

**Festiva Webshop**

Projektdokumentation, Projekte der Wirtschaftsinformatik

**Entwicklung eines Bestellsystems**

Fachhochschule der Wirtschaft Paderborn

4. Semester

Pfbw115a

WS 16/17

Marcus Becker

**Entwicklung durch:**

Alina Fankhänel

Nicola Kloke

Timo Schlüter

Inhaltsverzeichnis

[1 Einleitung 5](#_Toc468382408)

[1.1 Allgemein 5](#_Toc468382409)

[1.2 Systemumgebung 5](#_Toc468382410)

[1.2.1 Software 5](#_Toc468382411)

[1.2.2 Hardware 5](#_Toc468382412)

[1.2.3 Organisation und Hilfsmittel 5](#_Toc468382413)

[1.3 Zielgruppe 5](#_Toc468382414)

[2 Architektur der Anwendung 5](#_Toc468382415)

[2.1 Allgemeine Struktur 5](#_Toc468382416)

[2.2 Datenbank-Struktur 6](#_Toc468382417)

[2.3 Klassen-Struktur 7](#_Toc468382418)

[3 Oberfläche der Anwendung 11](#_Toc468382419)

[3.1 Planung 11](#_Toc468382420)

[3.1.1 Kundensicht 11](#_Toc468382421)

[3.1.2 Adminsicht 14](#_Toc468382422)

[3.1.3 Mobile Sicht 15](#_Toc468382423)

[3.2 Realisierung 16](#_Toc468382424)

[3.2.1 Kundensicht 16](#_Toc468382425)

[3.2.2 Adminsicht 18](#_Toc468382426)

[3.2.3 Mobile Sicht 18](#_Toc468382427)

[4 Funktionen der Anwendung 20](#_Toc468382428)

[4.1 Kundenfunktionen 20](#_Toc468382429)

[4.1.1 Planung 20](#_Toc468382430)

[4.1.2 Realisierung 21](#_Toc468382431)

[4.1.2.1 Kunden registrieren 22](#_Toc468382432)

[4.1.2.2 Kunden anmelden / abmelden 22](#_Toc468382433)

[4.1.2.3 Kundendaten ändern 23](#_Toc468382434)

[4.1.2.4 Festivals suchen 23](#_Toc468382435)

[4.1.2.5 Festivaldetails anzeigen 24](#_Toc468382436)

[4.1.2.6 Artikel zum Warenkorb hinzufügen 24](#_Toc468382437)

[4.1.2.7 Warenkorb anzeigen / bearbeiten 25](#_Toc468382438)

[4.1.2.8 Bestellung durchführen 25](#_Toc468382439)

[4.2 Administrationsfunktionen 26](#_Toc468382440)

[4.2.1 Planung 26](#_Toc468382441)

[4.2.2 Realisierung 28](#_Toc468382442)

[4.2.2.1 Kundenverwaltung 28](#_Toc468382443)

[4.2.2.2 Kategorienverwaltung 29](#_Toc468382444)

[4.2.2.3 Festivalverwaltung 29](#_Toc468382445)

[4.2.2.4 Artikelverwaltung für festivalunabhängige Artikel 30](#_Toc468382446)

[5 Testprotokoll 30](#_Toc468382447)

[6 Projektmanagement 36](#_Toc468382448)

[6.1 Soll-Ist-Vergleich 36](#_Toc468382449)

[6.1.1 Projektplan 36](#_Toc468382450)

[6.1.2 Muss/Kann-Kriterien 38](#_Toc468382451)

[6.2 Ressourcenzuordnung 39](#_Toc468382452)

[7 Fazit/Bewertung 39](#_Toc468382453)

[8 Sonstiges 40](#_Toc468382454)

[9 Eigenständigkeitserklärung 40](#_Toc468382455)

[10 Quellenverzeichnis 40](#_Toc468382456)

[11 Glossar 40](#_Toc468382457)

Abbildungsverzeichnis

[Abbildung 1 - Modell-View-Controll 6](#_Toc468382458)

[Abbildung 2 - Mock-Up: Startseite 11](#_Toc468382459)

[Abbildung 3 - Mock-Up: Shop 12](#_Toc468382460)

[Abbildung 4 - Mock-Up: Festivaldetailsicht 12](#_Toc468382461)

[Abbildung 5 - Mock-Up: Warenkorb 13](#_Toc468382462)

[Abbildung 6 - Mock-Up: Kasse 13](#_Toc468382463)

[Abbildung 7 - Mock-Up: Festivalverwaltung 14](#_Toc468382464)

[Abbildung 8 - Mock-Up: Festival ändern 15](#_Toc468382465)

[Abbildung 9 - Mock-Up: Mobile Sicht 15](#_Toc468382466)

[Abbildung 10 - Webshop: Startseite 16](#_Toc468382467)

[Abbildung 11 - Webshop: Ticket Shop 17](#_Toc468382468)

[Abbildung 12 - Webshop: Warenkorb 17](#_Toc468382469)

[Abbildung 13 - Webshop: Festivalverwaltung 18](#_Toc468382470)

[Abbildung 14 - Webshop: Festivaldetails und Warenkorb responsive 19](#_Toc468382471)

[Abbildung 15 - Use-Case Diagramm: Besucher 20](#_Toc468382472)

[Abbildung 16 - Use-Case Diagramm: Registrierter Kunde 21](#_Toc468382473)

[Abbildung 17 - Use-Case Diagramm: Kundenverwaltung 26](#_Toc468382474)

[Abbildung 18 - Use-Case Diagramm: Kategorienverwaltung 27](#_Toc468382475)

[Abbildung 19 - Use-Case Diagramm: Festivalverwaltung 27](#_Toc468382476)

[Abbildung 20 - Use-Case Diagramm: Artikelverwaltung 28](#_Toc468382477)

[Abbildung 21 – Carl-Steinweg Phasenmodell 36](#_Toc468382478)

**Änderungshistorie**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Verfasser | Grund/Inhalt der Änderung |
| 1.0 | 07.11.2016 | Timo Schlüter | Initialerstellung (Struktur) |

# Einleitung

## 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 zugleich die Rolle des Ansprechpartners während des Projekts.

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

## 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.

## 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.

## Systemumgebung

### Software

Der Onlineshop wurde mit Hilfe der Entwicklungsumgebung Eclipse Neon entwickelt und in Java, Java Script, sowie HTML 5 und CSS programmiert. Als Datenbanksystem wird MySQL und als Webserver Tomcat 7.0 aus der Software XAMPP verwendet. Somit kann eine lokale Datenbank sowie ein lokaler Webserver (Tomcat 7.0) simuliert werden.

Der Benutzer benötigt Google Chrome oder Mozilla Firefox um den Onlineshop bedienen zu können.

### 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 sowie Mozilla Firefox ausführen kann. Zudem kann ein Smartphone verwendet werden, dass ebenfalls einen der beiden genannten Browser besitzt.

### Organisation und Hilfsmittel

Der Ansprechpartner seitens des Auftraggebers ist der Dozent Marcus Becker. Durch ihn wird auch die Projektabnahme erfolgen. Projektverantwortlicher innerhalb des Projektes ist ebenfalls Marcus Becker. Er steht uns in sämtlichen Fragen, sei es organisatorisch oder fachlich, als Ansprechpartner zur Verfügung.

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.

## Zielgruppe

Die Zielgruppe für den Onlineshop sind 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.

# Architektur der Anwendung

## Allgemeine Struktur

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

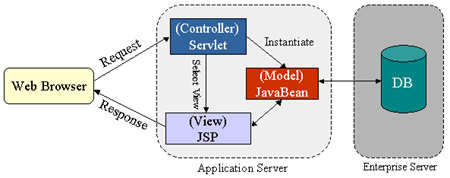


Abbildung 1 - Modell-View-Controller

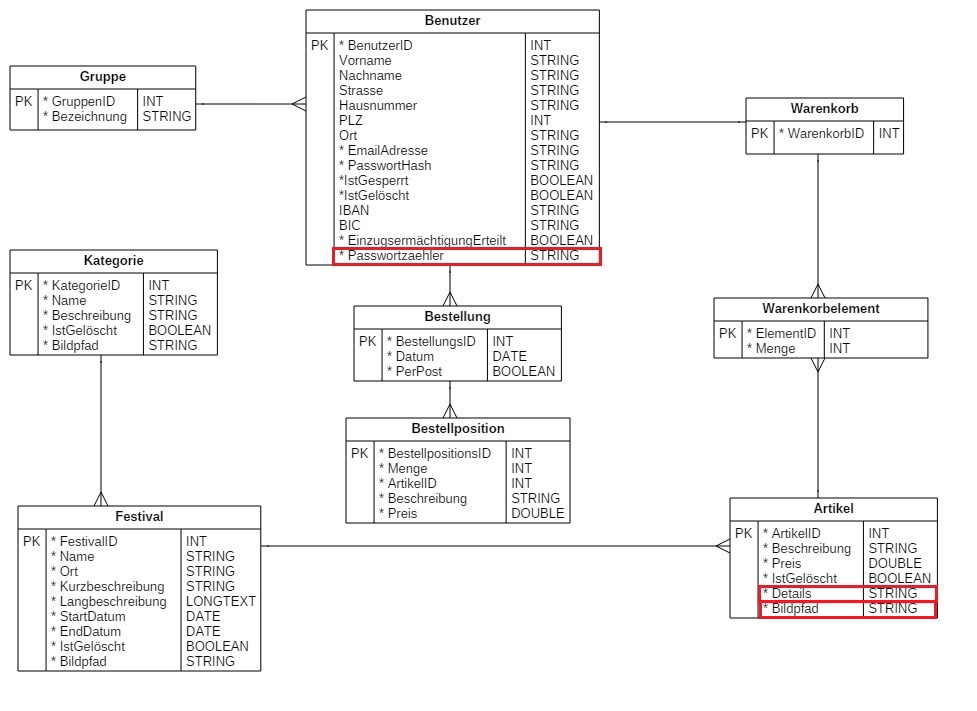
Da jede interaktive Anwendung aus Eingaben, Daten und der Darstellung der Daten besteht, wäre es problematisch, diese drei Elemente in einer Klasse (Formular) zusammenzufassen. Das Ziel des MVC-Konzepts liegt daher in der Trennung zwischen Datenverarbeitung, Repräsentation und Eingaben um das System 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]](#footnote-2)

