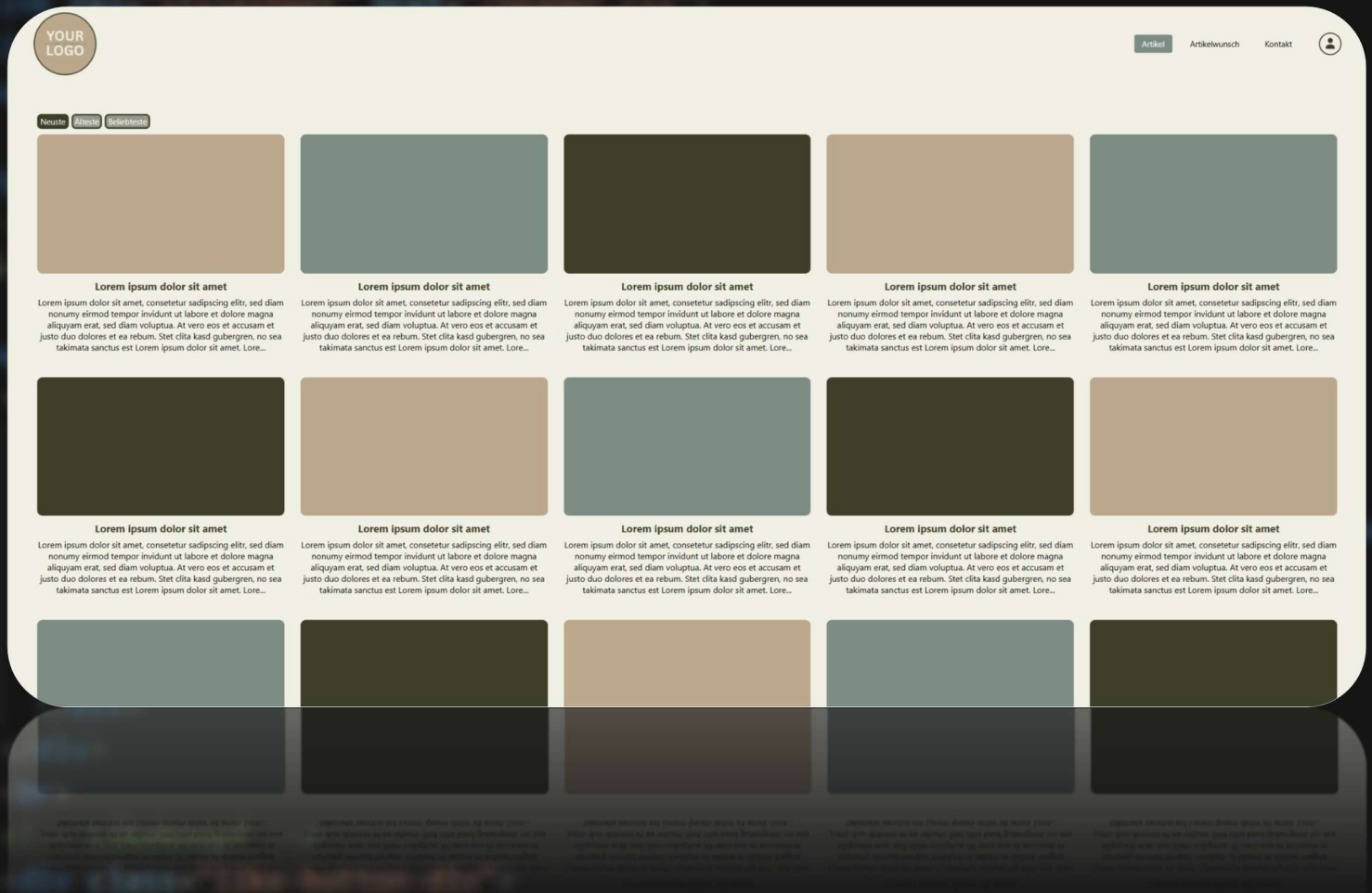


# Wie man einen Blog programmiert



Maturitätsarbeit, Kantonsschule Büelrain Winterthur

Verfasser: Jeremias Bisang, 4bW

Betreuung: Kaspar Jost

Korreferent: Robert Hofmann

Abgabedatum: 01.12.2025



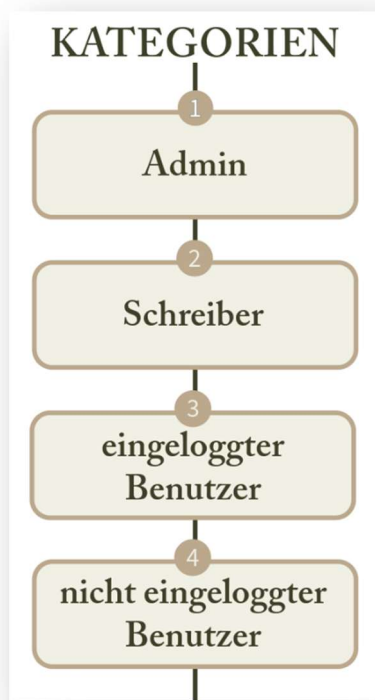
## **Abstract**

Websitebaukästen monatlich bezahlen – Nein, mein Blogtemplate ist eine unabhängige Webseite, selbst programmiert. Die 22 Unterseiten dieser Webseite sind in HTML und CSS programmiert, zusätzlich sorgt JavaScript im Hintergrund für allerlei Funktionen. Artikel veröffentlichen, wieder bearbeiten oder löschen, aber auch liken ist möglich. Der Inhaber hat die Kontrolle über seinen Blog, er sieht alle Benutzerkonten, kann diese schreibberechtigten und sieht eingereichte Artikelwünsche. Während dieser Maturitätsarbeit lernte ich mit «learning-by-doing» wie man richtige Webseiten programmiert.

# Inhaltsverzeichnis

1	Einleitung.....	- 2 -
1.1	Ausgangslage .....	- 2 -
1.2	Ziel.....	- 2 -
1.3	Konzept .....	- 2 -
2	Theoretischer Hintergrund.....	- 4 -
2.1	Webseiten im Internet.....	- 4 -
2.1.1	Das Internet .....	- 4 -
2.1.2	Webseiten .....	- 4 -
2.2	Wie Webseiten funktionieren.....	- 6 -
2.2.1	Frontend.....	- 6 -
2.2.2	Backend .....	- 8 -
2.2.3	Frameworks.....	- 10 -
3	Das Blogtemplate – Projektvorstellung.....	- 14 -
3.1	Sicht der Benutzer.....	- 14 -
3.1.1	System .....	- 14 -
3.1.2	Design und Layout.....	- 16 -
3.1.3	Ansicht und Handhabung.....	- 18 -
3.2	Blick hinter die Kulissen .....	- 22 -
3.2.1	Frontend.....	- 22 -
3.2.2	Backend .....	- 24 -
4	Projektverlauf.....	- 26 -
4.1	Ideenfindung .....	- 26 -
4.2	Planung .....	- 26 -
4.3	Umsetzung .....	- 28 -
4.4	Testung und Abschliessung.....	- 30 -
5	Rückblick.....	- 32 -
5.1	Evaluation .....	- 32 -
5.2	Reflexion .....	- 32 -
5.3	Fazit .....	- 34 -
5.4	Dank .....	- 34 -
5.5	Information zur Verwendung von künstlicher Intelligenz .....	- 34 -
6	Quellenverzeichnis.....	v
7	Anhang .....	b

Abb. 1: Kategorien von Benutzern



# 1 Einleitung

## 1.1 Ausgangslage

Ich interessiere mich schon lange für alles rund um den Computer und probierte schon vieles aus. Doch immer, wenn ich auf das Thema «Webseiten» stiess, kam ich nicht weiter, denn das Internet ist komplizierter als nur einen Code zu schreiben und auf den «Run-Button» zu drücken. Auch mit Webseitenbaukästen kam ich nicht weiter, denn bezahlen wollte ich nicht. Ich schaffte es, eine selbst programmierte, jedoch ausgesprochen unprofessionelle und an allen weitergehenden Funktionalitäten mangelnde Webseite – einen Blog – auf die Beine zu stellen.

So startete ich meine Maturaarbeit mit etwas breitem, aber nicht tiefem Computer- und Internetwissen und einer grundlegenden, doch unprofessionellen Erfahrung vom sogenannten «Frontend Webdesign». In anderen Worten: Einige Kenntnisse der Programmiersprachen HTML und CSS, mit denen man zwar Webseiten erstellen kann, jedoch fehlen alle weitergehenden Funktionen.

## 1.2 Ziel

Ziel dieser Maturaarbeit war es einen richtigen Blog beziehungsweise ein Blogtemplate, das heisst eine Vorlage, von Grund auf zu programmieren. Dieser soll völlig eigenständig funktionieren, unabhängig von einem Webseiten-Baukastenanbieter. Zusammengefasst: Selbst programmiert und volle Kontrolle über die Dateien. Die Idee ist, dass ich dieses Blogtemplate sowohl selbst nutzen als auch an andere weitergeben kann. Was das konkret für ein Blogtemplate werden sollte, konkretisierte sich im Verlauf des Projekts immer weiter. Doch einige grundlegende Ideen hatte ich von Anfang an, die nun auf die eine oder andere Weise im Endprodukt enthalten sind. Schlussendlich wurde es ein Blogtemplate mit seinen eigenen Funktionen und Eigenheiten.

## 1.3 Konzept

Das zugrundeliegende Konzept des Blogtemplates (vgl. Abb. 1) ist, dass der Admin den Blog verwaltet und über alle Rechte verfügt. Er kann Artikel veröffentlichen und hat Zugang zu den Verwaltungsoptionen des Blogs. Zusätzlich gibt es die sogenannten Schreiber, welche vom Admin ernannt werden. Sie können Artikel erstellen, diese bearbeiten und löschen. Jeder Artikel besteht aus einem Titelbild, welches während dem Erstellen eines Artikels hochgeladen werden kann, einem Titel und dem Blogartikel an sich. Auch die Leser des Blogs können sich einen Account erstellen, nur so können sie Blogartikel liken. Ausserdem ein wichtiges Feature ist die Computer- und Mobiletauglichkeit, sodass sich das Layout der Webseite anpasst an das Endgerät.

Abb. 2: Internet

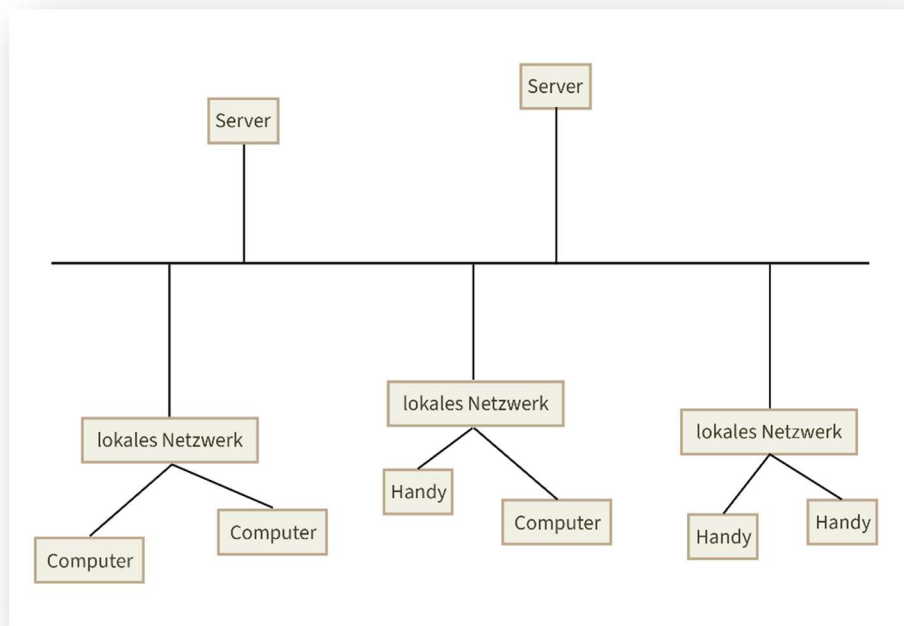


Abb. 3: Domain

Abb. 4: IP-Adresse

## 2 Theoretischer Hintergrund

In diesem Kapitel erkläre ich, wie das Internet und die Webseiten grundsätzlich funktionieren. Ausserdem stelle ich die in dieser Arbeit verwendeten Webtechnologien und Frameworks vor.

