Entwicklung eines Bestellsystems

Projektdokumentation, Projekte der Wirtschaftsinformatik

Der Webshop „Festiva“

Fachhochschule der Wirtschaft Paderborn
4. Semester
Pfbw115a
WS 16/17
Marcus Becker

Entwicklung durch:
Alina Fankhänel
Nicola Kloke
Timo Schlüter

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Allgemein

Das im Folgenden beschriebene Projekt wird im Rahmen des Dualen Studiums an der Fachhochschule der Wirtschaft Paderborn im Studienfach „Projekte der Wirtschaftsinformatik“ durchgeführt. Auftraggeber des Projekts ist der Dozent Marcus Becker. Dieser übernimmt im Projektverlauf zugleich die Rolle des technischen und fachlichen Ansprechpartners.

Inhalt des Projekts ist die Entwicklung eines Bestellsystems auf Basis von Java.

1.2 Ist-Zustand

Das fiktive Unternehmen „Festiva“ ist ein Anbieter von Festival-Artikeln, der aktuell nur Shops vor Ort besitzt. Da mehrere Kunden ihr Interesse an einer Möglichkeit zum Online-Shopping bekundet haben, möchte das Unternehmen ihren Kunden einen Webshop bereitstellen.

1.3 Soll-Zustand

Der Webshop „Festiva“ ist auf den Verkauf von Festival-Artikeln ausgelegt. Er bietet für registrierte Kunden die Möglichkeit, online einzukaufen. Dabei gliedert „Festiva“ die Festivals in bestimmte Kategorien (Rock, Schlager…). Für jedes Festival können verschiedene Artikel gekauft werden: Eintrittskarte für Freitag, Eintrittskarte für Samstag, Eintrittskarte für das gesamte Wochenende, etc.

Die Pflege der Daten innerhalb des Webshops findet über einen Administrator statt.

Der Webshop muss dabei die folgenden Funktionalitäten bereitstellen (Muss-Kriterien):

- Erfassen, anzeigen, bearbeiten, löschen der Kundendaten (Registrierung/Anmeldung/Verwaltung durch den Kunden selbst und durch den Admin)
- Der Admin kann den Kunden manuell bei der Bearbeitung der Kundendaten sperren
- Die Passwörter werden als Hash-Wert in der Datenbanktabelle abgelegt.
- Erfassen, anzeigen, bearbeiten, löschen der Kategoriendaten (durch den Admin)
- Erfassen, anzeigen, bearbeiten, löschen der Festivaldaten (durch den Admin)
- Erfassen, anzeigen, bearbeiten, löschen der Artikeldaten (durch den Admin)
- Die Suche nach Festivals kann über Name, Ort, Kategorie und Datum eingegrenzt werden.
- Es gibt eine direkte Navigation zur Festivalsuche über die Startseite. Dort kann eine Kategorie angeklickt werden. Anschließend sind weitere Eingrenzungen über die Kriterien (s.o.) möglich.
- Erfassen von Bestellungen und anzeigen von vergangenen Bestelldaten
- Artikel in den Warenkorb legen, Warenkorb anzeigen und bearbeiten
- Benutzerauthentifizierung, Benutzer mit unterschiedlichen Rechten (Kunden und Admin inkl. Login/Logout)
- Responsive Design auf Desktop und Android-Smartphone

Diese Funktionalitäten können zusätzlich von dem Webshop bereitgestellt werden (Kann-Kriterien):

- Kunden werden automatisch gesperrt (z.B. nach 3-mal falsch eingegebenem Kennwort)
- Bei der Festivalsuche kann in einem bestimmten Umkreis gesucht werden
- Prüfung, ob eine zulässige IBAN eingegeben wurde
- Kennwortrichtlinien festlegen und prüfen
- Automatisierte Möglichkeit um das Passwort zurückzusetzen
- Bewertungen für Artikel abgeben und einsehen können
- Bestandsführung der Artikel
- Verwaltung der Kundenbestellungen aus Administrator-Sicht
- Anlage von Admin-Konten über eine Oberfläche
- Einbinden von zusätzlichen Festival-Artikeln (Regencapes etc.)

1.4 Systemumgebung

1.4.1 Software

Der Onlineshop wird mit Hilfe der Entwicklungsumgebung Eclipse Neon entwickelt und in Java, Java Script, sowie HTML 5 und CSS programmiert.

Die Entwicklungsumgebung benötigt dabei zusätzliche Einstellungen. Zum Hochladen von Bildern müssen unter Window -> Preferences -> General -> Workspace „Refresh on access“ und „Refresh using native hooks or polling“ aktiviert sein. Zusätzlich muss die Environment Variable „myPath“ eingerichtet werden. Diese muss den Pfad zum Projekt beinhalten. Beispiel: C:\\Users\\MeinName\\Documents\\

Als Datenbanksystem wird MySQL und als Webserver Tomcat 7.0 aus der Software XAMPP verwendet. Somit können eine lokale Datenbank sowie ein lokaler Webserver simuliert werden.

Der Anwender benötigt Google Chrome oder Mozilla Firefox mit Java-Script um den Onlineshop ohne Einschränkungen bedienen zu können.

1.4.2 Hardware

Für die Entwicklung wird ein Rechner benötigt, der die Programme Eclipse Neon und XAMPP ausführen kann und installiert hat.

Als Hardware für den Endnutzer wird ein Rechner vorausgesetzt, der die Software Google Chrome oder Mozilla Firefox ausführen kann. Zudem kann ein Smartphone verwendet werden, das ebenfalls einen der beiden genannten Browser besitzt.

1.4.3 Organisation und Hilfsmittel

Der Ansprechpartner seitens des Auftraggebers ist der Dozent Marcus Becker. Er steht uns in sämtlichen Fragen, sei es organisatorisch oder fachlich, als Ansprechpartner zur Verfügung. Durch ihn wird auch die Projektabnahme erfolgen. Die Projektverantwortlichen des Projektes sind Nicola Kloke, Timo Schlüter und Alina Fankhänel.

Für die Planung, Organisation und Dokumentation werden MS Word, MS Excel, MS Visio, MS Project, StarUML sowie Moqups.com verwendet. Zur Versionsverwaltung und als Synchronisationstool für den Quellcode werden Git und GitHub.com verwendet.

1.5 Zielgruppe

Die Zielgruppe für den Onlineshop beinhaltet bisherige und auch neue Kunden von „Festiva“. Da eine Vielzahl an Festivals angeboten wird, sollen möglichst alle Altersgruppen angesprochen werden. Für den Administrationsbereich sind ausgewählte Mitarbeiter von „Festiva“ verantwortlich. Diese übernehmen die Rolle des Admins.

2 Architektur der Anwendung

2.1 Allgemeine Struktur

Die Anwendung wurde auf Grundlage des Model-View-Controller-Konzepts entwickelt.

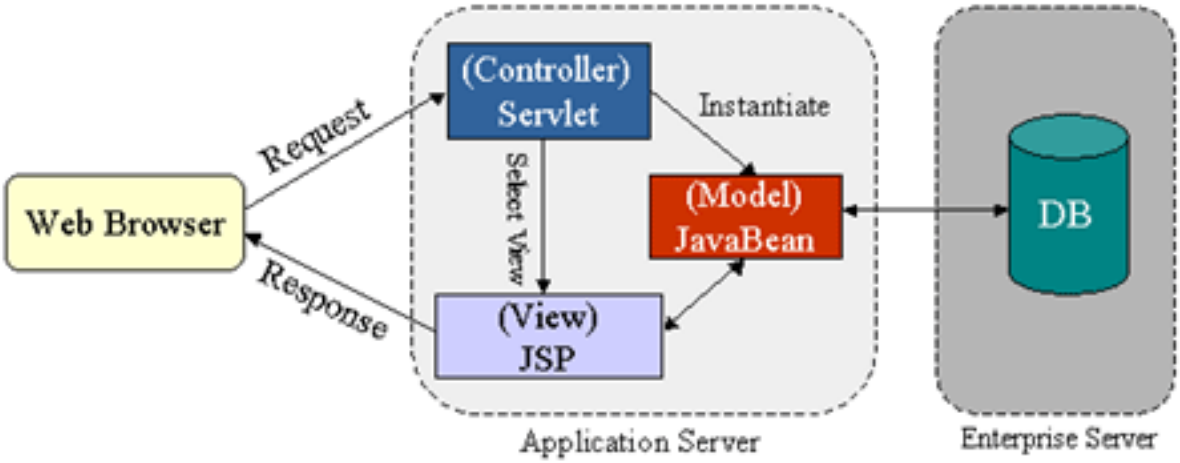

Abbildung 1 - Modell-View-Controller

Da jede interaktive Anwendung aus Eingaben, Daten und der Darstellung der Daten besteht, wäre es problematisch, diese drei Elemente in einer Klasse (Formular) zusammenzufassen. Das Ziel des MVC-Konzepts liegt daher in der Trennung zwischen Datenverarbeitung, Repräsentation und Eingaben, um das System insgesamt flexibler zu gestalten.

Das **Model** beinhaltet dabei den funktionalen Kern der Anwendung und ist unabhängig von der Darstellung am Bildschirm.

Die **View** ist für die aktuelle Darstellung der Eingangs- und Ausgangsdaten in den Anzeigeobjekten verantwortlich.

Der **Controller** ist für die Steuerung der Anwendung über den Benutzer verantwortlich. Durch ihn werden eingegebene Daten ausgewertet und weitergeleitet. Damit leitet der Controller die Änderungen der Modelldaten ein.[1]

[1] Vgl. zu diesem Absatz:
http://informatik.bildung-rp.de/fileadmin/user_upload/informatik.bildung-rp.de/Fortbildung/html/D1-MVCNotenstatistik/mvc0_0.html
(aufgerufen am 02.12.2016)

2.2 Datenbank-Struktur

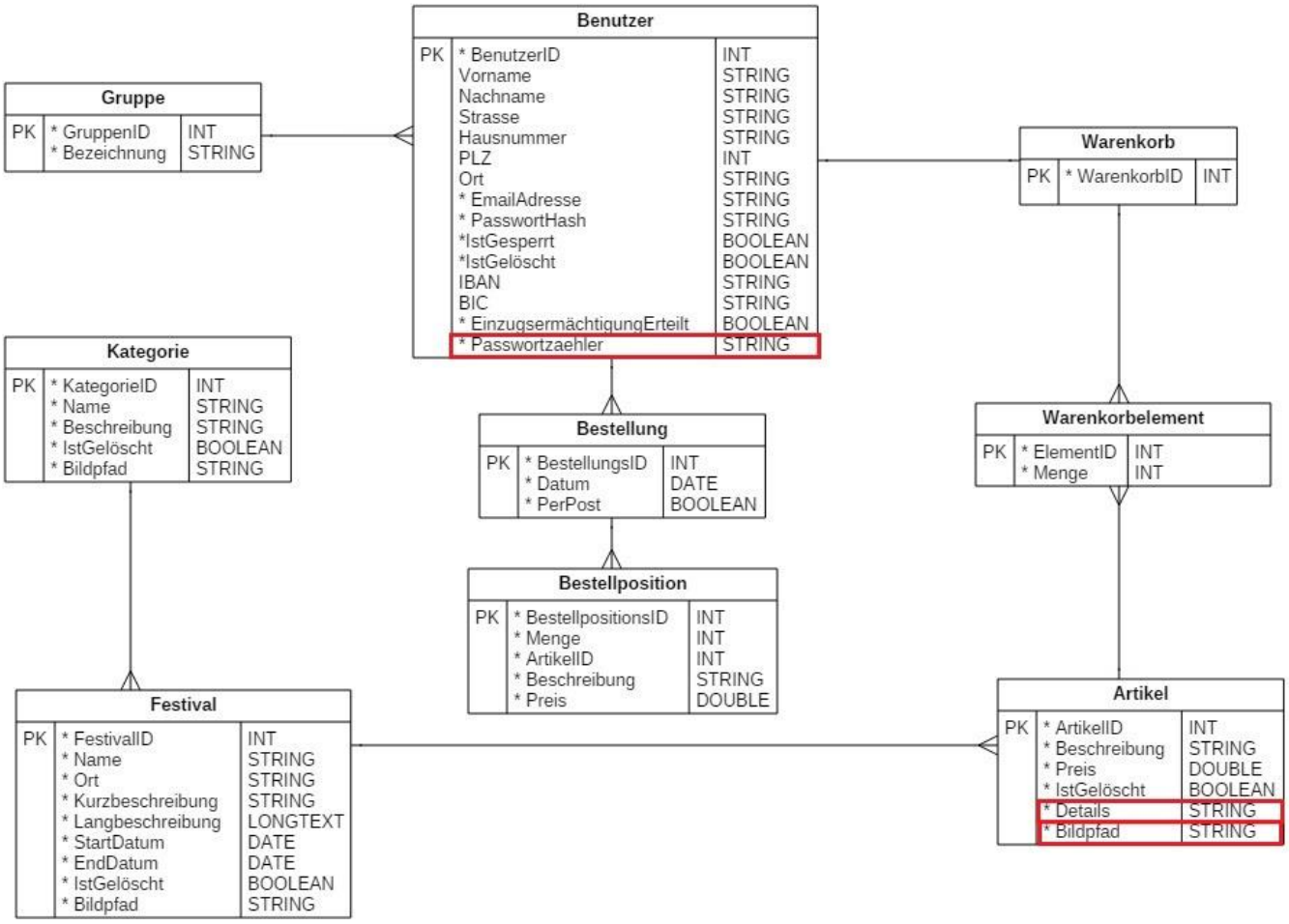

Abbildung 2 - Entity-Relationship-Modell

Die geplante Datenbankstruktur wurde im Verlauf des Projekts um einige Attribute erweitert. Dies liegt an der Realisierung von zwei Kann-Kriterien.

Das zusätzliche Attribut „Passwortzaehler“ wird benötigt, um die automatische Sperrung eines Kunden zu realisieren. Diese tritt nach dreimaliger Falscheingabe des Passworts ein. Die Anzahl der falschen Passworteingaben wird in diesem Feld gesichert.

Die weiteren Attribute „Details“ und „Bildpfad“ an der Entität „Artikel“ ergeben sich durch die Einbindung von festivalübergreifenden Artikeln. Zu diesen sollen weitere Informationen für eine Detailansicht im Zubehör-Shop bereitgestellt werden können.

2.3 Klassen-Struktur

Die Klassen innerhalb des Projekts sind in drei verschiedene Pakete eingeteilt. Auch die Klassendiagramme sind zur besseren Übersichtlichkeit in drei Diagramme aufgeteilt. Trotzdem bestehen natürlich Beziehungen zwischen den Klassen der unterschiedlichen Pakete, die zum einen durch die Namensgebung als auch durch die untenstehenden Beschreibungen deutlich werden sollen.

