Deckblatt

Inhaltsverzeichnis

[1. Einleitung 1](#_Toc177301813)

[1.1. Anforderungen 1](#_Toc177301814)

[1.2. Ziele und Aufbau der Arbeit 1](#_Toc177301815)

[2. Grundlagen 2](#_Toc177301816)

[3. Konzept 5](#_Toc177301817)

[3.1. Strategieentwicklung 5](#_Toc177301818)

[3.2. Design-Entwürfe 7](#_Toc177301819)

[3.3. Technische Entwürfe 10](#_Toc177301820)

[4. Umsetzung 15](#_Toc177301821)

[4.1. Systemvorbereitung 15](#_Toc177301822)

[4.2. Datenbank 15](#_Toc177301823)

[4.3. Front-End 15](#_Toc177301824)

[4.4. Back-End 15](#_Toc177301825)

[5. Systemtests 15](#_Toc177301826)

[5.1. Testfall 1 15](#_Toc177301827)

[5.2. Testfall 2 15](#_Toc177301828)

[5.3. Testfall 3 15](#_Toc177301829)

[5.4. Testfall 4 15](#_Toc177301830)

[6. Zusammenfassung 16](#_Toc177301831)

[6.1. Kritische Reflexion 16](#_Toc177301832)

[I. Quellenverzeichnis X](#_Toc177301833)

[II. Dateianhänge XI](#_Toc177301834)

Abbildungsverzeichnis

[Abbildung 1: Arbeitsprozess mit Git 2](#_Toc177301703)

[Abbildung 2: Wasserfallmodell 5](#_Toc177301704)

[Abbildung 3: Prozesslandkarte 6](#_Toc177301705)

Abbildung 4: Wireframe\_Admin\_Lading\_Page 8

[Abbildung 5: Wireframe\_User\_Landing\_Page 8](#_Toc176526479)

[Abbildung 6: Wireframe\_Login\_Page 8](#_Toc177301707)

[Abbildung 7: Rakete 9](#_Toc177301708)

[Abbildung 8: Upload-B. 9](#_Toc177301709)

[Abbildung 9: Enity Relationship Modell 10](#_Toc177301710)

[Abbildung 10: Use Case 12](#_Toc177301711)

[Abbildung 11: Programmaufbau – Grobübersicht (Anhang: Structure.jpg) 13](#_Toc177301712)

[Abbildung 12: Dateistruktur im Programm 14](#_Toc177301713)

# Einleitung

# Anforderungen

Die allgemeinen und speziellen Anforderungen dieses Projekt sind aus dem Dokument **WEB42 – Webprojekt I Labor 2024** zu entnehmen. Die speziellen Anforderungen ergeben sich aus der Wahl des Projektes und der Anzahl der Teilnehmer dieses Projekts.  
Die speziellen Projektanforderung in diesem Fall sind:

*„2.1 Einzel-/Gruppenaufgabe: Mediendatenbank mit Web-Frontend Ausgangssituation: Es soll ein Webfrontend mit Zugriff auf eine Medien-Datenbank entwickelt werden. Die Anzahl der Medienarten, die in der Datenbank gehalten werden, sowie die konkrete Funktionalität ist abhängig von folgenden Gruppengrößen:*

*…*

*Gruppenaufgabe drei Teilnehmer: Medienarten »Foto«, »Hörbücher«, »eBooks«, »Video«; ein Administrator, mind. drei Anwender; detailliertes Vorgehensmodell, inkl. Aufgabenzuweisung Teilnehmer (selbständiges [Unter-]Kapitel, mind. zwei Seiten); detailliertes ER-Diagramm (selbständiges [Unter-]Kapitel, mind. zwei Seiten); vier dokumentierte Testfälle; Ausgabe Liste aller Medien in DB (mind. Name und Art des Medienelements; Bildschirmanzeige ausreichend) Folgende grundlegende Funktionen bietet die Webanwendung:*