### 2.1 Webseiten im Internet

#### 2.1.1 Das Internet

Bevor man versteht, wie Webseiten funktionieren, muss man verstehen, wie das Internet funktioniert (vgl. Abb. 2). Das Wort Internet kommt vom englischen Begriff «*interconnected networks*». Es ist ein grosses Netzwerk bestehend aus vielen kleinen, lokalen Netzwerken. Auf diese Weise ist es möglich mit verschiedensten Computern auf der ganzen Welt zu kommunizieren.<sup>1</sup>

Das Internet hat kein Kontrollzentrum, sondern ist ein verteiltes Netzwerksystem, das nicht abhängig ist von einem zentralen Rechner, sondern jedes internetfähige Gerät ist ein Teil des Internets.<sup>2</sup>

#### 2.1.2 Webseiten

Eine Webseite ist im Grunde genommen ein Code, der an einem zugänglichen Ort im Internet, auf einem Server, ausgeführt wird. Ein Server ist ein Rechner, der seine Leistung für andere Computer bereitstellt.<sup>3</sup> «Betreten» werden kann ein Server über die Domain (vgl. Abb. 3) des Servers, das ist die Adresse, die wir in unserem Browser eingeben.<sup>4</sup> In meinem Fall ist das <https://maturaarbeit.jeremias-bisang.com>. Doch eigentlich ist die Domain eine Art Deckname für die IP-Adresse (vgl. Abb. 4), ausgeschrieben die Internetprotokoll-Adresse. Sie ist vergleichbar mit einer Postadresse, nur für Datenpakete anstatt physischer Pakete.<sup>5</sup>

---

<sup>1</sup> Vgl. Cloudflare Germany GmbH, «Wie funktioniert das Internet?», Internet

<sup>2</sup> Vgl. Cloudflare Germany GmbH, «Wie funktioniert das Internet?», Internet

<sup>3</sup> Vgl. Aschermann und Ohl, «Was ist ein Server? Definition und Funktion einfach zusammengefasst», Internet

<sup>4</sup> Vgl. P., «Was ist eine Website? Wie Websites funktionieren und vieles mehr», Internet

<sup>5</sup> Vgl. Wikifh et al., «IP-Adresse», Internet

Abb. 5: HTML-Beispiel-Code

```
1  <!DOCTYPE html>
2  <html lang="de">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Meine Website</title>
7  </head>
8  <body>
9    <main>
10     <p class="Titel">Das ist meine Website</p>
11     <button class="Knopf">Klicken</button>
12   </main>
13 </body>
14 </html>
```

Abb. 6: CSS-Beispiel-Code

```
1  .titel {
2    font-size: 18px;
3    font-weight: bold;
4    color: rgb(62, 63, 41);
5  }
6
7  .knopf {
8    padding: 3px 5px;
9
10   background-color: rgb(125, 141, 134);
11   border: none;
12   border-radius: 3px;
13
14   font-size: 16px;
15   color: white;
16 }
```

Abb. 7: Beispiel-Webseite



## 2.2 Wie Webseiten funktionieren

Die eigentliche Webseite, der Code, der auf dem Server ausgeführt wird, kann eingeteilt werden in zwei Teile. Das in der Fachsprache genannte Frontend und das Backend. Als Frontend bezeichnet man den sichtbaren Teil von Webseiten, der Teil, mit dem der Nutzer interagiert. Unter Backend versteht man sozusagen den hinteren Teil der Webseite, denn er läuft im Hintergrund ab und sorgt für die Funktionalität.<sup>6</sup>

### 2.2.1 Frontend

Das Frontend wird mithilfe der beiden Programmiersprachen HTML und CSS erstellt, ausserdem mit der Scripting-Sprache JavaScript,<sup>7</sup> welche ich im Frontend jedoch nur minimal verwendete. Mein Frontend besteht aus HTML-, CSS- und zusätzlich aus Bild-Dateien, beispielsweise für das Logo.

#### 2.2.1.1 HTML

Die Abb. 5 zeigt einen HTML-Beispiel-Code. Mit dieser Programmiersprache werden hauptsächlich die einzelnen Elemente, wie Texte oder Buttons, erstellt. Meine HTML-Dateien enden mit «.ejs» anstatt «.html», das ist ein Framework, welches es ermöglicht dynamische Inhalte anzuzeigen.<sup>8</sup> Also zum Beispiel Inhalte der Datenbank, die sich verändern können. Frameworks werden im späteren Verlauf dieses Kapitels erklärt.

#### 2.2.1.2 CSS

Nach dem Erstellen der einzelnen Elemente sind diese noch ohne Design, dieses wird mit Hilfe von CSS hinzugefügt, Proportionen wie Grösse oder Farbe werden angepasst. Auch komplexere Designs und Layouts sind möglich, beispielsweise Raster. Abb. 6 zeigt einen CSS-Beispiel-Code, anschliessend ist das Ergebnis aus beiden Programmiersprachen zusammen zu sehen (vgl. Abb. 7).

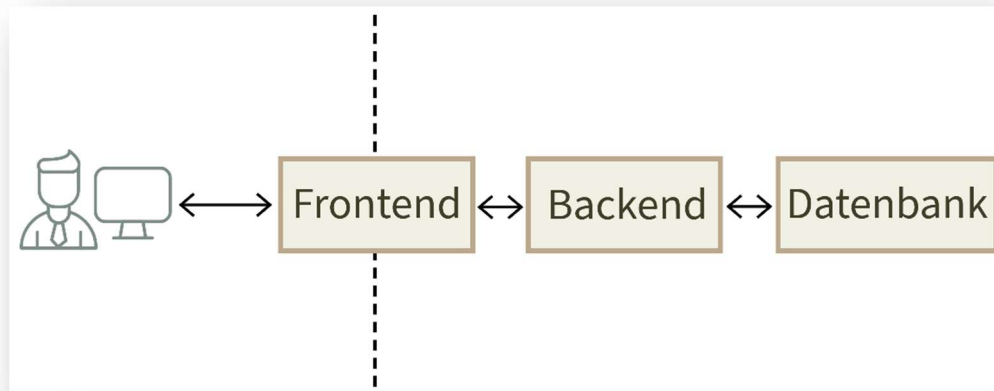
---

<sup>6</sup> Vgl. Tischlinger, «CMS-Revolution: Vom Backend zum Frontend», Internet

<sup>7</sup> Vgl. Tischlinger, «CMS-Revolution: Vom Backend zum Frontend», Internet

<sup>8</sup> Vgl. Ikechukwu, «Using EJS as a Template Engine in your Express App», Internet

Abb. 8: Serveraufbau



## 2.2.2 Backend

Das Backend ist für die Datenverarbeitung oder algorithmische Prozesse zuständig. Es definiert, was geschieht, wenn ein Benutzer eine Handlung ausführt, etwa einen Buttonklick.<sup>9</sup> In Abb. 8 ist der Aufbau zu sehen von Front- und Backend.

Das Backend kann in verschiedenen Programmiersprachen programmiert werden, häufig werden Programmiersprachen wie PHP, C oder Python verwendet.<sup>10</sup> Ich habe mein Backend jedoch in der Programmiersprache JavaScript programmiert. Dies tat ich in einem speziellen Environment, namens NodeJS.

Zusätzlich braucht es eine Datenbanksprache, die wohl bekannteste ist MySQL.<sup>11</sup> Ich benötigte jedoch bloss eine einfache Datenbank und importierte deshalb das Framework «better-sqlite3», mit welchem ich die vorprogrammierte Datenbank SQLite verwenden konnte. Dies gab mir die Möglichkeit, relativ unkompliziert eine Datenbank zu erstellen.

### 2.2.2.1 NodeJS

NodeJS oder Node.js ist eine sogenannte plattformübergreifende Laufzeitumgebung, die es ermöglicht, JavaScript-Code ausserhalb eines Webbrowsers auszuführen<sup>12</sup>, in meinem Fall als Webserver.

### 2.2.2.2 SQLite

SQLite ist eine Datenbank-Engine, also eine Art vorprogrammierte Datenbank. Sie ist in der Programmiersprache C geschrieben und wird als Bibliothek oder Framework importiert, um sie zu verwenden.

Im praktischen Sinne ist die Datenbank eine Datei mit der Endung «.db». Um die Datenbank visuell ansehen und auch manuell bearbeiten zu können, braucht man eine Desktop-Applikation wie «DB-Browser (SQLite)».

---

<sup>9</sup> Vgl. Tischlinger, «CMS-Revolution: Vom Backend zum Frontend», Internet

<sup>10</sup> Vgl. Tischlinger, «CMS-Revolution: Vom Backend zum Frontend», Internet

<sup>11</sup> Vgl. Tischlinger, «CMS-Revolution: Vom Backend zum Frontend», Internet

<sup>12</sup> Vgl. Valanagut et al., «Node.js», Internet



### 2.2.3 Frameworks

Beim Entwickeln einer Anwendung programmiert man in der Regel nie alles von Grund auf selbst. Zum einen ist da die Programmiersprache, die schon vieles übernimmt. Zum anderen gibt es viele vorgefertigte, sogenannte Frameworks. Das sind eine Art Entwicklungsrahmen, welche die Programmierung erleichtern.<sup>13</sup>

#### 2.2.3.1 Express.js<sup>14</sup>

Express ist ein Backend-Webapp-Framework. Es wird gesagt, dass es de facto das Standard-framework für NodeJS-Server ist. Es wird zum Beispiel von PayPal und Uber verwendet.<sup>15</sup> Express bietet sehr essenzielle Funktionen für ein Backend.

#### 2.2.3.2 bcrypt

Das ist ein Framework, das dabei hilft, Passwörter zu verschlüsseln.<sup>16</sup> Ich habe es verwendet, um die Benutzerpasswörter zu verschlüsseln, das erhöht die Sicherheit, da ein Hacker sich nicht einfach einloggen kann, selbst wenn er Zugang zu den Passwörtern hat.

#### 2.2.3.3 better-sqlite3

Dieses Framework macht es sehr einfach möglich, die vorprogrammierte Datenbank SQLite zu verwenden.<sup>17</sup> Man muss nur einige Details kennen, wie man sie handhabt, dann ist sie schon einsatzfähig. Das war sehr praktisch, da die Datenbank einen grossen Teil meiner Webseite ausmacht.

#### 2.2.3.4 cookie-parser<sup>18</sup>, jsonwebtoken<sup>19</sup> und dotenv<sup>20</sup>

Diese drei Frameworks benutzte ich zur Erzeugung und Verwendung von Cookies. Diese benötigte ich, um einen Benutzer einzuloggen. Mein Cookie ist eine lange Reihe an Buchstaben, Zahlen und Sonderzeichen, das bei jedem Benutzer einzigartig ist und in regelmässigen Abständen erneuert wird. Dieses wird auf dem Browser des Benutzers abgespeichert und gibt dem Server zu erkennen, in welchen Account ein Benutzer eingeloggt ist.

---

<sup>13</sup> Vgl. Business Systemhaus AG, «Was ist ein Framework?», Internet

<sup>14</sup> Vgl. OpenJS Foundation, «Express - Node.js Web Application Framework», Internet

<sup>15</sup> Vgl. Wikipedia, «Express.js», Internet

<sup>16</sup> Vgl. jfirebaugh et al., «node.bcrypt.js», Internet

<sup>17</sup> Vgl. joshuawise, «better-sqlite3», Internet

<sup>18</sup> Vgl. ulisesgascon, dougwilson und defunctzombie, «cookie-parser», Internet

<sup>19</sup> Vgl. charlesrea et al., «jsonwebtoken», Internet

<sup>20</sup> Vgl. ~jcbtlw et al., «dotenv», Internet



### 2.2.3.5 EJS

Wie zuvor erwähnt kann man mit diesem Framework im Frontend, im HTML-Code, Backend-JavaScript ausführen und so dynamische Inhalte anzeigen.<sup>21</sup> Ich benutzte es beispielsweise um Informationen aus der Datenbank anzuzeigen oder die richtigen Error-Meldungen einzublenken.

### 2.2.3.6 marked<sup>22</sup>

Mit diesem Framework war ich in der Lage, sehr schnell die Markdown-Formatierung bei der Artikelerstellung hinzuzufügen. So sind Funktionen wie Absätze, Fettschreibung oder Listen bei der Erstellung von Artikeln möglich.

### 2.2.3.7 multer<sup>23</sup>

Multer ermöglichte es mir, eine Bildhochladefunktion einzubauen. Ich benötigte diese Funktion für die Titelbilder der Artikel und für das Logo. Die Bilder werden als normale Bilddateien auf dem Server abgespeichert.

### 2.2.3.8 sanitize-html<sup>24</sup>

Sanitize-HTML bereinigt benutzergenerierte Inhalte von HTML-Code und schliesst so einige Sicherheitslücken.<sup>25</sup>

### 2.2.3.9 nodemon<sup>26</sup>

Nodemon ist ein sehr nützliches Framework während des Entwicklungsprozesses. Es startet den Server automatisch neu, wenn der Entwickler eine Änderung sichert, sonst müsste der Entwickler das jedes Mal manuell ausführen. Das ersparte mir viel Mühe.