1. standardPackage

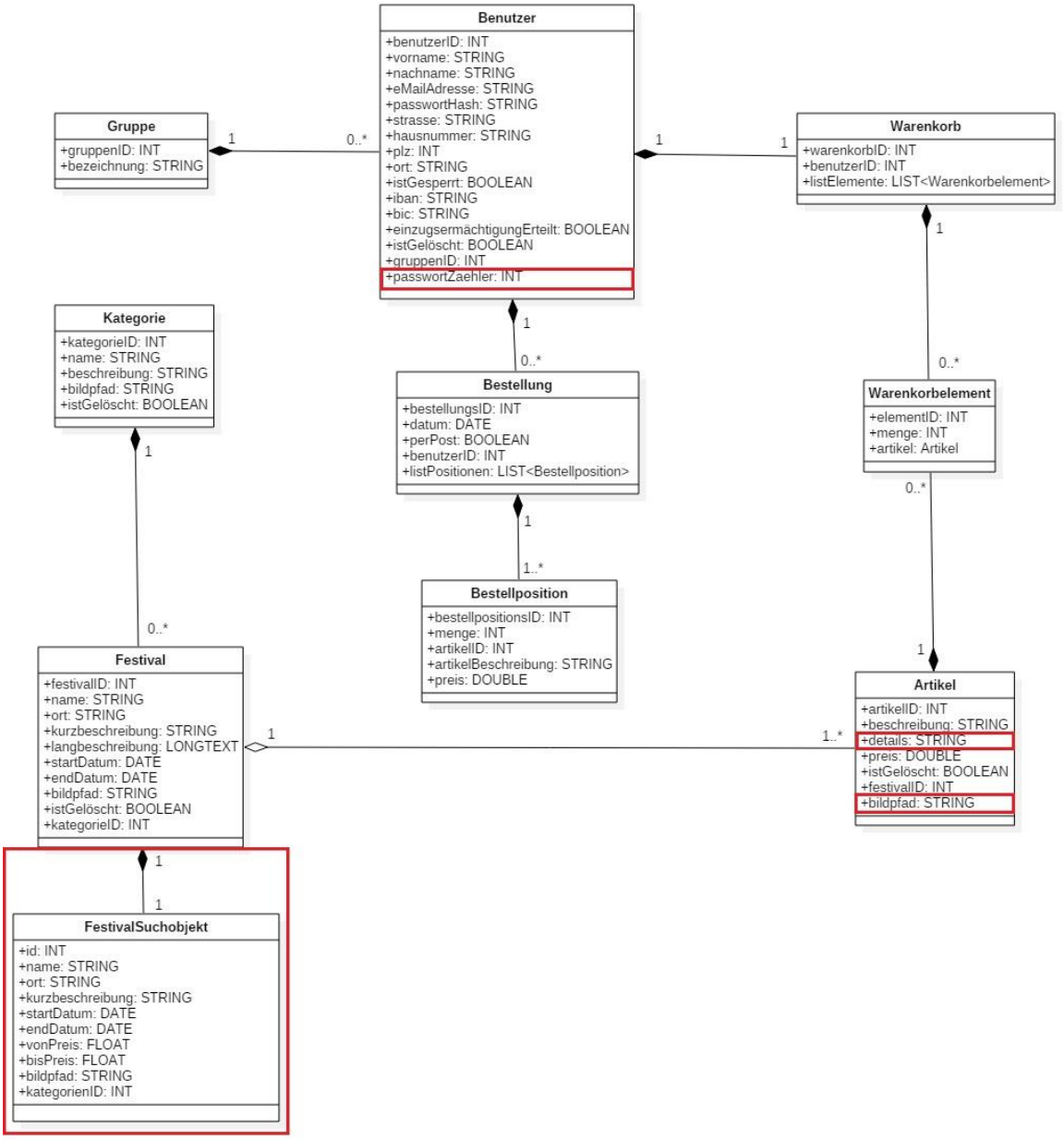

Abbildung 3 - Klassendiagramm: standardPackage

Das erste Paket beinhaltet alle Fachklassen, die im Rahmen des Webshops benötigt werden. Im Verlauf des Projekts haben sich gegenüber dem Entwurf im Fachkonzept einige Änderungen ergeben.

Die neue Klasse „FestivalSuchobjekt“ existiert, da im Bereich der Suche (Ticket Shop) zu einem Festival zusätzliche Informationen, wie beispielsweise ein Maximalpreis gespeichert werden müssen. Um die Übergabe an die JSP möglichst einheitlich zu gestalten, wird dann ein FestivalSuchobjekt, das alle nötigen Informationen, die dem Anwender in der Ticket-Suche angezeigt werden müssen, beinhaltet, übergeben.

Darüber hinaus wurde das Kann-Kriterium mit der Einbindung von festivalübergreifenden Artikeln realisiert. Da zu diesen auch Bilder verfügbar sein sollen, ist in der Klasse „Artikel“ das Attribut „bildpfad“ hinzugefügt worden. Für weiterreichende Informationen zu den Artikeln ist das Attribut „details“ integriert worden. Außerdem wurde für die Realisierung des Kann-Kriteriums mit der dreimaligen Sperre bei falscher Passworteingabe das zusätzliche Attribut „passwortZaehler“ bei der Klasse „Benutzer“ hinzugefügt.

2. managerPackage

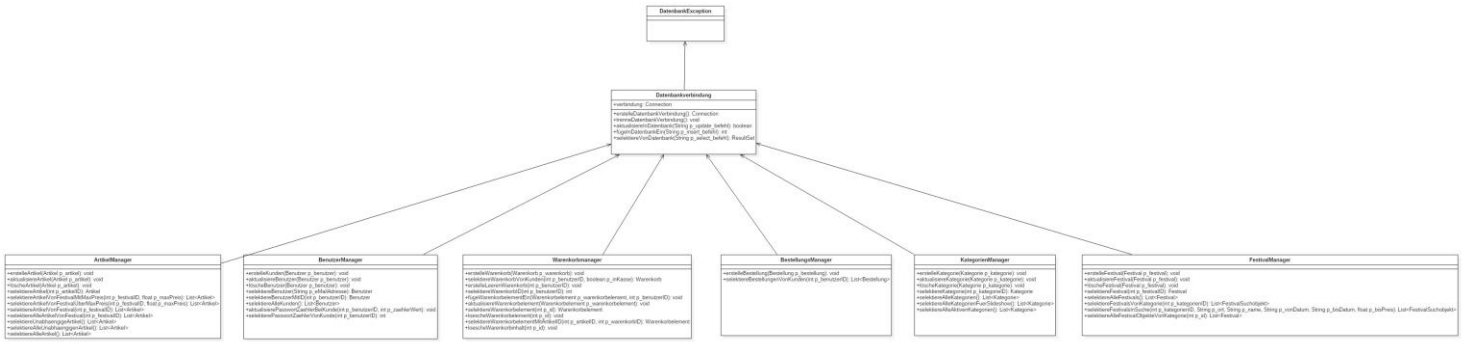

Abbildung 4 - Klassendiagramm: managerPackage (siehe Anhang)

Die Manager-Klassen beinhalten alle Funktionen, um direkt auf die Datenbank zugreifen zu können. Die Klasse „Datenbankverbindung“ ist dabei dafür zuständig, die Kommunikation mit der Datenbank zu steuern. Das bedeutet, dass in dieser Klasse sowohl die Erstellung der Verbindung als auch jeweils eine vorgefertigte Methode für das Durchführen von Select-, Insert- und Update-Befehlen existiert. Diese Schnittstellen werden in den tatsächlichen Manager-Klassen verwendet und bieten den Vorteil, dass dort nur der SQL-Befehl erstellt werden muss und die zurückgelieferten Ergebnisse (bei einem Select-Aufruf beispielsweise ein Result-Set) verarbeitet werden müssen. Der technische Teil zum Verbindungsaufbau ist damit in einer einzigen Klasse gekapselt.

Damit die Anwendung bei einer Nichtverfügbarkeit der Datenbank trotzdem benutzerfreundlich reagiert, gibt es die „DatenbankException“. Diese liefert dem Anwender in einem solchen Fall eine angemessene Fehlerseite und protokolliert den Fehler im Log.

Die Managerklassen nutzen die Fachklassen aus dem ersten Paket. Bei der Suche nach einem Artikel wird das zurückgegebene Ergebnis (Result-Set) beispielsweise verarbeitet, indem eine Instanz der entsprechenden Fachklasse erstellt wird, in der die Informationen dann abgelegt werden. Wenn mehrere Ergebnisse enthalten sind, wird eine ArrayList vom Typ der Fachklasse erstellt und mit den Daten gefüllt.

Es wurden sprechende Namen verwendet, sodass beispielsweise die Klasse „ArtikelManager“ alle Methoden zur Erstellung, Ändern und Löschen von Artikeln beinhaltet. Das bedeutet, dass hier vorrangig die Fachklasse Artikel instanziiert wird. In einigen Fällen, wie beispielsweise bei dem „BestellungsManager“ werden auch weitere Klassen als nur die Fachklasse „Bestellung“ instanziiert. Da das Objekt Bestellung unter Anderem aus einer Liste mit Bestellpositionen besteht, wird hier auch die Klasse „Bestellposition“ instanziiert.

Zur Übersichtlichkeit folgt der Aufbau aller Manager-Klassen innerhalb des Pakets dem erläuterten Prinzip.

3. servletPackage

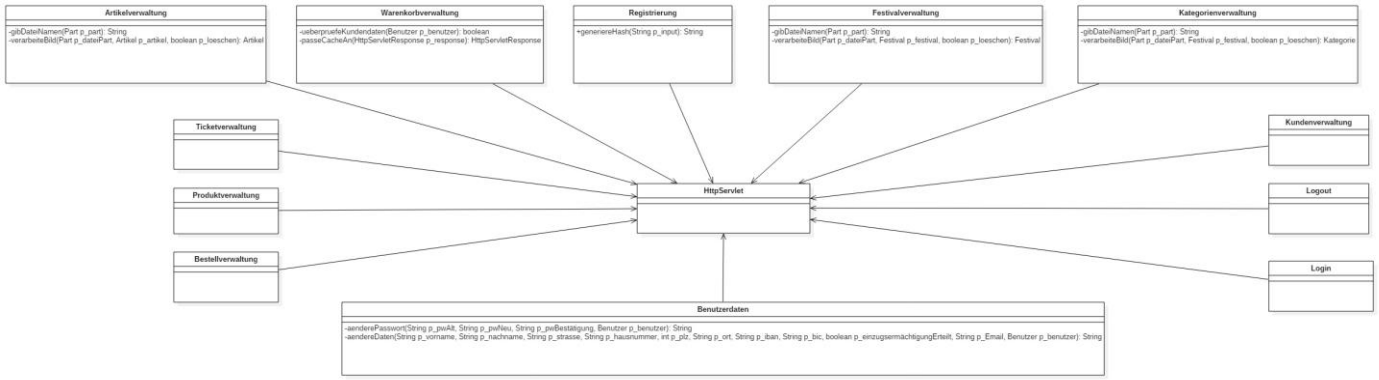

Abbildung 5 - Klassendiagramm: servletPackage (siehe Anhang)

Im Rahmen des MVC-Konzepts ist die Verwendung von Servlets zur Annahme und Beantwortung von Anfragen des Clients erforderlich. Im Klassendiagramm wurde auf die Darstellung der post()- und get()-Methoden verzichtet. Jedes Servlet enthält diese Methoden zur Übermittlung der benötigten Daten. Die Servlets arbeiten sowohl mit den Fachklassen (erstes Paket) als auch mit den Managerklassen (zweites Paket). Die Ermittlung eines Benutzers über die ID ist damit beispielsweise nur ein Aufruf einer Methode des „BenutzerManagers“ und die Speicherung des Rückgabewerts in einer Instanz der Klasse „Benutzer“.

Um nicht für jede Anfrage ein neues Servlet verwenden zu müssen, wird der Parameter „aktion“ verwendet. Dadurch werden in der Regel einzelne Use-Cases gemeinsam in einem Servlet bearbeitet. Das Servlet „Artikelverwaltung“ beinhaltet zum Beispiel die Möglichkeiten zur Anlage, Änderung, etc. von Artikeldaten durch den Administrator. Zur Anlage muss der Wert des Parameters in diesem Fall „anlegen“ haben. Dadurch entsteht eine logische Kapselung von Funktionalitäten, die vor Allem der Übersichtlichkeit dient. Der Parameter „aktion“ wird innerhalb des gesamten Projekts verwendet. Lediglich die Servlets Login und Logout werden nicht über diesen Parameter gesteuert. Das liegt daran, dass diese Servlets Grundfunktionen zur Session-Steuerung beinhalten und wir diese unabhängig von den fachlich-funktionalen Servlets steuern wollten.

Generell besitzen, durch den oben genannten Aufbau, alle Servlet-Klassen eine einheitliche Struktur.

3 Oberfläche der Anwendung

3.1 Kundensicht

Zum ersten Entwurf der Oberfläche des Webshops wurden Mock-Ups entwickelt. Im Folgenden sollen ausgewählte Mock-Ups dem tatsächlichen Design der Oberfläche gegenübergestellt werden.

3.1.1 Planung

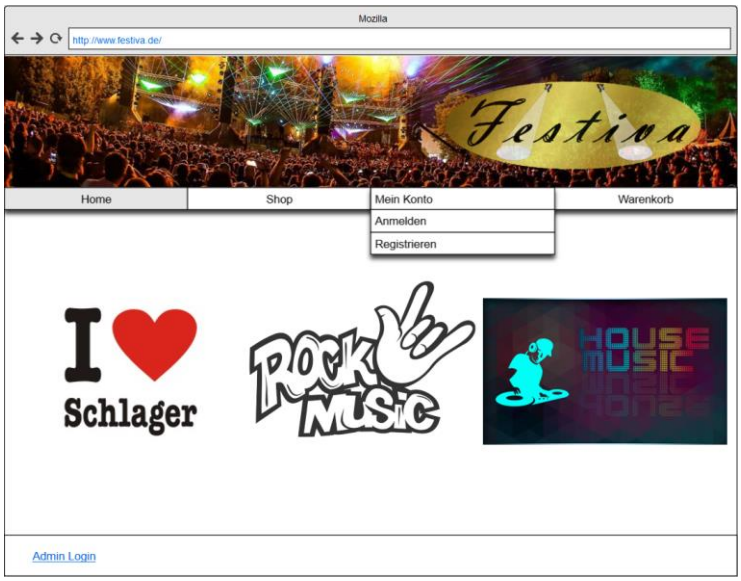

Abbildung 6 - Mock-Up: Startseite

In dem allgemeinen Aufbau des Webshops befindet sich ein Logo, welches jederzeit im Header zu sehen sein soll. Die Navigationsleiste soll horizontal unter dem Header positioniert werden. Abschließend wird der Inhaltsbereich mit dem Footer abgerundet. Ein Benutzer (nicht angemeldet) kann auf die Startseite, den Shop, auf sein Konto oder auf den Warenkorb navigieren. Auf der Startseite sollen Bilder von Festivalkategorien angezeigt werden. Ein Administrator soll über einen Link im Footer auf seine Anmeldemaske geleitet werden.

