

**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 4](#_Toc466844915)

[1.1 Allgemein 4](#_Toc466844916)

[1.2 Systemumgebung 4](#_Toc466844917)

[1.2.1 Software 4](#_Toc466844918)

[1.2.2 Hardware 4](#_Toc466844919)

[1.2.3 Organisation und Hilfsmittel 4](#_Toc466844920)

[1.3 Zielgruppe 4](#_Toc466844921)

[2 Architektur der Anwendung 4](#_Toc466844922)

[2.1 Allgemeine Struktur 4](#_Toc466844923)

[2.2 Datenbank-Struktur 5](#_Toc466844924)

[3 Funktionen der Anwendung 6](#_Toc466844925)

[3.1 Kundenfunktionen 6](#_Toc466844926)

[3.2 Administrationsfunktionen 7](#_Toc466844927)

[3.3 Technische Funktionen 8](#_Toc466844928)

[4 Testprotokoll 9](#_Toc466844929)

[5 Projektmanagement 14](#_Toc466844930)

[5.1 Soll-Ist-Vergleich 15](#_Toc466844931)

[5.1.1 Projektplan 15](#_Toc466844932)

[5.1.2 Muss/Kann-Kriterien 17](#_Toc466844933)

[5.2 Ressourcenzuordnung 18](#_Toc466844934)

[6 Fazit/Bewertung 18](#_Toc466844935)

[7 Sonstiges 18](#_Toc466844936)

[8 Eigenständigkeitserklärung 18](#_Toc466844937)

[9 Quellenverzeichnis 19](#_Toc466844938)

[10 Glossar 19](#_Toc466844939)

Abbildungsverzeichnis

[Abbildung 1 – Modell-View-Controll 5](#_Toc466844894)

[Abbildung 2 – ER-Modell (geplant) 5](#_Toc466844895)

[Abbildung 3 – Use-Case Diagramm (Besucher) 6](#_Toc466844896)

[Abbildung 4 – Use-Case Diagramm (registrierter Kunde) 7](#_Toc466844897)

[Abbildung 5 – Use-Case Diagramm (Admin – Kundenverwaltung) 7](#_Toc466844898)

[Abbildung 6 – Use-Case Diagramm (Admin - Kategorieverwaltung) 8](#_Toc466844899)

[Abbildung 7 – Use-Case Diagramm (Admin – Festivalverwaltung) 8](#_Toc466844900)

[Abbildung 8 – Carl-Steinweg Phasenmodell 14](#_Toc466844901)

**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. Konkret wurde ein Onlineshop für das fiktive Unternehmen Festiva entwickelt, welches Festivaltickets und -zubehör über eine Onlineplattform verkaufen möchte.

## 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

Zielgruppe für den Onlineshop sind Kunden von Festiva aller Altersgruppen, da eine große Bandbreite an Festivals angeboten ist. Für den Administrationsbereich sind ausgewählte Mitarbeiter von Festiva verantwortlich, die die Rolle des Admin übernehmen.

# Architektur der Anwendung

## Allgemeine Struktur

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

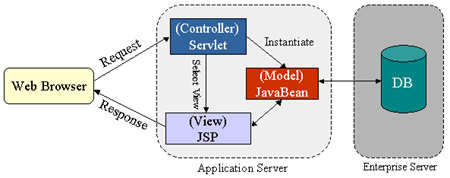


Abbildung 1 – Modell-View-Controll

Dabei wird der Quellcode entsprechend der verschiedenen Ebenen voneinander getrennt (3 Schichten Architektur).

## Datenbank-Struktur

Das ursprünglich geplante Datenmodell:

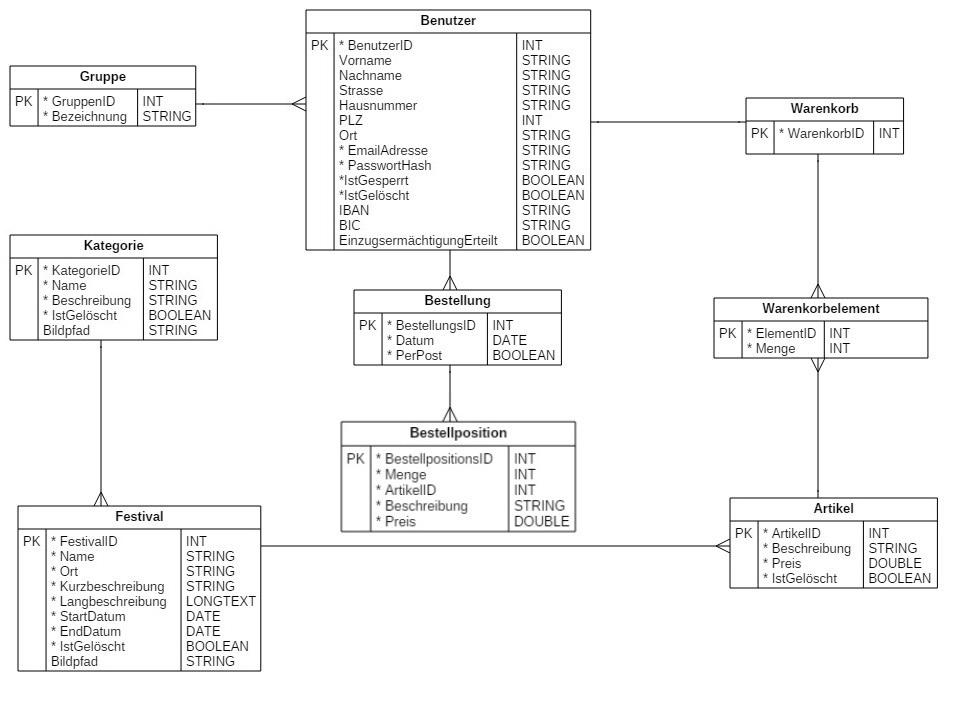


Abbildung 2 – ER-Modell (geplant)

Dieses wurde im Laufe des Projektes angepasst um weitere Kann-Kriterien, sowie die Suche nach dem maximalen Preis eines Festivals realisieren zu können.