---

<sup>21</sup> Vgl. Ikechukwu, «Using EJS as a Template Engine in your Express App», Internet

<sup>22</sup> Vgl. chjj et al., «Marked», Internet

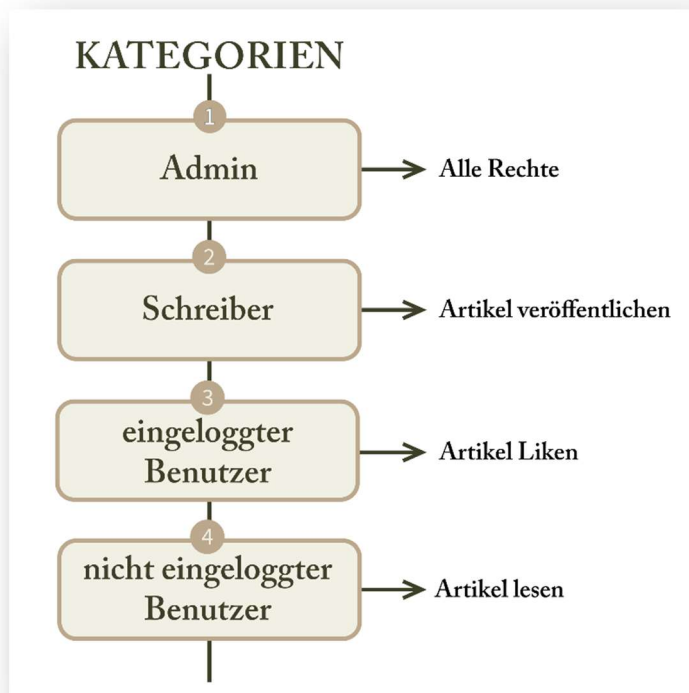
<sup>23</sup> Vgl. ctcpip et al., «Multer», Internet

<sup>24</sup> Vgl. alexgilbert et al., «sanitize-html», Internet

<sup>25</sup> Vgl. alexgilbert et al., «sanitize-html», Internet

<sup>26</sup> Vgl. remy, «nodemon», Internet

Abb. 9: Kategorien mit Rechten



## **3 Das Blogtemplate – Projektvorstellung**

In diesem Kapitel stelle ich mein Blogtemplate vor. Wie zuvor erwähnt, ist ein Template im Webentwicklungsbereich nichts anderes als eine Vorlage. Das heisst, mein Blogtemplate ist einfach gesagt ein leerer Blog, der nun von Blogautoren befüllt werden kann.

### **3.1 Sicht der Benutzer**

In diesem Unterkapitel stelle ich die Sicht der Benutzer vor, also alles, was der Benutzer direkt zu sehen bekommt. Wie das alles zustande kommt, zeige ich im nächsten Unterkapitel «Blick hinter die Kulissen».

#### **3.1.1 System**

Grundsätzlich funktioniert der Blog folgendermassen (vgl. Abb. 9): Jeder Benutzer der Webseite gehört zu einer von vier Kategorien. Die unterste Kategorie sind die Besucher ohne Account. Sie können Artikel lesen und Artikelwünsche einreichen. Die nächste Kategorie stellen die Besucher dar, die sich einen Account erstellt haben. Sie sind zusätzlich in der Lage, Blogartikel zu liken. Anschliessend kommen die Schreiber. Ihre zusätzlichen Rechte sind zum einen Blogartikel zu veröffentlichen, ausserdem diese zu bearbeiten und zu löschen. Auch können sie die zuvor erwähnten Artikelwünsche einsehen. Zuoberst ist der Blog-Admin beziehungsweise Bloginhaber. Dieser hat alle Rechte, das heisst er kann im Gegensatz zu den Schreibern auch fremde Artikel bearbeiten und löschen. Ausserdem ist er derjenige, der die Schreiber ernennt. Zusätzlich kann er ein Logo hochladen und die Kontaktdaten anpassen.

Abb. 10: Farbpalette



Abb. 11: Header- und Artikelraster

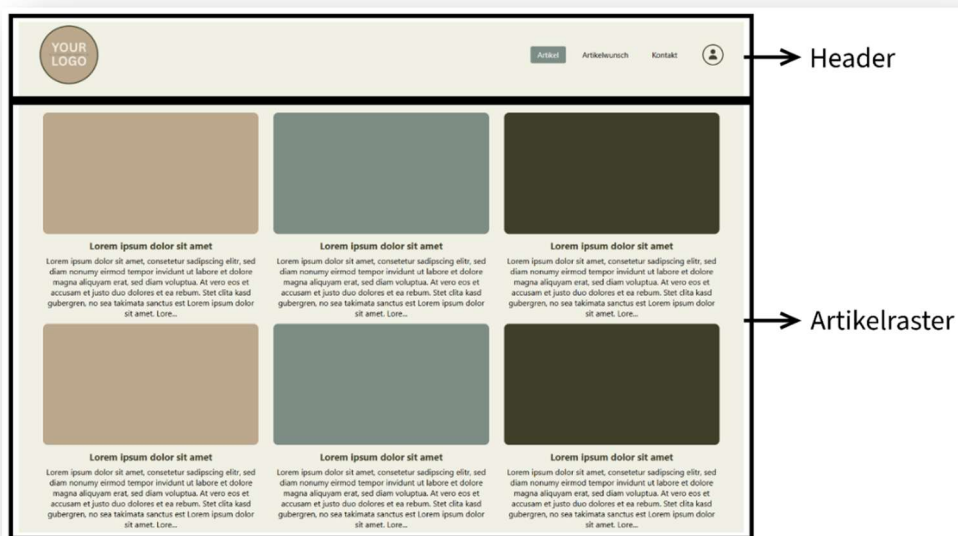


Abb. 12: Admin-Dashboard



### 3.1.2 Design und Layout

Es macht Sinn, sich auf eine Farbpalette festzulegen, bevor man überhaupt beginnt über Farben nachzudenken. So stellt man sicher, dass die Farben zusammenpassen und muss sich nicht den Kopf darüber zerbrechen. Im Verlauf der Entwicklung habe ich mich für eine pastellartige Grün-Blau-Braun-Mischung (vgl. Abb. 10) entschieden.

Bezüglich der Schriftart entschied ich mich für «Segoe UI». Es ist eine angenehme und gut leserliche Schriftart.

Farbe und Schriftart sind zwei Aspekte, die viel von der Webseite ausmachen und auch schnell geändert sind, deshalb werde ich diejenigen, die das Template verwenden möchten, fragen, ob ich ihnen diese zwei Proportionen anpassen soll entsprechend ihren Wünschen.

Für das Layout der Webseite habe ich allgemeine, häufig benutzte Layoutideen verwendet. Der sogenannte Header beziehungsweise die Navigationsbar befindet sich bei den allermeisten Webseiten heutzutage am oberen Rand der Webseite und bleibt auch sichtbar, wenn man herunterscrollt. Dementsprechend sieht auch der Header bei meinem Blogtemplate aus (vgl. Abb. 11). Die Anordnung von Logo und Navigationsbuttons sieht man häufig in etwa dieser Art, wie ich sie erstellt habe. Andere Teile der Webseite sind häufig in einer ähnlichen Art zu sehen, so zum Beispiel die Rasteranordnung der Blogartikel auf der Hauptseite (vgl. Abb. 11). In diesem Fall bin ich auf die Idee gekommen durch einen anderen Blog<sup>27</sup>. In der Regel habe ich mich jedoch nicht an einer Webseite orientiert und versucht das Layout nachzuahmen, sondern habe das Layout selbst entworfen, inspiriert durch gängige Webstandards. Bei einigen Teilen habe ich mir frei etwas ausgedacht, ohne Webstandards im Kopf zu haben, so zum Beispiel beim Dashboard (vgl. Abb. 12). Übersichtsseiten wie ein Dashboard sehen oft etwas anders aus, doch es schien mir gerade passend so, da die Übersicht auf diese Art kompakt ist.

---

1 <sup>27</sup> Vgl. The Daily Grace Co., «The Daily Grace Blog», Internet

Abb. 13: Artikel erstellen

The diagram illustrates the process of creating an article, starting from the Admin dashboard and moving to the article creation form and upload section.

**Dashboard von Admin**

- Email...
- 
- 
- 
- Benutzername...
- Benutzername...
- 
- 
- 

**Dashboard von Schreiber**

- 
- 
- 

**Article Creation Form**

- Titel...
- Artikel...
- 

**Upload Section**

- Titelbild mit Format 16:9 hochladen.
- 

A large curved arrow points from the 'Artikel erstellen' button in the Admin dashboard to the article creation form. A straight arrow points from the 'Veröffentlichen' button in the article creation form to the upload section.

### 3.1.3 Ansicht und Handhabung

Ein Besucher des Blogs kommt zuerst auf die Artikelseite. Hier sieht er alle Artikel aufgelistet, sortiert nach Veröffentlichungsdatum, der neuste Artikel zuoberst. Mit Sortierung-Buttons kann er die Sortierung jedoch ändern, sodass der älteste oder beliebteste Artikel zuoberst ist.

Mit einem Klick auf einen Artikel kann der Besucher nun einen Artikel lesen und wenn er eingeloggt ist auch liken.

Einloggen oder registrieren kann sich ein Besucher über das Benutzer-Icon in der oberen, rechten Ecke. Über dasselbe Icon kommt ein eingeloggter Besucher, ein Schreiber oder der Admin zu seinem Dashboard, welches mehr oder weniger Funktionen hat, je nach dem zu welcher Kategorie der Besucher gehört. Beim Admin sind alle Funktionen enthalten und er kann vom Dashboard aus den Blog verwalten.

Eine zentrale Funktion ist das Veröffentlichen eines Artikels (vgl. Abb. 13). Wenn der Admin oder die Schreiber über ihr Dashboard auf den «Artikel erstellen»-Button klicken, kommen sie zur «Artikel erstellen»-Seite. Hier können sie direkt den gewünschten Titel und den Text eingeben. Sie können den Inhalt auch in einem anderen Programm schreiben und ihn anschließend einfügen. Wenn sie auf «Veröffentlichen» klicken, kommen sie auf die «Titelbild-Upload»-Seite. Nach dem Bild-Upload ist der Prozess des Artikelerstellens auch schon abgeschlossen, der Artikel wird abgespeichert in der Datenbank und ist sofort öffentlich sichtbar.

Abb. 14: Datenschutzerklärung

### **Datenschutzerklärung**

Diese Website wird verwaltet von Jeremias Bisang.

Es werden bei der Registrierung eingegebene Emailadresse, Benutzername und Passwort auf unbestimmte Zeit abgespeichert.

Die Emailadresse wird verwendet, um Kontakt aufnehmen zu können mit Benutzer. Benutzername und Passwort werden verwendet, damit Benutzer ein Konto erstellen können.

Ausserdem wird ein Cookie auf dem Browser des Benutzers abgespeichert, um ihn einzuloggen.

Auch wurden verschiedene Externe Dienste, wie Node.js Frameworks oder ein Hostinganbieter, zur Erstellung und Veröffentlichung der Website verwendet. Es ist dem Inhaber nicht bekannt, ob diese Dienste auch Benutzerdaten sammeln.

Bei Fragen, kontaktieren Sie den Inhaber über die Emailadresse:  
bisangjeremias@gmail.com

### 3.1.3.1 Datenschutzerklärung

Auf jeder Seite findet man zuunterst einen Button «Datenschutzerklärung». Dieser führt entsprechend zur Datenschutzerklärung.

Rechtlich gesehen bin ich verpflichtet, eine Datenschutzerklärung auf meiner Webseite zu haben, denn die Webseite sammelt bei der Registrierung und zum Teil auch bei der Artikelwunscheingabe Daten der Benutzer.<sup>28</sup> Laut dem Schweizer Datenschutzgesetz muss ich den Benutzer über Folgendes aufklären:

1. Die Identität des Verantwortlichen;
2. den Bearbeitungszweck;
3. eine Beschreibung der Kategorien betroffener Personen und der Kategorien bearbeiteter Personendaten;
4. die Kategorien der Empfängerinnen und Empfänger;
5. wenn möglich die Aufbewahrungsdauer der Personendaten oder die Kriterien zur Festlegung dieser Dauer;
6. wenn möglich eine allgemeine Beschreibung der Massnahmen zur Gewährleistung der Datensicherheit nach Artikel 8;
7. falls die Daten ins Ausland bekanntgegeben werden, die Angabe des Staates sowie die Garantien nach Artikel 16 Absatz 2.<sup>29</sup>

In einer eigenhändig geschriebenen Datenschutzerklärung (vgl. Abb. 14) versuchte ich diese Punkte abzudecken. Zusätzlich informierte ich über die Verwendung von Cookies und gab einen Kontakt an für Fragen bezüglich des Datenschutzes.

Etwas ungewöhnlich für meine Webseite ist, dass die Benutzerdaten momentan öffentlich zugänglich sind. Denn ich hoste den Code auf GitHub und stelle ihn da zur Ansicht zur Verfügung. Unter anderem deshalb sind einige Punkte etwas frei formuliert, beispielsweise für Punkt sieben schrieb ich einfach, dass die Benutzerdaten weltweit, öffentlich zugänglich sind.

So ist der Datenschutz und die Datenschutzerklärung definitiv noch ausbaubar, jedoch belasse ich es für diese Arbeit dabei, denn es geht hauptsächlich um den Informatikaspekt des Blogtemplates.