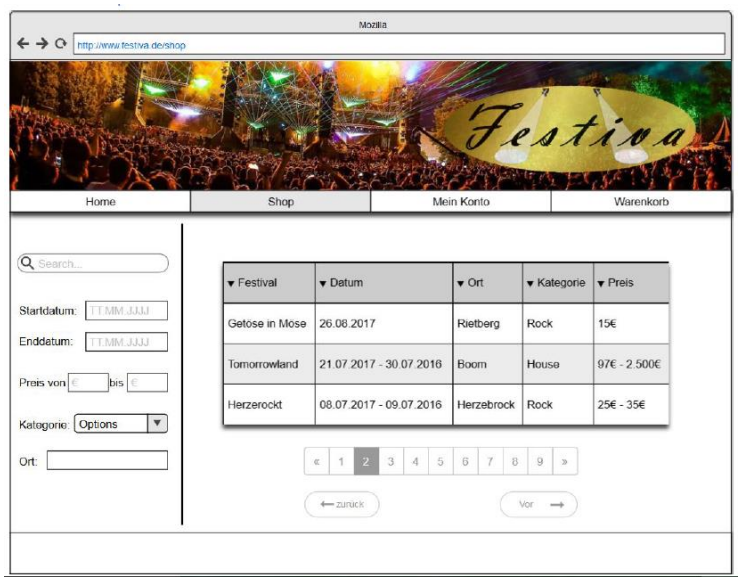

Abbildung 7 - Mock-Up: Shop

Im Shop soll durch verschiedenste Felder im linken Suchbereich möglich sein, die Suche einzugrenzen. Man kann nach Namen, Start- und Enddatum, Von- und Bis-Preis, Kategorie sowie dem Ort suchen. Die entsprechenden Festivals sollen daraufhin in einer Tabelle angezeigt werden.

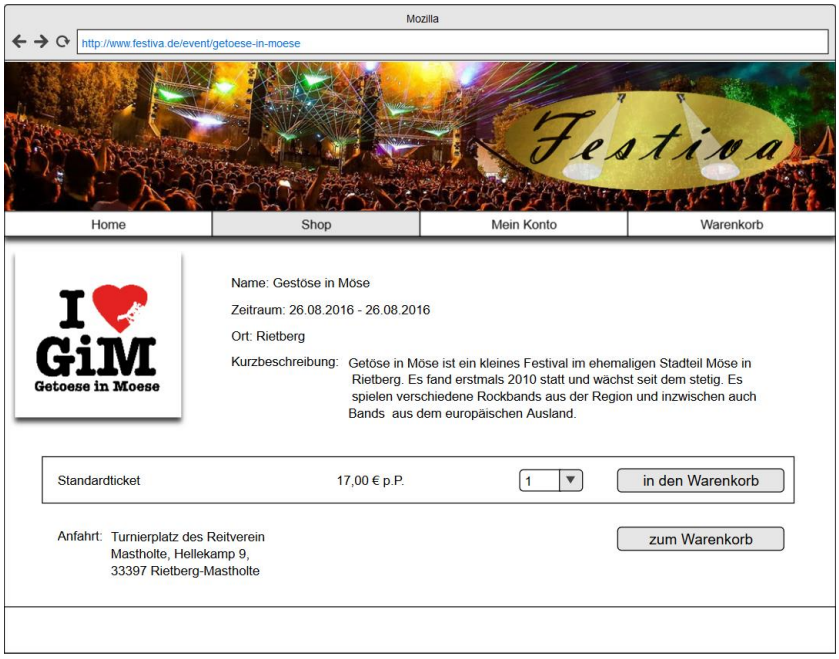

Abbildung 8 - Mock-Up: Festivaldetailsicht

Zur Festivaldetailsicht gelangt man, wenn im Shop ein Festival anklickt wird. In der Sicht werden weitere Informationen zu dem Festival sowie die bestellbaren Tickets angezeigt. Der Benutzer kann seine gewünschte Anzahl wählen, Tickets in den Warenkorb legen und den Button „zum Warenkorb“ klicken.

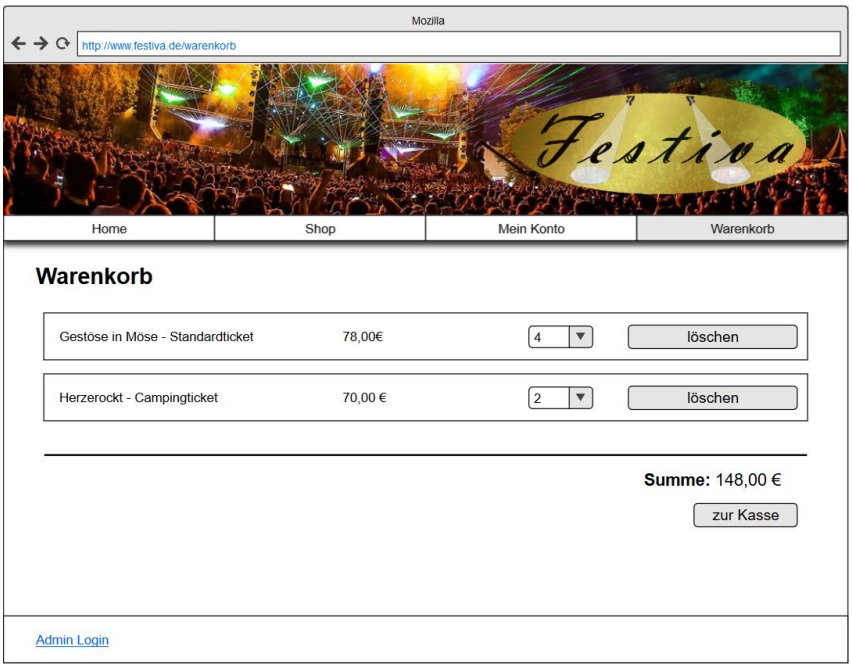

Abbildung 9 - Mock-Up: Warenkorb

Der Warenkorb beinhaltet alle Tickets, die zuvor in den Warenkorb gelegt wurden und die Gesamtsumme der Artikel. Im Warenkorb selbst kann über ein Drop-Down nochmals die Anzahl angepasst oder Positionen per Button gelöscht werden. Ebenso hat der Benutzer die Möglichkeit über einen Button zur Kasse zu gehen, um dort den Kauf durchzuführen.

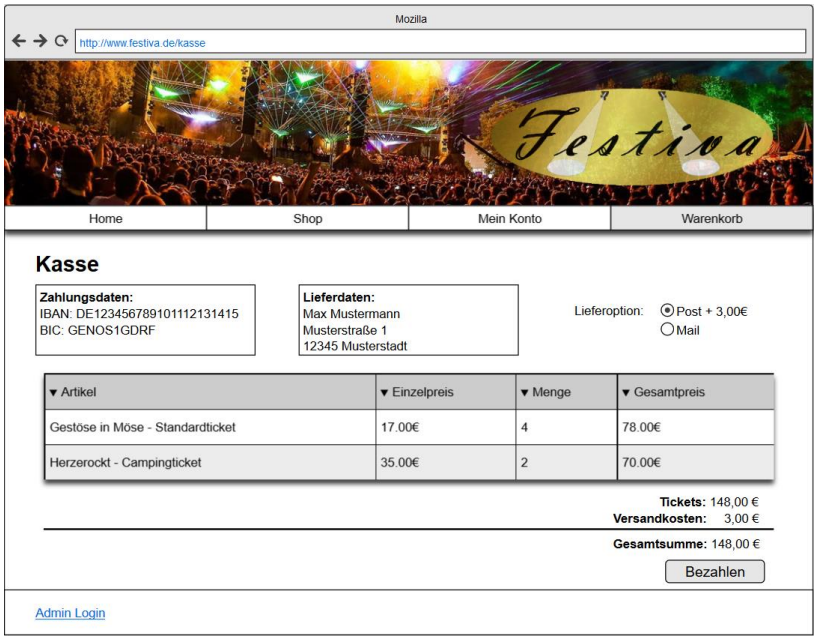

Abbildung 10 - Mock-Up: Kasse

In der Kasse werden die Zahlungs- und Lieferdaten des aktuell angemeldeten Kunden angezeigt. Er hat die Option, zwischen Post- und Mailversand zu wählen. Die im Warenkorb enthaltenen Tickets werden nochmals aufgeführt und am Ende eine Gesamtsumme inkl. Versandkosten angezeigt. Über den „Bezahlen“ - Button können die Artikel verbindlich bestellt werden.

3.1.2 Realisierung

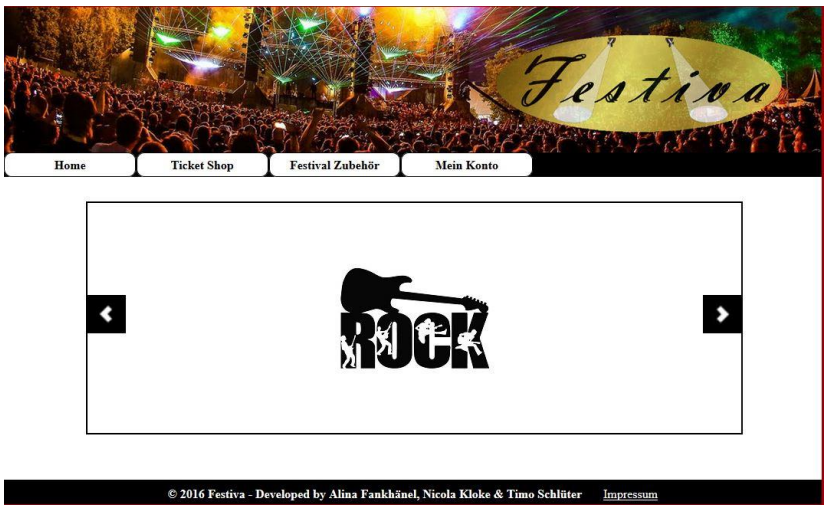

Abbildung 11 - Webshop: Startseite

Die Navigation aus den Mock-Ups wurde durch den Zubehörshop ergänzt und ändert sich je nachdem, ob der Kunde angemeldet ist oder nicht. Der Warenkorb des Kunden taucht beispielsweise erst nach der Anmeldung in der Navigation auf.

Die Bilder der Kategorien werden in einer Slideshow angezeigt. Wenn sich ein Festiva-Mitarbeiter als Admin anmelden möchte, kann er dies über die normale Navigation (Mein Konto) machen und muss nicht über einen Link im Footer gehen, wie zuvor in den Mock-Ups geplant wurde. Hinzugekommen ist außerdem das Impressum, welches der Benutzer im Footer anklicken kann.

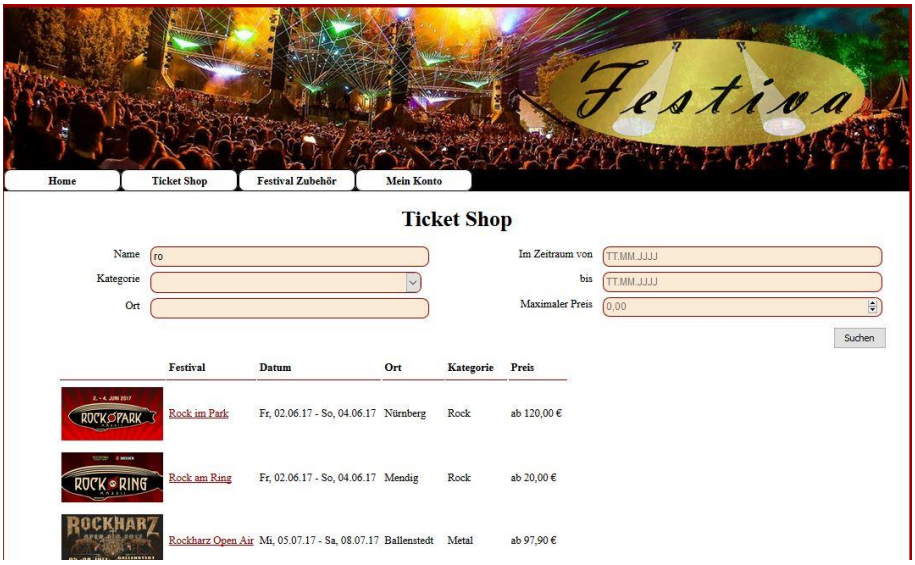

Abbildung 12 - Webshop: Ticket Shop

Die Suchfelder sind nun oberhalb der Festivals positioniert. Die Suchoptionen sind wie geplant umgesetzt worden. Eine Ausnahme ist jedoch, dass der Benutzer nur einen Maximal- und keinen Minimalpreis eingeben kann, da uns im Verlauf des Projekts bewusst wurde, dass der Maximalpreis das entscheidendere Kriterium ist. Um den Ticketshop ansprechender zu gestalten, sind nun auch Bilder zu den Festivals zu sehen.

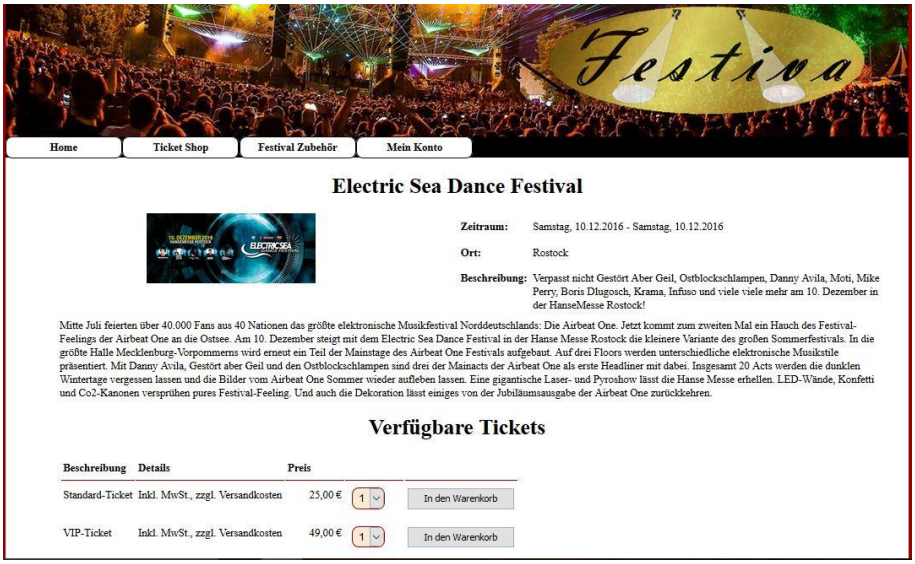

Abbildung 13 - Webshop: Festivaldetailsicht

In der Festivaldetailsicht wird die Beschreibung angezeigt, die zuvor im Shop noch nicht zu sehen war. Außerdem sind die zugehörigen Tickets aufgelistet, die, möglicherweise nach einer Änderung der Anzahl per Drop Down, über einen Button in den Warenkorb gelegt werden können. Diese Ansicht ist stark an das dafür vorgesehene Mock-Up angelehnt.