* *Datenbankanbindung (Erstellen/Bearbeiten/Löschen)*
* *Accountverwaltung (über Webfrontend; Passwortschutz frei wählbar)*
* *Upload der Medien erfolgt über das Webfrontend; Mehrfachauswahl ist möglich*
* *Verschlagwortung: verwendete Zeichenketten nach Ersterfassung als Drop--Down-Menü (inkl. nachträgliches Bearbeiten oder Löschen)*
* *kombinierbare Suchfunktion über die Menge aller Objekte und deren Attribute*
* *Bildergalerie (u. a. auch über Suchfunktion generiert)*
* *Berücksichtigung anerkannter aktueller Styleguides für das Webfrontend“*

*(Schumacher, 2024, S. 4)*

# Ziele und Aufbau der Arbeit

Die Projektziele sind eine funktionstüchtige Mediendatenbank mit Web-Frontend zu erstellen, die den gestellten Anforderungen entspricht. Darüber hinaus hat diese den Anspruch zusätzliche, sinnvolle Anforderungen zu erfüllen, die sich u.a. aus der Semantik der Variablen ergeben. Diese sind im Kapitel „4.5. Testfälle“ zu finden. Um ein strukturiertes und vorzeigbares Ergebnis zu erzielen, darf natürlich ein ausgeklügeltes Konzept (siehe Kapitel 3) nicht fehlen. In diesem finden sich, außer Projektmanagement-methoden, noch Entwürfe, die z.B. Hilfestellung bei der Strukturierung der Daten oder auch beim Design geben sollen, denn auch wenn man als Entwickler einige kreative Freiheiten hat, sollte man sich an gewisse Grundsätze halten, um dem menschlichen Auge nichts Unzumutbares vorzusetzen. Die ebenfalls notwendige Dokumentation zu einem Projekt stellt dieser Laborbericht dar.

# Grundlagen

**Git**Wir erwähnen im weiteren Text, dass wir Git als Tool zur Kollaboration genutzt haben, deshalb möchten wir es an dieser Stelle kurz erläutern. Git ist ein Versionskontrollsystem, das hauptsächlich zur Verwaltung von Programmcode in Software-projekten verwendet wird. Git ermöglicht es mehreren Ent-wicklern gleichzeitig an einem Projekt zu arbeiten, ohne sich dabei gegenseitig in die Quere zu kommen.

Abbildung 1: Arbeitsprozess mit Git

**Composer**  
Composer ist ein PHP-Paket-Manager, das heißt man kann mithilfe von Composer externe Bibliotheken herunterladen und im eigenen Programm verarbeiten. Es gibt allerdings noch eine andere Funktion, die von Composer zur Verfügung gestellt wird. Das sogenannte „autoloading“. Diese Funktion ermöglich es, einen „Namespace“, dauerhaft zu laden. Sie erleichtert die Kommunikation zwischen verschiedenen Klassen, die sich in verschiedenen Dateien befinden.

Ein Bild, das Text, Screenshot, Schrift, Multimedia-Software enthält.

Automatisch generierte Beschreibung

In der obigen Abbildung ist zu sehen, wie die „autoload“ Funktionalität konfiguriert wird. Bei „app/“ handelt es sich in diesem Fall um den Code dieses Projektes. Nachdem diese Datei manuell erstellt wurde. Danach kann im Terminal über das Kommando:

*„php composer.phar dump-autoload“*

Die Funktionalität aktiviert werden. Mehr zur Installation und Verwendung ist auf folgender Website zu finden:  https://getcomposer.org   
Vgl. ​(Composer.org, 2024)​

**Design Pattern**   
Design Pattern sind Prinzipien, die auf das gleiche Problem in verschiedener Ausführung angewandt werden können. Für viele solcher Probleme haben sich diese Design Pattern als grundlegende Lösung etabliert. Vgl. ​(Siebler, 2014, S. 1)​

Folgend werden drei Pattern, die in dieser Arbeit verwendet wurden, vorgestellt.

1. **MVC (Model, View und Controller)**   
   Es handelt sich dabei um ein Design Pattern für das Erstellen einer Anwendung mit optischer Benutzeroberfläche. In diesem Fall wird das Pattern für eine Website verwendet, kann aber auch auf jegliches Programm übertragen werden, das eine optische Benutzerschnittstelle hat.

Die grundlegende Idee ist, die Trennung von Daten, Logik und Nutzersicht.