## Datenbank-Struktur



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

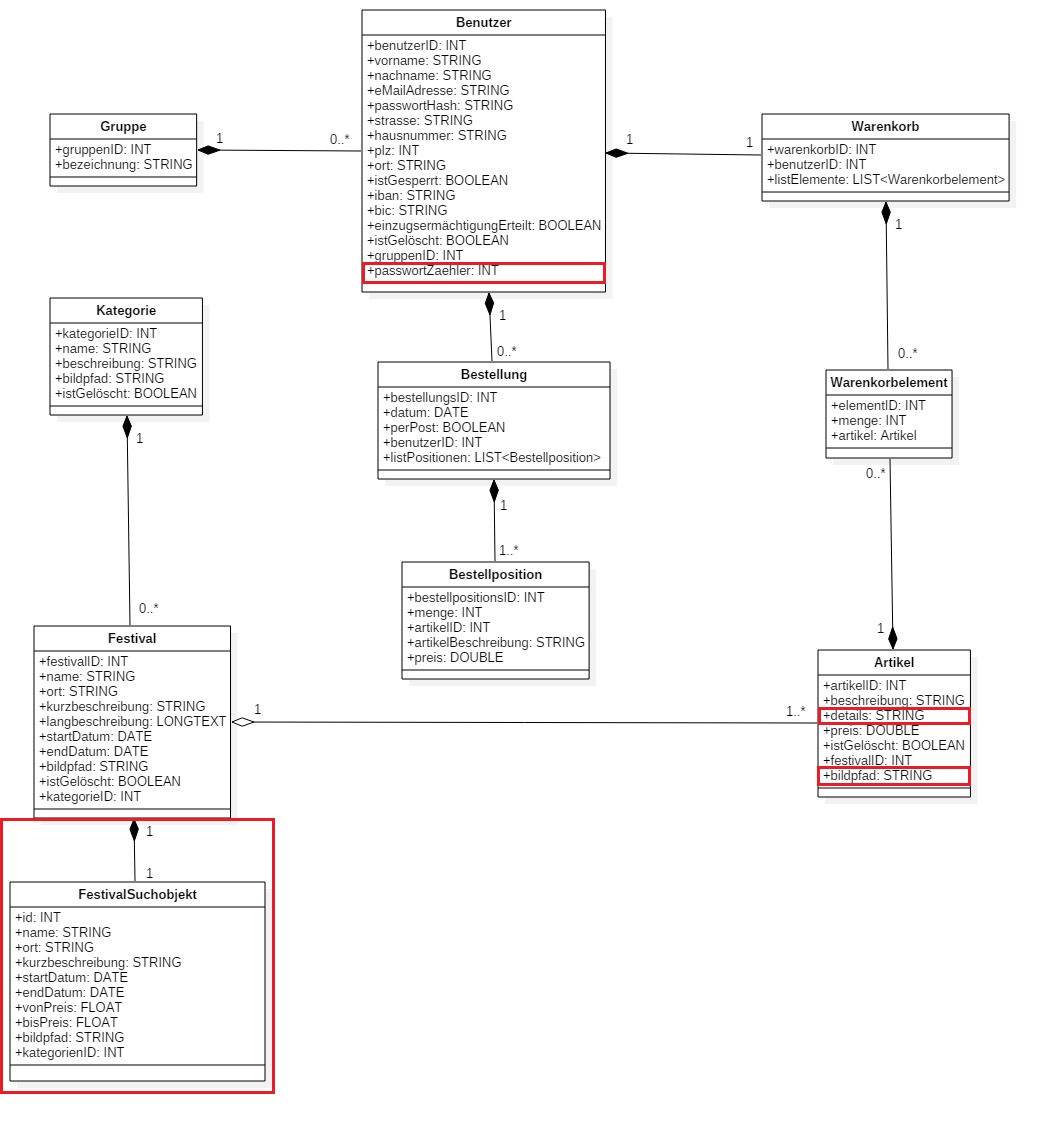
Das zusätzliche Attribut „Passwortzaehler“ wurde benötigt, um die automatische Sperrung eines Kunden zu realisieren. Diese tritt nach dreimaliger Falscheingabe des Passworts ein. Die Anzahl der falschen Passworteingaben wird in diesem Feld gesichert.

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

## Klassen-Struktur

Die Klassen innerhalb des Projekts wurden in drei verschiedene Pakete eingeteilt. Auch die Klassendiagramme wurden 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.

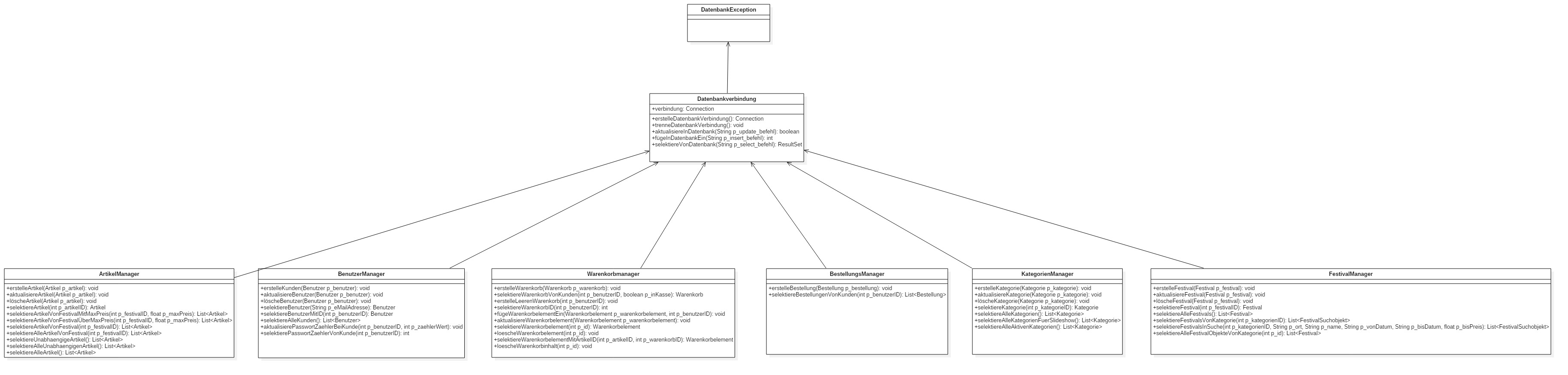
**1. standardPackage**



Das erste Paket beinhaltet alle Fachklassen, die im Rahmen des Webshops benötigt wurden. Im Verlauf des Projekts haben sich gegenüber dem Entwurf im Fachkonzept einige Änderungen ergeben. Die neue Klasse „FestivalSuchobjekt“ wurde erstellt, da im Bereich der Suche (Ticket Shop) zu einem Festival zusätzliche Informationen, wie beispielsweise ein Maximalpreis gespeichert werden muss. 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 sollten, wurde in der Klasse „Artikel“ das Attribut „bildpfad“ hinzugefügt. 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**