Bei der Tabelle Benutzer wurde das Attribut „Passwortzaehler“ hinzugefügt um ein automatisches Sperren des Users nach drei falschen Passworteingaben zu realisieren.

Der Tabelle Artikel wurde das Attribut „Bildpfad“ hinzugefügt, um einem Zubehörartikel ein Bild zu geben, welches für das Kann-Kriterium des Festival-Zubehörs erforderlich war. Artikel die keinem Festival zugeordnet sind, werden im „Zubehörshop“ und in der Artikeldetailansicht mit Bild angezeigt.

Aktuelle Datenbankstruktur:

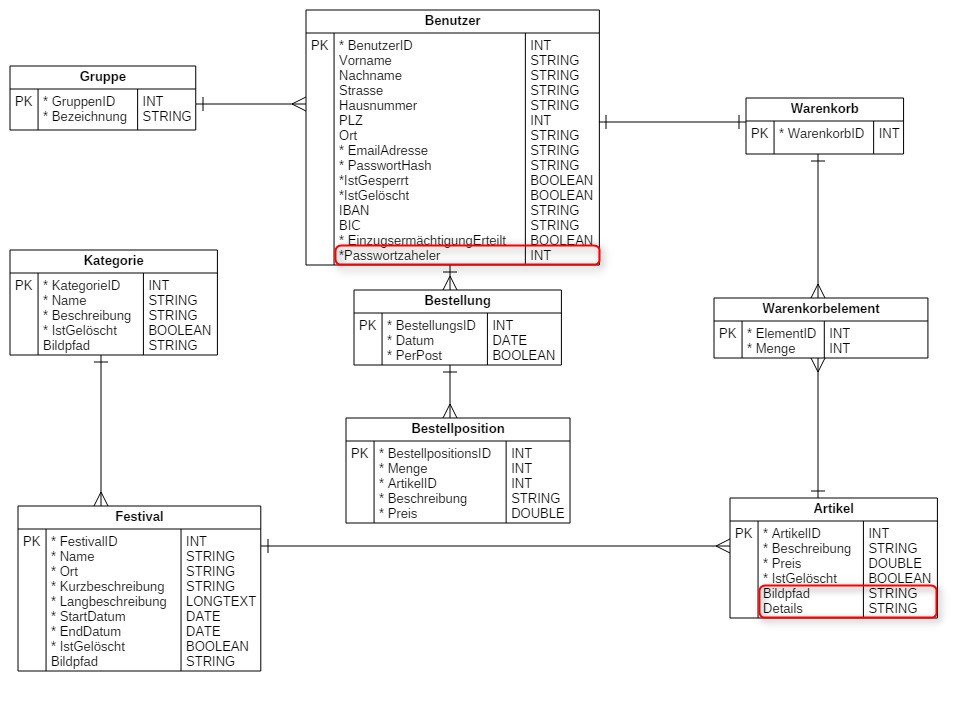


Abbildung – ER-Modell (neu)

# Oberfläche der Anwendung

## Planung

## Realisierung

# Funktionen der Anwendung

## Kundenfunktionen

### Planung

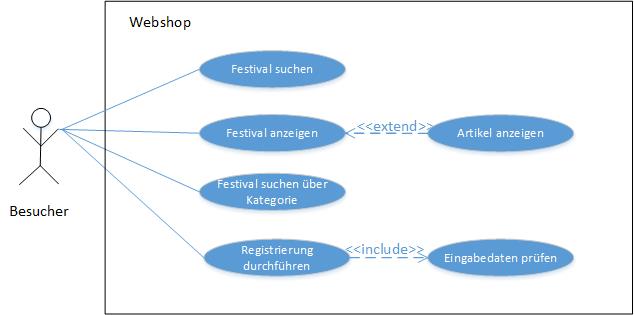


Abbildung 4 – Use-Case Diagramm (Besucher)

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.