---

<sup>28</sup> Vgl. Art. 2. Abs. 1 DSG

<sup>29</sup> Vgl. Art. 12 Abs. 2 DSG

Abb. 15: «index.ejs»-Ausschnitt

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Blog</title>
7     <!-- Import von CSS-Dateien -->
8     <link rel="stylesheet" type="text/css" href="/general.css">
9     <link rel="stylesheet" type="text/css" href="/header.css">
10    <link rel="stylesheet" type="text/css" href="/index.css">
11  </head>
12
13  <body>
14    <!-- Header -->
15    <%- include("includes/header") %> → Header
16
17    <main>
18      <!-- Sortierknöpfe -->
19      <div class="order-div">
20        <button class="order-buttons-active" onclick="location.href='/index.ejs';">Neuste</button>
21        <button class="order-buttons" onclick="location.href='/oldest-index.ejs';">Älteste</button>
22        <button class="order-buttons" onclick="location.href='/mostpopular-index.ejs';">Beliebteste</button>
23      </div>
24
25      <!-- Artikelübersicht -->
26      <div class="articles">
27        <% articles.forEach(article => { %>
28          <div class="article-div" onclick="location.href='/article/<%= article.id %>';">
29            
30            <div class="article-text">
31              <p class="article-title"><%= article.title %></p>
32              <p class="article-description"><%= article.article.slice(0, 300) %>...</p>
33            </div>
34          </div>
35          <% }) %>
36        </div>
37      </main>
38
39      <!-- Footer -->
40      <%- include("includes/footer") %>
41    </body>
42  </html>

```

Sortierbuttons

Artikel

Abb. 16: «index.css»-Ausschnitt

```

106 .article-title {
107   margin-top: 10px;
108
109   font-size: 19px;
110   font-weight: bold;
111   color: rgb(62, 63, 41);
112 }
113
114 .article-description {
115   margin-top: -12px;
116
117   font-size: 16px;
118 }
119
120 /* Bildschirmgrösseanpassung */
121
122 @media (max-width: 600px) {
123   .articles {
124     margin-right: 20px;
125     margin-left: 20px;
126
127     grid-template-columns: 1fr;
128   }

```

## 3.2 Blick hinter die Kulissen

In diesem Abschnitt gebe ich einen Einblick, wie die Webseite überhaupt zustande kommt und funktioniert. Wie im theoretischen Teil dieser Arbeit erklärt, kann man eine Webseite in zwei Teile, in das Front- und Backend einteilen. Hier gehe ich zuerst darauf ein, wie das Front- und danach das Backend zustande kommt.

### 3.2.1 Frontend

Viel vom Frontend habe ich schon beschrieben im vorhergehenden Kapitel «Sicht der Benutzer», denn das Frontend ist derjenige Teil, mit dem der Benutzer interagiert. Hier soll es um den Blick hinter die Kulissen gehen, also wie diese Designs und Layouts zustande kommen.

Mein Frontend besteht aus einigen Dateien, hauptsächlich sind das die EJS- und CSS-Dateien. Jede EJS-Datei steht für eine Unterseite. Die Hauptseite ist die «index.ejs», das ist die Übersicht über die Artikel. Sie ist so benannt, da das der Standardname für die Hauptseite ist. Je nach Hosting ist es automatisch so eingerichtet, dass der Besucher als erstes auf die Index-Seite kommt.

Wenn wir uns den Code ansehen (vgl. Abb. 15), sehen wir zuoberst den Head-Abschnitt (blau markiert), da sind diejenigen Informationen enthalten, die nicht auf der Seite sind. Also zum Beispiel Titelsetzung für den Browser oder Verlinkung mit CSS-Dateien.

Anschliessend folgt der Body-Teil (violett markiert), hier ist der Inhalt der Seite enthalten. Zuerst wird hier der Header aus einer anderen EJS-Datei importiert. Das handhabe ich auf diese Weise, weil der HTML-Header-Code auf jeder Unterseite exakt gleich aussieht. Wenn ich also diesen Code separat aufbewahre und ihn überall nur importieren muss, macht das den Code übersichtlicher.

In der CSS-Datei (vgl. Abb. 16) werden nun sogenannte Klassen aufgegriffen (gelb markiert), diese wurden in der EJS-Datei definiert. Nun können auf diesem Wege Proportionen zu den erstellten Elementen aus der EJS-Datei hinzugefügt werden. Zusätzlich werden unten in der CSS-Datei Proportionen erneut verändert, wenn die Bildschirmbreite verändert wird (grün markiert), so wird die Mobileansicht der Webseite angepasst.

Abb. 17: Get-Request-Verarbeitung

```

269 // Kontaktseite
270 app.get("/contact.ejs", (req, res) => {
271     // Auslesen des echten Adminnamens
272     searchAdmin = db.prepare("SELECT adminVisualName FROM adminVisualName WHERE id = 1")
273     admin = searchAdmin.get()
274
275     // Auslesen der Emailadresse des Admins
276     searchAdminEmail = db.prepare("SELECT email FROM users WHERE Rowid = 1")
277     adminEmail = searchAdminEmail.get()
278
279     res.render("contact", {admin, adminEmail})
280 })

```

Abb. 18: Post-Request-Verarbeitung

```

483 app.post("/register", (req, res) => {
484     // Errors
485     const errors = []
486     // Prüfung, ob alle Felder ausgefüllt sind
487     if (req.body.email.trim() == "") {errors.push("Bitte Emailadresse eingeben."); return res.render("register", {errors});}
488     if (req.body.username.trim() == "") {errors.push("Bitte Benutzername eingeben."); return res.render("register", {errors});}
489     if (req.body.password.trim() == "") {errors.push("Bitte Passwort eingeben."); return res.render("register", {errors});}
490
491     // Sicherheitslücke schließen und potentielle Fehlermeldungen vermeiden
492     if (typeof req.body.email != "string") {req.body.email = ""; errors.push("Die Emailadresse muss Textformat haben."); return res.render("register", {errors});}
493     if (typeof req.body.username != "string") {req.body.username = ""; errors.push("Der Benutzername muss Textformat haben."); return res.render("register", {errors});}
494     if (typeof req.body.password != "string") {req.body.password = ""; errors.push("Das Passwort muss Textformat haben."); return res.render("register", {errors});}
495
496     // Versöhnliche Leerzeichen am Anfang oder Ende entfernen
497     req.body.username = req.body.username.trim()
498     req.body.email = req.body.email.trim()
499
500     // Email-, Benutzernamen und Passwortprüfung
501     if (req.body.username.length == 1 || req.body.username.length == 2) {errors.push("Der Benutzername muss mindestens 3 Zeichen lang sein."); return res.render("register", {errors});}
502     if (req.body.username.length > 20) {errors.push("Der Benutzername darf nicht länger als 20 Zeichen sein."); return res.render("register", {errors});}
503     if (!req.body.username.match(/^[a-zA-Z0-9]{1,20}$/)) {errors.push("Nur Buchstaben und Zahlen sind erlaubt im Benutzernamen."); return res.render("register", {errors});}
504     if (req.body.password.length == 1 && req.body.password.length == 2 && req.body.password.length == 3 && req.body.password.length == 4) {errors.push("Das Passwort muss mindestens 5 Zeichen lang sein."); return res.render("register", {errors});}
505
506     // Versuchen, neuen Benutzer abzuspeichern -> falls Benutzername schon existiert -> Error -> springt zu catch {}
507     try {
508         // Passwort verschlüsseln
509         const salt = bcrypt.genSaltSync(10)
510         req.body.password = bcrypt.hashSync(req.body.password, salt)
511
512         // Neuer Benutzer in Datenbank abspeichern
513         const ourStatement = db.prepare("INSERT INTO users (email, username, password) VALUES (?, ?, ?)")
514         const result = ourStatement.run(req.body.email, req.body.username, req.body.password)
515
516         // Hinzugefügter Benutzer auslesen
517         const lookupStatement = db.prepare("SELECT * FROM users WHERE ROWID = ?")
518         const ourUser = lookupStatement.get(result.lastInsertRowid)
519
520         // Server neustarten, um Datenbank zu aktualisieren
521         let server = fork("server")
522         server.on("close", (code) => {
523             console.log("Restarted")
524         });
525
526         // Benutzer einloggen -> Cookie geben
527         const ourTokenValue = jwt.sign({exp: Math.floor(Date.now() / 1000) + 60 * 60 * 24, skyColor: "blue", userid: ourUser.id, username: ourUser.username}, process.env.JWTSECRET)
528
529         res.cookie("ourSimpleApp", ourTokenValue, {
530             httpOnly: true,
531             secure: true,
532             sameSite: "strict",
533             maxAge: 1000 * 60 * 60 * 24
534         })
535
536         res.redirect("/dashboard.ejs")
537     } catch {
538         errors.push("Dieser Benutzername existiert schon."); return res.render("register", {errors});
539     }
540 })

```

### 3.2.2 Backend

In diesem Abschnitt gebe ich einen Einblick in das sogenannte Backend des Blogtemplates. Dieses verarbeitet die eingehenden Daten und schickt diese oder andere zum Benutzer zurück, oft speichert es sie auch in der Datenbank ab.

Die verwendete Programmiersprache, Datenbank und die Frameworks wurden bereits im Theorieteil erläutert. Deshalb gehe ich hier nur auf den praktischen Teil ein.

Der gesamte JavaScript-Code ist in einer einzigen Datei enthalten, in der «server.js»-Datei. Der grösste Teil des Codes sind sogenannte Get- und Post-Request-Verarbeitungen. In erster Linie möchte der Browser des Benutzers bei Get-Requests gewisse Daten erhalten und bei Post-Requests sendet er Daten, damit der Server sie verarbeiten kann. Was genau bei einem Request passiert, das ist in diesen Get- oder eben Post-Request-Verarbeitungen definiert.

#### 3.2.2.1 Get-Request

Wenn wir uns die Get-Request-Verarbeitung für die Kontaktseite (vgl. Abb. 17) ansehen, dann wird zuerst der Get-Request genannt, auf den dieser Codeschnipsel reagiert. In diesem Fall ist das «/contact.ejs» (violett markiert). In der Funktion wird anschliessend der Name und die E-Mail-Adresse des Admins aus der Datenbank herausgesucht und in zwei Variablen gespeichert (blau markiert). Zum Schluss wird die Kontaktseite «contact.ejs» geladen (gelb markiert), die «.ejs»-Endung kann dabei beim «Render»-Befehl weggelassen werden. Zusätzlich werden die beiden Variablen dem Browser zugesandt, damit dieser sie dem Benutzer anzeigen kann.

#### 3.2.2.2 Post-Request

Post-Request-Verarbeitungen sind oft deutlich länger, beispielsweise beim Post-Request, der gesendet wird, wenn sich ein Benutzer registriert und so den Post-Request «/register» sendet (vgl. Abb. 18). Es werden die eingegebenen Daten geprüft, ob sie den vorgegebenen Formen entsprechen und im Falle des Benutzernamens auch, ob dieser schon vergeben ist.

Wenn alles passt, wird der neue Benutzer in der Datenbank abgespeichert und der Server neugestartet, denn erst dann ist der neue Benutzer aus der Datenbank abrufbar. Das Neustarten dauert nur einen Bruchteil einer Sekunde und bleibt für den Benutzer unbemerkt.

Zum Schluss wird der Benutzer automatisch in seinen neuen Account eingeloggt, indem ihm ein Cookie gegeben wird. Ausserdem wird er weitergeleitet auf sein Dashboard.

Abb. 19: Zeitplan

Github kennenlernen und einrichten - **10.07.2025**  
Projektskizze anfertigen; Kurzes dokumentieren der bisherigen Programmierlernfortschritte - **15.07.2025**  
HTML/CSS auffrischen und vertiefen; HTML/CSS Teil realisieren; Dokumentieren - **05.08.2025**

- HTML/CSS Must have (s. unten) optisch fertigstellen

Javascript lernen; Javascript Teil realisieren; Dokumentieren - **30.08.2025**

- JavaScript Must have (s. unten) fertig implementieren

Alles restliche lernen; Alles restliche realisieren; zusätzliche Funktionen - **15.09.2025**

- Restliche Must have Funktionen (s. unten) fertigstellen
- Je nach Zeit noch mehr Features implementieren

Web App fertigstellen; Dokumentieren - **01.10.2025**

- Testen
- Verbesserungen

Reflexionszeit - **03.10.2025**

- Was lief gut?
- Was war herausfordernd?
- Was hätte ich anders machen sollen?
- Was würde ich nächstes Mal genau gleich machen?

Konzept der schriftlichen Maturaarbeit - **05.10.2025**  
Maturaarbeit schreiben - **20.10.2025**  
Maturaarbeit korrigieren - **25.10.2025**  
Maturaarbeit testlesen lassen (Mama -> Deutschlehrerin) - **10.11.2025**  
Web App veröffentlichen; Maturaarbeit fertigstellen - **15.11.2025**  
-> Maturaarbeit drucken lassen

**Must have**  
*HTML/CSS*

- Startseite optisch: Übersicht über alle Artikel
- Creator-Seite optisch: Nur für mich zugänglich
- Artikelansichtsseite optisch
- Mobileansicht

*Javascript*

- Suchfunktion (Artikel, Wörter, Tags)
- Sortierung der Posts
- Zähler, der anzeigt, wie viele Posts vorhanden sind

*Restliches*

- Creator-Seite passwortschützen
- Noch nicht veröffentlichte Artikel mit JSON-Datei auf Server speichern
- Markdownformatierung
- Upload von Markdown-Dateien

**Optional**

- Loginfunktion
- Likefunktion
- Feedbackfunktion (automatisches Email)
- Seite für Artikelwünsche
- Designwechsel (Light-/Darkmode)
- Teilenfunktion
- Textanalysetool auf Creatorseite

## **4 Projektverlauf**

In diesem Kapitel geht es um den Entwicklungsprozess, wie ich dieses Projekt anging und wie ich es anschliessend durchführte. Im nächsten Kapitel werde ich auf das Projekt zurückblicken, evaluieren und reflektieren.

### **4.1 Ideenfindung**

Im ersten Schritt musste ich mich für ein Thema entscheiden. Damals überlegte ich mir, nach dem Gymnasium Informatik zu studieren, deshalb entschied ich mich für ein Thema in diesem Bereich. Ich dachte nach, hatte verschiedene Ideen und nach einer gewissen Zeit entschied ich mich für die Idee mit dem Blog. Die anfängliche Idee sah noch etwas anders aus, sie formte sich während des gesamten Prozesses zum jetzigen Ergebnis.

### **4.2 Planung**

Ich plante nicht so konkret. Das Einzige, was ich wirklich konkret machte, war der Zeitplan (vgl. Abb. 19), der sich jedoch im Verlauf ziemlich stark veränderte, weil ich zu Beginn nur teilweise wusste, welche Schritte es wirklich brauchen wird. Ansonsten schaute ich mir einige Blog-Webseiten an, um Inspirationen zu finden für Designs und Layouts. Ausserdem zeichnete ich grob ein Layout, wie der Blog aussehen könnte.

Ich dachte jedoch während des Prozesses immer wieder über die Planung nach und plante oft laufend in meinem Kopf, wie ich die nächsten Schritte angehe und setzte mir Zeitziele.

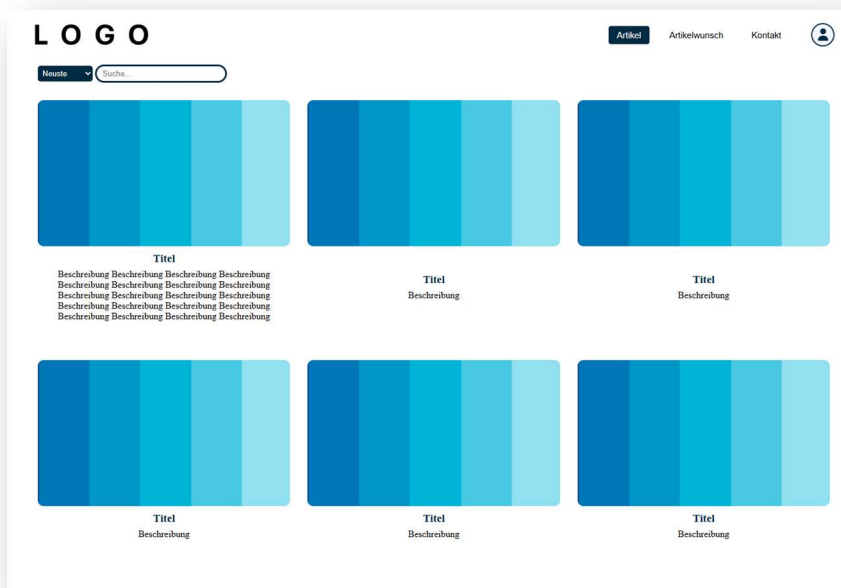
Abb. 20: HTML- und CSS-Zusammenfassung