**

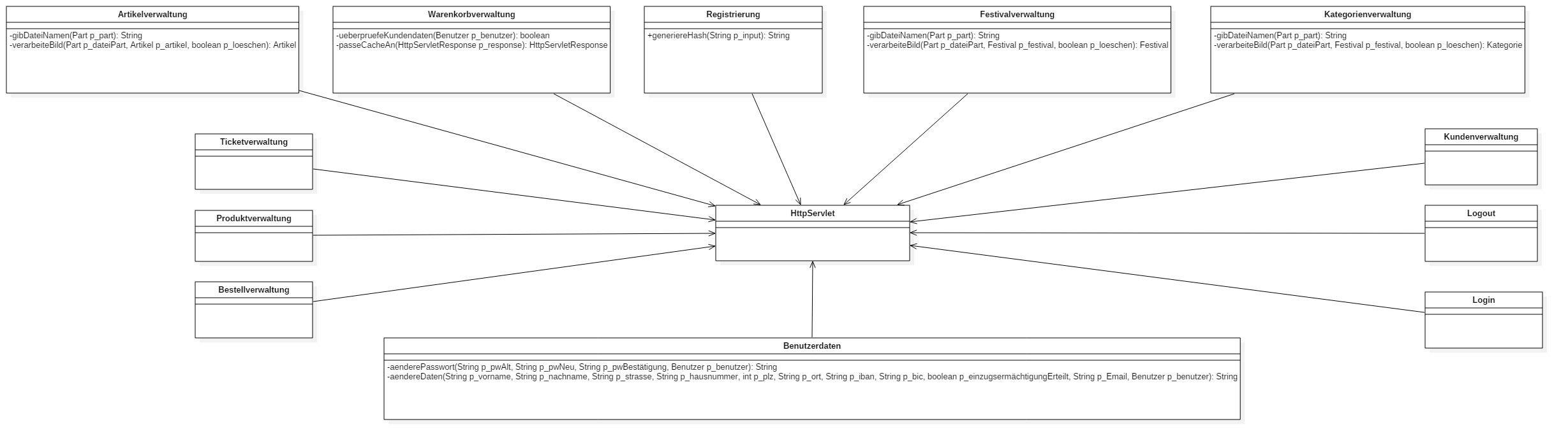
Die Manager-Klassen beinhalten alle Funktionen, um direkt auf die Datenbank zugreifen zu können. Die Klasse „Datenbankverbindung“ ist dabei dafür zuständig, die Kommunikation mit der Datenbank zu steuern. Das bedeutet, dass in dieser Klasse sowohl die Erstellung der Verbindung als auch jeweils eine vorgefertigte 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, wurde die „DatenbankException“ erstellt. 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) verarbeitet, indem eine Instanz der entsprechenden Fachklasse erstellt wird, in der die Informationen dann abgelegt werden. Wenn mehrere Ergebnisse enthalten sind, wird einsprechend eine ArrayList vom Typ der Fachklasse erstellt.

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**

**

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 innerhalb des „BenutzerManagers“ und die Speicherung des Rückgabewerts in einer Instanz der Klasse „Benutzer“.

Um nicht für jede Anfrage ein neues Servlet verwenden zu müssen, wurde 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. Dies liegt daran, dass die 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.

# Oberfläche der Anwendung

## Planung

Um einen Anhaltspunkt zu haben, wie wir uns unsere Webseitenoberfläche vorstellen, entwickelten wir Mock-Ups. Im Folgenden sollen ausgewählte Mock-Ups, dem tatsächlichen Design der Oberfläche gegenübergestellt werden.

### Kundensicht

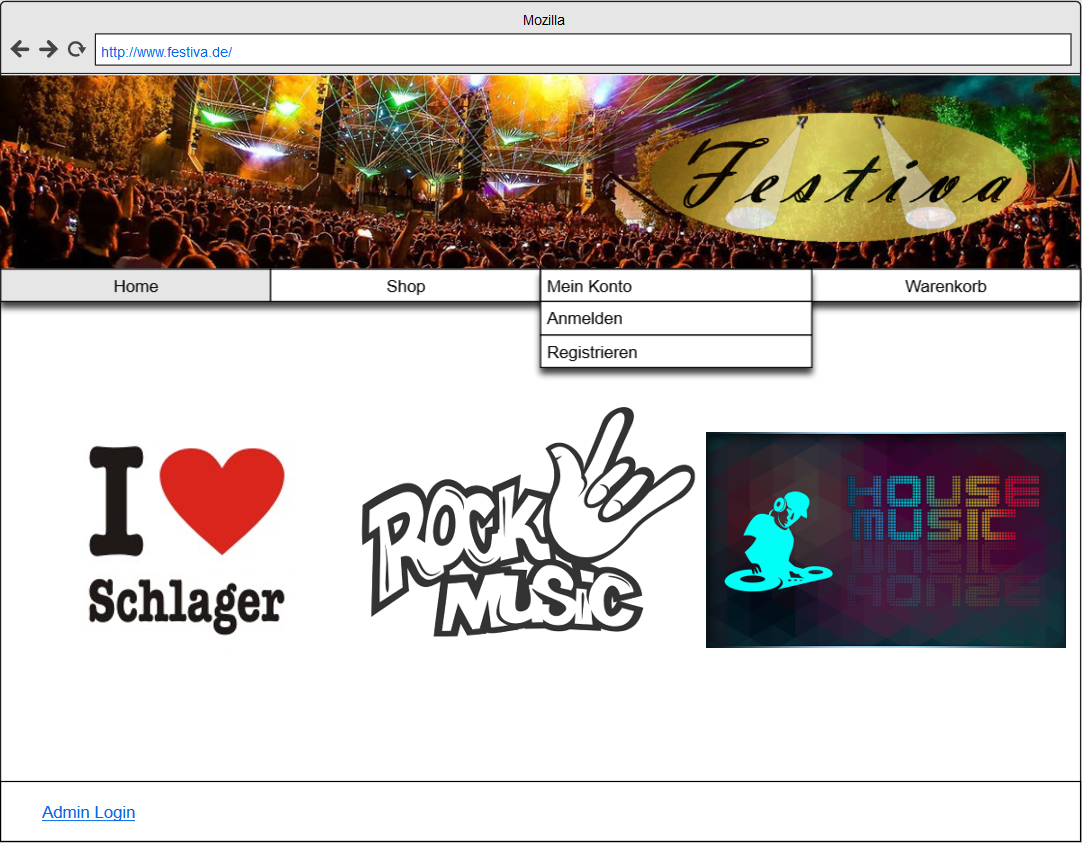


Abbildung 2 - Mock-Up: Startseite

Für den allgemeinen Aufbau des Webshops entwickelten wir 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 kann auf die Startseite, den Shop, auf sein Konto oder auf den Warenkorb navigieren. Auf der Startseite sollen Bilder von Festivalkategorien angezeigt werden. Der Admin soll über einen Link im Footer auf seine Anmeldemaske geleitet werden.



Abbildung 3 - Mock-Up: Shop

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

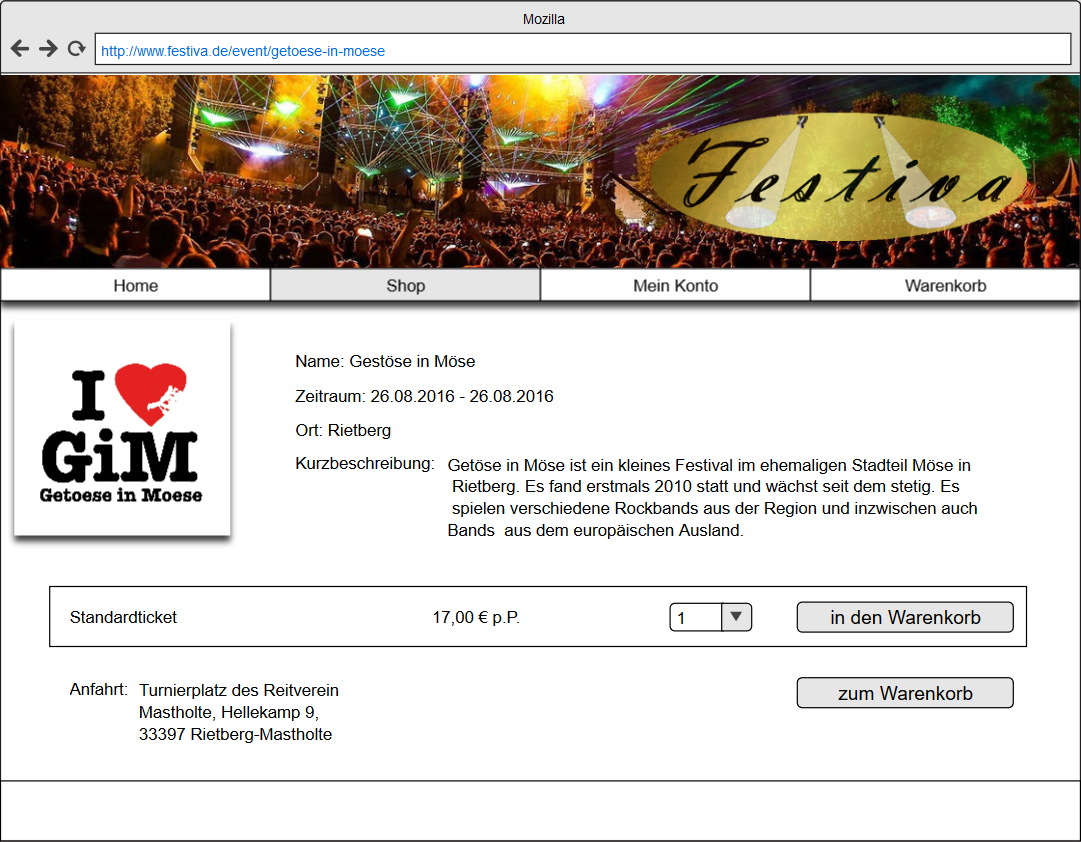


Abbildung 4 - Mock-Up: Festivaldetailsicht

Zur Festivaldetailsicht gelangt man, wenn im Shop ein Festival anklickt wird. Zu dem Festival werden weitere Informationen angezeigt, sowie die bestellbaren Tickets.