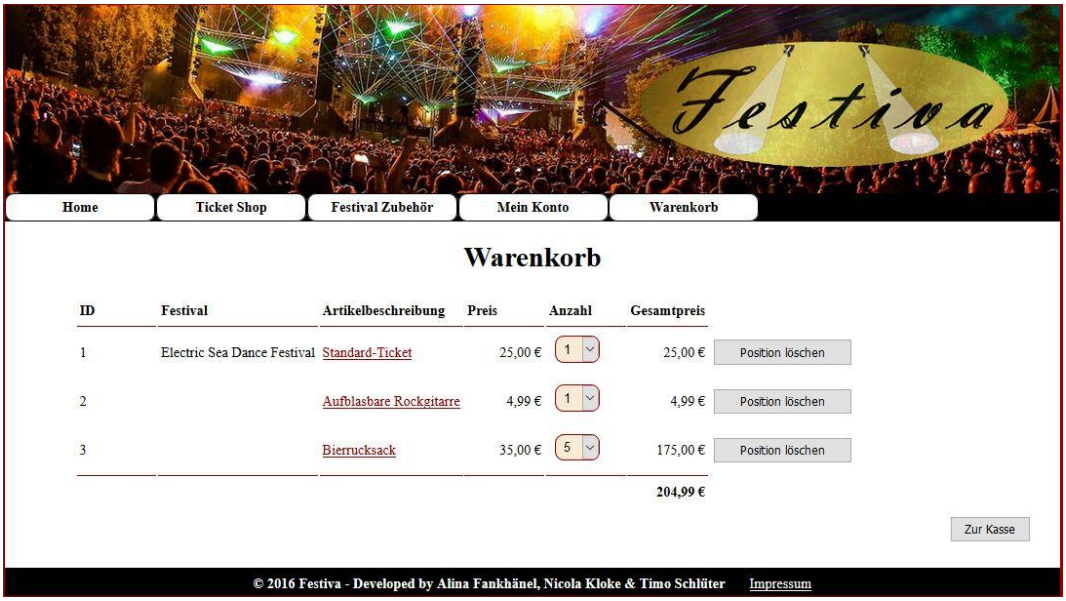

Abbildung 14 - Webshop: Warenkorb

Die Oberfläche des Warenkorbs gleicht dem zugehörigen Mock-Up. Pro Artikelposition wurde eine ID eingefügt und es gibt einen Gesamtpreis, der neben dem Einzelpreis eines Artikels den Preis zur gewählten Artikelanzahl ausgibt.

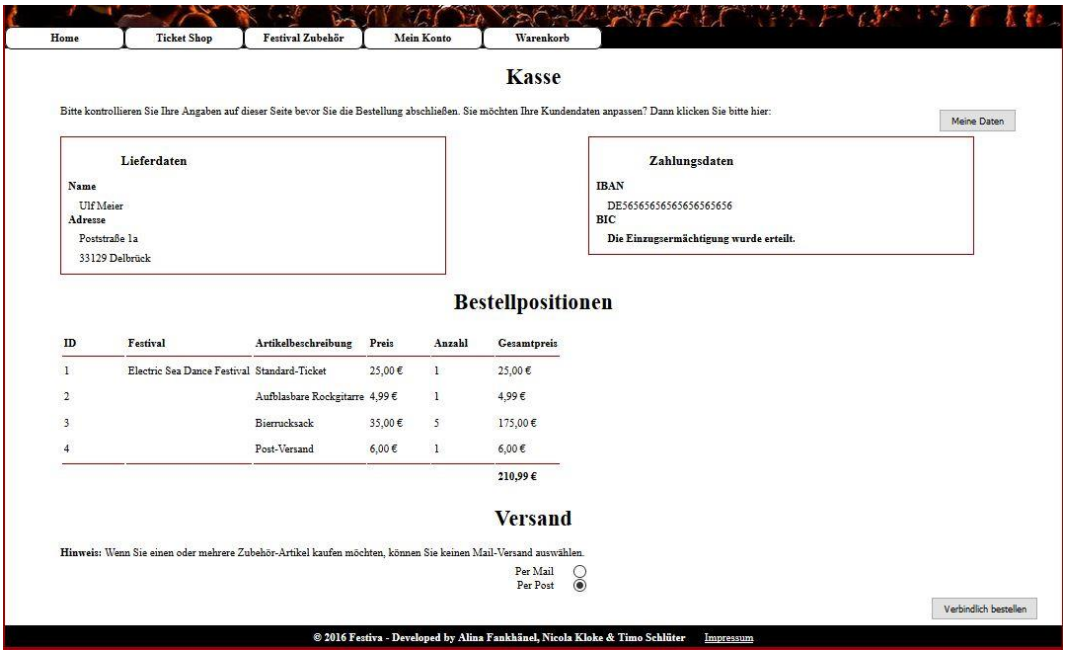

Abbildung 15 - Webshop: Kasse

Über den Button „Meine Daten“ gelangt der Kunde zu seinen Kundendaten und kann diese dort ändern, falls er bemerkt, dass die angezeigten Liefer- und Zahlungsdaten nicht korrekt oder unvollständig sind. Darunter werden die Tickets und Zubehörartikel aufgelistet, die zuvor aus den jeweiligen Shops in den Warenkorb gelegt wurden. Jeder Zeile wurde hier ebenfalls eine ID gegeben.

Ist die Spalte Festival gefüllt, handelt es sich um ein Ticket, ansonsten um einen Zubehörartikel. Diese Spaltentrennung wurde erst nach Implementierung des Zubehörshops realisiert. Analog zum Mock-Up kann auch hier zwischen Mail- und Postversand gewählt werden. Über den Button „Verbindlich bestellen“ wird die Bestellung erstellt.

3.2 Adminsicht

Das Navigationsmenü des Administrators besteht aus der Startseite, den Verwaltungen für Kunden, Festivals und Kategorien sowie „Mein Konto“.

3.2.1 Planung

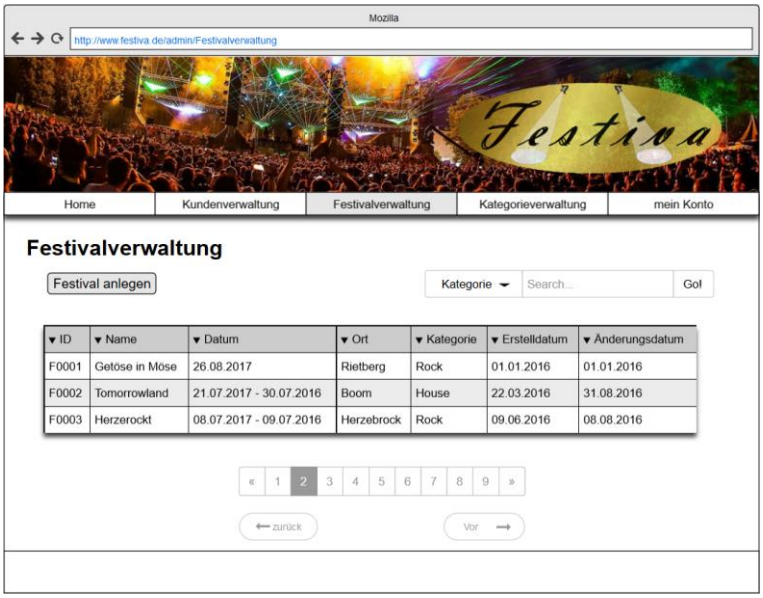

Abbildung 16 - Mock-Up: Festivalverwaltung

Beispielhaft für alle Verwaltungen soll hier anhand der Festivalverwaltung der generelle Aufbau erläutert werden.

In den Verwaltungen soll es möglich sein, über einen Button neue Kunden, Festivals oder Kategorien anzulegen. Darunter werden alle Elemente der jeweiligen Verwaltung angezeigt, die in der Datenbank existieren. Ebenso soll eine Suche über die Elemente implementiert sein.

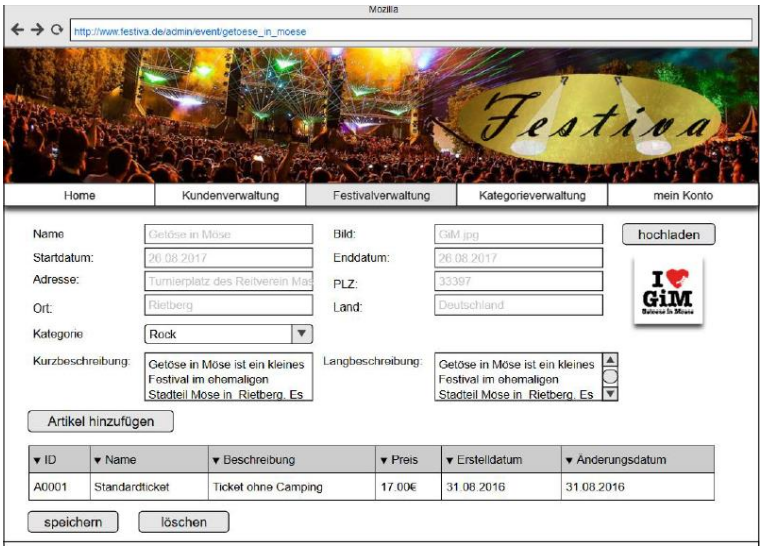

Abbildung 17 - Mock-Up: Festival ändern

Wählt man in der Verwaltung über die ID ein bestimmtes Element aus, gelangt man in den Änderungsmodus. Hier können die Daten des ausgewählten Elements angepasst werden oder das gesamte Element gelöscht werden.

Im Änderungsmodus eines Festivals ist es zusätzlich möglich, dem Festival neue Artikel hinzuzufügen oder ein Bild zu geben. Den Kategorien kann ebenfalls jeweils ein Bild zugeordnet werden.

3.2.2 Realisierung

Abbildung 18 - Webshop: Festivalverwaltung

Durch die Realisierung von Kann-Kriterien ist die festivalübergreifende Artikelverwaltung (für den Zubehörshop), sowie die Anlage neuer Administratoren über „Administration“ hinzugekommen. Hier kann der angemeldete Admin ebenso seine Daten ändern und sich abmelden.

Die Suche in den Verwaltungen sowie die Buttons zum Anlegen wurden ähnlich der Mock-Ups realisiert. Die entsprechenden Elemente werden ebenfalls tabellarisch aufgelistet. Anstatt Erstell- und Änderungsdatum wird angezeigt, ob das Element ein Bild besitzt (außer bei Kunden) oder gelöscht ist. Beim Kunden erscheint eine Spalte, die angibt, ob der Kunde gesperrt ist. Alle weiteren Spalten sind pro Verwaltung unterschiedlich und zeigen nur die für den Administrator relevanten Informationen an. Über die ID eines Elements gelangt man in den Änderungsmodus.

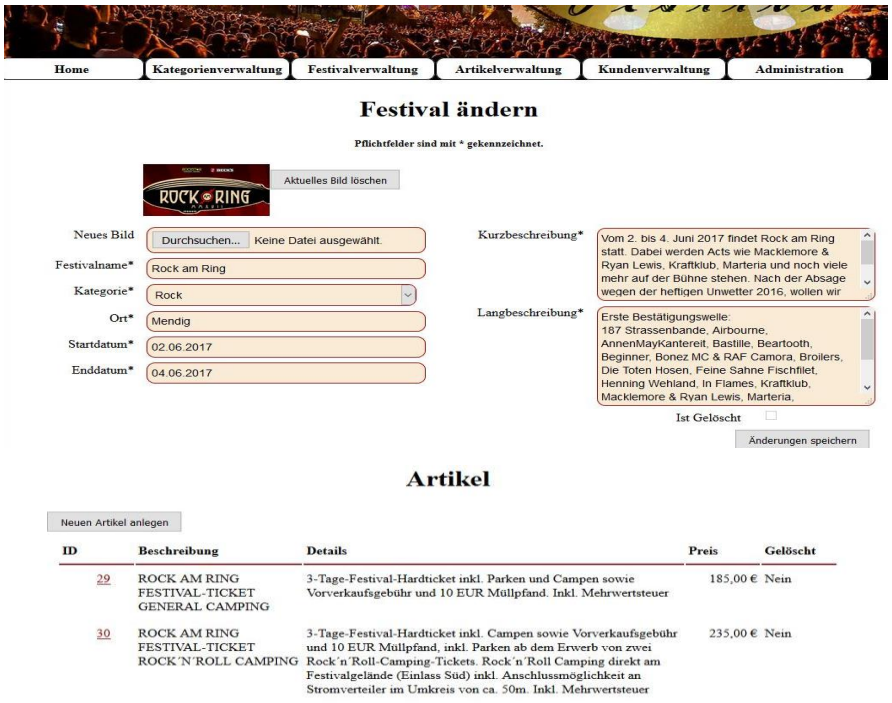

Abbildung 19 - Webshop: Festival ändern

Hier wurde die Checkbox „Ist Gelöscht“ hinzugefügt und Pflichtfelder durch „*“ gekennzeichnet. Je nach Verwaltung variieren die änderbaren Daten.

3.3 Mobile Sicht

3.3.1 Planung

Abbildung 20 - Mock-Up: Mobile Sicht Festivaldetailsicht

In der mobilen Sicht soll die Navigationsleiste nun vertikal angezeigt werden. Es werden neue Abschnitte geschaffen, um vertikales Scrollen vollständig zu vermeiden. Die Informationen sollen dem Benutzer in komprimierter Form dargestellt werden, aber die Funktionen sollen alle erhalten bleiben.

3.3.2 Realisierung

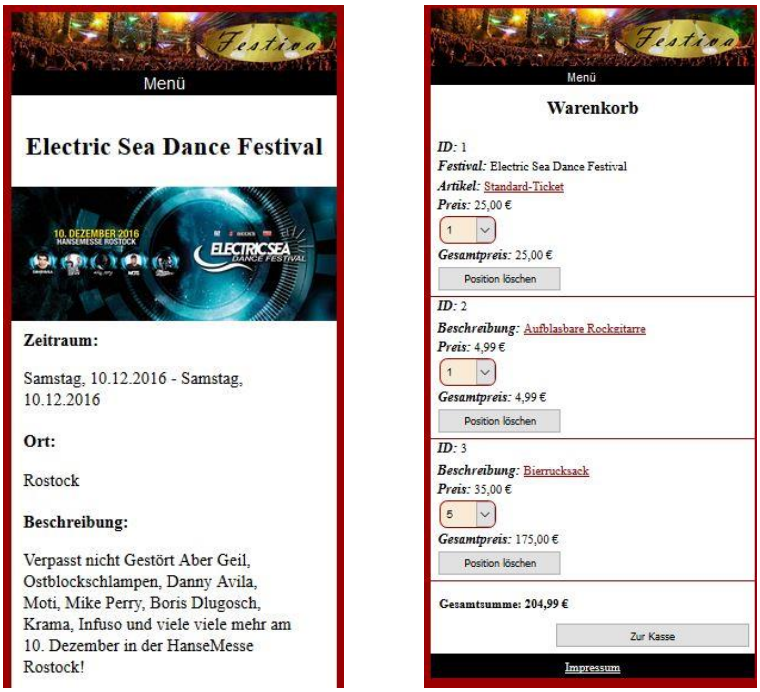

Abbildung 21 - Webshop: Festivaldetailsicht und Warenkorb responsive

Für das Responsive Design wurden Media Queries eingesetzt. Das Design passt sich an, sobald die Bildschirmgröße unter 650 Pixeln groß ist. Es öffnet sich ein vertikales Menü sobald „Menü“ angeklickt wird. Es werden unterschiedliche Breakpoints gesetzt, um normalerweise horizontal angeordnete Elemente, untereinander anzuordnen. Ebenso wird die Struktur von Tabellen vollständig aufgelöst. Die Funktionalitäten bleiben vollständig erhalten.