```

HTML
<html>: root element; --> lang="en" (language English)
<head>: children store metadata
  <meta>: void element; stores metadata not expressable in title, ...; --> charset="utf-8" (character encoding utf-8);
    name="viewport" (hints on the initial size of the viewport for mobile devices);
    content="width=device-width, initial-scale=1.0" (contains value for another attribute)
  <title>: metatitle of website
  <link>: links other files into current file
<body>: children store content of website
  <main>: main content of body element
    <div>: no meaning just a box; --> class="..." (naming for CSS/JS)
    <section>: somewhere inside body element; thematic grouping of content; should be defined (for ex. by h1-6 element)
    <article>: complete composition (for ex. forum post)
    <figure>: children store flow content
      <h1-6>: section heading from big to small;
      <p>: paragraph
      <hr>: void element; thematic break
      <a>: hyperlink; --> href="https..." (link); target="_blank" (opens link in new tab)
      <img>: void element; picture; --> src="https..." (defines image); alt="..." (alternative image description)
      <ul>: creates list (order not important)
        <li>: creates list item
      <figcaption>: caption or legend
      <ol>: creates list (order important)
      <strong>: signalises importance of content
      <form>: creates form; --> action="https..." (URL of programm which processes info)
      <fieldset>: children store form options
      <legend>: caption for the rest of the elements
      <label>: caption in userinterface
      <input>: data field for the user to fill in; --> id="..." (identifier); type="radio" (checkmark);
        name="..." (name of the form); value="..." (value of the specific input field); checked (prechecked)
      <button>: creates button; --> type="submit" (submits to server)
  <footer>: at the very bottom of the content