Der Benutzer kann seine gewünschte Anzahl wählen und Artikel in den Warenkorb legen, sowie „zum Warenkorb“ klicken.

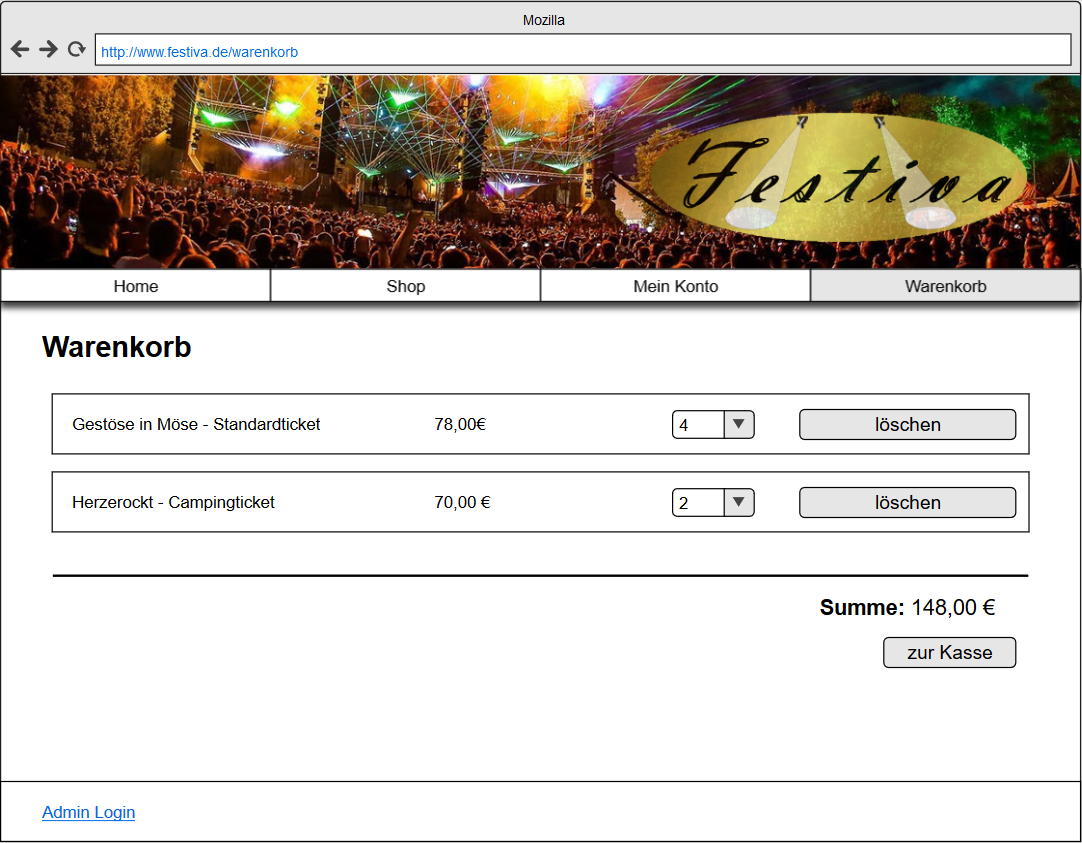


Abbildung 5 - Mock-Up: Warenkorb

Der Warenkorb beinhaltet alle Artikel, die zuvor in den Warenkorb gelegt wurden. 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 den Kauf dort abzuschließen.



Abbildung 6 - 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 Artikel werden nochmals aufgeführt und am Ende eine Gesamtsumme inkl. Versandkosten angezeigt. Über den „Bezahlen“ - Button können die Artikel verbindlich bestellt werden.

### Adminsicht



Abbildung 7 - 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.

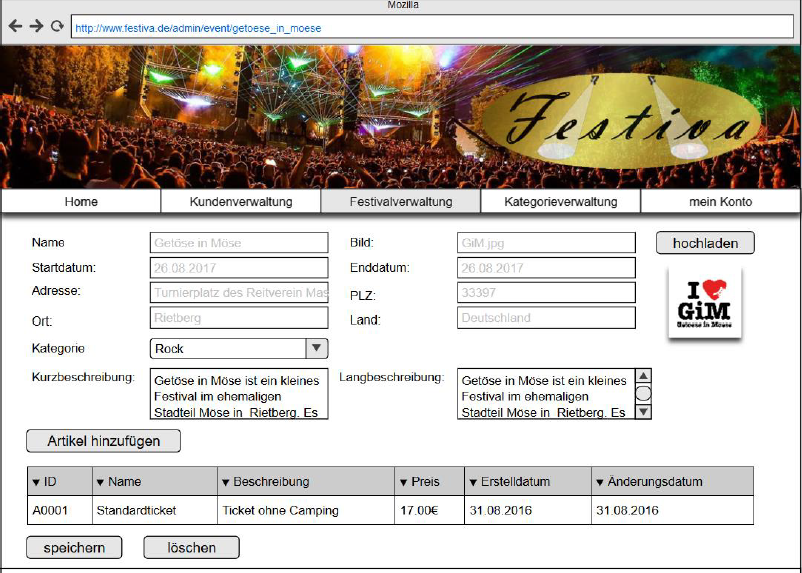


Abbildung 8 - Mock-Up: Festival ändern

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

In der Festivalverwaltung ist es zusätzlich möglich, neue Artikel dem Festival hinzuzufügen oder dem Festival ein Bild zu geben.

### Mobile Sicht



Abbildung 9 - Mock-Up: Mobile Sicht

In der mobilen Sicht, soll die Navigationsleiste nun vertikal angezeigt werden, um gut lesbar zu sein und trotzdem Platz zu sparen.

Mit Breakpoints, werden neue Abschnitte geschaffen, um den Text weiterhin leserlich zu gestalten und vertikales Scrollen vollständig zu vermeiden. Gegebenenfalls überflüssige Informationen werden ausgeblendet und klickbare Elemente vergrößert, um dem Benutzer eine einfache Bedienung zu gewährleisten.

## Realisierung

### Kundensicht

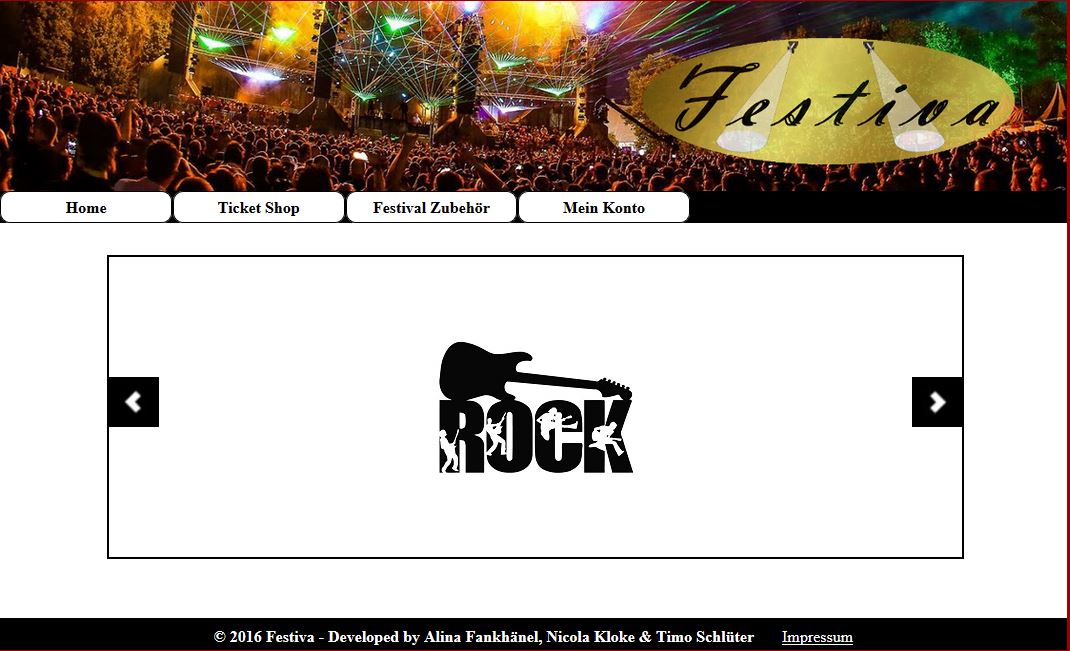


Abbildung 10 - Webshop: Startseite

Die Navigation aus den Mock-Ups, wurde durch den Zubehörshop ergänzt und ändert sich je nachdem, ob der Kunde angemeldet ist oder nicht oder ein Admin angemeldet ist. Die Bilder der Kategorien werden als Slideshow angezeigt. Wenn sich ein Festiva Mitarbeiter als Admin anmelden möchte, kann er dies über die normale Navigation machen und nicht über einen extra Link im Footer, wie in den Mock-Ups geplant.