* Im Model werden die Daten angelegt z.B. das Definieren von Objekten
* Im Controller wird die Logik implementiert. Eingaben vom Nutzer werden verarbeitet, benötigte Daten werden der View bereitgestellt und sensible Daten bleiben versteckt.
* Die View ist die Nutzerschnittstelle. Der Nutzer kann mit der Website interagieren und Anfragen an den Controller schicken.

Der Hauptvorteil dieses Patterns ist die Abstraktion. Das bedeutet, der Nutzer hat nur Zugriff auf die View und die in ihr vordefinierten Aktionen. Sensible Informationen und Logik sind in Model und Controller versteckt. Dadurch wird ein kontrollierter und sicherer Zugriff ermöglicht. Vgl. ​(Siebler, 2014, S. 48)​

1. **Singleton**   
   Um die Notwendigkeit dieser Patterns zu verstehen, muss kurz auf die weitverbreitete Praxis der „dependency Injection“ eingegangen werden. Wird eine Methode aus einer anderen Klasse benötigt, kann dies entweder über einen statischen Modifikator, oder eben die „dependency  Injection“ erreicht werden, dabei wird eine neue Instanz der benötigten Klasse übergeben. In den meisten Fällen ist das kein Problem, es gibt aber auch Klassen und Methoden, bei der das Erstellen von immer neuen Instanzen zu Problemen führt, Stichwort Datenbank.

Eine Datenbank, die mehrere offene Instanzen hat, die dann alle bearbeitet werden, kann zu Inkonsistenzen und schwerwiegenden Problemen führen. Das Singleton Pattern stellt immer eine einzige Instanz zur Verfügung. Das verhindert Zugriffe von verschiedenen Instanzen und gewährleisten einheitlichen Zugriff auf eine intakte Datenbank.     
Vgl. ​(Siebler, 2014, S. 10-11)​

1. **Repository**   
   Diesen Begriff haben Sie vielleicht schon mal irgendwo anders gehört, im Zusammenhang mit einem GitHub Repository. Es handelt sich aber hier um etwas anderes. Das Repository, im Kontext eines Design Patterns, ist eine Abstraktion des Datenbankzugriffs. Der Zugriff auf die Datenbank wird also auf die Repository Klasse limitiert.

Das Repository bietet alle grundlegenden Funktionen zum Erstellen, Löschen, Lesen und Bearbeiten der Datenbank an. Dieses Pattern bewirkt eine klare Aufgabentrennung, vermeidet Redundanz und macht den Code dadurch flexibler und einfacher zu warten.   
Vgl. ​(Microsoft)

# Konzept

Bevor die Umsetzung des Projekts beginnt, haben wir unsere Strategie festgelegt, den Ablauf geplant und diverse Entwürfe erstellt und somit die Grundsteine für das Projekt gelegt.

# Strategieentwicklung

Wir haben uns zur Beschreibung unserer Prozessphasen für das Wasserfallmodell (siehe Abbildung 2) entschieden. Da es sich bei unserem Projekt, um ein eher kleines Projekt handelt, mit einem klar definierten Endziel, bei dem sich die Anforderungen und Abläufe bereits in der Planungsphase genau beschreiben lassen und sich nachträglich kaum ändern, haben wir es einem Agilen-Ansatz vorgezogen.

Unsere Prozessphasen erstrecken sich von der Anforderungsformulierung bis hin zur Phase der Wartung, die wir hier aber nur Vollständigkeitshalber aufgeführt haben. Die Projektplanung umfasst die Phasen Anforderungen, Prozess-Design und Entwürfe. Die Umsetzung erfolgt in der Implementierungs- und Testphase (Tests & Fixes).

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 2: Wasserfallmodell

Während der Planung- & Umsetzungsphasen ist es möglich vorherigen Phasen repetitiv zu wiederholen. Dabei wird sukzessive, ausgehend von der Phase, in der ein Problem festgestellt wurde in die nächstgelegene Phase zurückgekehrt, um den Ursprung der Fehlerquelle zurückzuverfolgen. Nach der Übergabe an den Kunden ist eine Fehlerbehebung in diesem Ausmaß nicht mehr möglich, weshalb die mögliche Rückführung in den vorherigen Schritt nicht mehr angezeigt ist.