CSS
background-image
background-color
font-family
font-size
font-style
padding(-top, etc)
margin(-top, etc)
width
max-width
text-align
display
border-color
color
  
```

Abb. 21: Erstes Layout und Design



### 4.3 Umsetzung

So startete ich bald mit der Umsetzung und fing an mit dem Frontend, also mit HTML und CSS. Während den Sommerferien repetierte ich meine bereits vor dem Projekt erlernten HTML- und CSS-Fähigkeiten indem ich eine kleine Zusammenfassung (vgl. Abb. 20) erstellte aus dem Gedächtnis und mithilfe der Erklärungen aus der «Visual Studio Code»-App<sup>30</sup>. Anschliessend erstellte ich das Frontend der Webseite beziehungsweise von diesen Teilen, von denen ich zu diesem Zeitpunkt schon wusste, dass ich sie sicher einbauen werde. Ich hatte viele optionale Funktionen auf meinem Plan, da ich nicht einschätzen konnte, was wie viel Zeit in Anspruch nehmen wird. So legte ich den Grundbaustein für das Frontend (vgl. Abb. 21), gewisse Aspekte änderte ich später noch wie die Farbpalette. Anschliessend kam auf meinem Plan der nächste Schritt: JavaScript lernen. Ich fing damit an und merkte schnell, dass es keine schnelle Aufgabe ist, eine Programmiersprache wirklich zu lernen, einige Grundlagen sind jedoch trotzdem nicht allzu zeitaufwendig. Auch fiel mir auf, dass ich gar nicht so genau wusste, welche Aspekte von JavaScript ich brauchte und ob ich ein JavaScript-Tutorial benötigte, das in etwa das zeigt, was ich effektiv benötigte. Sonst würde ich nur viel Zeit damit verbringen, diese Programmiersprache zu lernen, ohne am Ende zu wissen, was ich nun genau tun muss.

Lange Rede, kurzer Sinn: Schlussendlich bemerkte ich, dass ich so gut wie kein JavaScript in meinem Frontend benötigte, für das ich eigentlich einiges an Zeit eingeplant hatte. Dafür würde das Backend den allergrössten Teil der Zeit beanspruchen und auch dieses wollte ich in JavaScript programmieren, also musste ich nach Backend-JavaScript-Tutorials suchen. So machte ich mich auf die Suche nach diesen und schaute in verschiedene hinein, die mir hilfreich erschienen. Doch plötzlich stiess ich auf ein Video, das zu einem Teil genau zeigte, was ich vorhatte. Also begann ich, dieses dreistündige Video zu schauen und programmierte den Code nebenbei ab, um ihn mir besser einprägen zu können. Mehr und mehr verstand ich, wie man so ein Backend programmieren kann, wie man eine Datenbank erstellt, wie man Daten darin abspeichert oder abruft und vieles mehr. Nach ca. einer Stunde des Tutorials kam ich auf die Idee, den Code nicht mehr abzuschreiben, sondern direkt so in meine eigene Webseite zu implementieren, wie ich ihn brauchte. Vieles konnte ich eins zu eins übernehmen, einiges musste ich etwas abändern. Dazu war ich in der Lage, da ich schon genug Wissen gesammelt hatte in dieser einen Stunde des Tutorials. So schritt ich voran und war nach deutlich mehr Zeit als die Laufzeit des Tutorials beträgt damit durch, denn die meiste Zeit hatte ich das Video auf Pause und versuchte den Code zu implementieren. Manchmal ging es ganz rasch, manchmal tauchte auch ein Error auf, bei dem ich je nachdem viel oder gar nichts verstand, was er mir zu sagen versuchte. Nichtsdestotrotz fand ich eine Lösung für diese Errors und lernte so noch viel mehr den Umgang mit meinem nun wachsenden Backend. Der Prozess begann richtig

---

<sup>30</sup> Vgl. Microsoft Corporation, «The open source AI code editor», Internet



Spass zu machen, da ich mein Projekt fortschreiten sah und bemerkte, dass ich immer mehr auch selbst wusste, was zu tun war.

Nach Abschluss des Videos fing das richtige Programmieren erst an, denn es fehlten noch viele Funktionen, die ich beabsichtigte einzubauen und nun war ich auf mich selbst gestellt. Ich fing an, mein Wissen ganz praktisch anzuwenden und konnte so viele der noch geplanten Funktionen realisieren. Es war nicht immer die eleganteste Lösung, da mir die praktische Erfahrung mit JavaScript schlichtweg fehlte, doch für alles fand ich auf die eine oder andere Art eine Lösung. Einen grossen Teil der Zeit investierte ich in das Lösen von Errors, gerade wenn ich eine Funktion in Angriff nahm, für die ich ein neues Framework implementierte.

Es kamen mir immer mehr Funktions- und Erweiterungsideen, doch gegen Ende des Projekts musste ich eine Grenze setzen, da es den Rahmen der Maturaarbeit gesprengt hätte und die Zeit dazu fehlte.

## **4.4 Testung und Abschliessung**

Zwei Wochen vor Abgabe schloss ich die Webseite vorerst ab, auch wenn es noch einiges zu tun gab. Ausserdem tauchten mit dem Hosting zuerst einige Schwierigkeiten auf, da das Hochladen auf GitHub nicht funktionierte. So schrieb ich erst einmal an meiner schriftlichen Arbeit.

In der Zwischenzeit schaute sich mein Vater das Blogtemplate an und gab mir eine ganze Liste von Verbesserungsmöglichkeiten und Ergänzungen, die ich noch angehen könnte.

Anschliessend setzte ich mich nochmals an mein Blogtemplate, fügte wenige kleine Funktionen hinzu, korrigierte den Code bei Error-Meldungen und verbesserte das Design vor allem in der Mobileansicht. Auf diese Weise konnte ich einen, jedoch eher kleinen Teil der Verbesserungsmöglichkeiten umsetzen, denn irgendwann musste ich einen Schlussstrich ziehen, da Stunden vorbeiziehen beim Einbau neuer Funktionen. Zuallerletzt kamen noch das Auskommentieren und Sortieren des Codes, um ihn verständlicher zu machen. Schlussendlich löste ich das Hosting-Problem und das Hochladen auf GitHub funktionierte, anschliessend verband ich mein Repository mit Render<sup>31</sup>, wo die Webseite nun sogar kostenlos gehostet wird.

---

<sup>31</sup> Vgl. Osano International Compliance Services Limited, «Render», Internet



## 5 Rückblick

In diesem Kapitel blicke ich auf das Werk meiner Maturaarbeit zurück. Ich betrachte, wie das Endprodukt nun herausgekommen ist, was die Limitierungen sind und was ich noch einbauen hätte können. Anschliessend reflektiere ich, was gut lief, was ich besser hätte ausarbeiten können und welche der anfänglichen Ziele ich erreicht habe.

### 5.1 Evaluation

Das Endprodukt ist ein vollständig funktionstüchtiges Blogtemplate mit einigen Funktionen, die man nicht bei jedem Blog findet. Mit einem Blick auf die anfänglichen Ziele können wir sehen, dass viele dieser erreicht wurden, jedoch nicht alle. Einige Ziele habe ich in einer etwas anderen Form erfüllt, zum Beispiel das Ziel «Creator-Seite passwortschützen». Ich habe es dadurch gelöst, dass nur der Admin und die Schreiber auf die «Artikel erstellen»-Seite Zugriff haben. Ausserdem habe ich viele andere Funktionen eingebaut, die gar nicht auf der anfänglichen Liste standen, da mir die Idee dazu erst während des Prozesses gekommen ist.

Einige der nicht erreichten Ziele sind nicht wirklich relevant, beispielsweise der Zähler, der anzeigt, wie viele Posts vorhanden sind. Doch es gibt auch einige Funktionen, die das Blogtemplate nochmals aufwerten würden, wie die Suchfunktion oder das Speichern von noch nicht veröffentlichten Artikeln.

Ein weiteres Problem, das sich im Prozess ergab, sind die Sicherheitslücken. Ich habe nicht das Knowhow, eine vor Angriffen sichere Webseite zu programmieren, das würde den Rahmen der Maturaarbeit sprengen. Schön wäre die Funktion gewesen, dass der Admin in regelmässigen Zeitabständen ein Backup automatisch per E-Mail bekommt. Für das reichte jedoch die Zeit nicht mehr am Ende und ich löste das Problem noch einfacher mit einer Sicherheitsinformation, in welcher der Admin erfährt, wie er Backups machen kann und sie aktivieren könnte im Falle eines Angriffs.

### 5.2 Reflexion

Grundsätzlich bin ich zufrieden mit meiner Maturaarbeit und kann zurückblickend sagen, dass meine Herangehensweise funktionierte. Mit diesem Tutorial war ich extrem zufrieden, es hat mir sehr geholfen, genau das zu lernen, was ich auch brauchte. Zusammen mit dem «Learning-by-doing» brachte es mich ans Ziel und ich habe ein Endprodukt, welches als Blog verwendbar ist mit einigen Funktionen, die mir echt gefallen. Klar, es hat Verbesserungspotenzial, doch mir wurde während dem Programmieren relativ schnell bewusst, dass ich irgendwo eine Grenze ziehen musste. Wenn ich den Blog wirklich verwenden möchte, kann ich auch nach der Maturaarbeit Anpassungen vornehmen. Ich gehe davon aus, dass es nicht mein letztes



Informatik-Projekt war. Ich hätte jedoch konkreter planen können, das hätte mir mehr Struktur gegeben, Stress herausgenommen und ich hätte mir die Arbeit besser einteilen können. Ich arbeitete recht konzentriert, im Sinne von nicht verteilt, sondern viel auf einen Schlag. Es gab zwei intensive Arbeitsphasen: In den Herbstferien arbeitete ich viel am Werk und vor der Abgabe war ich mit dem schriftlichen Teil der Maturaarbeit beschäftigt.

Ich kann gut intensiv arbeiten, trotzdem ist es angenehmer, die Aufgaben besser einzuteilen. Es nimmt Stress heraus und ist angenehmer, da ich die Zeit regelmässiger einteilen kann.

### **5.3 Fazit**

Anstelle gedanklicher Pläne würde ich häufiger schriftliche Pläne erstellen. Ausserdem würde ich mit der Arbeitsphase früher beginnen, denn ich habe relativ kurzfristig mit der schriftlichen Arbeit gestartet und unterschätzte sie etwas. Jedoch ist es schwierig, richtig mit dem Schreiben zu beginnen, ohne schon grosse Teile der Webseite programmiert zu haben.

Dafür würde ich die Vorgehensweise so beibehalten. Ich brauchte dieses zu meinem Werk passende Tutorial, um mir das benötigte Wissen anzueignen. Ich war erstaunt darüber, wie viel mit diesem Wissen möglich ist.

### **5.4 Dank**

Ich danke herzlich meinem Betreuer Kaspar Jost für die Unterstützung während des Projekts, insbesondere den sehr ausführlichen Antworten auf meine Fragen. Auch danke ich meinem Vater Matthias Bisang für die Verbesserungs- und Erweiterungsvorschläge in Bezug auf mein Werk. Bei meiner Mutter Cristina Bisang bedanke ich mich für das zweifache Korrekturlesen meiner schriftlichen Arbeit. Auch meiner Freundin Salome Jenni danke ich für das Korrekturlesen der schriftlichen Arbeit, ausserdem für ihre Unterstützung bei der Ideenfindung zu Beginn des Projekts. Zum Schluss danke ich Andrin Blass für das Bereitstellen seiner Maturaarbeit in der Mediothek der Kantonsschule Büelrain. Sie diente mir etwas als Orientierung, gerade durch das Layout seiner schriftlichen Arbeit liess ich mich inspirieren.<sup>32</sup>

### **5.5 Information zur Verwendung von künstlicher Intelligenz**

Ich habe mich aus ethischen Gründen bewusst gegen die Verwendung von künstlicher Intelligenz entschieden bei der Erstellung dieser Arbeit, sowohl beim Werk als auch bei der schriftlichen Arbeit. Ich fand heraus, dass alle grossen KI-Unternehmen ihre Modelle mit Hilfe von unterbezahlten Arbeitskräften, zum Teil auch noch unter schlechten Arbeitsbedingungen, trainieren. Gerade was das Herausfiltern von verstörenden Inhalten anbelangt, wird nicht viel Rücksicht genommen auf diese Menschen. Das bedeutet, dass ich alle Error-Meldungen selbst

---

<sup>32</sup> Vgl. Blass, «Timely – Entwicklung eines Stundenplantools», I + 8-9 + 16 + 37 + III + IV



oder mit Hilfe des Internets, beispielsweise [stackoverflow.com](https://stackoverflow.com), löste. Auch wurde die schriftliche Arbeit nicht mit KI korrigiert, sondern von Hand von meiner Mutter und mir selbst



## 6 Abbildungsverzeichnis

Quellen: Titelbild-Hintergrund, Titelbild-Vordergrund, Abb. 1-9, 11-26: Eigene Darstellung

Quelle: Titelbild-QR-Code: Erstellt mit foundata GmbH; <https://goqr.me/de/>

Quelle: Abb. 10: Shir; <https://colorhunt.co/palette/3e3f297d8d86bca88df1f0e4>

Quelle: Abb. 27: Fonticons, Inc.; <https://fontawesome.com/icons/user?f=classic&s=solid&pc=%233e3f29&sc=%233e3f29>

Titelbild-Hintergrund: HTML-Code

Titelbild-Vordergrund: Screenshot Blogtemplate

Titelbild-QR-Code: Blogtemplate-Link

Abb. 1 : Kategorie von Benutzern

Abb. 2 : Internet

Abb. 3 : Domain

Abb. 4 : IP-Adresse

Abb. 5 : HTML-Beispiel-Code

Abb. 6 : CSS-Beispiel-Code

Abb. 7 : Beispiel-Webseite

Abb. 8 : Serveraufbau

Abb. 9 : Kategorien mit Rechten

Abb. 10 : Farbpalette

Abb. 11 : Header- und Artikelraster

Abb. 12 : Admin-Dashboard

Abb. 13 : Artikel erstellen

Abb. 14 : Datenschutzerklärung

Abb. 15 : «index.ejs»-Ausschnitt

Abb. 16 : «index.css»-Ausschnitt

Abb. 17 : Get-Request-Verarbeitung

Abb. 18 : Post-Request-Verarbeitung

Abb. 19 : Zeitplan

Abb. 20 : HTML- und CSS-Zusammenfassung

Abb. 21 : Erstes Layout und Design

Abb. 22 : Ordnersystem

Abb. 23 : Grünes Platzhalter-Titelbild

Abb. 24 : Blaues Platzhalter-Titelbild

Abb. 25 : Braunes Platzhalter-Titelbild

Abb. 26 : Platzhalter-Logo

Abb. 27 : User-Icon



## 7 Quellenverzeichnis

Akamai DevRel, *Uploading Files to the Web with HTML* | *Learn Web Dev with Austin Gil*, YouTube, 2023. Zugriffen 29. November 2025. [https://youtu.be/s2TTck1sj4s?si=pkB\\_ZQYPB3jaKKZ-](https://youtu.be/s2TTck1sj4s?si=pkB_ZQYPB3jaKKZ-)

alexgilbert et al.; «sanitize-html». Juni 2025. Zugriffen 23. November 2025, <https://www.npmjs.com/package/sanitize-html>

Aschermann, Tim, und Ohl, Lisa; «Was ist ein Server? Definition und Funktion einfach zusammengefasst». 27. September 2025. Zugriffen 24. November 2025, [https://praxistipps.chip.de/was-ist-ein-server-definition-und-funktion-einfach-zusammengefasst\\_12282](https://praxistipps.chip.de/was-ist-ein-server-definition-und-funktion-einfach-zusammengefasst_12282)

Blass, Andrin. «Timely – Entwicklung eines Stundenplantools», 2. Dezember 2024, I + 8-9 + 16 + 37 + III + IV

Business Systemhaus AG; «Was ist ein Framework?». Zugriffen 22. November 2025, <https://bsh-ag.de/it-wissensdatenbank/framework/>

charlesrea et al.; «jsonwebtoken». 2023. Zugriffen 22. November 2025, <https://www.npmjs.com/package/jsonwebtoken>

chjj et al.; «Marked». 14. November 2025. Zugriffen 22. November 2025, <https://www.npmjs.com/package/marked>

Cloudflare Germany GmbH; «Wie funktioniert das Internet?». Zugriffen 20. November 2025, <https://www.cloudflare.com/de-de/learning/network-layer/how-does-the-internet-work/>

CoR und Paul Roub; «JavaScript or jQuery "Are you sure?" dialog for <A> link?». 5. Januar 2016. Zugriffen 29. November 2025, <https://stackoverflow.com/questions/12531324/javascript-or-jquery-are-you-sure-dialog-for-a-link>

ctcpip et al.; «Multer». August 2025. Zugriffen 22. November 2025, <https://www.npmjs.com/package/multer>

Ikechukwu, Linda; «Using EJS as a Template Engine in your Express App». 8. November 2018. Zugriffen 23. November 2025, [https://medium.com/@Linda\\_Ikechukwu/https-](https://medium.com/@Linda_Ikechukwu/https-)



[medium-com-linda-ikechukwu-using-ejs-as-a-template-engine-in-your-express-app-cb3d82c15e17](https://medium.com/@linda_ikechukwu/using-ejs-as-a-template-engine-in-your-express-app-cb3d82c15e17)

James Q Quick, *How to Upload Files in Node.js Using Express and Multer*, YouTube, 2023. Zugegriffen 29. November 2025. <https://youtu.be/i8yxx6V9UdM?si=kTut3i6LKsh8NiAj>

jfirebaugh et al.; «node.bcrypt.js». Mai 2025. Zugegriffen 22. November 2025, <https://www.npmjs.com/package/bcrypt>

joshuawise; «better-sqlite3». 15. November 2025. Zugegriffen 22. November 2025, <https://www.npmjs.com/package/better-sqlite3>

LearnWebCode, *Back-End Web Development (Tutorial for Beginners)*, YouTube, 2024. Zugegriffen 11. Oktober 2025. <https://youtu.be/1oTuMPIwHmk?si=IioqHcYHHiAI6ESe>

Microsoft Corporation; «The open source AI code editor ». Zugegriffen 29. November 2025, <https://code.visualstudio.com/>

OpenJS Foundation ; «Express - Node.js Web Application Framework». Zugegriffen 22. November 2025, <https://expressjs.com/>

Osano International Compliance Services Limited; «Render». Zugegriffen 26. November 2025, <https://render.com>

P., Vera; «Was ist eine Website? Wie Websites funktionieren und vieles mehr». 14. Mai 2025. Zugegriffen 20. November 2025, <https://www.hostinger.com/de/tutorials/was-ist-eine-web-site>

remy; «nodemon». 11. November 2025. Zugegriffen 22. November 2025, <https://www.npmjs.com/package/nodemon>

The Daily Grace Co.; «The Daily Grace Blog». Zugegriffen 5. August 2025, <https://thedailygraceco.com/blogs/the-daily-grace-blog>

thinktt; «Node.js: What's a good way to automatically restart a node server that's not responding?». 8. Februar 2016. Zugegriffen 29. November 2025, <https://stackoverflow.com/questions/35185522/node-js-whats-a-good-way-to-automatically-restart-a-node-server-thats-not-res>

Tischlinger, David; «CMS-Revolution: Vom Backend zum Frontend». 20. Januar 2023. Zugegriffen 20. November 2025, <https://blog.hubspot.de/website/frontend-backend>



ulisesgascon, dougwilson und defunctzombie; «cookie-parser». 2024. Zugriffen 22. November 2025, <https://www.npmjs.com/package/cookie-parser>

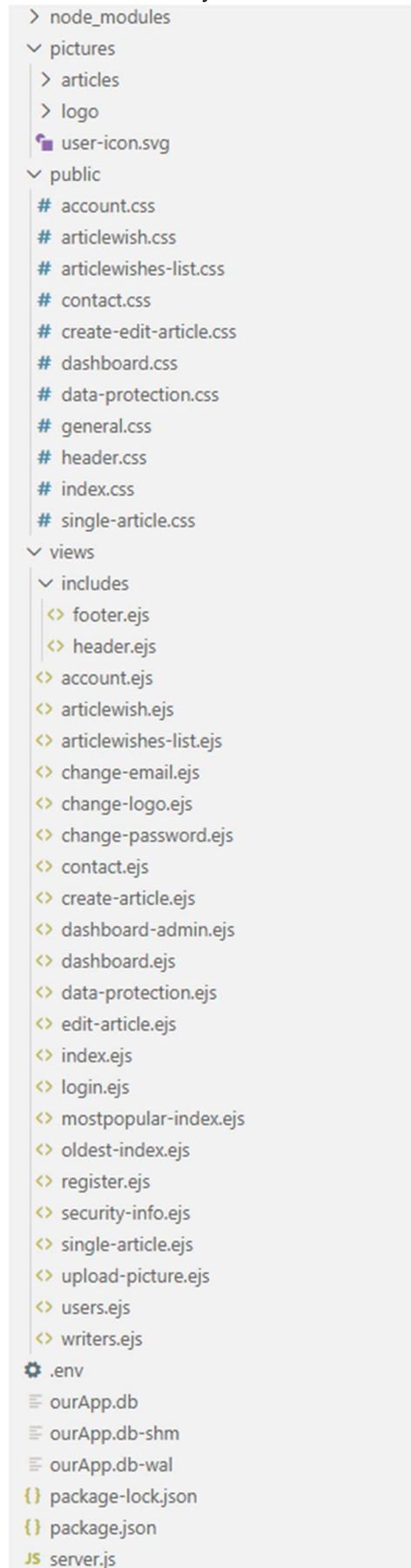
Valanagut et al.; «Node.js». 5. November 2025. Zugriffen 23. November 2025, <https://de.wikipedia.org/w/index.php?title=Node.js&oldid=261271161>

Wikifh et al.; «IP-Adresse». 3. November 2025. Zugriffen 20. November 2025, <https://de.wikipedia.org/w/index.php?title=IP-Adresse&oldid=261198662>

Wikipedia; «Express.js». 8. Oktober 2025. Zugriffen 23. November 2025, <https://en.wikipedia.org/w/index.php?title=Express.js&oldid=1315811220>

~jcblw et al.; «dotenv». September 2025. Zugriffen 22. November 2025, <https://www.npmjs.com/package/dotenv>

Abb. 22: Ordnersystem



Im hier zugeklappten «/articles/»-Ordner befinden sich die Titelbilder der Artikel.

Abb. 23: Grünes Platzhalter-Titelbild



Abb. 24: Blaues Platzhalter-Titelbild



Abb. 25: Braunes Platzhalter-Titelbild



Im «/logo/»-Ordner befindet sich das Logo, auch hier ist es ein Platzhalter, der ersetzt wird, sobald der Admin ein eigenes Logo hochlädt.

Abb. 26: Platzhalter-Logo



Abb. 27: User-Icon



## 8 Anhang

Im Anhang ist der Code meiner Webseite zu sehen. Ich ordne ihn entsprechend meinem Ordnersystem, das heisst alphabetisch sortiert. In den obersten Ordner «node\_modules», da befinden sich die heruntergeladenen Frameworks, und in die Datei package-lock.json gebe ich keinen Einblick, da diese vollkommen automatisch erstellt und befüllt worden sind.

Auch bei der Programmierung habe ich Quellen verwendet. Gerade vom Backend-Tutorial, das mir so half, konnte ich vieles übernehmen. Aber vereinzelt beneutzte ich auch andere Quellen, um Probleme zu lösen oder Funktionen zu ermöglichen. Um diese zu deklarieren, markierte ich den Code in Farbe, Kommentare wären umständlich und nicht übersichtlich.

Backend-Tutorial<sup>33</sup>

Tutorials zur Bildupload-Funktion<sup>34</sup>

Codeschnipsel zum automatischen Server-Restart<sup>35</sup>

Bestätigungs-Popup beim Löschen eines Artikels<sup>36</sup>

### 8.1 CSS-Dateien

#### 8.1.1 account.css