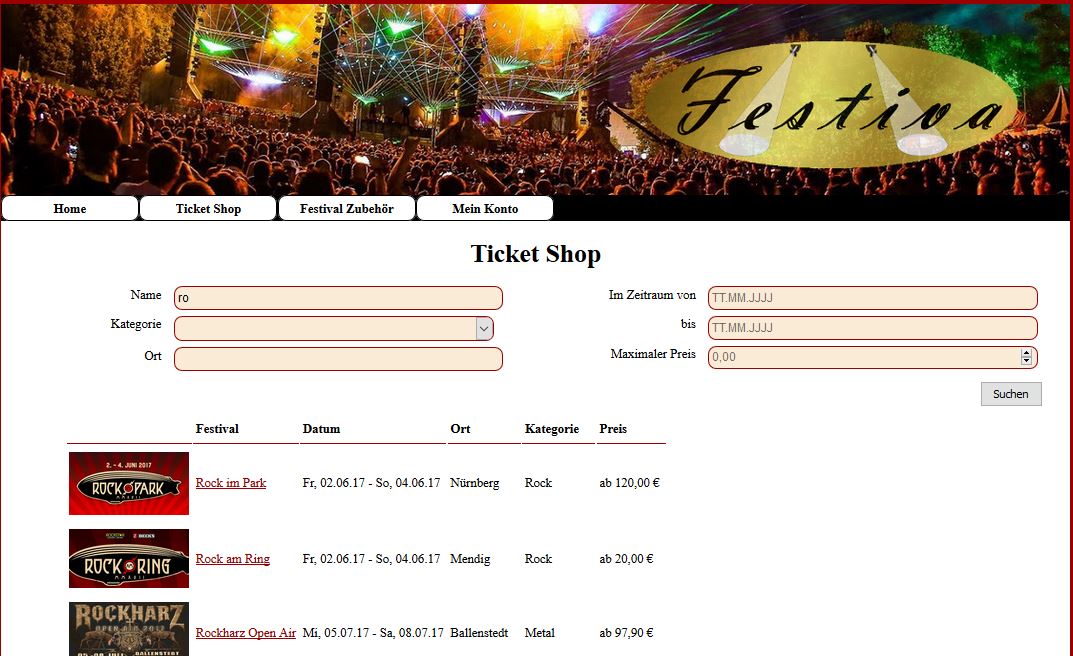


Abbildung 11 - Webshop: Ticket Shop

Die Suchfelder sind nun oberhalb der Festivals positioniert, jedoch sind die Suchoptionen wie geplant umgesetzt worden außer, dass der Benutzer nur einen Maximal- und keinen Minimalpreis eingeben kann. Um den Ticketshop ansprechender zu gestalten, fügten wir die Festivalbilder hinzu.

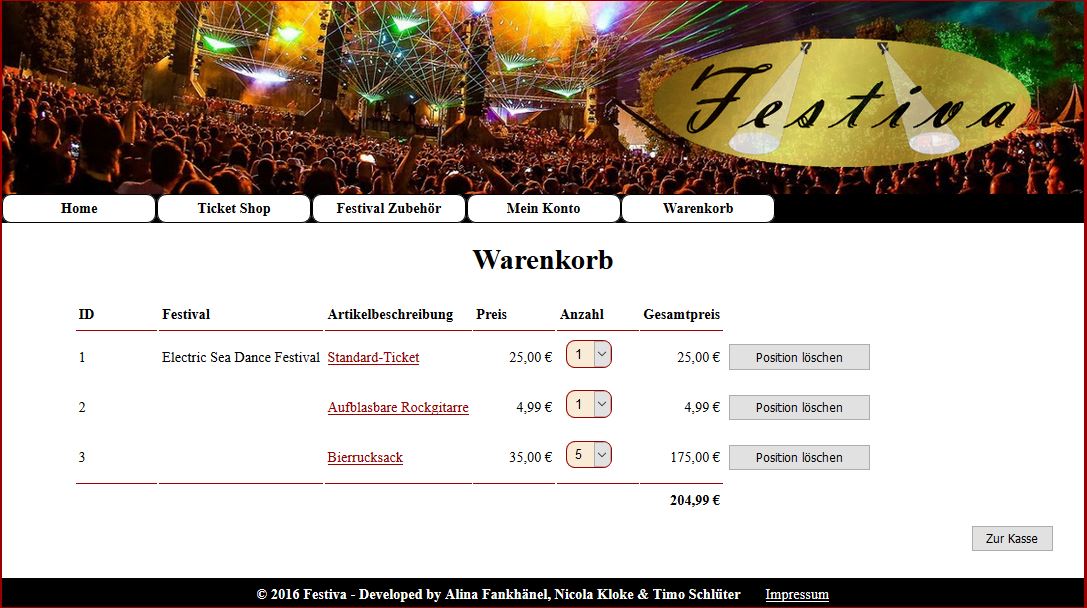


Abbildung 12 - Webshop: Warenkorb

Die Oberfläche des Warenkorbs gleicht dem zugehörigen Mock-Up nahezu. Den einzelnen Artikeln wurde jedoch noch eine Positions-ID gegeben.

### Adminsicht



Abbildung 13 - Webshop: Festivalverwaltung

Die Suche, sowie ein Button zum Anlegen wurden ähnlich der Mock-Ups realisiert. Die entsprechenden Elemente werden ebenfalls tabellarisch aufgelistet, jedoch mit leicht abweichenden Spalteninformationen. Über die ID gelangt man in den Änderungsmodus.

### Mobile Sicht

Abbildung 14 - Webshop: Festivaldetails 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 werden unterschiedliche Breakpoints gesetzt, um horizontal angeordnete Elemente, untereinander anzuordnen. Ebenso wird die Struktur von Tabellen vollständig aufgelöst. Die Funktionalitäten bleiben gleich.

# Funktionen der Anwendung

Für die Realisierung orientierten wir uns an den zuvor erstellten Use-Case Diagrammen. Während des Projektes gab es kleinere Änderungen, sowie Erweiterungen, welche in den Use-Case Diagrammen rot gekennzeichnet sind.

## Kundenfunktionen

### Planung

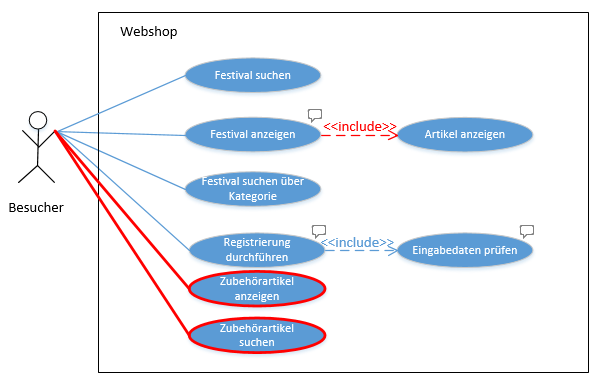


Abbildung 15 - Use-Case Diagramm: Besucher

Ein Besucher hat im Webshop nur eine begrenzte Sicht. Er kann sich alle Festival mit ihren Artikeln ansehen und auch nach ihnen 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 kann.

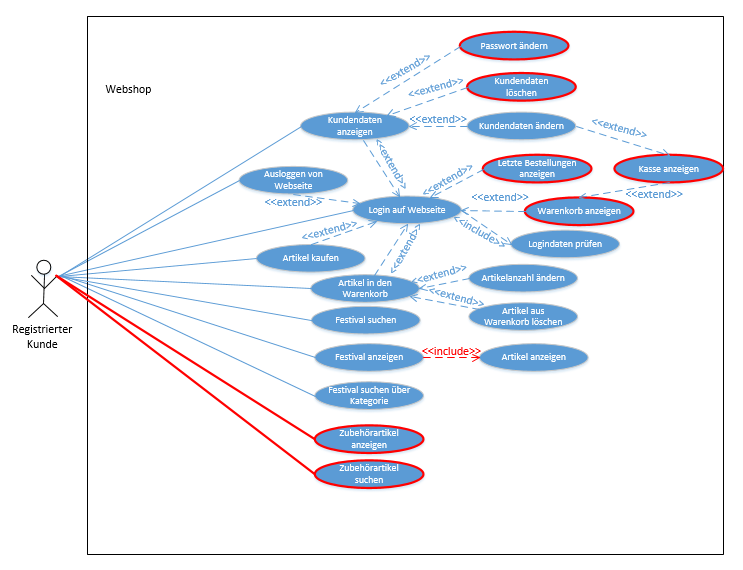


Abbildung 16 - Use-Case Diagramm: Registrierter Kunde

Zusätzlich zu den Benutzerfunktionen 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 zu jeder Zeit die Möglichkeit seine persönlichen Daten einzusehen und zu ändern.

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

### 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.

#### 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 richtig 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 auf die Startseite 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.

#### 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, wird 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.

#### 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 der Kunde 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

#### 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 finden 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.

#### 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.

#### 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.

#### 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.

#### 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. Fall bekommt der Kunde entsprechendes Feedback und die Menge wird auf zehn gesetzt.

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 der Menge des Artikels.