4 Funktionen der Anwendung

Im Rahmen der Realisierung orientierten wir uns an den Use-Case Diagrammen aus der Designphase. Während des Projektes gab es kleinere Änderungen sowie Erweiterungen, welche in den Diagrammen rot gekennzeichnet sind.

4.1 Kundenfunktionen

4.1.1 Planung

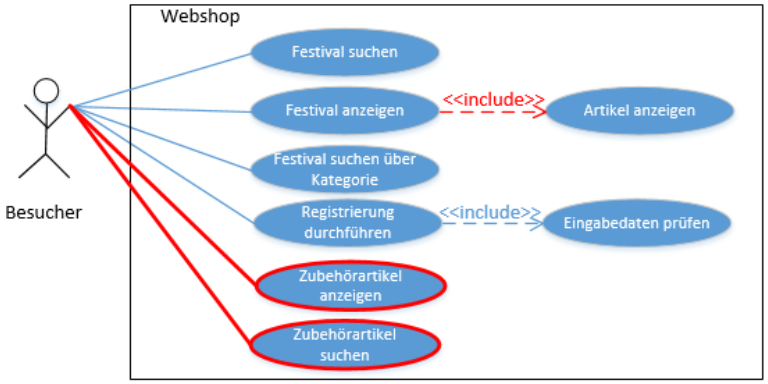

Abbildung 22 - Use-Case Diagramm: Besucher

Ein Besucher hat im Webshop nur eine begrenzte Sicht. Er kann sich alle Festival mit ihren Artikeln ansehen und diese auch über diverse Suchfunktionen suchen. Wenn er die Möglichkeit zum Kauf eines Artikels haben möchte, muss er sich registrieren. Im Laufe des Projekts wurde der Zubehörshop realisiert, in dem der Besucher ebenfalls suchen und sich einzelne Artikel anzeigen lassen können.

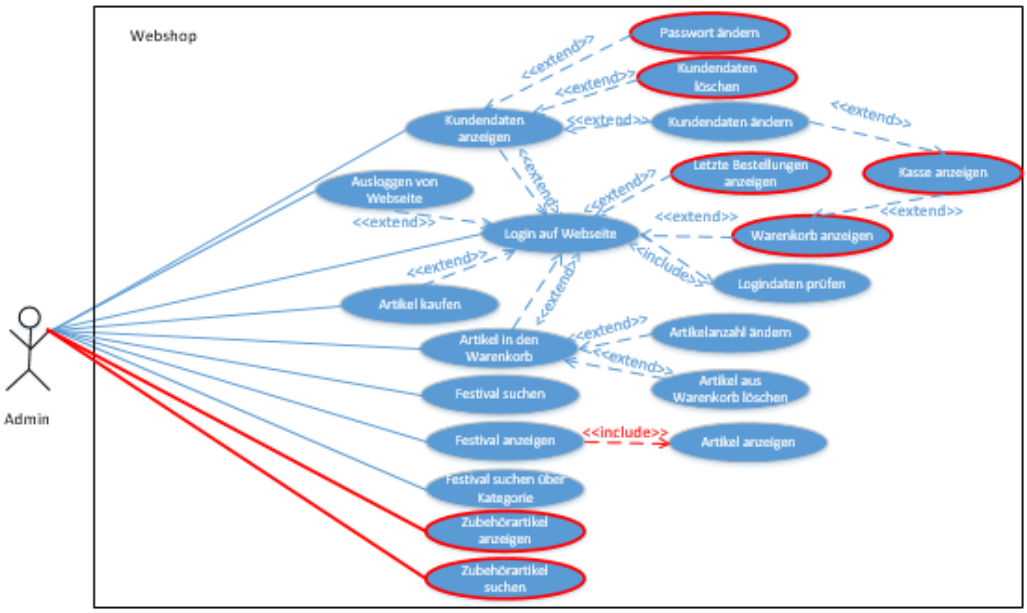

Abbildung 23 - Use-Case Diagramm: Registrierter Kunde

Zusätzlich zu den Besucherfunktionen hat ein angemeldeter Kunde weitere Möglichkeiten. Sobald man registriert ist und sich mit seinen Benutzerdaten angemeldet hat, kann eine Bestellung durchgeführt werden. Dafür können Artikel in den Warenkorb gelegt und auch wieder gelöscht werden sowie die gewünschte Artikelanzahl ausgewählt werden. Der Kunde hat zudem jederzeit die Möglichkeit, seine persönlichen Daten einzusehen und zu ändern.

Analog zum Use-Case Diagramm des Besuchers wurden auch hier weitere Use-Cases für den Zubehörshop ergänzt. Außerdem hat der Kunde nun die Möglichkeit, sich seine letzten Bestellungen anzeigen zu lassen und sein eigenes Kundenkonto zu löschen.

4.1.2 Realisierung

Im Folgenden wird die konkrete Art der Realisierung an zentralen Anwendungsfällen betrachtet. Dies soll verdeutlichen, wie der Webshop insgesamt aufgebaut ist und wie die Zusammenhänge der Klassen (siehe 2.3) gestaltet wurden.

Dieser Teil umfasst nur die grundlegenden Funktionalitäten, die der Webshop bietet. Weitere Informationen zur konkreten Implementierung können in der Java-Doc eingesehen werden.

4.1.2.1 Kunden registrieren

Verwendete JSPs:	k_registrieren, k_startseite, (k_festivaldetails, k_artikeldetails)
Verwendete Servlets:	Registrierung, Login, Produktverwaltung, (Ticketverwaltung)
Verwendete Managerklassen:	BenutzerManager, WarenkorbManager, KategorienManager, (ArtikelManager, FestivalManager)

Damit sich ein Anwender registrieren kann, werden sowohl die E-Mail-Adresse als auch das gewünschte Passwort benötigt. Beides muss zweimal eingegeben werden, damit Fehleingaben möglichst vermieden werden können. Die Prüfung auf die Kennwortrichtlinie geschieht vor dem Absenden des Formulars mit Hilfe von HTML5-Eingabefeldern und Patterns. Sobald der Anwender auf den Button „Registrieren“ klickt, wird das Servlet „Registrierung“ aufgerufen. Hier wird geprüft, ob die beiden E-Mail-Adressen und die beiden Passwörter übereinstimmen. Zusätzlich wird geprüft, ob die eingegebene E-Mail-Adresse bereits bei einem anderen Benutzer verwendet wird. Sollten die Eingaben zulässig sein, werden anschließend ein neuer Kunde sowie ein leerer Warenkorb für den Kunden erstellt. Das Passwort wird dabei als Salted SHA1-Hash in der Datenbank gespeichert. Anschließend findet eine Weiterleitung zum Login-Servlet statt und der Anwender wird automatisch angemeldet. Dann wird das Servlet „Produktverwaltung“ mit dem Parameter „**aktion=s_anzeigen**“ aufgerufen, sodass die aktuellen Kategorien ermittelt werden können und diese dem Kunden dann auf der Oberfläche für die Startseite (k_startseite) in der Slideshow präsentiert werden können.

Falls der unangemeldete Kunde durch einen Klick auf den Button „In den Warenkorb“ zu der Registrierungs-Seite gelangt ist, wird er wieder zu genau dieser Seite zurückgeleitet, damit er die Warenkorb-Aktion nun durchführen kann. In diesem Fall kommen auch die JSPs, Servlets und Manager-Klassen, die oben in Klammern angegeben wurden, zum Einsatz.

Bei einem erfolgreichen Registrierungs-Vorgang erhält der Kunde in jedem Fall eine Meldung über die erfolgreiche Registrierung und wird weitergeleitet.

Sollte der Anwender zwei ungleiche E-Mail-Adressen oder Passwörter eingegeben haben oder seine eingegebene E-Mail-Adresse bereits im System verwendet werden, erhält er eine entsprechende Fehlermeldung.

4.1.2.2 Kunden anmelden / abmelden

Verwendete JSPs:	k_anmelden, k_startseite, (k_festivaldetails, k_artikeldetails)
Verwendete Servlets:	Login, Logout, Produktverwaltung, (Ticketverwaltung)
Verwendete Managerklassen:	BenutzerManager, KategorienManager, (ArtikelManager, FestivalManager, WarenkorbManager)

Damit sich ein Kunde anmelden kann, werden die E-Mail-Adresse und das zugehörige Passwort benötigt. Wenn der Kunde diese in der Anmeldemaske (k_anmelden) eingegeben hat und auf den Button „Anmelden“ klickt, wird das Servlet „Login“ aufgerufen. Hier werden auf Grundlage der E-Mail-Adresse die Daten des Benutzers ermittelt. Sollte kein Benutzer mit dieser E-Mail-Adresse gefunden werden oder das eingegebene Passwort nicht zu dem in der Datenbank gespeicherten passen, erhält der Benutzer eine passende Fehlermeldung. Darüber hinaus erhält der Kunde auch eine Fehlermeldung, wenn er gesperrt ist oder sein Benutzerkonto gelöscht wurde. Wenn die Eingaben korrekt waren, wird

eine neue Session erstellt und der Kunde standardmäßig auf die Kundenstartseite geleitet. Dies findet analog zu dem in 4.1.2.1 beschriebenen Verfahren statt. Auch an dieser Stelle hat der Kunde die Möglichkeit, auf die Seite des Artikels zurückzukehren, wenn er unangemeldet einen Artikel in den Warenkorb legen wollte.

Beim Abmelden wird die Session des Kunden geschlossen und dieser zurück auf die Kundenstartseite geleitet.

4.1.2.3 Kundendaten ändern

Verwendete JSPs:	k_kundendaten
Verwendete Servlets:	Benutzerdaten, Logout, Produktverwaltung
Verwendete Managerklassen:	BenutzerManager, KategorienManager

Wenn der Kunde über „Mein Konto“ zu „Meine Daten“ navigiert, wird das Servlet „Benutzerdaten“ mit dem Parameter „**aktion=anzeigen**“ aufgerufen. Zu dem aktuellen Kunden werden dann die Daten ermittelt und der Kunde wird auf die passende Oberfläche (k_kundendaten), auf der seine aktuellen persönlichen Daten angezeigt werden, weitergeleitet. Möchte der Kunde seine Daten ändern, tut er dies in der Benutzeroberfläche und klickt dann auf den Button „Änderungen speichern“. Dadurch wird das Servlet „Benutzerdaten“ mit dem Parameter „**aktion=aendern**“ aufgerufen. Anschließend wird im Servlet ein grober Teil zur Validierung (Der Großteil findet bereits zuvor mit HTML5-Eingabefeldern und Patterns statt) und die Verarbeitung der aktuellen Daten durchgeführt. Danach wird der Kunde wieder zur Anzeige der aktuellen Daten geleitet. Hier erhält er eine Rückmeldung. Entweder konnten die eingegebenen Daten erfolgreich gespeichert werden oder es erscheint eine Fehlermeldung, weil beispielsweise eine E-Mail-Adresse, die bereits einem anderen Kunden zugeordnet wurde, eingegeben wurde und die Änderung deswegen nicht durchgeführt werden konnte.

Zusätzlich hat der Kunde die Möglichkeit, sein Passwort zu ändern. Erforderlich dafür sind die Eingabe seines alten Passworts und eine zweimalige Eingabe des neuen, gewünschten Passworts. Mit dem Klick auf den Button „Passwort ändern“, wird das Servlet „Benutzerdaten“ mit dem Parameter „**aktion=p_aendern**“ aufgerufen. Die Prüfung auf die Kennwortrichtlinie geschieht dabei wieder vor dem Absenden des Formulars. Im Servlet findet dann die Prüfung statt, ob das eingegebene alte Passwort korrekt ist und ob die beiden Eingaben für das neue Passwort gleich sind. Wenn die Eingaben korrekt sind, werden die Änderungen durchgeführt und der Kunde erhält eine entsprechende Rückmeldung. Wenn eine Prüfung einen Fehler wirft, erhält der Kunde eine passende Fehlermeldung.

Darüber hinaus hat der Kunde die Möglichkeit, sein Konto zu löschen. Mit dem Klick auf den Button „Mein Benutzerkonto löschen“ und der anschließenden Bestätigung des Pop-Ups wird das Servlet „Benutzerdaten“ mit dem Parameter „**aktion=loeschen**“ aufgerufen. Im Servlet wird der Kunde dann logisch gelöscht.

Danach findet eine Weiterleitung zu dem Logout-Servlet statt, sodass die aktuelle Session des Kunden auch automatisch beendet wird. Der Kunde wird zum Schluss wieder auf die Kunden-Startseite geleitet. Die Anzeige der Startseite folgt dem Verfahren in 4.1.2.1.

4.1.2.4 Festivals suchen

Verwendete JSPs:	k_ticketShop
Verwendete Servlets:	Ticketverwaltung
Verwendete Managerklassen:	FestivalManager, KategorienManager

Bei der Suche kann der Kunde verschiedene Suchkriterien (Name, Kategorie, Ort, Maximalpreis, Von- & Bis-Datum) in der Suchmaske der Oberfläche (k_ticketShop) eingeben und miteinander kombinieren. Mit dem Klick auf den Button „Suchen“ wird das Servlet „Ticketverwaltung“ mit dem Parameter „**aktion=t_anzeigen**“ aufgerufen. Die Validierung der Benutzer-Eingaben findet wieder primär vor dem Absenden des Formulars statt. Im Servlet wird nur noch geprüft, ob das Startdatum vor dem Enddatum liegt oder gleich dem Enddatum ist. Ist dem nicht so, wird die Suche nicht ausgeführt

und der Kunde erhält eine passende Fehlermeldung. Sollten die Eingaben korrekt sein, werden die Festivals, die den Eingaben entsprechen, ermittelt und an die JSP zurückgegeben, sodass diese auf der Oberfläche des Ticket-Shops angezeigt werden können.