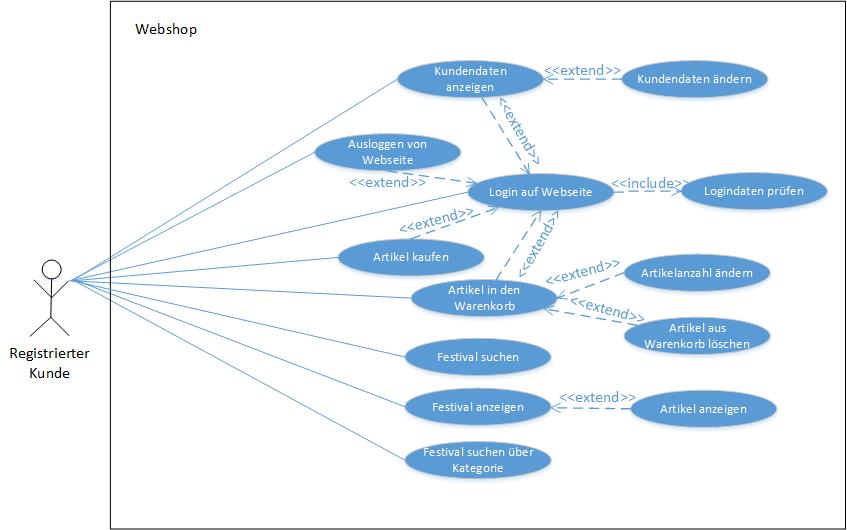


Abbildung 5 – Use-Case Diagramm (registrierter Kunde)

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.

### Realisierung

#### Kunde registrieren

JSP: k\_registrieren

Servlet: Registrierung

Java-Klasse: BenutzerManager

Um einen Kunden zu registrieren werden die E-Mail-Adresse sowie das gewünschte Passwort benötigt. Beides muss 2-mal eingegeben werden um ein Vertippen zu verhindern. Sobald man auf den Registrieren-Button wird das Servlet „Registrierung“ aufgerufen. Hier werden zunächst die Parameter der Seite übergeben und dann geprüft. Zunächst wird geprüft ob die E-Mail-Adresse bereits registriert ist. Dieses geschieht in der Java-Klasse „BenutzerManager“ in der Methode „selektiereBenutzer(String email)“. Ist der Benutzer noch nicht registriert – wird also nicht in der Datenbank gefunden – wird null zurückgegeben. Danach wird geprüft, ob die beiden eingegebenen E-Mail-Adressen identisch sind und anschließend dasselbe mit den Passwörtern. Abschließend wird ein neues Objekt der Klasse „Benutzer“, mit einem gehashten Passwort erstellt und durch die Methode „erstelleKunde(Benutzer benutzer)“ in der Klasse „BenutzerManager“ in die Datenbank geschrieben. Der Benutzer bekommt eine Rückmeldung zu seiner Registrierung und wird bei erfolgreicher Registrierung angemeldet und auf die Startseite geleitet.

#### Kunde anmelden/abmelden

JSP: k\_anmelden

Servlet: Login, Logout

Java-Klasse: BenutzerManager

Um sich als Kunde anzumelden wird die E-Mail-Adresse und das zugehörige Passwort benötigt. Hat man diese in der Anmeldemaske (k\_anmelden) eingegeben und auf den Anmelde-Button geklickt wird das Servlet „Login“ aufgerufen. Hier werden die Daten des Benutzers durch die E-Mail-Adresse mit Hilfe der Methode „selektiereBenutzer“ der Klasse „BenutzerManager“ aus der Datenbank geholt. Wird der Benutzer gefunden wird geprüft, ob dieser gesperrt ist und falls ja warum (Passwort 3-mal falsch oder von Admin gesperrt) und entsprechend Feedback an den Kunden gegeben. Ist der Benutzer nicht gesperrt wird zusätzlich geprüft, ob dieser gelöscht wurde und ggf. entsprechendes Feedback gegeben. Wenn der Kunde nicht gelöscht und nicht gesperrt ist, wird das Passwort geprüft. Hierfür wird das eingegebene Passwort gehasht und mit dem in der Datenbank liegenden Passwort-Hash verglichen. Sollten die Passwörter übereinstimmen wird eine neue Session eröffnet, der Passwortzähler auf 0 gesetzt und der Benutzer auf eine von drei Seiten geleitet. Sollte es sich um einen Kunden handeln (GruppenID=2), wird die Startseite angezeigt oder falls er zuvor einen Artikel unangemeldet in den Warenkorb legen wollte, auf die entsprechende Artikelseite. Ist der Benutzer ein Admin wird er auf die Admin-Startseite geleitet.

Beim Abmelden wird die Session des Benutzers geschlossen und der Benutzer auf die Startseite geleitet.

#### Kundendaten ändern

JSP: k\_kundendaten

Servlet: Benutzerdaten

Java-Klasse: BenutzerManager

Öffnet der Kunde „Meine Daten“ wird das Servlet „Benutzerdaten“ ausgeführt. Hier wird zunächst das JSP „k\_kundendaten“ aufgerufen, welches die aktuellen Benutzerdaten des Kunden anzeigen lässt. Möchte der Kunde seine Daten ändern, tut er dies in der Benutzeroberfläche und klickt auf „Änderungen speichern“. Dadurch wird wieder das Servlet „Benutzerdaten“ ausgeführt. Allerdings dieses Mal mit dem Attribut „ändern“. In diesem werden zunächst die Parameter des Formulars ausgelesen und zwischengespeichert. Daraus wird in der Methode „aendereDaten“ ein neues Objekt der Klasse Benutzer erstellt, welcher die gleiche ID hat wie der Benutzer. Anschließend wird über die Methode „aktualisiereBenutzer“ der Klasse „BenutzerManager“ die Daten des Benutzers geändert. Abschließend wird das JSP „k\_kundendaten“ mit den neuen Daten aufgerufen und der Kunde erhält ein entsprechendes Feedback ob die Änderung erfolgreich war oder z.B. durch eine bereits verwendete E-Mail-Adresse abgebrochen wurde.