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

#### 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.

#### Bestellung durchführen

**Verwendete JSPs:** k\_warenkorb, k\_kasse

**Verwendete Servlets:** Warenkorbverwaltung, Bestellverwaltung

**Verwendete Managerklassen:** BestellungsManager, WarenkorbManager

Von dem Warenkorb 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.

## Administrationsfunktionen

### Planung

Ein Admin meldet sich über den Login an und erhält andere Sichten als ein Kunde.

Er hat die drei Menüpunkte Kunden-, Kategorie- und Festivalverwaltung.



Abbildung 17 - 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.

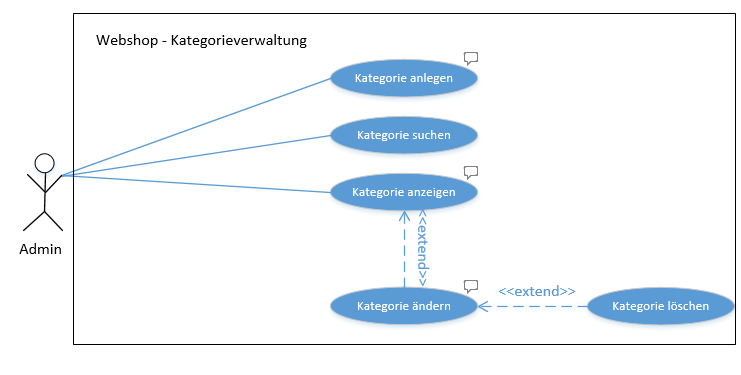


Abbildung 18 - Use-Case Diagramm: Kategorienverwaltung

Ein Admin kann eine Kategorie anlegen, suchen, anzeigen und desweitern über den Änderungsmodus löschen.

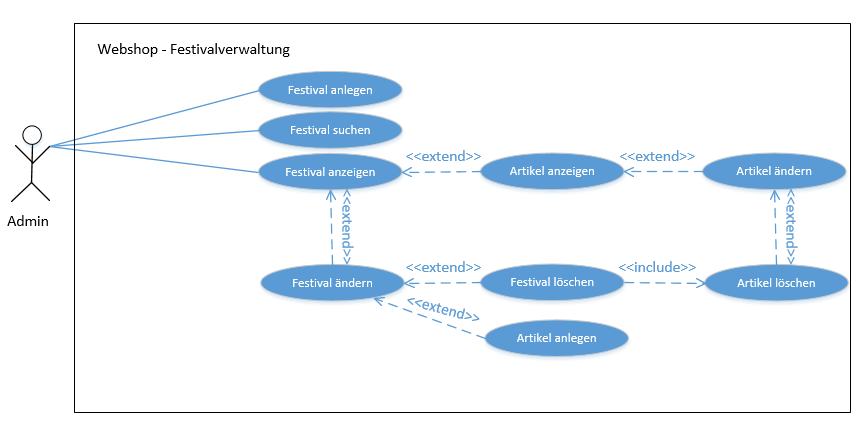


Abbildung 19 - Use-Case Diagramm: Festivalverwaltung

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.

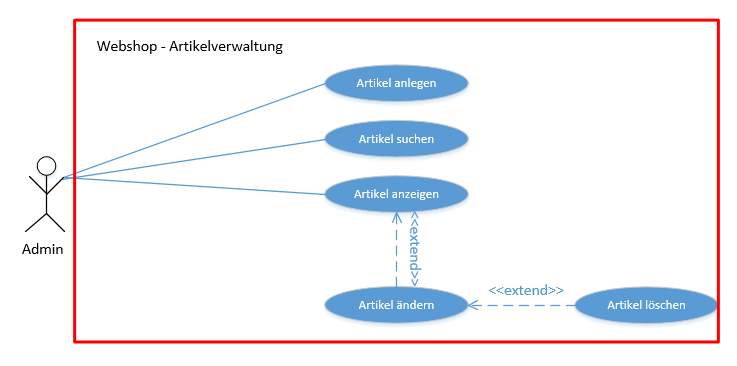


Abbildung 20 - Use-Case Diagramm: Artikelverwaltung

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 zusätzlich zu implementieren. Die Funktionen sind identisch zu denen der Kategorienverwaltung. Es kann nach Artikeln gesucht werden, neue können angelegt, sowie angezeigt, geändert und gelöscht werden.

### 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.

#### Kundenverwaltung

**Verwendete JSPs:** a\_kundenverwaltung, a\_kundeAendern, a\_kundeAnlegen

**Verwendete Servlets:** Kundenverwaltung, Registrierung

**Verwendete Managerklasse**n: 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 Ansicht über die JSP „a\_kundenAendern“.

#### Kategorienverwaltung

**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 diese 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 nur um Kategorien erweitert oder verkleinert werden, beinhaltet aber immer mindestens die vier Standardkategorien.