Mit dem Fokus auf den Prozessen bzw. der Prozesskette haben wir uns zusätzlich noch für eine Prozesslandkarte entschieden, um unsere Abläufe zu optimieren, die Effizienz zu steigern und letztendlich unsere Ziele zu erreichen.

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

Abbildung 3: Prozesslandkarte

Es empfiehlt sich der Top down Ansatz, dem wir gefolgt sind, d.h. wir schauen erst auf die Unternehmung als Ganzes um anschließend in die Prozessketten zu gehen. Da es bei einer Prozesslandkarte darum geht, „wie“ etwas gemacht wird und nicht „wer“ genau „was“ macht, haben wir zusätzlich noch einen Zeitplan (siehe Zeitplan.pdf der anhängten Dokumente) erstellt. In diesem haben wir die identifizierten Aufgaben festgehalten und die Ressourcen zugewiesen. In diesem Fall heißt das, wir haben die Teammitgliederkapazität den offenen To-Do´s zugeteilt. Die drei größten Wertschöpfungsprozesse (Oberfläche design (Front-End), Funktionen integrieren (Back-End) und die Erstellung der Datenbank) haben wir grob gesehen jeweils einem Teammitglied zugeteilt. In der Spalte „Pers.“ Des Zeitplanes ist zu sehen, wie wir die identifizierten Aufgaben dazu aufgeteilt haben. Die Koordination der Kollaboration haben wir gemeistert, in dem wir uns zweimal die Woche ausgetauscht haben. Bei diesem Austausch wurden Ergebnisse besprochen ggf. Veränderungen vorgenommen und das weitere Vorgehen geplant. Bei der Planung wurden konkrete To-Do´s benannt, deren Umsetzung jeweils bis zum nächsten Meeting geplant war. Die Meetings entsprechen demnach unseren Meilensteinen. Die Zeit bis dahin könnte man als Sprint betrachten. Einer unserer wertvollsten Tools der Unterstützungsprozesse ist die Nutzung von Git, was zum einen unsere Ergebnisse versioniert sichert und zusätzlich beim Austausch gemeinsamer Daten unterstützt. Der geplante zeitliche Verlauf bis zur Fertigstellung der konkreten Aufgaben ist im Zeitplan durch die farbliche Markierung der Wochenspalten eindeutig erkennbar. Beides zusammen, Prozesslandkarte und Zeitplan, bilden eine gute Kombination, um das „wie“, „was“ und „wer“ abzubilden.

# Design-Entwürfe

Für das Thema des Designs haben wir uns entschieden, dass unser „Space“, also unsere Mediendatenbank das Thema „Space“ bekommen soll, weil uns die Analogie marketing-technisch passend vorkam. Bevor wir konkrete Entwürfe zu diesem Thema entwerfen können, müssen aber noch grundlegendere Aspekte geklärt werden z.B. welche Ansichten gibt es? Welche Elemente werden benötigt und wo befinden sie sich?  
Die Frage, die wir zuerst beantworten müssen, ist, welche Ansichten es gibt. Bei uns gibt es eine Ansicht für alle (normalen) Benutzer. Diese werden nicht weiter unterschieden. Und die Ansicht des Administrators. Sie unterscheiden sich u.a. in den Aktionen, die die Nutzeroberfläche bietet. Weitere Informationen dazu unter „3.3. Technische Entwürfe“ zu finden. Hier möchten wir aber weiter auf die Design-Entwürfe eingehen. Wireframes sind ein nützliches Hilfsmittel, um die Anordnung der benötigen Elemente der verschiedenen Ansichten zu visualisieren.

Abbildung 4 – 6 geben einen Vorgeschmack auf diese Wireframes. Um sie besser betrachten zu können, nutzen Sie bitte die Dokumente **Wireframe\_Login\_Page.pdf**, **Wireframe\_Admin\_Landing\_Page.pdf** und **Wireframe\_User\_Landping\_Page.pdf** im Anhang.