Möchte der Kunde sein Passwort ändern, benötigt er sein aktuelles Passwort und muss sein neues Passwort zweimal bestätigen. Klickt der Kunde auf „Passwort ändern“, wird das Servlet „Benutzerdaten“ mit dem Attribut „p\_aendern“ aufgerufen. Dabei werden zunächst die Parameter des Formulars ausgelesen und entsprechend gehasht. Anschließend wird die Methode „aenderePasswort“ aufgerufen. Diese prüft ob das aktuelle Passwort richtig eingegeben wurde, die beiden neuen Passwörter übereinstimmen und aktualisiert das Passwort bei fehlerfreier Eingabe das Passwort. Das JSP „k\_kundendaten“ wird aufgerufen und der Kunde bekommt ein entsprechendes Feedback je nachdem ob die Änderung erfolgreich war oder nicht.

Möchte der Kunde sein Konto löschen kann er dieses über den Button „Mein Benutzerkonto löschen“ machen. Nach einem Klick erscheint ein über Javascript erstelltes Popup mit der Frage, ob man sich sicher ist, dass das Konto gelöscht werden soll. Wird dieses bejaht, wird das Servlet „Benutzerdaten“ mit dem Attribut „loeschen“ ausgeführt. Dabei wird der Boolean des Kunden „istGeloescht“ auf true gesetzt, und durch die Methode „löscheBenutzer“ in der Klasse „BenutzerManager“ logisch gelöscht. Abschließend wird das Servlet „Logout“ aufgerufen, welches die Session des Benutzers schließt und auf die Startseite weiterleitet.

#### Festival suchen

JSP: k\_ticketShop

Servlet: Ticketverwaltung

Java-Klasse: FestivalManager, WarenkorbManager

Bei der Suche kann der Kunde verschiedene Suchkriterien in einem Formular eingeben (JSP: k\_ticketShop). Es kann nach dem Namen, der Kategorie, dem Ort, dem Startdatum, dem Enddatum und dem maximalen Preis gesucht werden. Klickt der Kunde auf den „Suchen“-Button wird das Servlet „Ticketverwaltung“ mit dem Attribut „t\_anzeigen“ aufgerufen. Zunächst werden die Parameter des Formulars ausgelesen und zwischengespeichert. Dabei wird geprüft, ob Suchkriterien eingegeben wurden und ob das Startdatum vor dem Enddatum liegt. Ist dem nicht so wird die Suche nicht ausgeführt und der Kunde erhält entsprechendes Feedback. Anschließend wird die Methode „selektiereFestivalsInSuche()“ mit den zwischengespeicherten Parametern ausgeführt. Diese erstellt ein je nach Suchkriterium spezifisches SELECT-Statement und führt dieses aus. Es wird eine Liste von Festivals an das Servlet zurückgegeben, die dann wiederum an das JSP zurückgegeben und angezeigt wird.

#### Festival anzeigen

JSP: k\_festivaldetails

Servlet: Ticketverwaltung

Java-Klasse: FestivalManager, WarenkorbManager

Hat der Kunde ein Festival ausgewählt, wird das Servlet „Ticketverwaltung“ mit dem Attribut „f\_anzeigen“ ausgeführt. Zunächst werden die Parameter „festivalid“ und „maxpreis“ zwischengespeichert und die Warenkorbelemente des Kunden ebenfalls. Dieses ist nötig, um den Kunden ggf. darauf hinzuweisen, dass sich der Artikel bereits im Warenkorb befindet, wenn dieser hinzugefügt wird (4.1.2.6). Anschließend wird das Festival durch die Methode „selektiereFestival()“ der Klasse FestivalManager aus der Datenbank geholt und als Festival-Objekt erzeugt. Danach wird eine komplette Liste aller Artikel des Festivals durch die Methode „selektiereArtikelVonFestival()“ erzeugt.

Wird ein Maximalpreis angegeben werden zwei Artikellisten erzeugt. Die eine Liste beinhaltet alle Artikel an, die kleiner oder gleich dem Maximalpreis sind und die andere Artikelliste alle Artikel, die über dem maximalen Preis liegen. Das JSP „k\_festivaldetails“ wird mit dem Festival und den erstellten Listen aufgerufen.

Im JSP werden dann die Daten ausgelesen und aufgelistet.

