxy erhalten, somit ist die HTTP-Methode GET. In den beiden Folgeschritten werden die Daten des Warenkorbs im Rumpf der HTTP Nachricht als JSON-Objekt übertragen, die verwendete Methode ist daher POST.

Dialogschritt	Http-Methode	URL	Eingabedaten	Ausgabedaten
1	GET	/api/v1/initialdata	-	Statische Produktdaten
2	GET	/api/v1/pricesandavailabilities	An- und Abreisedatum (werden im Kopf übertragen)	Dynamische Produktdaten (Preise und Verfügbarkeiten)
3	POST	/api/v1/totalprice	Warenkorb	Gesamtpreis
4	POST	/api/v1/order	Warenkorb mit Nutzerdaten	O.K.

5 Realisierungsphase

5.1 Aufbau der Repositorystruktur und Installation Symfony3

Herr Thierfelder richtete einen Webordner auf einem extern betriebenen Server für das Projekt ein, welcher global via Internet zugänglich ist. Hierzu wurde dort ein leeres Git-Repository via Terminal angelegt und anschließend per clone-Operation auf den lokalen Entwicklungsrechner des Autors kopiert. Alle nun folgenden Schritte bezüglich des Erstellens der Anwendung wurden lokal durchgeführt. Der Entwicklungsprozess wurde regelmäßig in das lokale Repository übertragen und anschließend per push-Operation auf den externen Server geschoben.

Die Installation von Symfony3 erfolgte durch herunterladen des Pakets und anschließendem Erzeugen eines neuen Symfony3-Projekts mittels new-Anweisung im Terminal. Da Symfony einen eigenen kleinen Entwicklungsserver mitbringt ist hier keine zusätzliche Webserver-Software nötig. Sowohl der externe Server als auch der lokale Entwicklungsrechner verfügten bereits über MySQL Installationen, welche von Symfony3 als DBMS benötigt werden. Die Konfiguration von Symfony3 wurde auf die jeweilige MySQL Instanz vorgenommen.

5.2 Implementierung der Modelle

Die Modelle verzichten weitgehend auf Funktionalität und sind im Wesentlichen mit privaten Eigenschaften (Attribute) und öffentlichen Getter- und Settermethoden ausgestattet. Die Funktionalität findet weitgehend in der Steuerung (Controller) statt, für aufwändigere Datenbankabfragen wurden sogenannte Entity-Repositories verwendet.

Hierbei wurden die Attribute der Modelle mit entsprechenden Anmerkungen (Annotationen), welche sich an Doctrine richten, versehen. Auch die Klasse wurde mit Annotationen versehen um dem ORM die erforderlichen Informationen, wie die Klasse auf die Datenbank abgebildet werden soll, mitzugeben. Die erforderlichen Getter- und Settermethoden wurden mittels der Kommandozeilenschnittstelle von Doctrine mit der Konsole automatisch erzeugt.

Während der Erstellung der Modelle wurde darauf geachtet, daß bei one-to-many Assoziationen von Klassen die many-Seite ein entsprechendes Attribut vom Typ ArrayCollection enthalten muß. Ein Objekt dieser Klasse wird im Konstruktor erzeugt.

5.3 Implementierung der Steuerung und Abfragen

Es wurden insgesamt vier Controller implementiert, welche den jeweiligen URLs der definierten Endpoints entsprechen. Ein Controller erhält die Daten der Anfrage als Request-Objekt und gibt anschließend ein Response-Objekt an die anfragenden Apps zurück.

Der erste Controller, welcher die statischen Produktdaten liefert, bekommt von den Apps keine Informationen geliefert und besteht aus einer DQL-Abfrage die ein Array aller Produkte ausgibt. Daraufhin werden diese Produkte serialisiert, in eine JSON-Struktur gebracht und an die Apps ausgeliefert.