4.1.2.5 Festivaldetails anzeigen

Verwendete JSPs:	k_ticketShop, k_festivaldetails
Verwendete Servlets:	Ticketverwaltung
Verwendete Managerklassen:	FestivalManager, WarenkorbManager, ArtikelManager

Hat der Kunde ein Festival ausgewählt (Klick auf den Namen oder das Bild eines Suchergebnisses aus 4.1.2.4), wird das Servlet „Ticketverwaltung“ mit dem Parameter „**aktion=f_anzeigen**“ aufgerufen. Hier werden alle nötigen Daten zu dem Festival und auch (falls ein Kunde in der Session eingeloggt ist) der Warenkorbinhalt des aktuellen Kunden. Danach wird eine Liste aller verfügbaren Artikel des Festivals ermittelt.

Wird ein Maximalpreis angegeben, werden zwei Artikellisten erzeugt. Die eine Liste beinhaltet alle Artikel, die kleiner oder gleich dem Maximalpreis sind und die andere Artikelliste alle Artikel, die über dem maximalen Preis liegen.

Die JSP „k_festivaldetails“ wird anschließend mit den ermittelten Daten aufgerufen und kann dem Anwender das Ergebnis anzeigen.

4.1.2.6 Zubehör-Artikel suchen

Verwendete JSPs:	k_zuebhoerShop
Verwendete Servlets:	Produktverwaltung
Verwendete Managerklassen:	ArtikelManager

Im Zubehör-Shop kann der Kunde nur eine einfache Suche nach der Beschreibung durchführen. Diese wurde mittels Java-Script realisiert. Mit dem initialen Klick auf den Zubehör-Shop wird zunächst das Servlet „Produktverwaltung“ mit dem Parameter „**aktion=z_anzeigen**“ aufgerufen, sodass alle Artikel, die zum Verkauf stehen, ermittelt werden können und an die Oberfläche „k_zubehoerShop“ übergeben werden können.

4.1.2.7 Zubehör-Artikel anzeigen

Verwendete JSPs:	k_zubehoerShop, k_artikeldetails
Verwendete Servlets:	Produktverwaltung
Verwendete Managerklassen:	WarenkorbManager, ArtikelManager

Hat der Kunde einen Zubehör-Artikel ausgewählt (Klick auf den Namen oder das Bild eines Suchergebnisses aus 4.1.2.6), wird das Servlet „Produktverwaltung“ mit dem Parameter „**aktion=a_anzeigen**“ aufgerufen. Hier werden alle nötigen Daten zu dem Artikel und auch (falls ein Kunde in der Session eingeloggt ist) der Warenkorbinhalt des aktuellen Kunden.

Die JSP „k_artikeldetails“ wird anschließend mit den ermittelten Daten aufgerufen und kann dem Anwender das Ergebnis anzeigen.

4.1.2.8 Artikel zum Warenkorb hinzufügen

Verwendete JSPs:	k_festivaldetails, k_artikeldetails
Verwendete Servlets:	Warenkorbverwaltung, Ticketverwaltung; Produktverwaltung
Verwendete Managerklassen:	WarenkorbManager, ArtikelManager, FestivalManager

Möchte der Kunde einen Artikel in den Warenkorb legen, muss er die gewünschte Menge auswählen und klickt auf den Button „In den Warenkorb“. Sollte der Anwender noch nicht eingeloggt sein, wird er zu der Anmelden-Seite weitergeleitet (siehe 4.1.2.2). Anschließend wird geprüft, ob sich der gewünschte Artikel bereits im Warenkorb befindet.

1. Ist dieses der Fall erhält der Kunde ein Pop-Up mit der Nachfrage, ob der Artikel trotzdem dem Warenkorb hinzugefügt werden soll. Wird dies bestätigt, wird das Servlet „Warenkorbverwaltung“ mit dem Parameter „**aktion=aktualisieren**“ aufgerufen. Im Servlet wird anschließend der passende Artikel und das Warenkorbelement des Kunden, welches sich auf den Artikel bezieht, ermittelt. Anschließend wird die Anzahl um die gewünschte Menge erhöht und der Kunde erhält eine Rückmeldung über das Aktualisieren der Anzahl des Artikels. Sollte bei dem Vorgang die maximale Anzahl von 10 überschritten werden, wird der Vorgang abgebrochen und der Kunde erhält eine passende Fehlermeldung.

2. Wenn der Artikel sich noch nicht im Warenkorb befindet, wird das Servlet „Warenkorbverwaltung“ mit dem Parameter „**aktion=hinzufuegen**“ aufgerufen. Im Servlet wird dem Warenkorb des aktuellen Kunden anschließend ein neues Warenkorbelement mit der ausgewählten Menge hinzugefügt. Anschließend erhält der Kunde eine Rückmeldung zu dem erfolgreichen Hinzufügen des Artikels.

Der Kunde wird in jedem Fall zurück zu der Detail-Seite (k_festivaldetails) des Festivals geleitet und erhält dort eine Rückmeldung.

4.1.2.9 Warenkorb anzeigen / bearbeiten

Verwendete JSPs:	k_warenkorb, k_festivaldetails, k_artikeldetails
Verwendete Servlets:	Warenkorbverwaltung, Ticketverwaltung, Produktverwaltung
Verwendete Managerklassen:	WarenkorbManager, ArtikelManager, FestivalManager

Zur Anzeige des Warenkorbs wird das Servlet „Warenkorbverwaltung“ mit dem Parameter „**aktion=anzeigen**“ aufgerufen. Anschließend werden alle Warenkorbelemente des Warenkorbs des aktuell angemeldeten Kunden ermittelt. Dann findet eine Weiterleitung an die JSP „k_warenkorb“ statt. Diese stellt dann die übermittelten Daten dar und bietet dem Kunden beispielsweise die Möglichkeit durch einen Klick auf den Artikelnamen wieder in die Detail-Informationen zu dem Artikel navigieren zu können.

Wenn ein Kunde den Warenkorb bearbeiten möchte, muss er auf der Warenkorb-Oberfläche entweder die Menge des Artikels über das Dropdown-Menü ändern oder auf den „Löschen“-Button klicken. Ändert er die Menge, wird das Servlet „Warenkorbverwaltung“ mit dem Parameter „**aktion=aendern**“ aufgerufen. Dann wird bei dem betroffenen Warenkorbelement die Menge angepasst. Klickt der Kunde auf den „Löschen“-Button, wird das Servlet „Warenkorbverwaltung“ mit dem Parameter „**aktion=loeschen**“ aufgerufen. Dann wird das betroffene Warenkorbelement aus dem Warenkorb des Kunden gelöscht.

In jedem Fall wird der Kunde nach einer Änderung oder Löschung wieder zurück zur Warenkorb-Oberfläche mit den aktuellen Daten (Aufruf von „Warenkorbverwaltung“ mit dem Parameter „**aktion=anzeigen**“) geleitet.

4.1.2.10 Bestellung durchführen

Verwendete JSPs:	k_warenkorb, k_kasse
Verwendete Servlets:	Warenkorbverwaltung, Bestellverwaltung
Verwendete Managerklassen:	BestellungsManager, WarenkorbManager

Von dem Warenkorb aus kann der Kunde mit dem Button „Zur Kasse“ weiter navigieren. Diese Möglichkeit bietet sich dem Kunden allerdings nur, wenn er Artikel in seinem Warenkorb hat. Anderenfalls ist der Button deaktiviert.

Mit dem Klick auf den Button wird das Servlet „Warenkorbverwaltung“ mit dem Parameter „**aktion=k_anzeigen**“ aufgerufen. An dieser Stelle wird geprüft, ob die Bestellung festivalübergreifende Artikel beinhaltet. In diesem Fall wird der Post-Versand festgesetzt. Andernfalls wird der kostenlose Mail-Versand als Standard vorgeschlagen und kann später noch durch den Kunden geändert werden. Zusätzlich wird geprüft, ob alle Kundendaten für den Abschluss einer Bestellung vorliegen. In einem letzten Schritt werden alle Warenkorbelemente ermittelt und alle Daten an die JSP „k_kasse“ übergeben. Hier werden die Daten dann dargestellt und es wird beispielsweise im Fall von unvollständigen Daten ein entsprechender Hinweis eingeblendet sowie die Möglichkeit zur Änderung der Daten geboten. Der Button „Verbindlich bestellen“ ist in diesem Fall deaktiviert.

Wenn jedoch alle Daten vorliegen, der Button „Verbindlich bestellen“ verfügbar ist und der Kunde auf den Button klickt und das erscheinende Pop-Up-Fenster bestätigt, wird das Servlet „Bestellverwaltung“ mit dem Parameter „**aktion=anlegen**“ aufgerufen. Aus den übermittelten Daten wird eine Bestellung mit den entsprechenden Bestellpostionen angelegt. Anschließend wird das Servlet „Bestellverwaltung“ mit dem Parameter „**aktion=anzeigen**“ aufgerufen. Hier werden alle bisherigen Bestellungen des aktuell angemeldeten Kunden ermittelt und an die JSP „k_bestellungen“ übergeben, um diese darzustellen.

4.2 Administrationsfunktionen

Zur Planung: Der Admin hat nach seiner Anmeldung die Möglichkeit zur Kunden-, Kategorien-, Festival- und Artikelverwaltung sowie zur Administration zu navigieren. Zur Übersicht wurde für jede Möglichkeit ein neues Use-Case Diagramm erstellt.

Zur Realisierung: Die konkrete Art der Realisierung wird an zentralen Anwendungsfällen betrachtet. Dies soll verdeutlichen, wie der Webshop insgesamt aufgebaut ist und wie die Zusammenhänge der Klassen (siehe 2.3) gestaltet wurden. Dieser Teil umfasst nur die grundlegenden Funktionalitäten, die der Webshop bietet. Weitere Informationen zur konkreten Implementierung können in der Java-Doc eingesehen werden.

4.2.1 Kundenverwaltung

4.2.1.1 Planung

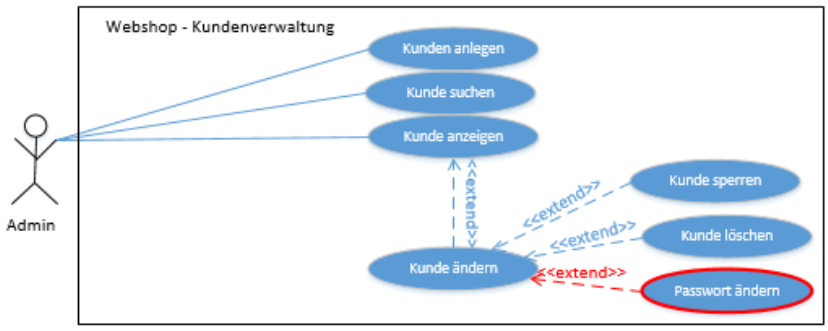

Abbildung 24 - Use-Case Diagramm: Kundenverwaltung

Ein Admin kann einen Kunden anlegen, suchen, anzeigen und über den Änderungsmodus sperren und löschen. Des Weiteren wurde eine Funktion zum Passwort ändern implementiert.

4.2.1.2 Realisierung

Verwendete JSPs:	a_kundenverwaltung, a_kundeAendern, a_kundeAnlegen
Verwendete Servlets:	Kundenverwaltung, Registrierung
Verwendete Managerklassen:	BenutzerManager

In der Kundenverwaltung kann der Admin alle Kunden einsehen. Dazu wird zu Beginn das Servlet „Kundenverwaltung“ mit dem Parameter „**aktion=anzeigen**“ aufgerufen. Alle Kunden werden ermittelt und an die JSP „a_kundenverwaltung“ übergeben. Das Suchen nach Kunden wurde mittels Javascript realisiert. Sobald der Admin auf den Button „Neuen Kunden anlegen“ klickt, wird die JSP „a_kundeAnlegen“ aufgerufen. Nach der Eingabe der benötigten Daten (Validierung zunächst direkt auf der JSP) kann das Formular mit dem Klick auf den Button „Anlegen“ abgeschickt werden. Dadurch wird das Servlet „Registrierung“ aufgerufen. Der Ablauf hier gleicht dem in 4.1.2.1. Der Admin wird anschließend allerdings wieder zurück zu der JSP „a_kundeAnlegen“ geleitet und erhält hier eine Information zur Anlage des Kunden. Diese Weiterleitung wird auf Basis der Existenz und des Werts des Session-Attributs „gruppenid“ entschieden.

Sollen Kundendaten geändert werden, muss der Admin in der Kundenverwaltung auf die ID des Kunden klicken. Dabei wird das Servlet „Kundenverwaltung“ mit dem Parameter „**aktion=aendern**“ aufgerufen. Alle Daten des gewünschten Kunden werden ermittelt und an die JSP „a_kundenAendern“ übergeben. Hier hat der Admin die Möglichkeit, Werte zu ändern und diese mit dem Button „Änderungen speichern“ zu sichern (Erste Validierungen finden direkt in der JSP statt). Dabei wird das Servlet „Kundenverwaltung“ mit dem Parameter „**aktion=datenaendern**“ aufgerufen. Nach einer weiteren Validierung werden die Daten gesichert und die neuen Daten sowie eine Rückmeldung zu der Änderung in der JSP „a_kundenAendern“ angezeigt. Das Passwort des Kunden kann durch die zweimalige Eingabe des neuen Passworts mit dem Klick auf den Button „Passwort ändern“ geändert werden. Es wird das Servlet „Kundenverwaltung“ mit dem Parameter „**aktion=pw_aendern**“ aufgerufen.

In der Oberfläche zur Bearbeitung des Kunden hat der Admin außerdem die Möglichkeit, den Kunden durch einen Klick auf den Button „Kunden löschen“, zu löschen. Nach Bestätigung des Pop-Ups wird das Servlet „Kundenverwaltung“ mit dem Parameter „**aktion=loeschen**“ aufgerufen und damit der Kunde logisch gelöscht. Der Admin erhält eine Rückmeldung über das Löschen des Kunden und gelangt wieder in die Kunden-Ändern-Ansicht über die JSP „a_kundenAendern“.

4.2.2 Kategorienverwaltung

4.2.2.1 Planung

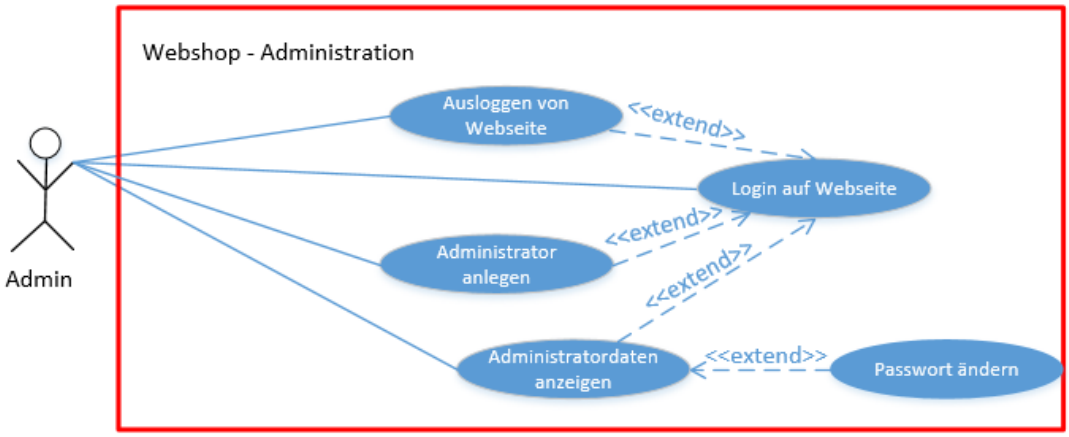

Abbildung 25 - Use-Case Diagramm: Kategorienverwaltung

Ein Admin kann eine Kategorie anlegen, suchen, anzeigen und über den Änderungsmodus löschen. Nachträglich wurde noch eine „Bild löschen“-Funktion implementiert.