Ein Bild, das Text, Screenshot, Rechteck, Diagramm enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

Abbildung 4: Wireframe\_Admin\_Landing\_Page Abbildung 5: Wireframe\_User\_Landping\_Page

Die Admin Landing Page soll Aktionen bereitstellen zur Verwaltung von Nutzern und Schlagwörtern und außerdem ein Dashboard zur Visualisierung von nützlichen Informationen, wogegen die User Ansicht den Fokus auf den Umgang mit den Medien legt.

Ein Bild, das Text, Screenshot, Rechteck, Diagramm enthält.

Automatisch generierte BeschreibungEinige Seiten, wie zum Beispiel die Login Page sind bei beiden Sichten identisch. Diese ehergroben Entwürfe der Seiten lassen sich durch Mockups weiter ausbauen. Man orientiert sich dabei an den Wireframes und fügt Details hinzu. Man sollte sich dabei vorab damit auseinander-setzen, welches Farbschema verwendet werden soll.Das Farbschema unseres Projekts hat sich daraus ergeben, dass wir von einem Hintergrund überzeugt waren (siehe Anhang **Background.jpg**) und die restlichen Elemente darauf abgestimmt haben. Dadurch, dass der Hintergrund sehr dunkel ist, ist es für die Lesbarkeit essenziell, dass eine helle Schriftfarbe verwendet wird. Für Akzente bieten sich Grün, Blau oder Schwarz an, da diese Farben im Hintergrund enthalten sind. Wenn man etwas besonders hervorheben will, könnte man auch Komplementärfarben nutzen. Von der Farbe Blau wäre das beispielsweise Orange. Da unser Space ein eher ruhiges, homogenes Gesamtbild liefern soll, haben wir uns für Akzente im selben Farbton entschieden. Unseren Styleguide im Überblick zeigt Tabelle 1.

Abbildung 6: Wireframe\_Login\_Page

|  |  |  |  |
| --- | --- | --- | --- |
| **Akzente** | **Farbe** | **Farbcode** | **Deckkraft** |
| Rahmen | Türkis | #02BFBF | 100% |
| Buttons | Türkis | #02BFBF | 50% |
|  | | | |
| Schrift | Farbe | Farbcode | Schriftart |
| Allgemein | Weiß | #FFFFFF | PangMenZhengDao\* |
| Ausgewählt | Türkis | #02BFBF | PangMenZhengDao\* |
| Auf weißen Hintergrund | Grau | Türkis | #555555 | #02BFBF | PangMenZhengDao\* |
| \*bei einigen Elementen wie dem Auswahlmenu auch Roboto | | | |
|  | | | |
| Bilder | Größe (in px) | Name im Anhang | Bild |
| Logo | 100x100 | Rakete.png | Abbildung 7: Rakete |
| Upload-Button | 200x200 | Upload-Button.png | Abbildung 8: Upload-B. |

Tabelle 1: Styleguide

Kernstück dieses Gesamtbildes soll der Upload-Button sein, weshalb wir uns bei diesem für ein auffälliges Design entschieden haben, statt eines türkisen Standard-Buttons. Die dazugehörigen Mockups sind im Anhang zu finden unter den Namen   
**Login-Page\_Mockup.jpg, Admin-Landinpage\_Mockup.jpg** und **User-Landingpage  
\_Mockup.jpg**. Die Mockups wurden mit der Seite https://rp.mockplus.com/ erstellt. Bei der Implementierung ergab sich leider das Problem, dass uns die Schriftarten nicht zur Verfügung standen, weshalb wir auf die nächstähnliche Schriftart zurückgreifen mussten.

# Technische Entwürfe

Wir arbeiten in unserem Projekt mit der Datenbanksprache SQL (näheres dazu unter Systemvorbereitung). Es handelt sich also um eine relationale Datenbank. Um eine relationale Datenbank zu erstellen, eignet sich ein E/R-Modell hervorragend um vorab alle Entitäten, also Dinge der realen Welt, in Entitätstypen zusammenzufassen und die Beziehung zwischen den einzelnen Entitätstypen zu beschreiben.