#### Artikel zu Warenkorb hinzufügen

JSP: k\_festivaldetails

Servlet: Warenkorbverwaltung

Java-Klasse: WarenkorbManager, ArtikelManager

Möchte der Kunde einen Artikel in den Warenkorb legen, gibt er die gewünschte Menge ein und klickt auf den entsprechenden Button. Damit führt er die Javascript Funktion „einfuegen“ im JSP aus. Diese führt den Kunden, sollte er noch nicht angemeldet sein, zum Login-Screen. Anschließend wird geprüft, ob sich der Artikel bereits im Warenkorb befindet. Ist dieses der Fall wird gefragt, ob der Artikel trotzdem hinzugefügt werden soll. Wird dieses bejaht wird das Servlet „Warenkorbverwaltung mit dem Attribut „aktualisieren“ ausgeführt. Dabei werden zunächst die Parameter „userid“, „artikelid“ und „menge“ zwischengespeichert und ein Objekt vom Typ Artikel mit Hilfe der Methode „selektiereArtikel()“ der Klasse „ArtikelManager“ erstellt. Anschließend wird der Parameter „warenkorbid“ eingelesen und das entsprechende Warenkorbelement durch die Methode „selektiereWarenkorbelementMitArtikelID()“ erzeugt. Anschließend wird die Warenkorbmenge um die gewünschte Anzahl erhöht und geprüft, ob durch das Hinzufügen die von uns festgelegte maximale Menge von zehn überschritten werden würde. Ist dies der Fall bekommt der Kunde entsprechendes Feedback und die Menge wird auf zehn gesetzt. Wird die Maximalanzahl nicht erreicht, wird die Anzahl durch die gewünschte Menge erhöht und entsprechendes Feedback an den Kunden gegeben. In beiden Fällen wird das Warenkorbelement durch die Methode „aktualisiereWarenkorbelement()“ der Klasse „WarenkorbManager“ aktualisiert. Abschließend wird das JSP „k\_festivaldetails“ mit den entsprechenden Parametern und Feedbacks geöffnet.

#### Warenkorb bearbeiten

JSP: k\_warenkorb

Servlet: Warenkorbverwaltung, Ticketverwaltung, Produktverwaltung

Java-Klasse: WarenkorbManager, ArtikelManager

Soll der Warenkorb bearbeitet werden, muss der Kunde im JSP „k\_warenkorb“ 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 Attribut „aendern“ ausgeführt. Hier wird sich das aktuelle Warenkorbelement durch die Methode „selektiereWarenkorbelement()“ der Klasse „Warenkorbmanager“ aus der Datenbank geholt und ein Warenkorbelement-Objekt erzeugt, die Menge geändert und durch die Methode „aktualisiereWarenkorbelement()“ in der Datenbank aktualisiert.

Klickt der Kunde auf den „Löschen“-Button, wird das Servlet „Warenkorbverwaltung“ mit dem Attribut „loeschen“ ausgeführt. Hier wird die Methode „loescheWarenkorbelement()“ der Klasse „WarenkorbManager“ ausgeführt und der Eintrag in der Datenbank gelöscht.

#### Bestellung ausführen

JSP: k\_kasse

Servlet: Bestellverwaltung

Java-Klasse: BestellungsManager, WarenkorbManager

Befindet sich der Kunde im JSP „k\_kasse“ wird zunächst geprüft, ob der Kunde die Bestellung überhaupt ausführen kann. Die Benutzer- sowie Zahlungsdaten müssen dafür vollständig sein. Klickt der Kunde auf den nun klickbaren Button „Verbindlich bestellen“ wird das Servlet „Besetellverwaltung“ mit dem Attribut „anlegen“ ausgeführt. Hier wird erst der User und Versand ausgelesen und zwischengespeichert und dann geprüft, ob der User per Post oder Mail bestellt. Der Mailversand kann nur ausgewählt werden, wenn keine Zubehörartikel im Warenkorb sind, sondern nur Tickets. Anschließend wird der Warenkorb des Kunden als neuen Warenkorb erzeugt (WarenkorbManager.selektiereWarenkorbVonKunden()“) und dann eine neues Bestell-Objekt erstellt, und durch die Methode „erstelleBestellung()“ der Klasse „Bestellungsmanager“ in die Datenbank geschrieben. Danach wird der aktuelle Warenkorb mit der Methode „loescheWarenkorbinhalt“ der Klasse „WarenkorbManager“ geleert. Abschließend wird die Bestellübersicht durch das Servlet „Bestellverwaltung“ mit dem Attribut „anzeigen“ angezeigt.

## Administrationsfunktionen

### Planung

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

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

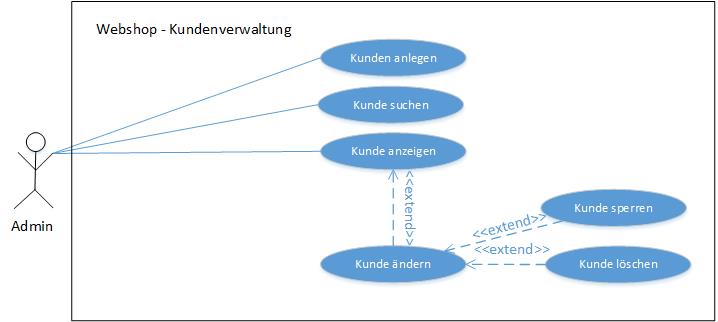


Abbildung 6 – Use-Case Diagramm (Admin – Kundenverwaltung)

Ein Admin kann einen Kunden anlegen, suchen, anzeigen und desweitern über den Änderungsmodus sperren und löschen.