#### Festivalverwaltung

**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 innerhalb des Servlets „Artikelverwaltung“.

#### Artikelverwaltung für festivalunabhängige Artikel

**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.

#### Administration

**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

# Testprotokoll

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Funktion** | **M/K** | **Beschreibung** | **Erwartetes Ergebnis** | **Ergebnis** |
| T01 | Erfassen von Kundendaten (Kunde selbst) | M | Der Kunde registriert sich mit der E-Mail-Adresse [max.mustermann@test.de](mailto:max.mustermann@test.de) und dem Passwort „123456“ | Der Kunde ist registriert und kann sich mit den Anmeldedaten anmelden | Ging auf Grund der Kennwortrichtlinie nicht (siehe T02) |
| T02 | Kennwortrichtlinien prüfen | K | Der Kunde benutzt zunächst das Passwort aus T01 und dann „Test123!“ | Der Kunde bekommt zunächst eine Fehlermeldung auf Grund eines unzulässigen Passworts und kann sich beim 2. Passwort registrieren |  |
| T03 | Passwort zurücksetzen | K | Der Kunde klickt auf Passwort zurücksetzen | Der Kunde gelangt in eine Maske in der er sein Passwort zurücksetzen kann |  |
| T04 | Kunde anmelden | M | Der Kunde meldet sich mit den in T01 bzw. T02 genannten Anmeldedaten an | Der Kunde gelangt auf die Startseite und ist angemeldet |  |
| T05 | Kunde abmelden | M | Der Kunde klickt auf abmelden | Der Kunde befindet sich auf der Startseite und ist abgemeldet |  |
| T06 | Admin anmelden | M | Der Admin meldet sich mit den Admindaten an | Der Admin gelangt ins Adminmenü und ist angemeldet |  |
| T07 | Admin abmelden | M | Der Admin klickt auf abmelden | Der Admin befindet sich auf der normalen Startseite und ist abgemeldet |  |
| T08 | Anzeigen und bearbeiten von Kundendaten (Kunde selbst) | M | Der Kunde klickt auf „mein Konto“ und gibt als Daten ein:  Vorname: Max  Nachname: Mustermann  IBAN: DE 1234567891011121314  BIC: TESTDJF14F  Einzugsermächtigung: check  Adresse: Musterstraße 1  PLZ: 12345  Ort: Musterstadt | Nachdem der Kunde auf „Speichern“ geklickt hat befinden sich die Daten in der Datenbank und werden unter „Mein Konto“ angezeigt. |  |
| T09 | Löschen von Kundendaten (Kunde selbst) | M | Der Kunde klickt in „Mein Konto“ auf „löschen“ und bestätigt die Löschung mit „OK“ | Der Kunde wird in der Datenbank logisch gelöscht und kann sich nicht mehr anmelden |  |
| T10 | Erfassen von Kundendaten (Admin) | M | Der Admin geht in die Kundenverwaltung, klickt auf „Kunde anlegen“ und legt einen Kunden mit den in T01 bzw. T02 genannten Daten einen Kunden an mit der eMailadresse max.mustermann1@test.de | Der Kunde kann sich mit den passenden Anmeldedaten anmelden |  |
| T11 | Anlegen eines Adminkonto | K | Der Admin legt einen Admin mit den Daten [admin@festiva.de](mailto:admin@festiva.de) und dem Passwort „123Test!“ an | Der Admin kann sich mit den Anmeldedaten in den Adminbereich anmelden |  |
| T12 | Anzeigen und bearbeiten von Kundendaten (Admin) | M | Der Admin ändert die Kundendaten von [max.mustermann1@test.de](mailto:max.mustermann1@test.de) gemäß T08 um | Die Kundendaten sind entsprechend geändert |  |
| T13 | Löschen von Kundendaten (Admin) | M | Der Admin löscht den Account des Kunden max.mustermann1@test.de | Der Kunde ist logisch gelöscht und kann sich nicht mehr anmelden |  |
| T14 | Kundenkonto Sperrung manuell durch den Admin | M | Der Admin setzt bei dem Kunden [max.mustermann@test.de](mailto:max.mustermann@test.de) das „Sperren-Häkchen“ | Der Kunde ist gesperrt und kann sich nicht mehr anmelden |  |
| T15 | Kundenkonto Sperrung durch 3-malig falschem Passwort | K | Der Kunde gibt 3-mal das falsche Passwort ein | Der Kunde wird gesperrt und kann sich nicht mehr anmelden |  |
| T16 | Kundenentsperrung durch den Admin | M | Der Admin entsperrt den gesperrten Kunden in dem er das „Sperren-Häkchen“ entfernt | Der Kunde ist wieder entsperrt und kann sich wieder anmelden |  |
| T17 | Anlegen einer Kategorie (Admin) | M | Der Admin klickt in der Kategorienverwaltung auf Kategorie anlegen und gibt folgende Daten ein:  Name: Jazz  Beschreibung: Jazz Sessions  Bild: Jazz.jpg | Die Kategorie ist angelegt und wird bei der Kategorienverwaltung angezeigt |  |
| T18 | Anzeigen und bearbeiten einer Kategorie (Admin) | M | Die Kategorie „Jazz“ wird ausgewählt und bearbeitet. Die Beschreibung wird in „Jazz Sessions und Festivals“ geändert und gespeichert | Die Beschreibung ist geändert und in der Kategorienverwaltung zu finden |  |
| T19 | Anlegen eines Festivals (Admin) | M | Der Admin geht in die Festivalverwaltung, klickt auf „Festival anlegen“, gibt folgende Daten ein und speichert diese:  Name: Jazzed Off Session  Bild: Jazz\_off.jpg  Startdatum: 27.01.2017  Enddatum: 27.01.2017  Ort: Gütersloh  Kategorie: Jazz  Kurzbeschreibung: Jazz Off Session in der Weberei  Langbeschreibung: Jazz Off Session in der Weberei. Einlass: 17:00 Uhr Beginn 17:30 Uhr  Bogenstraße 1-8  33330 Gütersloh | Das Festival ist angelegt und in der Festivalverwaltung, sowie im Shop zu sehen |  |
| T20 | Anzeigen und bearbeiten eines Festivals (Admin) | M | Der Admin wählt das Festival „Jazz off Session“ aus und bearbeitet dieses indem er die Langbeschreibung um die „Endzeit 22:00 Uhr“ ergänzt | Die Änderung ist auf der Eventseite sichtbar |  |
| T21 | Anlegen eines Artikels (Admin) | M | Der Admin wählt das Festival „Jazz off Session“ aus und geht in den Änderungsmodus. Hier klickt der Admin auf Artikel hinzufügen und fügt folgende Daten ein und speichert diese:  Name: Standardticket  Beschreibung: freie Platzwahl  Preis: 17,95 | Der Artikel ist angelegt und sowohl in der Festivalsicht des Admins als auch in der Eventsicht des Kunden zu sehen. |  |
| T22 | Anlegen eines Merchandise Artikels (Admin) | K | Der Admin geht in die Artikelverwaltung, klickt auf „neuen Artikel anlegen“ und fügt folgende Daten ein und speichert diese:  Name: Regencape  Bild: Regencape.jpg  Preis: 25,99 | Der Artikel ist angelegt und sowohl in der Festivalsicht des Admins als auch in der Eventsicht des Kunden zu sehen. |  |
| T23 | Anzeigen und bearbeiten eines Artikels(Admin) | M | Der Admin geht zunächst auf das Festival „Jazz off Session“ und dann auf den Artikel Standardticket um ihn zu bearbeiten. Er ändert den Preis auf 15,95 und speichert diese | Die Änderung ist gespeichert und sowohl in der Festivalsicht des Admins als auch in der Eventsicht des Kunden zu sehen |  |
| T24 | Löschen eines Artikels (Admin) | M | Der Admin geht zunächst auf das Festival „Jazz off Session“ und dann auf den Artikel Standardticket um ihn zu bearbeiten. Er klickt auf „löschen“ und bestätigt dieses durch „OK“ um den Artikel zu löschen | Der Artikel ist logisch gelöscht und sowohl in der Festivalsicht des Admins als auch in der Eventsicht des Kunden nicht mehr zu sehen |  |
| T25 | Löschen eines Festivals (Admin) | M | Der Admin wählt das Festival „Jazz off Session“ aus um es zu bearbeiten. Er klickt anschließend auf „löschen“ und bestätigt dieses anschließend durch „OK“ um das Festival zu löschen | Das Festival ist logisch gelöscht und sowohl in der Festivalsicht des Admins als auch in der Eventsicht des Kunden nicht mehr zu sehen |  |
| T26 | Löschen einer Kategorie (Admin) | M | Die Kategorie „Jazz“ wird zum Bearbeiten geöffnet und durch einen Klick auf „löschen“ sowie die Bestätigung durch „OK“ gelöscht | Die Kategorie „Jazz“ wird logisch gelöscht und ist nicht mehr in der Kategorieverwaltung zu finden |  |
| T27 | Festivalsuche nach Name | M | Es wird im Shop nach dem Begriff „Session“ gesucht | Mindestens das Festival „Jazz off Session“ ist gefunden worden und Festivals auf dem das Suchkriterium passt |  |
| T28 | Festivalsuche nach Ort | M | Es wird im Shop nach dem Ort „Gütersloh“ gesucht | Mindestens das Festival „Jazz off Session“ ist gefunden worden und Festivals auf dem das Suchkriterium passt |  |
| T29 | Festivalsuche nach Kategorie | M | Es wird im Shop nach der Kategorie „Jazz“ gesucht | Mindestens das Festival „Jazz off Session“ ist gefunden worden und Festivals auf dem das Suchkriterium passt |  |
| T30 | Festivalsuche nach Datum | M | Es wird nach dem Zeitraum 22.01.2016 – 31.01.2016 gesucht | Mindestens das Festival „Jazz off Session“ ist gefunden worden und Festivals auf dem das Suchkriterium passt |  |
| T31 | Direktnavigation von der Startseite nach Kategorie | M | Es wird auf der Startseite auf die Kategorie „Jazz“ geklickt | Es wird in den Shop gewechselt und die Kategoriesuche ist auf „Jazz“ voreingestellt. Mindestens das Festival „Jazz off Session“ ist gefunden worden und Festivals auf dem das Suchkriterium passt |  |
| T32 | Artikel zum Warenkorb hinzufügen | M | Es wird das Festival „Jazz off Session“ ausgewählt und der Artikel „Standartticket“ mit 2 Tickets in den Warenkorb gelegt | 2 Standarttickets für das Festival „Jazz off Session“ liegen in dem Warenkorb |  |
| T33 | Artikel im Warenkorb anzeigen und bearbeiten | M | Es wird in den Warenkorb gegangen und die Anzahl der Standarttickets für das Festival „Jazz of Session“ auf 4 erhöht | Die Ticketanzahl liegt jetzt bei 4 |  |
| T34 | Artikel aus dem Warenkorb löschen | M | Der Artikel „Standartticket“ wird aus dem Warenkorb durch einen Klick auf „löschen“ gelöscht | Der Artikel befindet sich nicht mehr im Warenkorb |  |
| T35 | Bestellung ausführen | M | Nachdem der Artikel „Standartticket“ wieder hinzugefügt wurde, auf „zur Kasse“ geklickt, Post als Lieferoption ausgewählt und dann auf „bezahlen geklickt. | Alle Informationen werden richtig angezeigt und berechnet. Die Bestellung ist ausgeführt, der Warenkorb gelehrt und die Bestellung in der Datenbank abgelegt. |  |
| T36 | Bestellungen einsehen (Kunde) | M | Der Kunde klickt auf „Meine Bestellungen“ | Die eben abgegebene Bestellung wird angezeigt |  |
| T37 | Seite auf Mobile Device starten | M | Die Seite wird über einen Emulator gestartet und angezeigt | Die Seite entspricht den Anforderungen eines Responsive Designs |  |

Der Testplan beschreibt den Abnahmetest in Form eines Black-Box Test. Hier wurden alle Funktionen ohne Kenntnis bzw. Berücksichtigung des Programmcodes. Zusätzlich zum Testplan wurde der Webshop von außenstehenden Personen ohne einen Plan wild durchgeklickt und probiert um auch ungewöhnliche Konstellationen zu erreichen. Es wurden beim Abschlusstest bis auf Konfigurationsfehler keine größeren Fehler gefunden.