Der zweite Controller erhält von den Apps An- und Abreisedatum im Header der Nachricht. Nach erfolgreicher Validierung der eingegangenen Zeitstempel werden die zeitabhängigen Daten berechnet, serialisiert und als JSON-Objekt an die Apps übertragen. Elementare Abfragen wurden in Entity-Repository Klassen ausgelagert und funktional gekapselt. Diese Abfragen lassen sich wiederverwenden, auch wird der Controller dadurch übersichtlicher.

Im dritten Controller wird der von den Apps gelieferte Warenkorb als JSON-Objekt angenommen und gibt den Gesamtpreis zurück.

Der vierte und letzte Controller erhält ebenfalls den Warenkorb im Rumpf der Nachricht, ergänzt um persönliche Informationen des Nutzers, und versendet einerseits eine E-Mail mit den Reservierungsdaten an das Hotel und gibt andererseits einen Status an die Apps zurück. Alle Controller geben zusätzlich entsprechende HTTP Statusmeldungen an die Apps zurück.

5.4 Implementierung der Testdaten

Die Datenbank wurde mit Testdaten gefüllt die den Abbildungen des grafischen Konzepts entsprechen. In die Preistabelle wurden probenhalber Änderungen eingefügt um die Zeitabhängigkeit der Daten darstellen und die Korrektheit der Algorithmen darstellen und testen zu können. Das Symfony3 Framework bietet hierfür die sogenannten DataFixtures an, über welche die Daten via Kommandozeilenschnittstelle aus vorher erstellten Klassen in die Datenbank geladen werden können.

Die Daten entsprechen in etwa einem kleinen Hotel mit unterschiedlichen Zimmerkategorien, verschiedenen Verpflegungsarten und einigen Zusatzprodukten.

6. Qualitätsmanagement

Getestet wurden die Schnittstellen manuell mit dem Programm Postman. Die Validierung der JSON-Objekte wurde manuell mittels der Webseite <http://jsonlint.com/> vorgenommen. Eine Validierung der JSON-Objekte gegen die entsprechende Schemadefinition erfolgte manuell auf der Seite <http://jsonschema.net/>.

6.1 Fehlerbehebung

Das Teilprojekt wurde mittels der IDE Netbeans 8.1 erstellt. Da die IDE Rechtschreibfehler zeitnah meldet, ist eine syntaktisch fehlerfreie Erstellung des Quelltextes möglich gewesen.

Symfony3 bietet ebenfalls einen eingebauten Debugger, der Fehler und Unstimmigkeiten während der Ausführung meldet. Somit konnten Fehlerbehebungen während des Erstellens der Anwendung durchgeführt werden.

6.2 Codereview

In diesem Phasenabschnitt wurde Herr Blohm zur Durchsicht des Quelltextes mit einbezogen. Dadurch zeigten sich einige formelle Unstimmigkeiten, insbesondere bei den Datentypen der Datenbank sowie bei der Benennung von Funktionen und Variablen. Durch entsprechende Korrekturen wurde der Quelltext wesentlich lesbarer und übersichtlicher.

7. Einführungsphase

7.1 Übergabe der Software und Inbetriebnahme

Der Code wurde laufend auf den Produktivserver geschoben. Dadurch daß dort ein Git-Repository vorhanden ist, ist eine einfache push Operation seitens des Entwicklungsrechners erforderlich. Git

baut hierbei eine sichere ssh-Verbindung mit dem Server auf und aktualisiert das entfernte Repository auf dem Server. Da lediglich ein Entwicklungszweig im Repository vorhanden ist, wird gleichfalls das Verzeichnis mit allen Dateien aktualisiert.

Mittels eines Terminalemulators erfolgte anschließend eine Verbindung mit dem Server, ein kleines selbstgeschriebenes Skript wurde daraufhin ausgeführt und richtete die Datenbank mit den Testdaten ein.

Herr Korth, welcher die App für das Betriebssystem iOS entwickelt, wurde informiert und kann nun die Schnittstellen nutzen.

8. Projektergebnisse

Am Ende des Teilprojekts ist ein dauerhafter und zuverlässiger Serverbetrieb eingerichtet worden.