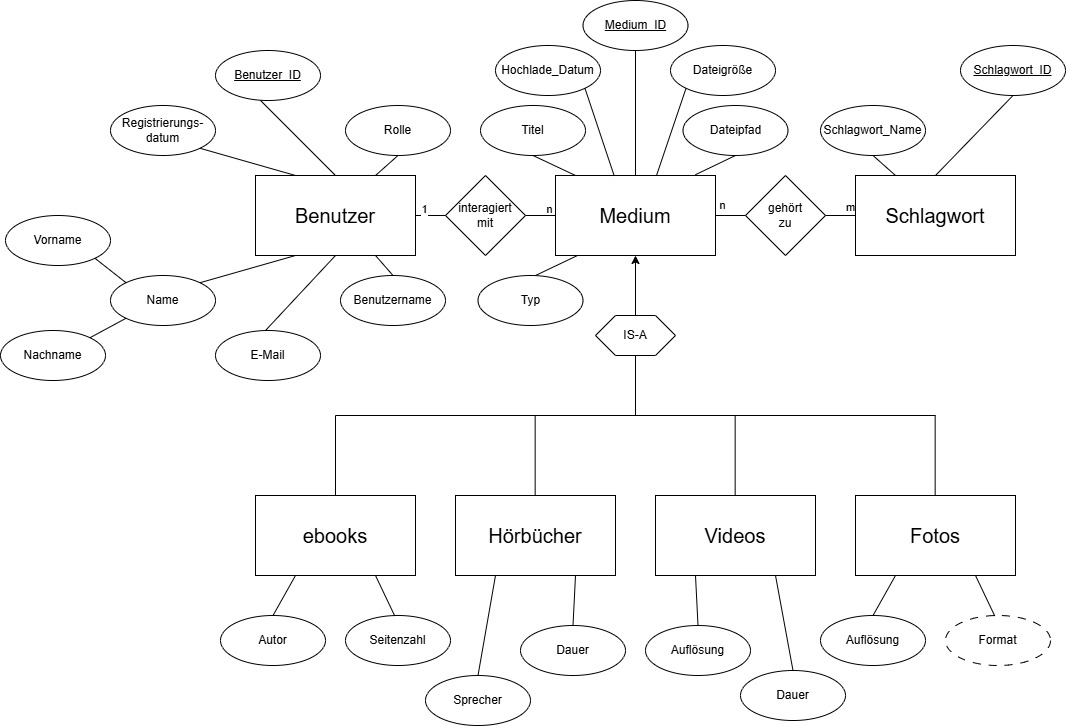


Abbildung 9: Entity Relationship Modell

Für eine größere Darstellung nutzen Sie bitte ER-Diagramm.jpg im Anhang.   
 Das E/R Diagramm beschreibt unsere Entitätstypen, wobei „ebooks“, „Hörbücher“, „Videos“ und „Fotos“ Spezialisierungen des Typs „Medium“ sind. Jede dieser Spezialsierungen besitzt die Attribute des General-Typs „Medium“ und zusätzliche noch ihre individuellen Attribute. Das individuelle Attribut „Format“ des Typs „Fotos“ ist ein abgeleitetes Attribut. Es ist kein für sich selbst stehendes Attribut, sondern es lässt sich aus dem Attribut Auflösung berechnen. Von Medium ausgehend stehen zwei weitere Entitätstypen in Beziehung zu diesem mit jeweils den Kardinalitäten einer 1 zu n und einer n zu m Beziehung. „Benutzer“ steht zu „Medium“ in einer 1 zu n Beziehung, da 1 Benutzer mit mehreren Medien interagieren kann (genaueres zu den Interaktionen der Benutzer im „Use Case“ im übernächsten Abschnitt dieses Kapitels) aber ein Medium nur von einem Benutzer verwendet werden kann. Anders verhält sich das zwischen Schlagwörtern und Medien. Hier handelt es sich um eine n zu m Beziehung, denn ein Schlagwort kann zu mehreren Medien gehören und ein Medium kann mehrere Schlagwörter besitzen. Bei den unterstrichenden Attributen handelt es sich jeweils um die Primärschlüssel des Entitätstyps. Wir haben uns dafür entschieden diesen immer als ID, also eine fortlaufende, einzigartige Nummer, auszuführen, obwohl z.B. bei Benutzer ebenfalls der „Benutzername“ in Frage kommen würde, da man in SQL mit dem „auto increment“ Parameter arbeiten kann, was bei der Implementierung einiges erleichtert.