Zusätzlich zu dem oben zusehenden Testplan wurden während der Realisierungsphase White-Boxtests – Testen des Programms mit Kenntnis des Programmcodes (Debuggen) - der Komponenten von den Entwicklern durchgeführt

# Projektmanagement

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

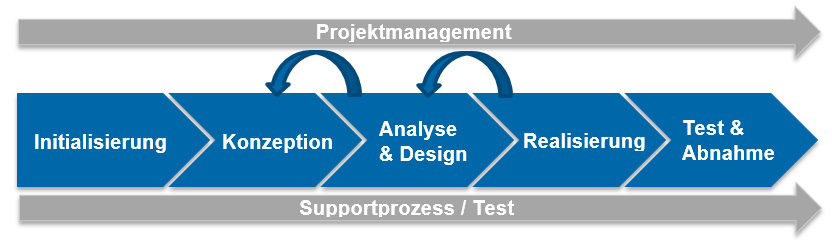


Abbildung 21 – Carl-Steinweg Phasenmodell

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

## Soll-Ist-Vergleich

### Projektplan

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vorgangsname | Dauer | Anfang | Ende | Ressourcennamen | Geplante Arbeit | Benötigte Arbeit | Abweichung Arbeit (Std.) |
| **Projektinitialisierung** | **1 Tag** | **Mit 05.10.16** | **Mit 05.10.16** |  | **3 Std.** | **3 Std.** | **0** |
| Kickoff-Meeting | 1 Tag | Mit 05.10.16 | Mit 05.10.16 | Alina; Nicola; Timo | 3 Std. | 3 Std. | 0 |
| **Konzeption** | **2 Tage** | **Don 06.10.16** | **Fre 07.10.16** |  | **22 Std.** | **22 Std.** | **0** |
| Pflichtenheft | 1 Tag | Don 06.10.16 | Don 06.10.16 | Alina; Nicola; Timo | 6 Std. | 7 Std. | 1 |
| Projektplan | 1 Tag | Fre 07.10.16 | Fre 07.10.16 | Alina; Nicola; Timo | 16 Std. | 15 Std. | -1 |
| **Design** | **6 Tage** | **Fre 07.10.16** | **Fre 14.10.16** |  | **53 Std.** | **54 Std.** | **1** |
| Festlegen der Namenskonventionen | 1 Tag | Fre 07.10.16 | Fre 07.10.16 | Alina; Nicola; Timo | 3 Std. | 3 Std. | 0 |
| Erstellung ER-Diagramm | 4 Tage | Fre 07.10.16 | Mit 12.10.16 | Alina | 7 Std. | 6 Std. | -1 |
| Erstellung Klassendiagramm | 4 Tage | Fre 07.10.16 | Mit 12.10.16 | Alina | 13 Std. | 13 Std. | 0 |
| Erstellung Use-Case-Diagramme | 4 Tage | Fre 07.10.16 | Mit 12.10.16 | Nicola | 8 Std. | 9 Std. | 1 |
| Erstellung von Mock-Ups für die Oberfläche | 5 Tage | Fre 07.10.16 | Don 13.10.16 | Timo | 10 Std. | 11 Std. | 1 |
| Entwurf Testplan | 5 Tage | Fre 07.10.16 | Don 13.10.16 | Alina; Nicola; Timo | 6 Std. | 5 Std. | -1 |
| Zusammenfassung Diagramme / Anforderungen / Testplan | 1 Tag | Fre 14.10.16 | Fre 14.10.16 | Alina; Nicola; Timo | 6 Std. | 7 Std. | 1 |
| **Realisierung** | **20 Tage** | **Fre 14.10.16** | **Don 10.11.16** |  | **176 Std.** |  |  |
| Installation und Konfiguration von Eclipse, Tomcat, MySQL… | 2 Tage | Fre 14.10.16 | Mon 17.10.16 | Alina; Nicola; Timo | 15 Std. | 12 Std. | -3 |
| Erstellung der Tabellendefinitionen in der Datenbank | 1 Tag | Die 18.10.16 | Die 18.10.16 | Alina | 4 Std. |  |  |
| Datenbankanbindung | 1 Tag | Die 18.10.16 | Die 18.10.16 | Alina; Nicola; Timo | 6 Std. |  |  |
| Umsetzung Oberfläche | 8 Tage | Die 18.10.16 | Don 27.10.16 | Nicola | 35 Std. |  |  |
| Oberflächen-Logik | 10 Tage | Die 18.10.16 | Mon 31.10.16 | Alina; Nicola; Timo | 21 Std. |  |  |
| Backend-Entwicklung | 13 Tage | Mit 19.10.16 | Fre 04.11.16 | Alina; Timo | 50 Std. |  |  |
| Verbindung Frontend & Backend | 3 Tage | Mon 07.11.16 | Mit 09.11.16 | Alina; Nicola | 10 Std. |  |  |
| Responsive Design | 6 Tage | Fre 28.10.16 | Fre 04.11.16 | Timo | 29 Std. |  |  |
| Dokumentation / Refactoring | 1 Tag | Don 10.11.16 | Don 10.11.16 | Alina; Nicola; Timo | 6 Std. |  |  |
| **Test** | **5 Tage** | **Fre 11.11.16** | **Don 17.11.16** |  | **33 Std.** |  |  |
| Testkonzeption | 1 Tag | Fre 11.11.16 | Fre 11.11.16 | Alina; Nicola; Timo | 15 Std. |  |  |
| Testdaten pflegen | 2 Tage | Mon 14.11.16 | Die 15.11.16 | Nicola | 8 Std. |  |  |
| Integrationstest | 2 Tage | Mit 16.11.16 | Don 17.11.16 | Timo | 10 Std. |  |  |
| **Dokumentation** | **21 Tage** | **Fre 18.11.16** | **Fre 16.12.16** |  | **45 Std.** |  |  |
| Soll- / Ist-Vergleich | 1 Tag | Fre 18.11.16 | Fre 18.11.16 | Alina; Nicola; Timo | 5 Std. |  |  |
| Präsentation | 2 Tage | Fre 18.11.16 | Mon 21.11.16 | Timo | 15 Std. |  |  |
| Dokumentation | 19 Tage | Die 22.11.16 | Fre 16.12.16 | Alina; Nicola; Timo | 25 Std. |  |  |
| **dauernde Projektleitung (Abstimmungen, Entscheidungen,…)** | **52 Tage** | **Don 06.10.16** | **Fre 16.12.16** | **Alina; Nicola; Timo** | **28 Std.** | **2 Std.** | **-26** |

### Erfüllung Muss/Kann-Kriterien

**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. Log-In und Log-Out)
* Responsive Design auf Desktop und Android-Smartphone

**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.)

## Ressourcenzuordnung

Wer hat wie viele Stunden gemacht

|  |  |  |
| --- | --- | --- |
| **Name** | **Soll** | **Ist** |
| Alina Fankhänel | 120 |  |
| Nicola Kloke | 120 |  |
| Timo Schlüter | 120 |  |

# Fazit/Bewertung

Zusammenfassend lässt sich sagen, dass das Projekt fristgerecht, mit einer leichten Überschreitung der zuvor kalkulierten 360 Stunden, abgeschlossen wurde. Des Weiteren konnten neben den Muss-Kriterien, einige Kann-Kriterien erfolgreich umgesetzt werden.

Die gestalteten Diagramme in der Designphase bildeten eine gute Orientierung und entsprachen am Ende des Projekts mit wenigen Anpassungen dem Endresultat.

Zeitverluste mussten unter anderem durch Einarbeitung in Tools und fehlenden Programmierkenntnissten, sowie Differenzen innerhalb des Teams in Kauf genommen werden. Insgesamt entstand jedoch ein großer Lerneffekt, dadurch dass vieles von Grund auf erlernt werden musste. Ebenfalls wurde uns bewusst, wie wichtig eine gute Planung innerhalb des Teams ist.

*Alles in allem sind wir trotz leichter Abweichungen im Projektplan mit dem Endprojekt zufrieden. Wir haben alle Muss-Kriterien und auch einige Kann-Kriterien erfüllen können. Auch wenn zum Beginn der Realisierungsphase wir hinter dem Plan lagen, konnten wir durch einen stetigen Lernprozess den Rückstand zum Ende hin gut aufholen. Zu Beginn hat das fehlende Wissen den Projektplan ins Schwanken gebracht. Ein Problem während des Projekts war die Kommunikation innerhalb des Teams. Hier wurden Absprachen falsch verstanden, was zu Missverständnissen und Unmut geführt hat. Das hatte zur Folge, dass Termine nicht eingehalten worden sind und Aufgaben nicht richtig erfüllt wurden. Dieses hätte eventuell durch eine bessere und regelmäßigere Kommunikation zwischen den Mitgliedern des Teams verhindert werden können. Letztendlich haben wir den Unmut und die Probleme durch ein Gespräch zusammen mit Marcus Becker lösen können. In Zukunft müssen wir darauf achten, dass ähnliches nicht nochmal vorkommt. Zusammenfassend kann man sagen, dass das Projekt das gemacht hat was es machen sollte. Unsere Fähigkeiten haben sich in verschiedenen Disziplinen wie Projektplanung oder auch Programmierung über das Model-View-Control Konzept verbessert. Auch wurde unsere Fähigkeit Konflikte lösen zu können durch unser zwischenzeitiges Tief gestärkt.*

# Sonstiges

Sollten sich Screenshots oder Funktionalitäten ändern, sowie anderer Fehler in dem Dokument auftauchen, informieren Sie bitte den zuletzt genannten Autor.

Bei Verbesserungsvorschlägen wenden Sie sich bitte, an dem zuletzt genannten Autor.

# 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 |

# Quellenverzeichnis

* Brandt-Pook | Kollmeier - Softwareentwicklung kompakt und verständlich (1. Auflage 2008) für die Projektor-ganisation
* http://www.oracle.com/technetwork/articles/javase/servlets-jsp- 140445.html

# Glossar

|  |  |
| --- | --- |
| **Begriff** | **Beschreibung** |
|  |  |

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) [↑](#footnote-ref-2)