```
/* Header */

.article-button,
.articlewish-button,
.contact-button {
  background-color: rgb(241, 240, 228);

  color: black;
}

/* Seite herunterschieben */

body {
  padding-top: 50px;
}

/* Login/Registrieren Buttons */

.account-button-div {
  text-align: center;
```

---

<sup>33</sup> Vgl. LearnWebCode, *Back-End Web Development (Tutorial for Beginners)*, 00:00:00-02:25:50

<sup>34</sup> Vgl. Akamai DevRel, *Uploading Files to the Web with HTML | Learn Web Dev with Austin Gil*, 00:00-04:54; James Q Quick, *How to Upload Files in Node.js Using Express and Multer*, 00:00-06:51

<sup>35</sup> Vgl. thinktt, «Node.js: What's a good way to automatically restart a node server that's not responding? », Internet

<sup>36</sup> Vgl. CoR und Paul Roub, «JavaScript or jQuery "Are you sure?" dialog for <A> link? », Internet

```

}

.account-buttons {
  padding: 7px 10px;

  background-color: rgb(62, 63, 41);

  align-self: center;

  font-size: 15px;
  color: white;

  border: none;
  border-radius: 8px;

  cursor: pointer;
  transition: opacity 0.2s;
}
.account-buttons:hover {
  opacity: 0.8;
}
.account-buttons:active {
  opacity: 0.6;
}

/* Registrieren/Login Seiten */

.register-div,
.login-div {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.login-form,
.register-form {
  height: 150px;
  width: 500px;
  padding-top: 30px;
  padding-bottom: 30px;
  padding-left: -20px;
  padding-right: -20px;

  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-between;

  border: solid;
  border-width: 3px;
  border-radius: 20px;
}

.register-form {
  height: 200px;

```

```

}

.account-input {
  width: 300px;
  padding: 5px 2px;
  margin-top: 15px;

  font-size: 17px;

  border: solid;
  border-width: 3px;
  border-color: rgb(62, 63, 41);
  border-radius: 7px;
}

.accept-data-protection {
  margin-bottom: 0;

  text-align: center;
  font-size: 11px;
}

.data-protection-link {
  color: black;
}

.data-protection-link:hover {
  opacity: 0.8;
}

.data-protection-link:active {
  opacity: 0.6;
}

.send-button {
  padding: 8px 12px;
  margin-top: 15px;
  margin-bottom: -6px;

  background-color: rgb(62, 63, 41);

  font-size: 17px;
  color: white;

  border: none;
  border-radius: 50px;

  transition: opacity 0.2s;
}

.send-button:hover {
  opacity: 0.7;
}

.login-register-legend {
  margin-top: -47px;

  font-size: 21px;
  font-weight: bold;
}

```

```

    background-color: rgb(241, 240, 228);
}

/* Errormeldung */

.notice {
    margin-top: -10px;
    padding: 5px 5px;

    background-color: rgba(188, 168, 141, 0.3);

    text-align: center;

    border-radius: 6px;
}

/* Bildschirmanpassung */

@media (max-width: 370px) {
    .account-buttons {
        margin-top: 10px;
    }
}

@media (max-width: 535px) {
    .login-form,
    .register-form {
        width: 400px;
    }
}

@media (max-width: 425px) {
    .login-form,
    .register-form {
        width: 340px;
    }
}

```

## 8.1.2 articlewish.css

```

/* Header */

.articlewish-button {
    background-color: rgb(125, 141, 134);

    color: white;
}

.article-button,
.contact-button {
    background-color: rgb(241, 240, 228);

    color: black;
}

```

```

/* Artikelwunsch Seite */

.articlewish-div {
    margin-top: 200px;

    display: flex;
    flex-direction: column;
    align-items: center;
}

.articlewish-form {
    height: 340px;
    width: 500px;

    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-between;
}

.email-input {
    width: 300px;

    font-size: 15px;

    border: solid;
    border-width: 3px;
    border-color: rgb(62, 63, 41);
    border-radius: 7px;
}

.articlewish-input {
    width: 300px;
    padding-bottom: 190px;

    font-size: 15px;

    border: solid;
    border-width: 3px;
    border-color: rgb(62, 63, 41);
    border-radius: 7px;
}

.accept-data-protection {
    text-align: center;
    font-size: 11px;
}

.data-protection-link {
    color: black;
}

.data-protection-link:hover {
    opacity: 0.8;
}

.data-protection-link:active {

```

```

    opacity: 0.6;
}

.send-button {
    padding: 5px 10px;

    background-color: rgb(62, 63, 41);

    font-size: 15px;
    color: white;

    border: none;
    border-radius: 50px;

    transition: opacity 0.2s;
}

.send-button:hover {
    opacity: 0.7;
}

/* Errormeldung */

.notice {
    width: 300px;
    margin-top: -10px;
    padding: 5px 5px;

    background-color: rgba(175, 20, 0, 0.1);

    text-align: center;

    border-radius: 6px;
}

/* Bildschirmgrösseanpassung */

@media (max-width: 600px) {
    .articlewish-div {
        margin-top: -30px;
    }

    .articlewish-form {
        width: 300px;
        height: 350px;
    }
}

```

### 8.1.3 articlewishes-list.css

```

/* Header */

.article-button,
.articlewish-button,
.contact-button {

```

```

    background-color: rgb(241, 240, 228);

    color: black;
}

/* Artikelwunschliste Seite */

.articlewishes-list-div {
    text-align: center;
}

.articlewishes-list-email,
.articlewishes-list-articlewish {
    font-size: 17px;
}

.articlewishes-list-email {
    color: rgb(62, 63, 41);
    font-weight: bold;
}

.articlewishes-list-articlewish {
    margin-bottom: 40px;
    margin-top: 0;
}

/* Bildschirmgrösseanpassung */

@media (max-width: 720px) {
    body {
        padding-top: 0;
        margin-top: 110px;
    }
}

```

### 8.1.4 contact.css

```

/* Header */

.contact-button {
    background-color: rgb(125, 141, 134);

    color: white;
}

.article-button,
.articlewish-button {
    background-color: rgb(241, 240, 228);

    color: black;
}

/* Kontakt Seite */

.contact-div {

```

```

    margin-top: 160px;

    text-align: center;
}

.contact-picture {
    width: 200px;
    height: 200px;

    border-radius: 10px;
}

.contact-name {
    font-size: 20px;
    color: rgb(62, 63, 41);
    font-weight: bold;
}

.contact-email {
    font-size: 17px;
    color: rgb(62, 63, 41);
}

.maturaarbeit-link {
    color: black;
}
.maturaarbeit-link:hover {
    opacity: 0.8;
}
.maturaarbeit-link:active {
    opacity: 0.6;
}

/* Bildschirmgrösseanpassung */

@media (max-width: 600px) {
    .contact-div {
        margin-top: -50px;
    }
}

```

### 8.1.5 create-edit-article.css

```

/* Header */

.article-button,
.contact-button,
.articlewish-button {
    background-color: rgb(241, 240, 228);

    color: black;
}

/* Artikel erstellen/bearbeiten Seite */

```

```
.create-article-div {
  margin-top: 200px;

  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
}

.create-article-form {
  height: 550px;
  width: 500px;

  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
}

.title-input {
  width: 363px;

  font-size: 15px;

  border: solid;
  border-width: 3px;
  border-color: rgb(62, 63, 41);
  border-radius: 7px;
}

.article-input {
  width: 600px;
  height: 350px;
  padding-bottom: 100px;

  font-size: 15px;

  border: solid;
  border-width: 3px;
  border-color: rgb(62, 63, 41);
  border-radius: 7px;
}

.create-article-button {
  padding: 5px 10px;

  background-color: rgb(62, 63, 41);

  font-size: 15px;
  color: white;

  border: none;
  border-radius: 50px;

  transition: opacity 0.2s;
}
```

```

.create-article-button:hover {
  opacity: 0.8;
}
.create-article-button:active {
  opacity: 0.6;
}

/* Bildhochladeseite */

.picture-input {
  width: 220px;
  padding: 1px;

  font-size: 13px;

  border: solid;
  border-width: 3px;
  border-color: rgb(62, 63, 41);
  border-radius: 7px;
}

.picture-upload-legend {
  margin-bottom: 10px;

  font-size: 18px;
  font-weight: bold;
}

/* Errormeldung */

.notice {
  width: 360px;
  padding: 5px 5px;
  margin-top: -10px;

  background-color: rgba(188, 168, 141, 0.3);

  text-align: center;

  border-radius: 6px;
}

/* Bildschirmgrösseanpassung */

@media (max-width: 650px) {
  .article-input {
    width: 500px;
  }
}

@media (max-width: 550px) {
  .create-article-div {
    margin-top: -90px;
  }

  .article-input {

```

k

```

        width: 400px;
    }
}

@media (max-width: 450px) {
    .create-article-div {
        margin-top: -90px;
    }

    .title-input {
        width: 300px;
    }

    .article-input {
        width: 300px;
    }
}

@media (max-width: 400px) {
    .create-article-div {
        margin-top: -100px;
    }

    .picture-input {
        width: 170px;
    }
}

```

### 8.1.6 dashboard.css

```

/* Header */

.article-button,
.articlewish-button,
.contact-button {
    background-color: rgb(241, 240, 228);

    color: black;
}

/* Dashboard Seite */

.dashboard-div {
    text-align: center;
}

.dashboard-title,
.writerlist-title {
    text-align: center;

    font-size: 25px;
    color: rgb(62, 63, 41);
    font-weight: bold;
    text-decoration: underline;
}

```

```

.dashboard-email {
  margin-top: -20px;
  margin-bottom: 20px;

  text-align: center;

  font-size: 15px;
  color: black;
}

.dashboard-buttons {
  padding: 5px 7px;
  margin-top: 10px;

  background-color: rgb(62, 63, 41);

  font-size: 15px;
  color: white;

  border: none;
  border-radius: 8px;

  cursor: pointer;
  transition: opacity 0.2s;
}
.dashboard-buttons:hover {
  opacity: 0.8;
}
.dashboard-buttons:active {
  opacity: 0.6;
}

.dashboard-forms {
  display: inline-block;
  text-align: center;
}

.logout-button {
  margin-top: 10px;
  padding: 7px 7px;

  background-color: rgb(188, 168, 141);

  font-size: 15px;
  color: white;

  border: none;
  border-radius: 8px;

  cursor: pointer;
  transition: opacity 0.2s;
}
.logout-button:hover,
.logout-button:active {
  opacity: 0.8;
}

```

m

```

}
.logout-button:active {
  opacity: 0.6;
}

/* Logoändernseite */

.change-logo-form-div {
  text-align: center;
}

.logo-input {
  padding: 2px 2px;
}

.change-logo-legend,
.security-info {
  margin-bottom: 10px;

  display: block;
  text-align: center;

  font-size: 18px;
  font-weight: bold;
}

/* Sicherheitsinformationsseite */

.security-info-div {
  display: flex;
  flex-direction: row;
  justify-content: center;
}
.security-info {
  width: 500px;
  padding-left: 10px;
  padding-right: 10px;
  font-size: 16px;
}

/* Email-/Passwortänderungsseite */

.change-email-input,
.change-password-input,
.dashboard-input {
  width: 300px;
  padding: 4px 2px;

  font-size: 15px;

  border: solid;
  border-width: 3px;
  border-color: rgb(62, 63, 41);
  border-radius: 7px;
}

```

```

/* Schreiberlistenseite */

.writer-list {
  display: block;
  text-align: center;

  font-size: 20px;
}

/* Errormeldung */

.error-div {
  display: flex;
  flex-direction: row;
  justify-content: center;
}