8.1 Soll- Istvergleich

Die oben genannten Ziele des Teilprojekts sind erreicht worden. Es wurde eine Analyse des Dialogs zwischen Apps und Proxy auf Grundlage des grafischen Konzepts vorgenommen, die Schnittstellen definiert und die Daten in einem JSON-Schema festgelegt. Der Proxy wurde mittels PHP-Framework Symfony3 implementiert. Die Anwendung läuft zuverlässig und performant auf einem extern betriebenen Server.

Es ergaben sich Differenzen in der Zeitplanung des Projektablaufs gegenüber den Angaben des Projektantrags. Der Zeitaufwand für die Erstellung der UML Diagramme wurde mit zusätzlichen zwei Stunden heraufgesetzt, der zeitliche Umfang zur Erstellung der Modelle wurde signifikant gekürzt, da die Funktionalitäten in die Steuerungen eingingen. Zusätzlich wurde Zeit für die Implementierung der Testdaten eingeplant.

Auch ergaben sich Abweichungen zwischen Zeitplanung und tatsächlich aufgewendeter Zeit. Die Modelle konnten etwas zügiger implementiert werden (-1h), die Dokumentation benötigte zusätzliche Zeit. (+2h).

8.2 Fazit

Der Autor hat durch das Teilprojekt wesentliche Erfahrungen auf dem Gebiet der systematischen Softwareentwicklung sammeln können. Hierbei wurde auch deutlich, daß eine sorgfältige Dokumentation und Zeitplanung wichtige Komponenten bei der erfolgreichen Durchführung von Softwareprojekten ist.

Während der Umsetzung des Teilprojekts wurden online-Dokumente rege genutzt. Insbesondere sei hier auf die hervorragenden Seiten des Doctrine-Projekts (<http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/>) und Symfony

(<https://symfony.com/doc/current/book/index.html>) verwiesen.

Es stellten sich auch Kenntnislücken des Autors heraus, insbesondere auf dem Gebiet der Softwaretechnik.

Die Vermutung, in der vorangegangenen Schulphase der Ausbildung alles Wissen vermittelt bekommen zu haben, um ein erfolgreicher Anwendungsentwickler zu sein, erwies sich als naiv.

Somit stellt die beständige Weiterbildung eine wichtige Komponente im Beruf des Anwendungsentwicklers dar.