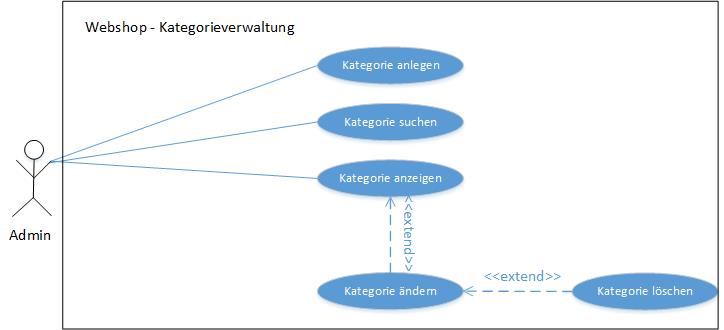


Abbildung 7 – Use-Case Diagramm (Admin - Kategorieverwaltung)

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

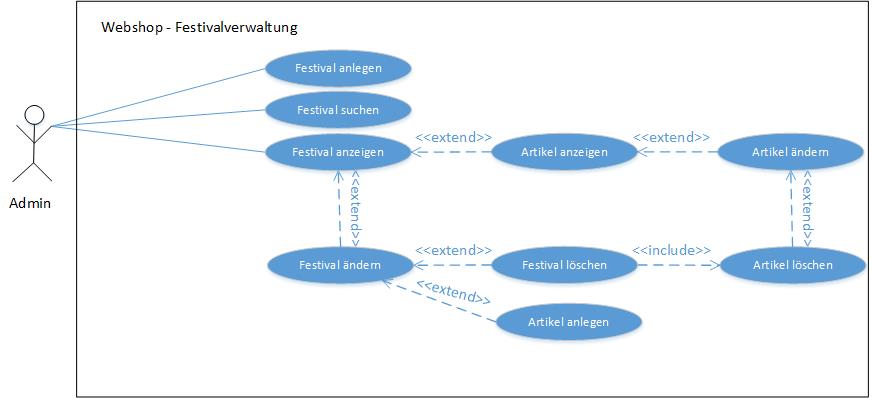


Abbildung 8 – Use-Case Diagramm (Admin – Festivalverwaltung)

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.

### Realisierung

#### Kundenverwaltung

JSP: a\_kundenverwaltung, a\_kundeAendern, a\_kundeAnlegen

Servlet: Kundenverwaltung

Java-Klasse: BenutzerManager

In der Kundenverwaltung kann der Admin Kunden anlegen und Kundendaten ändern. Klickt der Admin auf den Button „Neuen Kunden anlegen“ wird das JSP „a\_kundeAnlegen“ aufgerufen. Hier kann der Admin die Daten eingeben und auf „anlegen“ klicken. Dadurch wird das Servlet „Registrierung“ aufgerufen. Es geht weiter wie in 4.1.2.1, nur, dass am Ende das JSP „a\_kundenAnlegen“ aufgerufen wird.

Sollen Kundendaten geändert werden, muss der Admin auf die ID des Kunden klicken. Dabei wird das Servlet „Kundenverwaltung“ mit dem Attribut „aendern“ aufgerufen, die wiederum das JSP „a\_kundenAendern“ aufruft. Hier werden alle Daten des Kunden angezeigt. Diese können wie in 4.1.2.3 geändert werden. Auch der Servletaufruf ist derselbe, nur, dass am Ende das JSP „a\_kundenAendern“ aufgerufen wird.

#### Kategorienverwaltung

JSP: a\_kategorieverwaltung, a\_kategorieAendern, a\_kategorieAnlegen

Servlet: Kategorienverwaltung

Java-Klasse: Kategorienmanager

In der Kategorienverwaltung kann der Admin Kategorien anlegen, ändern und löschen. Klickt der Admin auf den Button „Neue Kategorie anlegen“ wird das JSP „a\_kategorieAnlegen“ aufgerufen. Hier kann der Admin die Daten eingeben, ein Bild hochladen und auf „anlegen“ klicken. Dadurch wird das Servlet „Kategorieverwaltung“ mit dem Attribut „anlegen“ ausgeführt. Es werden die Parameter eingelesen, zwischgespeichert und auf Vollständigkeit geprüft. Sind diese nicht vollständig wird ein entsprechendes Feedback zurückgegeben. Sind die Daten vollständig, wird zunächst eine neue Kategorie ohne den Bildpfad mit der Mehtode „erstelleKategorie“ der Klasse „KategorienManager“ angelegt. Anschließend wird das Bild in den Ordner „WebContent\Bilder“ hochgeladen und die Kategorie mit der Methode „aktualisiereKategorie“ aktualisiert. Abschließend wird das JSP „a\_kategorieAnlegen“ mit einem entsprechenden Feedback aufgerufen.

Möchte der Admin eine Kategorie ändern klickt er im JSP „a\_kategorieverwaltung“ auf die ID der entsprechenden Kategorie und führt das Servlet „Kategorieverwaltung“ mit dem Attribut „aendern“ aus. Hier werden zunächst die Parameter eingelesen und es wird geprüft, ob die notwendigen Informationen eingetragen wurden. Ist dem nicht so, wird ein entsprechendes Feedback an den Kunden gegeben und nichts geändert. Liegen alle Informationen vor, wird zunächst das Bild hochgeladen und geändert – sollte ein neues Bild vorliegen. Danach werden die anderen Informationen aktualisiert und durch die Methode „aktualisiereKategorie()“ auch in der Datenbank aktualisiert. Abschließend wird wieder das JSP „a\_kategorieAendern“ zusammen mit dem entsprechenden Feedback angezeigt.

Auch kann der Admin im JSP „a\_kategorieAendern“ das aktuelle Bild löschen. Durch Klick auf den Button wird das Servlet „Kategorieverwaltung“ mit dem Attribut „b\_loeschen“ aufgerufen. In diesem wird das Bild gelöscht und der Bildpfad der Kategorie auf „“ gesetzt. Dieses wird dann durch die Methode „aktualisiereKategorie()“ der Klasse „Kategoriemanager“ auch in die Datenbank geschrieben.