4.2.2.2 Realisierung

Verwendete JSPs:	a_kategorienverwaltung, a_kategorieAendern, a_kategorieAnlegen
Verwendete Servlets:	Kategorienverwaltung
Verwendete Managerklassen:	KategorienManager

In der Kategorienverwaltung kann der Admin alle Kategorien einsehen. Da alle Verwaltungen im Admin-Bereich einheitlich gestaltet wurden, findet die Verwendung des Parameters „**aktion**“ mit den Werten „**anzeigen**, **anlegen**, **aendern**, **datenaendern** & **loeschen**“ analog statt. Das verwendete Servlet ist „Kategorienverwaltung“.

Zusätzlich findet sich im Bereich der Kategorienverwaltung die Möglichkeit, das Bild einer Kategorie löschen zu können. Dazu muss der Admin in der Änderungs-Ansicht für Kategorien auf den Button „Aktuelles Bild löschen“ klicken. Dann wird das Servlet „Kategorienverwaltung“ mit dem Parameter „**aktion=b_loeschen**“ aufgerufen. Nach der Durchführung wird der Admin dann wieder zurück zu der Änderungs-Ansicht der Kategorie geleitet, erhält eine Rückmeldung zum Löschen des Bildes und sieht, dass kein Bild mehr verfügbar ist.

Besonders zu beachten ist, dass die ersten vier Kategorien (ID 1-4) nicht gelöscht werden können, da dies Standardkategorien sind. Aus diesem Grund kann bei diesen Kategorien auch kein Bild gelöscht werden, da dieses für die Darstellung innerhalb der Slideshow benötigt wird. Die Slideshow kann daher zwar um Kategorien erweitert oder verkleinert werden, beinhaltet aber immer mindestens die vier Standardkategorien.

4.2.3 Festivalverwaltung

4.2.3.1 Planung

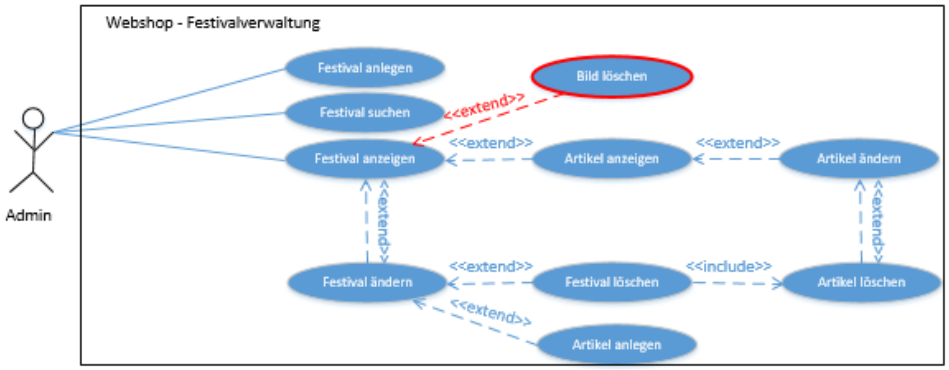

Abbildung 26 - Use-Case Diagramm: Festivalverwaltung

Die Festivalverwaltung weist ebenfalls die Standardfunktionen zum Anlegen, Suchen, Anzeigen und Ändern und Löschen von Festivals und ihren zugehörigen Artikeln auf. Hinzukommt, dass automatisch alle zu dem Festival zugehörigen Artikel gelöscht werden sobald das Festival gelöscht wird. Im Änderungsmodus eines Festivals wurde ebenfalls noch die Funktion „Bild löschen“ implementiert.

4.2.3.2 Realisierung

Verwendete JSPs:	a_festivalverwaltung, a_festivalAendern, a_festivalAnlegen, a_artikelAnlegen, a_artikelAendern
Verwendete Servlets:	Festivalverwaltung, Kategorienverwaltung, Artikelverwaltung
Verwendete Managerklassen:	FestivalManager, KategorienManager, ArtikelManager

In der Festivalverwaltung kann der Admin alle Festivals einsehen. Auch diese Verwaltung verwendet die bereits erläuterte Verwendung des Parameters „**aktion**“ mit den Werten „**anzeigen**, **anlegen**, **aendern**, **datenaendern, b_loeschen** & **loeschen**“ innerhalb des Servlets „Festivalverwaltung“.

Zusätzlich findet sich im Bereich der Festivalverwaltung die Möglichkeit, Artikel mit Verbindung zu einem Festival anlegen, ändern und löschen zu können. Dazu muss sich der Admin in der Änderungs-Ansicht für Festivals befinden. Dort befindet sich eine „kleine Artikelverwaltung“. Das bedeutet, dass auch hier der Parameter „**aktion**“ mit den Werten „**anlegen**, **aendern** & **datenaendern**“ verwendet wird. Dies geschieht dann innerhalb des Servlets „Artikelverwaltung“.

4.2.4 Artikelverwaltung für festivalübergreifende Artikel

4.2.4.1 Planung

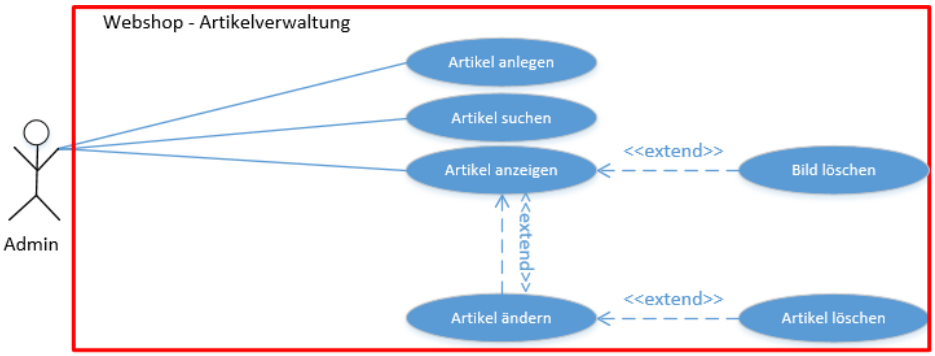

Abbildung 27 - Use-Case Diagramm: Artikelverwaltung

In der Artikelverwaltung können festivalübergreifende Artikel verwaltet werden. Dieses Use-Case wurde zu Beginn nicht modelliert, da ein weiterer Shop für Zubehör ein Kann-Kriterium war. Im Laufe des Projektes entschieden wir uns dazu, den Shop zu implementieren.

4.2.4.2 Realisierung

Verwendete JSPs:	a_artikelverwaltung, a_artikelAendern, a_artikelAnlegen
Verwendete Servlets:	Artikelverwaltung
Verwendete Managerklassen:	ArtikelManager

In der Artikelverwaltung kann der Admin alle festivalunabhängigen Artikel einsehen. Auch diese Verwaltung verwendet die bereits erläuterte Verwendung des Parameters „**aktion**“ mit den Werten „**anzeigen**, **anlegen**, **aendern**, **datenaendern, b_loeschen** & **loeschen**“ innerhalb des Servlets „Artikelverwaltung“.

Das Hochladen und Löschen des Bildes sowie das generelle Löschen des Artikels mit der ID 6 ist nicht möglich, da dieser unseren Postversand darstellt. Daher wird der Artikel auch nicht in dem „Zubehör-Shop“ zur Auswahl dargestellt.

4.2.5 Administration

4.2.5.1 Planung

Abbildung 28 - Use-Case Diagramm: Administration

Für die Administration wurde zu Beginn kein Use-Case Diagramm modelliert. Dadurch, dass die Funktion einen Admin anzulegen hinzukam, wurde dieses Diagramm im Nachgang modelliert.

4.2.5.2 Realisierung

Verwendete JSPs:	a_adminKonto, a_adminAnlegen
Verwendete Servlets:	Benutzerdaten, Registrierung
Verwendete Managerklassen:	BenutzerManager

Um administrative Tätigkeiten durchführen zu können, hat der Admin die Möglichkeit, in den Administrations-Bereich zu navigieren. Hier kann er sein eigenes Passwort ändern oder einen neuen Admin anlegen.

Bei der Anlage eines neuen Passworts in der Oberfläche „a_adminKonto“ muss der Admin dieselben Daten eingeben, wie der Kunde auch (siehe 4.1.2.3). Es wird das Servlet „Benutzerdaten“ mit dem Parameter „**aktion=p_aendern**“ aufgerufen. Der Ablauf gleicht also der Passwortänderung eines Kunden. Die Zurückleitung auf die richtige Seite nach der Änderung findet über die Ermittlung der Gruppe, der der aktuell angemeldete User angehört, statt.

Die Anlage eines neuen Admins findet in der Oberfläche „a_adminAnlegen“ statt. Es werden sowohl die E-Mail-Adresse als auch das gewünschte Passwort benötigt. Beides muss zweimal eingegeben werden, damit Fehleingaben möglichst vermieden werden können. Die Prüfungen gleichen denen, die bei der Registrierung eines Kunden (siehe 4.1.2.1) durchgeführt werden. Sobald der Admin auf den Button „Anlegen“ klickt, wird das Servlet „Registrierung“ mit dem Parameter „**aktion=a_anlegen**“ aufgerufen. Hier wird geprüft, ob die beiden E-Mail-Adressen und die beiden Passwörter übereinstimmen. Zusätzlich wird geprüft, ob die eingegebene E-Mail-Adresse bereits bei einem anderen Benutzer verwendet wird. Sollten die Eingaben richtig sein, wird anschließend ein neuer Admin erstellt. Das Passwort wird dabei als Salted SHA1-Hash in der Datenbank gespeichert

5 Test

Das Testen durchzog das gesamte Projekt. Während der Realisierung konnten Fehler im Sinne von White-Box Tests gefunden und behoben werden. Unter Kenntnis des Programmcodes und mit Hilfe des Eclipse Debuggers wurde bereits in dieser Phase ein Großteil des Testaufwands abgewickelt.

Mit Hilfe des Testplans (s. Anhang) wurden zum Abschluss der Realisierung Black-Box Tests durchgeführt. Hier wurden alle Funktionen ohne Kenntnis bzw. ohne Berücksichtigung des Programmcodes getestet.

Außenstehende Personen testeten den Webshop ebenfalls, da man als Projektverantwortliche oftmals eine eingeschränkte Sicht hat. Insgesamt traten jedoch keine größeren Bugs auf.

6 Projektmanagement

Im Rahmen der Projektplanung wurde das Carl-Steinweg-Phasenmodell eingesetzt:

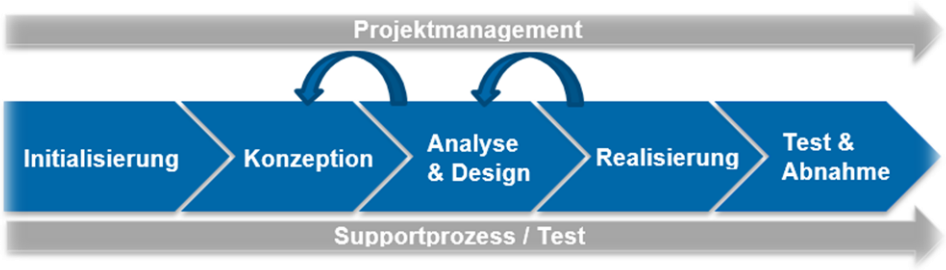

Abbildung 29 - Carl-Steinweg Phasenmodell

Der Projektmanagementbereich diente während des gesamten Projektzeitraumes der Leitung und Steuerung des Projektes. Der blau dargestellte Kernbereich war in mehrere Phasen, in denen es gegebenenfalls Rücksprünge geben konnte, gegliedert. Parallel dazu verlief der Support- bzw. Testprozess, der insbesondere eine frühe Fehlererkennung ermöglicht hat.

6.1 Soll-/ Ist-Vergleich

6.1.1 Erfüllung Muss/Kann-Kriterien

Während der Projektlaufzeit konnten alle Muss-Kriterien und einige Kann-Kriterien implementiert werden. Folgende Kann-Kriterien wurden umgesetzt:

- Kunden werden automatisch gesperrt (z.B. nach 3-mal falsch eingegebenem Kennwort)
- Kennwortrichtlinien festlegen und prüfen
- Anlage von Admin-Konten über eine Oberfläche
- Einbinden von zusätzlichen Festival-Artikeln (Regencapes etc.)

6.1.2 Projektinitialisierung

Vorgangsname	Anfang	Ende	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Projektinitialisierung	**05.10.16**	**05.10.16**	**3 Std.**	**3 Std.**	**0**
Kickoff-Meeting	05.10.16	05.10.16	3 Std.	3 Std.	0

Das Projekt begann mit der Phase Projektinitialisierung, in der der Grundstein für den späteren Projekterfolg gelegt wurde. Dazu wurde die Projektidee in einem Kickoff-Meeting ausgearbeitet und präzisiert.
Das Ergebnis dieser Phase war somit die Festlegung von Rahmenbedingungen und Projektzielen. Der dabei geplante Aufwand von 3 Stunden konnte eingehalten werden.

6.1.3 Konzeption

Vorgangsname	Anfang	Ende	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Projektinitialisierung	**05.10.16**	**05.10.16**	**3 Std.**	**3 Std.**	**0**
Konzeption	**06.10.16**	**07.10.16**	**22 Std.**	**22 Std.**	**0**
Pflichtenheft	06.10.16	06.10.16	6 Std.	7 Std.	+1
Projektplan	07.10.16	07.10.16	16 Std.	15 Std.	-1

Nach dem erfolgreichen Abschluss der Phase Projektinitialisierung gingen wir in die Phase Konzeption über. In dieser Phase erstellten wir ein Pflichtenheft und einen Projektplan. Durch das gemeinsame Verständnis nach dem Gespräch in der vorherigen Phase konnte der geplante Zeitraum trotz einzelner Abweichungen auch in dieser Phase insgesamt eingehalten werden.