Anhang

A.1 Detaillierte Zeitplanung

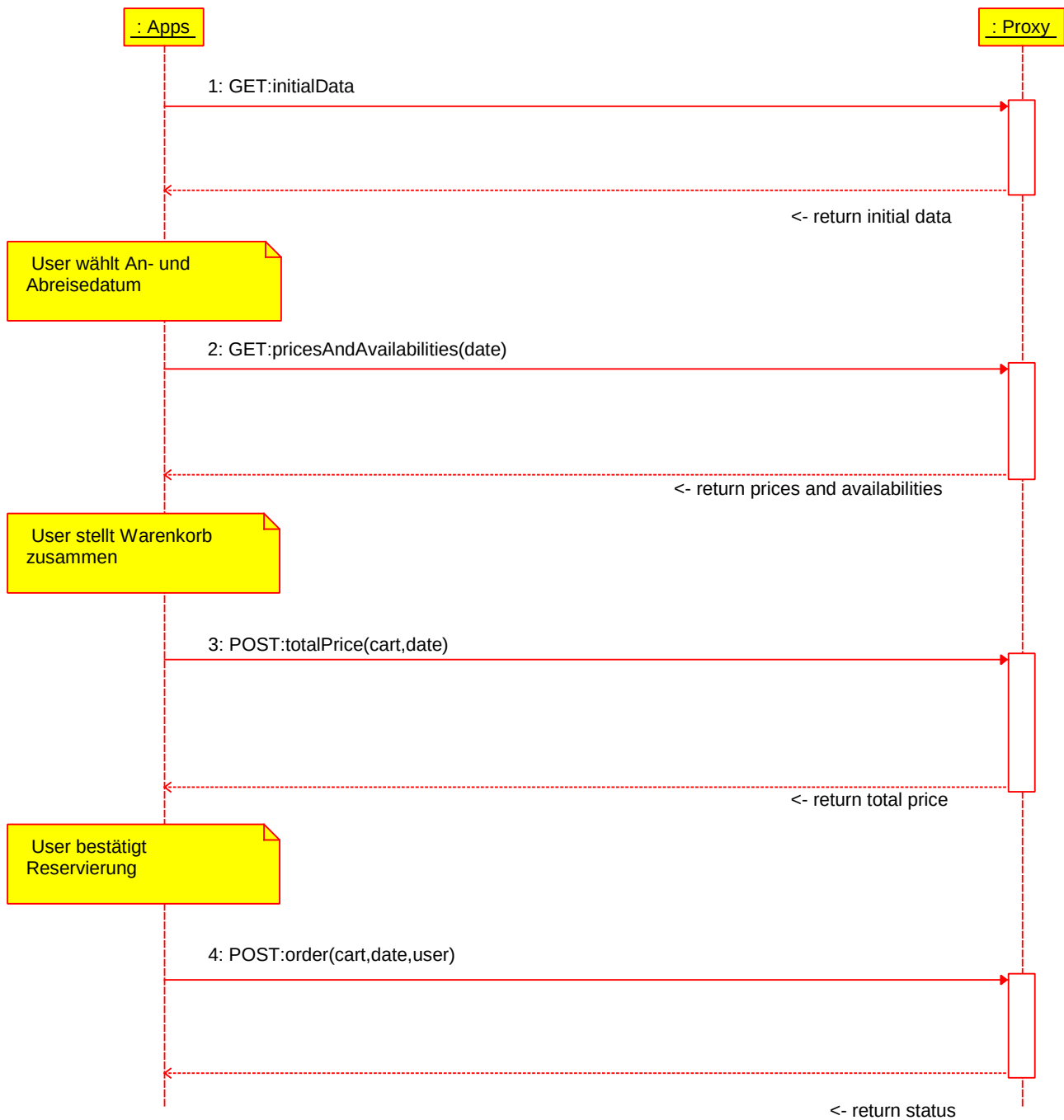
Phase	Tätigkeit	Dauer
I) Analysephase (8h)	Grafischen Entwurf Analysieren	2h
	Beteiligte Entitäten Ermitteln	2h
	Beziehung der Entitäten Ermitteln	2h
	Dialog zwischen Apps und Proxy Analysieren	2h
II) Entwurfsphase (11h)	Erstellen des UML - Sequenzdiagramms	2h
	Erstellen des UML - Klassendiagramms	2h
	Entwurf des Datenbankmodells	2h
	Definition QS - Maßnahmen	2h
	Definition der JSON Objekte	2h
	Definition der Endpoints	1h
III) Realisierungsphase (30h)	Aufbau der Repositorystruktur	1h
	Installation Symfony3	1h
	Implementierung der Modelle	7h
	Implementierung der Steuerung	14h
	Implementierung der Testdaten	7h
IV) Qualitätsmanagement (9h)	Manuelle Tests	3h
	Fehlerbehebung	3h
	Codereview	3h
V) Einführungsphase (2h)	Übergabe der Software und Inbetriebnahme	2h
VI) Dokumentation (10h)	Erstellen der Dokumentation	10h
	Gesamtzeit:	70h