Gleiches gilt für den Button „Kategorie löschen“. Hier wird das Servlet „Kategorieverwaltung“ mit dem Attribut „loeschen“ aufgerufen. Zunächst werden alle Festivals der Kategorie mittels der Methode „selektiereAlleFestivalObjekteVonKategorie“ der Klasse „FestivalManager“ in eine Liste geschrieben und geprüft ob keine aktiven Festivals mehr verfügbar sind. Ist dies der Fall, wird die Kategorie durch die Methode „löscheKategorie()“ der Klasse „KategorieManager“ aus der Datenbank logisch gelöscht. Sollten noch aktive Festivals verfügbar sein, kann die Kategorie nicht gelöscht werden. Abschließend wird auf das JSP „a\_kategorieAendern“ mit einem entsprechenden Feedback gegangen.

Besonders muss beachtet werden, dass die ersten 4 Kategorien (ID 1-4) nicht gelöscht werden können, da diese unsere Standardkategorien sind. Gleiches gilt für die Löschung des Bildes.

#### Festivalverwaltung

JSP: a\_festivalverwaltung, a\_festivalAendern, a\_festivalAnlegen

Servlet: Festivalverwaltung

Java-Klasse: FestivalManager

In der Festivalverwaltung kann der Admin Festivals bzw. deren Artikel anlegen, ändern und löschen. Klickt der Admin auf den Button „Neues Festival anlegen“ wird das Servlet „Festivalverwaltung“ mit dem Attribut „anlegenanzeigen“ aufgerufen, das wiederum das JSP „a\_festivalAnlegen“ mit einer Liste der Kategorien (für die Kategorienzuordnung nötig) öffnet. Hier kann der Admin die Daten eingeben, ein Bild hochladen und auf „anlegen“ klicken. Dadurch wird das Servlet „Festivalverwaltung“ mit dem Attribut „anlegen“ ausgeführt. Es werden die Parameter eingelesen, zwischengespeichert und auf Vollständigkeit geprüft. Sind diese nicht vollständig wird ein entsprechendes Feedback zurückgegeben. Sind die Daten vollständig, wird zunächst ein neues Festival ohne den Bildpfad mit der Methode „erstelleFestival“ der Klasse „FestivalManager“ angelegt. Anschließend wird das Bild in den Ordner „WebContent\Bilder“ hochgeladen und das Festival mit der Methode „aktualisiereFestival“ aktualisiert. Abschließend wird das JSP „a\_festivalAnlegen“ mit einem entsprechenden Feedback aufgerufen.

Möchte der Admin ein Festival ändern, klickt er im JSP „a\_festivalverwaltung“ auf die ID des entsprechenden Festivals und führt das Servlet „Festivalverwaltung“ mit dem Attribut „aendern“ aus. Hier werden zunächst die Parameter eingelesen und geprüft, ob die notwendigen Informationen eingetragen wurden. Ist dem nicht so, wird ein entsprechendes Feedback an den Kunden gegeben und nichts geändert. Liegen alle Informationen vor, wird zunächst das Bild hochgeladen und geändert – sollte ein neues Bild vorliegen. Danach werden die anderen Informationen aktualisiert und durch die Methode „aktualisiereFestival()“ auch in der Datenbank aktualisiert. Abschließend wird wieder das JSP „a\_festivalAendern“ zusammen mit dem entsprechenden Feedback angezeigt.

Auch kann der Admin im JSP „a\_festivalAendern“ das aktuelle Bild löschen. Durch Klick auf den Button wird das Servlet „Festivalverwaltung“ mit dem Attribut „b\_loeschen“ aufgerufen. In diesem wird das Bild gelöscht und der Bildpfad des Festivals auf „“ gesetzt. Dieses wird dann durch die Methode „aktualisiereFestival()“ der Klasse „FestivalManager“ auch in die Datenbank geschrieben.

Für den Button „Festival löschen“ gilt, dass das Servlet „Festivalverwaltung“ mit dem Attribut „loeschen“ aufgerufen wird, nachdem man den Hinweis bestätigt hat, dass alle Artikel automatisch mitgelöscht werden. Zunächst wird das Festival durch die Methode „löscheFestival“ der Klasse „FestivalManager“ in der Datenbank logisch gelöscht. Anschließend werden alle Artikel des Festivals durch die Methode „löscheArtikel“ der Klasse „ArtikelManger“ logisch gelöscht. Abschließend wird das JSP „a\_festivalAendern“ mit einem entsprechenden Feedback für den Kunden aufgerufen.

Zusätzlich können noch Artikel zu den Festivals erstellt, geändert und gelöscht werden. Dieses läuft äquivalent zu dem Kapitel 4.2.2.4, nur, dass eine Festival-ID mit berücksichtigt wird und keine Bilder hinzugefügt werden können. Die Funktionsweise wird im folgenden Kapitel erklärt.