Aus diesem abstrakten Modell lässt sich ein konkreteres Modell ableiten: das relationale Modell. Aufgrund der Größe steht es nur im Anhang zur Verfügung (**relationales Datenmodell.xlsx**). Während der Konzeptphase haben wir im Team über die drei Lösungsvarianten debattiert.

Lösung 1: Eine Tabelle für jede Spezialisierung mit Basistabelle

Lösung 2: Eine Tabelle für jede Spezialisierung ohne Basistabelle

Lösung 3: Eine Tabelle für alle Entitäten

Wir haben uns dabei für Lösung 2 entschieden, da es eine gute Kompromisslösung aus den Nachteilen von Lösung 1 und 3 ist. Lösung 1 hat den Nachteil, dass relativ aufwändige Joins notwendig sind, wenn ein oder mehrere bestimmte Tupel ausgewählt werden sollen. Lösung 3 hat den Nachteil, dass die spezialisierten Attribute mit „null“ ausgefüllt werden müssen, wenn sie nicht von der Entität verwendet wird. Lösung 2 reduziert die Anzahl der Joins durch das Ausbleiben der Basistabelle, muss dafür zwar redundante Attribute in Kauf nehmen, kommt aber dafür ohne leere Felder aus.

Ein weiterer technischer Entwurf widmet sich den bereits erwähnten Interaktionen. Einen abstrakten, aber dafür übersichtlichen Überblick bietet der „Use Case“ siehe Abb. 10.

Ein Bild, das Text, Fastfood, Essen enthält.

Automatisch generierte Beschreibung

Abbildung 10: Use Case

Wie wir in den Design-Entwürfen bereits festgestellt haben, gibt es in dem Projekt zwei Sichten, die sich durch die Verfügbarkeit unterschiedlicher Aktionen unterscheiden. Der Admin verfügt über die Admin-Konsole, in der Aktionen zur Nutzer- und globalen Schlagwortverwaltung auswählen kann. Er kann auf die Benutzer-Konsole wechseln, um die gleichen Aktionen ausführen zu können wie ein Benutzer. Dem Benutzer stehen zahlreiche Interaktionsmöglichkeiten mit den eigenen Medien und dem eigenen Account zur Verfügung inklusive eigens gewählter Schlagwörter. Eine zentrale Datenbank stellt das Herzstück dieser Interaktion dar. Sie wird verwaltet von einem Datenbankadministrator.

Das Ablaufdiagramm (siehe Anhang **Ablaufdiagramm.jpg**) zeigt diese Interaktionen im Detail und außerdem sonstige Interaktionen, die der User mit der graphischen Oberfläche eingehen kann. Der Startpunk befindet sich im Login Bereich. Hier hat der Nutzer die Möglichkeit sich mit seinem vorhandenen Account einzuloggen oder sich zu registrieren. Handelt es sich um einen Admin-Account gibt es einen eigenen Funktionenbereich (Adminspace), der bei der Admin-Landingpage beginnt. Von dort aus kann der Admin verschiedene Funktionen auswählen, die nur dem Admin zur Verfügung stehen oder in die Nutzeransicht wechseln, um sich die Funktionen des Nutzers anzeigen zu lassen, was u.a. zu Testzwecken nützlich ist. Ein „normaler“ User hat äquivalent dazu ebenfalls eine Landingpage mit auswählten Funktionen z.B. zur Verwaltung der eigenen Medien. Beide verlassen die jeweilige Ansicht über den Logoutbutton und kehren zurück zur Loginpage.

Ein Bild, das Text, Screenshot, Diagramm enthält.

Automatisch generierte BeschreibungDes Weiteren haben wir Entwürfe erstellt, um unseren Programmaufbau zu planen.

Abbildung 11: Programmaufbau – Grobübersicht (Anhang: Structure.jpg)

Die obige Abbildung fasst den Planungsprozess zum Design der Web-Applikation zusammen. Hier werden das MVC und Repository Pattern eingebaut, welche in den Grundlagen vorgestellt wurden. Dieser Erste Planungsschritt legt die Dateistruktur und die Vorgehensweise zum Programmieren der Applikation fest.