A.2 Kosten-Nutzen-Analyse

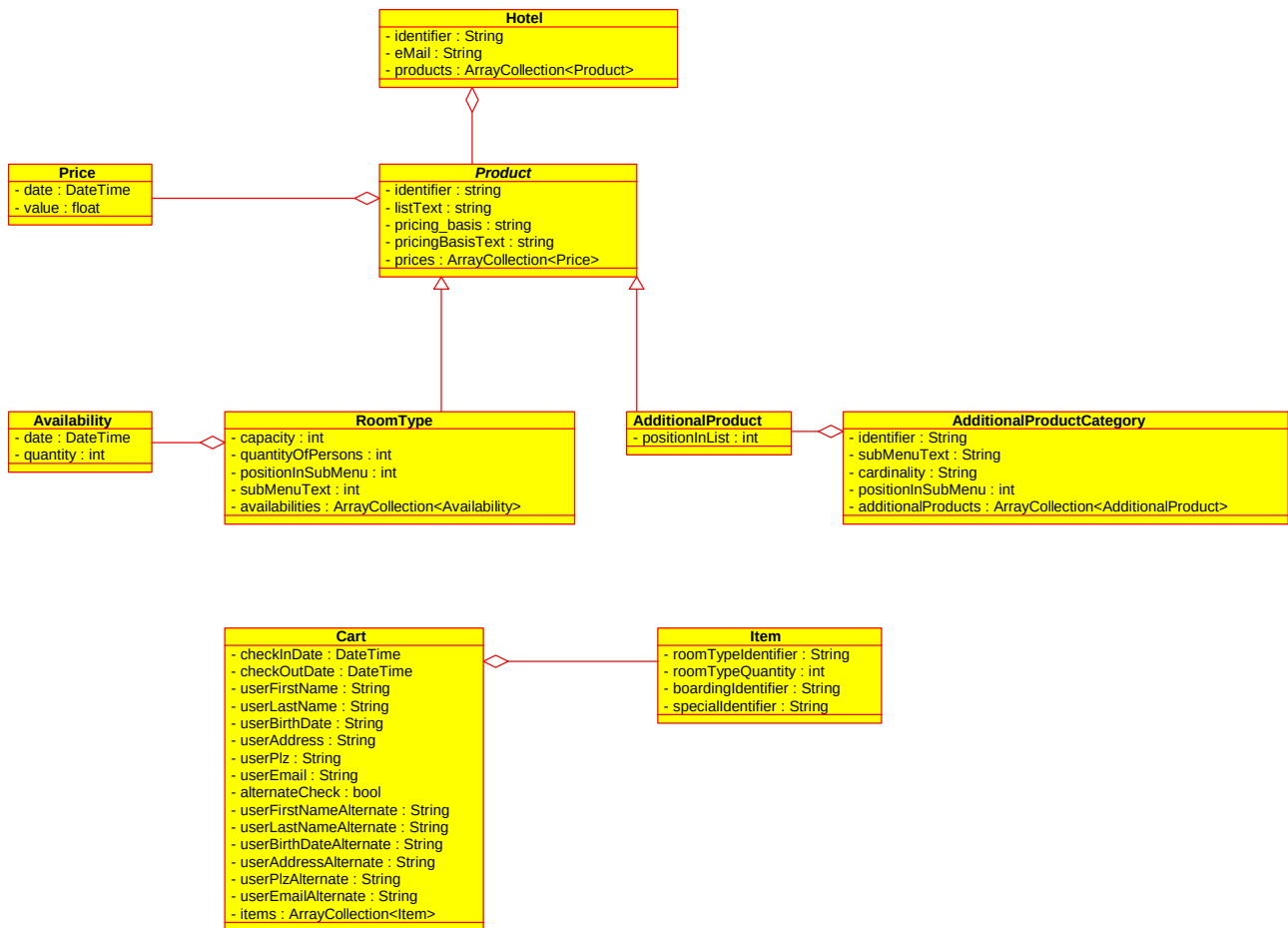
Da die Gesamtkosten des Projekts noch nicht geplant werden konnten kennen wir bisher nur die Kosten des Teilprojekts. Diese wurden in Abschnitt 2.4 dargestellt.

Die Geschäftsleitung der MAM erwartet durch den Vertrieb der Apps einen monatlichen Umsatz von ca. 50 Euro pro Monat und Kunde.