6.1.4 Design

Vorgangsname	Anfang	Ende	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Projektinitialisierung	**05.10.16**	**05.10.16**	**3 Std.**	**3 Std.**	**0**
Konzeption	**06.10.16**	**07.10.16**	**22 Std.**	**22 Std.**	**0**
Design	**07.10.16**	**14.10.16**	**53 Std.**	**54 Std.**	**+1**
Festlegen der Namens-konventionen	07.10.16	07.10.16	3 Std.	3 Std.	0
Erstellung ER-Diagramm	07.10.16	12.10.16	7 Std.	6 Std.	-1
Erstellung Klassen-diagramm	07.10.16	12.10.16	13 Std.	13 Std.	0
Erstellung Use-Case-Diagramme	07.10.16	12.10.16	8 Std.	9 Std.	+1
Erstellung von Mock-Ups für die Oberfläche	07.10.16	13.10.16	10 Std.	11 Std.	+1
Entwurf Testplan	07.10.16	13.10.16	6 Std.	5 Std.	-1
Zusammenfassung Diagramme / Anfor-derungen / Testplan	14.10.16	14.10.16	6 Std.	7 Std.	+1

Nach Abschluss der Phase Konzeption starteten wir mit der Phase Design. Hier erarbeiteten wir den Aufbau des Webshops und damit die Basis für das Vorgehen in der Phase Realisierung. Dabei haben wir uns insgesamt bemüht, sehr sauber zu modellieren. Aus diesem Grund haben wir eine Stunde länger als geplant benötigt. Wir erhofften uns, dadurch möglichst viele Fehler im Voraus vermeiden zu können.

6.1.5 Realisierung

Vorgangsname	Anfang	Ende	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Projektinitialisierung	**05.10.16**	**05.10.16**	**3 Std.**	**3 Std.**	**0**
Konzeption	**06.10.16**	**07.10.16**	**22 Std.**	**22 Std.**	**0**
Design	**07.10.16**	**14.10.16**	**53 Std.**	**54 Std.**	**+1**
Realisierung	**14.10.16**	**10.11.16**	**176 Std.**	**190 Std.**	**+14**
Installation & Konfi-guration von Eclipse, Tomcat, MySQL…	14.10.16	17.10.16	15 Std.	12 Std.	-3
Erstellung der Tabel-lendefinitionen in der Datenbank	18.10.16	18.10.16	4 Std.	4 Std.	0
Datenbank-anbindung	18.10.16	18.10.16	6 Std.	4 Std.	-2

Umsetzung Oberfläche	18.10.16	27.10.16	35 Std.	45 Std.	+10
Oberflächen-Logik	18.10.16	31.10.16	21 Std.	25 Std.	+4
Backend-Entwicklung	19.10.16	04.11.16	50 Std.	50 Std.	0
Verbindung Frontend & Backend	07.11.16	09.11.16	10 Std.	15 Std.	+5
Responsive Design	28.10.16	04.11.16	29 Std.	29 Std.	0
Dokumentation / Refactoring	10.11.16	10.11.16	6 Std.	6 Std.	0

Nach der ausführlichen Planung und Modellierung in den vorherigen Phasen starteten wir in die Realisierung. Zuerst begannen wir mit der Realisierung und Gestaltung der Oberflächen sowie der Realisierung der reinen Backend-Logik. Nach und nach stellten wir eine Verbindung zwischen den Oberflächen und den konkreten Daten her, indem wir eine Kommunikation über Servlets integrierten.

In dieser Phase wurde uns bewusst, dass wir die Einarbeitungszeit im Vorfeld nicht ausreichend bedacht hatten. Zusätzlich entstanden Probleme in Bezug auf die Planung und Absprachen innerhalb des Teams. Darüber hinaus haben wir, wie im Carl-Steinweg-Phasenmodell vorgesehen, sehr viel mit Hilfe von White-Box-Tests getestet, um Fehler möglichst früh erkennen zu können und haben auch hier zusätzliche Zeit gebraucht.

Aus den oben genannten Gründen benötigten wir in dieser Phase 14 Stunden länger als geplant. Wir rechneten jedoch damit, diese Zeit in der Phase Test durch die bereits durchgeführten Tests und die frühen Fehlerbehandlungen in dieser Phase wieder relativieren zu können.

6.1.6 Test

Vorgangsname	Anfang	Ende	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Projektinitialisierung	**05.10.16**	**05.10.16**	**3 Std.**	**3 Std.**	**0**
Konzeption	**06.10.16**	**07.10.16**	**22 Std.**	**22 Std.**	**0**
Design	**07.10.16**	**14.10.16**	**53 Std.**	**54 Std.**	**+1**
Realisierung	**14.10.16**	**10.11.16**	**176 Std.**	**190 Std.**	**+14**
Test	**11.11.16**	**17.11.16**	**33 Std.**	**28,5 Std.**	**-4,5**
Testkonzeption	11.11.16	11.11.16	15 Std.	12 Std.	-3
Testdaten pflegen	14.11.16	15.11.16	8 Std.	8,5 Std.	+0,5
Integrationstest	16.11.16	17.11.16	10 Std.	8 Std.	-2

Im Anschluss an die Realisierung konnten wir in die Phase Test übergehen. Diese konnte schneller abgeschlossen werden als ursprünglich geplant. Das lag daran, dass während der Realisierung viele White-Box Tests durchgeführt wurden. Beim Back-Box Test traten daher nur noch kleinere, schnell behebbare Fehler auf. Insgesamt konnten wir in dieser Phase 4,5 Stunden einsparen.

6.1.7 Dokumentation

Vorgangsname	Anfang	Ende	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Projektinitialisierung	**05.10.16**	**05.10.16**	**3 Std.**	**3 Std.**	**0**
Konzeption	**06.10.16**	**07.10.16**	**22 Std.**	**22 Std.**	**0**

Design	**07.10.16**	**14.10.16**	**53 Std.**	**54 Std.**	**+1**
Realisierung	**14.10.16**	**10.11.16**	**176 Std.**	**190 Std.**	**+14**
Test	**11.11.16**	**17.11.16**	**33 Std.**	**28,5 Std.**	**-4,5**
Dokumentation	**18.11.16**	**05.12.16**	**45 Std.**	**45 Std.**	0
Soll- / Ist-Vergleich	18.11.16	18.11.16	5 Std.	5 Std.	0
Präsentation	18.11.16	21.11.16	15 Std.	10 Std.	-5
Dokumentation	22.11.16	05.12.16	25 Std.	30 Std.	+5

Nachdem wir auch die Phase Test abgeschlossen hatten, konnten wir in die Phase Dokumentation starten. Innerhalb dieser Phase konnten zunächst fünf Stunden im Rahmen der Erstellung der Präsentation eingespart werden. Auf Grund des Umfangs des Gesamtprojekts benötigten wir im Rahmen der Erstellung der Dokumentation jedoch fünf Stunden länger als zuvor geplant. Weiterer Aufwand entstand durch die verteilte Kenntnis des Systems innerhalb des Teams. Insgesamt konnten wir die Phase jedoch mit dem geplanten Aufwand von 45 Stunden abschließen.

6.1.8 Gesamtbetrachtung Projektplan

Vorgangsname	Anfang	Ende	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Projektinitialisierung	**05.10.16**	**05.10.16**	**3 Std.**	**3 Std.**	**0**
Konzeption	**06.10.16**	**07.10.16**	**22 Std.**	**22 Std.**	**0**
Design	**07.10.16**	**14.10.16**	**53 Std.**	**54 Std.**	**+1**
Realisierung	**14.10.16**	**10.11.16**	**176 Std.**	**190 Std.**	**+14**
Test	**11.11.16**	**17.11.16**	**33 Std.**	**28,5 Std.**	**-4,5**
Dokumentation	**18.11.16**	**05.12.16**	**45 Std.**	**45 Std.**	**0**
dauernde Projektleitung (Abstimmungen, Entscheidungen, …)	**06.10.16**	**05.12.16**	**28 Std.**	**28 Std.**	**0**
Gesamt			**360 Std.**	**370,5 Std.**	**+ 10,5**

Zusätzlich zu den dargestellten Phasen hatten wir einen Vorgang, den wir als „dauernde Projektleitung“ parallel verlaufen ließen. Hier wurden wichtige Abstimmungen und Entscheidungen, die sich im Verlauf des Projekts ergeben haben, diskutiert. Diesen Aufwand planten wir für wöchentliche und auch zwischenzeitliche Absprachen optimal.

Insgesamt lässt sich feststellen, dass wir insbesondere aufgrund der Abweichungen in der Phase Realisierung den geplanten Aufwand um 10,5 Stunden überschritten haben. Dies hätte vermieden werden können, wenn wir Risikofaktoren innerhalb der Teamkommunikation besser einkalkuliert hätten.

6.1.9 Ressourcenzuordnung

Ressourcenname	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Alina Fankhänel	**121 Std.**	**156 Std.**	**+35**
Kickoff-Meeting	1 Std.	1 Std.	0
Pflichtenheft	2 Std.	2 Std.	0
Projektplan	8 Std.	10 Std.	+2
Festlegen der Namenskonventionen	1 Std.	1 Std.	0
Erstellung ER-Diagramm	7 Std.	6 Std.	-1
Erstellung Klassendiagramm	13 Std.	13 Std.	0
Entwurf Testplan	2 Std.	0 Std.	-2

Zusammenfassung Diagramme / Anforderungen / Testplan	2 Std.	2 Std.	0
Installation und Konfiguration von Eclipse, Tomcat, MySQL…	5 Std.	4 Std.	-1
Erstellung der Tabellendefinitionen in der Datenbank	4 Std.	4 Std.	0
Datenbankanbindung	2 Std.	3 Std.	+1
Oberflächen-Logik	7 Std.	25 Std.	+18
Backend-Entwicklung	40 Std.	50 Std.	+10
Verbindung Frontend & Backend	5 Std.	10 Std.	+5
Dokumentation / Refactoring	2 Std.	3 Std.	+1
Testkonzeption	5 Std.	4 Std.	-1
Soll- / Ist-Vergleich	1 Std.	1 Std.	0
Präsentation	0 Std.	2 Std.	+2
Dokumentation	5 Std.	5 Std.	0
dauernde Projektleitung (Abstimmungen, Entscheidungen,…)	9 Std.	10 Std.	+1

Ressourcenname	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Nicola Kloke	**119 Std.**	**143**	**+24**
Kickoff-Meeting	1 Std.	1 Std.	0
Pflichtenheft	2 Std.	3 Std.	+1
Projektplan	4 Std.	3 Std.	-1
Festlegen der Namenskonventionen	1 Std.	1 Std.	0
Erstellung Use-Case-Diagramme	8 Std.	9 Std.	+1
Entwurf Testplan	2 Std.	0 Std.	-2
Zusammenfassung Diagramme / Anforderungen / Testplan	2 Std.	3 Std.	+1
Installation und Konfiguration von Eclipse, Tomcat, MySQL…	5 Std.	4 Std.	-1
Datenbankanbindung	2 Std.	0,5 Std.	-1,5
Umsetzung Oberfläche	35 Std.	45 Std.	+10
Oberflächen-Logik	9 Std.	0	-9
Verbindung Frontend & Backend	5 Std.	5 Std.	0
Responsive Design	0 Std.	29	+29
Dokumentation / Refactoring	2 Std.	3 Std.	+1
Testkonzeption	5 Std.	5 Std.	0
Testdaten pflegen	8 Std.	8,5 Std.	+0,5
Integrationstest	0 Std.	2 Std.	+2
Soll- / Ist-Vergleich	3 Std.	3 Std.	0
Präsentation	0 Std.	3 Std.	+3
Dokumentation	15 Std.	5 Std.	-10
dauernde Projektleitung (Abstimmungen, Entscheidungen,…)	10 Std.	10 Std.	0

Ressourcenname	Geplante Arbeit	Benötigte Arbeit	Abweichung Arbeit (Std.)
Timo Schlüter	**120 Std.**	**71,5**	**-48,5**
Kickoff-Meeting	1 Std.	1 Std.	0
Pflichtenheft	2 Std.	2 Std.	0
Projektplan	4 Std.	2 Std.	-2
Festlegen der Namenskonventionen	1 Std.	1 Std.	0
Erstellung von Mock-Ups für die Oberfläche	10 Std.	11 Std.	+1
Entwurf Testplan	2 Std.	5 Std.	+3
Zusammenfassung Diagramme / Anforderungen / Testplan	2 Std.	2 Std.	0
Installation und Konfiguration von Eclipse, Tomcat, MySQL…	5 Std.	4 Std.	-1
Datenbankanbindung	2 Std.	0,5 Std.	-1,5
Oberflächen-Logik	5 Std.	0 Std.	-5
Backend-Entwicklung	10 Std.	0 Std.	-10
Responsive Design	29 Std.	0 Std.	-29
Dokumentation / Refactoring	2 Std.	0 Std.	-2

Testkonzeption	5 Std.	3 Std.	-2
Integrationstest	10 Std.	6 Std.	-4
Soll- / Ist-Vergleich	1 Std.	1 Std.	0
Präsentation	15 Std.	5 Std.	-10
Dokumentation	5 Std.	20 Std.	+15
dauernde Projektleitung (Abstimmungen, Entscheidungen,…)	9 Std.	8 Std.	-1

7 Fazit/Bewertung

Zusammenfassend lässt sich sagen, dass das Projekt fristgerecht und erfolgreich abgeschlossen wurde. Es konnten neben den Muss-Kriterien weitere Kann-Kriterien erfolgreich umgesetzt werden.

Die gestalteten Diagramme in der Designphase bildeten eine gute Orientierung und erleichterten spätere Entscheidungsprozesse.

Zeitverluste mussten unter anderem durch Einarbeitung in Tools, fehlende Kompetenzen sowie zwischenzeitliche Differenzen innerhalb des Teams in Kauf genommen werden. Insgesamt entstand jedoch ein großer Lerneffekt und uns wurde bewusst, wie wichtig eine gute Strukturierung, Planung und Kommunikation innerhalb des Teams ist.

Für weitere Projekte haben wir gelernt, dass wir die Einarbeitungsdauer sowie die Möglichkeiten für Differenzen innerhalb des Teams besser in die Kalkulierung integrieren sollten. Das Projekt hat uns damit auch die Wichtigkeit der Betrachtung von Risikofaktoren nähergebracht.

8 Eigenständigkeitserklärung

Persönliche Erklärung der Studenten:
Wir versichern durch unsere Unterschriften, dass wir das Projekt und die dazugehörige Dokumentation selbstständig und ohne fremde Hilfe angefertigt haben. Alle Stellen, die wir wörtlich aus Veröffentlichungen entnommen haben, sind als solche kenntlich gemacht worden. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

....
Ort, Datum Unterschrift der Studenten

9 Quellenverzeichnis

1. Brandt-Pook & Kollmeier: Softwareentwicklung kompakt und verständlich (1. Auflage, 2008)
2. Ullenboom, Christian: Java ist auch eine Insel (10. Auflage)
3. http://www.oracle.com/technetwork/articles/javase/servlets-jsp- 140445.html (aufgerufen am 02.12.2016)
4. http://informatik.bildung-rp.de/fileadmin/user_upload/informatik.bildung-rp.de/Fortbildung/html/D1-MVCNotenstatistik/mvc0_0.html (aufgerufen am 02.12.2016)