#### Artikelverwaltung (unabhängige Artikel)

JSP: a\_artikelverwaltung, a\_artikelAendern, a\_artikelAnlegen

Servlet: Artikelverwaltung

Java-Klasse: ArtikelManager

In der Artikelverwaltung kann der Admin festivalunabhängige Artikel, wie beispielsweise Regencapes und Zelte anlegen, ändern und löschen. Klickt der Admin auf den Button „Neuen Artikel anlegen“ wird das JSP „a\_artikelAnlegen“ aufgerufen. Hier kann der Admin die Daten eingeben und auf „anlegen“ klicken. Dadurch wird das Servlet „Artikelverwaltung“ mit dem Attribut „anlegen“ ausgeführt. Es werden die Parameter eingelesen, zwischengespeichert und auf Vollständigkeit geprüft. Sind diese nicht vollständig wird ein entsprechendes Feedback zurückgegeben. Sind die Daten vollständig, wird zunächst ein neuer Artikel ohne den Bildpfad mit der Methode „erstelleArtikel“ der Klasse „ArtikelManager“ angelegt. Anschließend wird das Bild in den Ordner „WebContent\Bilder“ hochgeladen und der Artikel mit der Methode „aktualisiereArtikel“ aktualisiert. Abschließend wird das JSP „a\_ArtikelAnlegen“ mit einem entsprechenden Feedback aufgerufen.

Möchte der Admin einen Artikel ändern, klickt er im JSP „a\_artikelverwaltung“ auf die ID des entsprechenden Artikels und führt das Servlet „Artikelverwaltung“ mit dem Attribut „aendern“ aus. Hier werden zunächst die Parameter eingelesen und es wird geprüft, ob die notwendigen Informationen eingetragen wurden. Ist dem nicht so, wird ein entsprechendes Feedback an den Kunden gegeben und nichts geändert. Liegen alle Informationen vor, wird zunächst das Bild hochgeladen und geändert – sollte ein neues Bild vorliegen. Danach werden die anderen Informationen aktualisiert und durch die Methode „aktualisiereArtikel()“ auch in der Datenbank aktualisiert. Abschließend wird wieder das JSP „a\_artikelAendern“ zusammen mit dem entsprechenden Feedback angezeigt.

Auch kann der Admin im JSP „a\_artikelAendern“ das aktuelle Bild löschen. Durch Klick auf den Button wird das Servlet „Artikelverwaltung“ mit dem Attribut „b\_loeschen“ aufgerufen. In diesem wird das Bild gelöscht und der Bildpfad des Artikels auf „“ gesetzt. Dieses wird dann durch die Methode „aktualisiereArtikel()“ der Klasse „ArtikelManager“ auch in die Datenbank geschrieben.

Für den Button „Artikel löschen“ gilt, dass das Servlet „Artikelverwaltung“ mit dem Attribut „loeschen“ aufgerufen wird, nachdem man den Hinweis bestätigt hat, dass der Artikel unwiderruflich gelöscht wird. Dadurch wird das Festival durch die Methode „löscheArtikel“ der Klasse „ArtikellManager“ in der Datenbank logisch gelöscht. Abschließend wird das JSP „a\_festivalAendern“ mit einem entsprechenden Feedback für den Kunden aufgerufen. Das Löschen des Artikels mit der ID 6 ist nicht möglich, da dieses unseren Postversand darstellt.

## Technische Funktionen

Technische Klassen, Datenbankzugriffe etc.

# Testprotokoll

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Funktion** | **M/K** | **Beschreibung** | **Erwartetes Ergebnis** | **Ergebnis** |
| T01 | Erfassen von Kundendaten (Kunde selbst) | M | Der Kunde registriert sich mit der eMailadresse [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 unzulässigem Passwort 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 Kategorieverwaltung auf Kategorie anlegen und gibt folgende Daten ein:  Name: Jazz  Beschreibung: Jazz Sessions  Bild: Jazz.jpg | Die Kategorie ist angelegt und wird bei der Kategorieverwaltung 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 Kategorieverwaltung 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 Gütersoh | 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 diese indem sie in der Langbeschreibung um die „Endzeit 22:00 Uhr“ ergänzt und speichert | 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 Bearbeitenmodus. 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. | Fehlermeldung „Seite nicht verfügbar“ |
| 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 | Bestellungen einsehen (Admin) | K | Der Admin klickt auf „Bestellungen“. | Die eben abgegebene Bestellung wird angezeigt | Wurde nicht implementiert, da die Anwendungsfälle unrealistisch sind |
| T38 | Seite auf Mobile Device starten | M | Die Seite wird über einen Emulator gestartet und angezeigt | Die Seite entspricht den Anforderungen eines Responsiv Designs |  |

Zusätzlich zu dem oben zusehenden Testplan wurden während der Realisierungsphase White-Boxtests der Komponenten von den Entwicklern durchgeführt

# Projektmanagement

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

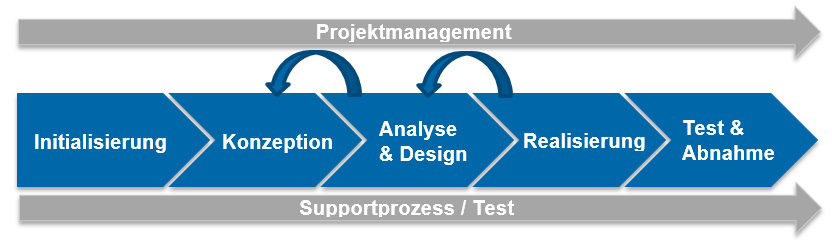


Abbildung 9 – Carl-Steinweg Phasenmodell

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** |

### 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** |
|  |  |