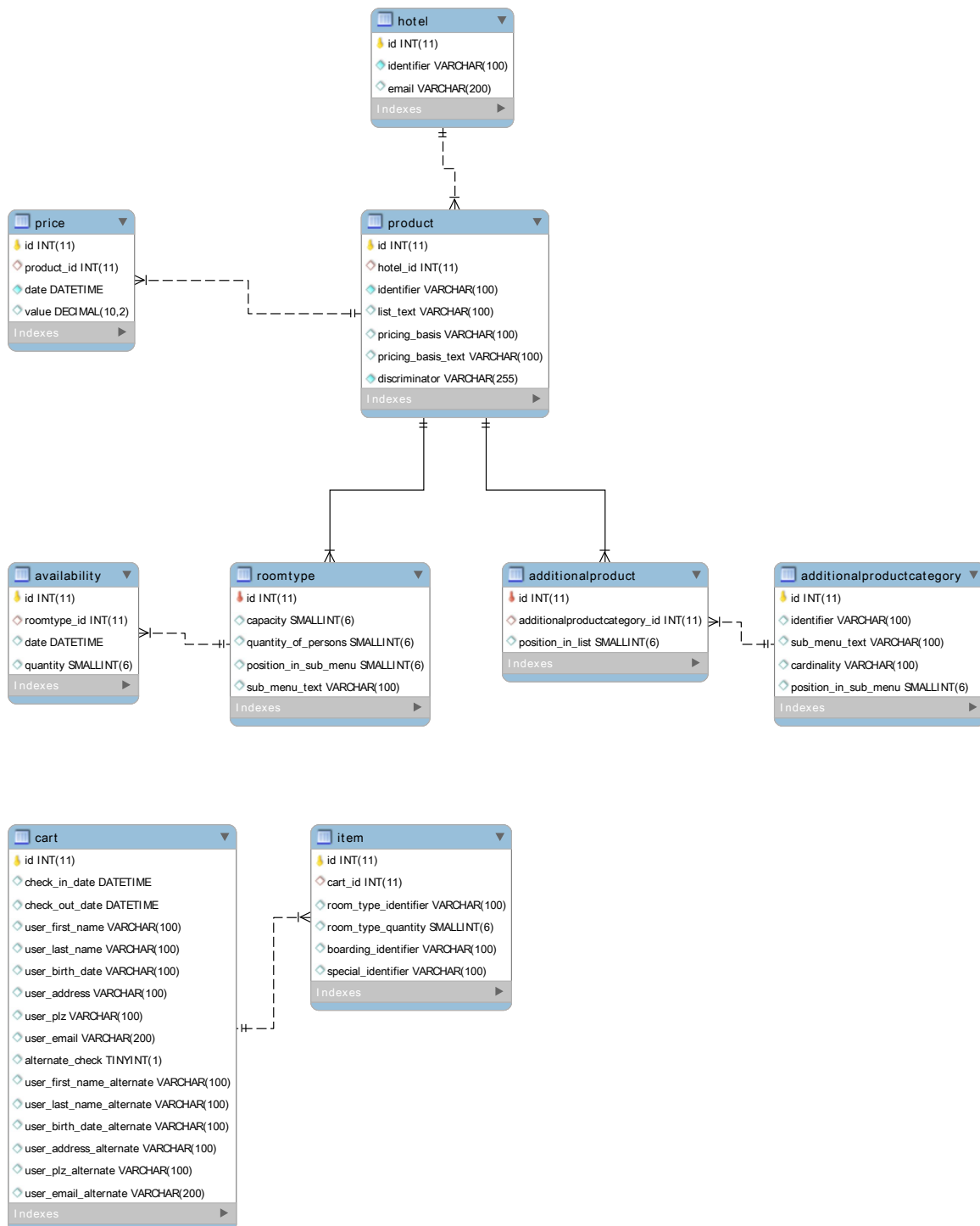
A.3 Sequenzdiagramm









A.4 Klassendiagramm (ohne Getter- und Settermethoden)



A.5 Datenbankdiagramm



A.6 Ausschnitte des grafischen Konzepts

		
Auswahl des Anreisedatums	Auswahl der Zimmerkategorie	Auswahl der Verpflegung
		
Auswahl der Specials	Übersicht der Bestelldetails	Eingabe der persönl. Daten