* Die **View** beinhaltet dabei die verschiedenen Ansichten programmiert in HTML
* Im **Controller** werden alle Anfragen, das Hochladen der Medien, die Nutzerverwaltung und gegebenenfalls der Zugriff auf das Repository durchgeführt.
* Im **Model** werden alle Datenstrukturen definiert.
* Das **Repository** kümmert sich um alle benötigten Datenbankzugriffe.

Hier können Sie noch die Dateistruktur sehen, die aus dieser Planung abgeleitet wurde

Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte Beschreibung

Abbildung 12: Dateistruktur im Programm

Im Weiteren Schritt der Programmaufbauplanung wurden die benötigten Klassen, Methoden und Attribute festgelegt. Wir haben uns dabei für die Erstellung eines Klassendiagramms entschieden. Aus Platzgründen möchten wir auch hier auf den Anhang verweisen (**Klassendiagramm.jpeg**). Je besser die Struktur des Codes, desto einfacher ist die Wartung und Erweiterung des Codes. Es zahlt sich also aus, sich im Vorhinein zwei- und dreimal zu überlegen, wie man das benötigte möglichst sauber umsetzt. Zusätzlich zu der Abstraktion, die mit dem MVC- und Repository Pattern erreicht wurden, wird für jede Objektart eine Abtrennung vorgenommen. Grundsätzlich wird für jede Klasse eine eigene Datei angelegt, Aufgrund der Struktur, die in der groben Programmplanung stattgefunden hat.

Das Diagramm verrät auch schon, die spätere Herangehensweise an das Routing und die Navigation, dass Sie sich beginnend bei der Klasse Index ansehen können. Die technische Umsetzung wird später detailliert erklärt.

# Umsetzung

# Systemvorbereitung

# Datenbank

# Front-End

# Back-End

# Systemtests

# Testfall 1

# Testfall 2

# Testfall 3

# Testfall 4

# Zusammenfassung

# Kritische Reflexion

# Quellenverzeichnis

Schumacher, Lutz (2024)  
WEB42 – Webprojekt I Labor 2024 (Zugriff 04.09.2024)

Prof. Plate, Jürgen; aktualisiert von Prof. Dr. Harwardt, Mark; Willamowski, Sonja (o.J.)  
Studienbrief INT102, Einführung in die Internetprogrammierung (Zugriff 04.09.2024)

Prof. Dr. Olderog, Torsten; Slama, Tim (o.J.)  
Studienbrief WEB602, Grundlagen der Programmierung mit PHP (Zugriff 04.09.2024)

Prof. Dr.-Ing. Timm, Heiko (o.J)  
Studienbrief DBA201, Einführung in die Datenbank-Programmierung   
mit MySQL und PHP (Zugriff 04.09.2024)

Sarah Laoyan (2024)  
Wasserfallmodell: Definition, Anwendung und Vorteile!   
URL: https://asana.com/de/resources/waterfall-project-management-methodology  
Zugriff am 19.08.2024

Julia Kupke (2023)  
Prozesslandkarten: Der Schlüssel zur Effizienzsteigerung in Unternehmen  
URL: https://www.d-velop.de/blog/prozesse-gestalten/prozesslandkarte/  
Zugriff am 19.08.2024

Simone Glitsch (o.J.)  
Wie Expert:innen eine Prozesslandkarte erstellen  
URL: https://prozessoptimierung-sprung.de/prozesslandkarte-erstellen/  
Zugriff am 19.08.2024

**Quellen der Abbildungen:**

**Hier fehlt noch einiges**  
Abbildung 1  
https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F  
Figure 6. Working tree, staging area, and Git directory (letzter Zugriff 09.09.2024)

„Abbildung 2:

# Dateianhänge

Eidesstattliche Erklärung

„Ich versichere, dass ich die beiliegende Ausarbeitung

selbstständig verfasst, keine anderen als die angegebenen Quellen und

Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der

Arbeit gekennzeichnet habe.“

Hamburg, xx.09.2024