.notice {
  width: 407px;
  padding: 5px 5px;
  margin-top: -10px;

  background-color: rgba(188, 168, 141, 0.3);

  border-radius: 6px;
}

/* Benutzerliste Seite */

.users-list-div {
  text-align: center;
}

.users-list-email,
.users-list-username {
  font-size: 17px;
}

.users-list-email {
  color: rgb(62, 63, 41);
  font-weight: bold;
}

.users-list-username {
  margin-bottom: 40px;
  margin-top: 0;
}

/* Bildschirmgrösseanpassung */

@media (max-width: 510px) {

  .dashboard-input {
    width: 170px;
  }
}

```

```

.security-info {
    width: 320px;
}

.change-password-input {
    margin-top: 8px;
}

.dashboard-title {
    margin-top: -80px;
}

.writerlist-title {
    margin-top: -80px;
}

.users-list-div {
    margin-top: -80px;
}

.security-info-div {
    margin-top: -100px;
}

.notice {
    width: 300px;
}
}

```

### 8.1.7 data-protection.css

```

/* Header */

.article-button,
.articlewish-button,
.contact-button {
    background-color: rgb(241, 240, 228);

    color: black;
}

/* Datenschutzseite */

main {
    display: flex;
    flex-direction: row;
    justify-content: center;
}

.data-protection-text {
    width: 500px;

    text-align: center;
}

```

```

/* Bildschirmgrösseanpassung */

@media (max-width: 600px) {
  .data-protection-text {
    width: 320px;
    margin-top: -100px;
  }
}

```

## 8.1.8 general.css

```

/* Allgemeine CSS-Datei -> Verändert alle Seiten */

/* Seitenformatierung */

body {
  margin-top: 200px;

  background-color: rgb(241, 240, 228);
}

/* Textformatierung */

p, button, legend {
  font-family: 'Segoe UI';
}

/* Footer */

footer {
  bottom: 0;
  margin-top: 1200px;

  text-align: center;
}

.footer-buttons {
  background-color: rgba(188, 168, 141, 0.9);

  border: none;
  border-radius: 3px;

  font-size: 15px;
  color: white;

  cursor: pointer;
  transition: opacity 0.2s;
}

.footer-buttons:hover {
  opacity: 0.8;
}

.footer-buttons:active {
  opacity: 0.6;
}

```

```

/* Bildschirmgrösseanpassung */

@media (max-width: 600px) {
  footer {
    margin-top: 450px;
  }
}

```

## 8.1.9 header.css

```

/* Allgemeine Header CSS-Datei -> Beeinflusst alle Seiten */

header {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
}

.header-div {
  padding: 9px 50px;

  background-color: rgb(241, 240, 228);

  display: flex;
  flex-direction: row;
  justify-content: space-between;
}

/* Logo */

.logo-div {
  display: flex;
  justify-content: center;
}

.logo {
  width: 120px;

  cursor: pointer;
}

/* Header Links */

.header-buttons {
  width: 370px;

  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}

.article-button,
.articlewish-button,

```

```

.contact-button {
  padding: 7px 14px;
  margin-left: -17px;

  background-color: rgb(241, 240, 228);

  font-size: 15px;

  border: none;
  border-radius: 4px;

  cursor: pointer;
  transition: background-color 0.2s;
}

.article-button:hover,
.articlewish-button:hover,
.contact-button:hover {
  background-color: rgb(125, 141, 134);
  color: white;
}

/* Benutzer-Icon */

.user-icon-button {
  width: 42px;
  height: 42px;

  background-color: rgb(241, 240, 228);

  border: solid;
  border-color: rgb(62, 63, 41);
  border-width: 2px;
  border-radius: 25px;

  cursor: pointer;
  transition: opacity 0.2s;
}

.user-icon-button:hover {
  opacity: 0.8;
}

/* Bildschirmgrösseanpassung */

@media (max-width: 720px) {
  .logo {
    width: 0;
  }

  .header-div {
    padding-top: 25px;

    justify-content: center;
  }

  .header-buttons {

```

```

    margin-left: -10px;

    justify-content: space-between;
}
}

@media (max-width: 430px) {
    .user-icon-button {
        margin-right: -25px;
    }

    .article-button,
    .articlewish-button,
    .contact-button {
        font-size: 15px;

        transition: background-color 0s;
    }

    .user-icon-button {
        transition: opacity 0s;
    }
}

```

### 8.1.10 index.css

```

/* Header */

.article-button {
    background-color: rgb(125, 141, 134);

    color: white;
}

.articlewish-button,
.contact-button {
    background-color: rgb(241, 240, 228);

    color: black;
}

/* Artikelsortierungsknöpfe */

.order-div {
    margin-left: 50px;

    display: flex;
    flex-direction: row;
    column-gap: 5px;
}

.order-buttons {
    padding: 1px 3px;
}

```

```

background-color: rgba(62, 63, 41, 0.5);

font-size: 15px;
color: white;

border: solid;
border-color: rgb(62, 63, 41);
border-width: 3px;
border-radius: 8px;

cursor: pointer;
transition: background-color 0.2s, opacity 0.2s;
}.order-buttons:hover {
    background-color: rgb(62, 63, 41);
}
.order-buttons:active {
    opacity: 0.6;
}

.order-buttons-active,
.search-button {
    padding: 4px 6px;

    background-color: rgb(62, 63, 41);

    font-size: 15px;
    color: white;

    border: none;
    border-radius: 8px;

    cursor: pointer;
    transition: opacity 0.2s;
}
.order-buttons-active:hover,
.search-button:hover {
    opacity: 0.8;
}
.order-buttons-active:active,
.search-button:active {
    opacity: 0.6;
}

/* Artikelübersicht */

.articles {
    margin-top: 10px;
    margin-right: 50px;
    margin-left: 50px;

    display: grid;
    column-gap: 30px;
    row-gap: 30px;
}

.article-image {

```

u

```

width: 100%;

object-fit: contain;

border-radius: 10px;
}

.article-div {
  max-height: fit-content;

  display: grid;
  grid-template-columns: 1fr;

  cursor: pointer;
}

.article-text {
  text-align: center;
}

.article-title {
  margin-top: 10px;

  font-size: 19px;
  font-weight: bold;
  color: rgb(62, 63, 41);
}

.article-description {
  margin-top: -12px;

  font-size: 16px;
}

/* Bildschirmgrösseanpassung */

@media (max-width: 600px) {
  .articles {
    margin-right: 20px;
    margin-left: 20px;

    grid-template-columns: 1fr;
  }
  .order-div {
    margin-top: -80px;
    margin-bottom: 20px;
    margin-left: 20px;
  }
}

@media (min-width: 601px) and (max-width: 1150px) {
  .articles {
    grid-template-columns: 1fr 1fr;
  }
}

```

```

@media (min-width: 1151px) and (max-width: 1500px) {
  .articles {
    grid-template-columns: 1fr 1fr 1fr;
  }
}

@media (min-width: 1501px) and (max-width: 2000px) {
  .articles {
    grid-template-columns: 1fr 1fr 1fr 1fr;
  }
}

@media (min-width: 2001px) {
  .articles {
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr;
  }
}

```

### 8.1.11 single-article.css

```

/* Header */

.article-button,
.articlewish-button,
.contact-button {
  background-color: rgb(241, 240, 228);

  color: black;
}

/* Artikelansichtsseite */

.article-div {
  display: flex;
  flex-direction: column;
  align-items: center;
}

/* Artikelüberschrift */

.article-title {
  font-size: 24px;
  font-weight: bold;
  color: rgb(62, 63, 41);
}

.name-date {
  margin-top: 8px;

  font-weight: bold;
}

/* Artikel */

```

```

.article-text-div {
  width: 700px;

  text-align: justify;
}

.article-text {
  font-size: 15px;
}

/* Artikeländerungs-, Artikellöschungs- und Likeknopf */

.change-article-buttons-div-div {
  display: flex;
  flex-direction: row;
  justify-content: center;
}

.change-article-buttons-div {
  width: 300px;

  display: flex;
  flex-direction: row;
  justify-content: space-evenly;
}

.like-button-div {
  text-align: center;
}

.change-article-buttons,
.like-button {
  padding: 7px 7px;

  background-color: rgb(62, 63, 41);

  font-size: 14px;
  color: white;

  border: none;
  border-radius: 8px;

  cursor: pointer;
  transition: opacity 0.2s;
}

.change-article-buttons:hover,
.like-button:hover {
  opacity: 0.8;
}

.change-article-buttons:active,
.like-button:active {
  opacity: 0.6;
}

/* Bildschirmgrösseanpassung */

```

```

@media (max-width: 550px) {
  .article-text-div {
    width: 400px;
  }
  .article-div {
    margin-top: -100px;
  }
}

@media (max-width: 450px) {
  .article-text-div {
    width: 300px;
  }

  .article-div {
    margin-top: -100px;
  }
}

```

## 8.2 EJS-Dateien

### 8.2.1 «/includes/»-Ordner

#### 8.2.1.1 footer.ejs

```

<footer>
  <!-- Datenschutz Knopf -->
  <button class="footer-buttons" onclick="location.href='/data-protection.ejs';">Daten-
schutz</button>
</footer>

```

#### 8.2.1.2 header.ejs

```

<header>
  <div class="header-div">
    <!-- Logo -->
    <div class="logo-div">
      
    </div>

    <div class="header-buttons">
      <!-- Header Knöpfe -->
      <button class="article-button" onclick="location.href='/';">Artikel</button>
      <button class="articlewish-button" onclick="location.href='/articlewish.ejs';">Arti-
kelwunsch</button>
      <button class="contact-button" onclick="location.href='/contact.ejs';">Kontakt</but-
ton>

      <!-- Benutzer-Icon Knopf mit verschiedenem Link, je nach Benutzer -->
      <% if (user.username == "admin") {%>
        <button class="user-icon-button" onclick="location.href='/dashboard-admin.ejs';">
        
      <% } else if (user) { %>
        <button class="user-icon-button" onclick="location.href='dashboard.ejs';">
        
      }
    }
  </div>

```

```

        <% } else { %>
        <button class="user-icon-button" onclick="location.href='account.ejs';">
        
        <%}%>
    </button>
</div>
</div>
</header>

```

## 8.2.2 account.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/account.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- Login/Registrieren Knöpfe -->
      <div class="account-button-div">
        <button class="account-buttons" onclick="location.href='/login.ejs';">Einloggen</but-
ton>
        <button class="account-buttons" onclick="location.href='/register.ejs';">Registrie-
ren</button>
      </div>
    </main>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

## 8.2.3 articlewish.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/articlewish.css">
  </head>

```

```

<body>
  <!-- Header -->
  <%- include("includes/header") %>

  <main>
    <div class="articlewish-div">
      <!-- Errormeldung -->
      <% errors.forEach(errors => { %>
        <p class="notice"><%= errors %></p>
      <% }) %>
      <!-- Artikelwunsch Eingabe -->
      <form class="articlewish-form" action="/articlewish-input" method="POST">
        <input class="email-input" id="email" name="email" placeholder="Email... (optional)">
        <textarea class="articlewish-input" id="articlewish" name="articlewish" placeholder="Artikelwunsch..."></textarea>
        <p class="accept-data-protection">Mit Klick auf "Senden" akzeptieren Sie die <a class="data-protection-link" href="/data-protection.ejs">Datenschutzerklärung</a>.</p>
        <button class="send-button">Senden</button>
      </form>
    </div>
  </main>

  <!-- Footer -->
  <%- include("includes/footer") %>
</body>
</html>

```

## 8.2.4 articlewishes-list.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/articlewishes-list.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- Artikelwunschliste -->
      <div class="articlewishes-list-div">
        <% articlewishes.forEach(articlewishes => { %>
          <p class="articlewishes-list-email"><%= articlewishes.email %></p>
          <p class="articlewishes-list-articlewish"><%= articlewishes.articlewish %></p>
          <hr>
          <br>
        <% }) %>
      </div>
    </main>
  </body>
</html>

```

```

    </div>
  </main>

  <!-- Footer -->
  <%- include("includes/footer") %>
</body>
</html>

```

## 8.2.5 change-email.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/dashboard.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <div class="dashboard-div">
        <!-- Errormeldungen -->
        <div class="error-div">
          <% errors.forEach(error => { %>
            <p class="notice"><%= errors %></p>
          <% }) %>
        </div>
        <!-- Emailänderungseingabe -->
        <form action="/change-email" method="POST">
          <input class="change-email-input" id="newemail" name="newemail" placeholder="Neue
Emailadresse...">
          <button class="dashboard-buttons">Email ändern</button>
        </form>
      </div>
    </main>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

## 8.2.6 change-logo.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Blog</title>
<!-- Import von CSS-Dateien -->
<link rel="stylesheet" href="/general.css">
<link rel="stylesheet" href="/header.css">
<link rel="stylesheet" href="/dashboard.css">
</head>

<body>
  <!-- Header -->
  <%- include("includes/header") %>

  <main>
    <!-- Logoänderungsupload -->
    <div class="change-logo-form-div">
      <form class="dashboard-forms" action="/upload-logo" method="POST" enctype="multi-
part/form-data">
        <label class="change-logo-legend">Bitte quadratisches Bild hochladen:</label>
        <input class="dashboard-input logo-input" type="file" id="logo" name="logo"
placeholder="Logo...">
        <button class="dashboard-buttons">Logo hochladen</button>
      </form>
    </div>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

## 8.2.7 change-password.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/dashboard.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <div class="dashboard-div">
        <!-- Errormeldungen -->
        <div class="error-div">
          <% errors.forEach(error => { %>
            <p class="notice"><%= errors %></p>
          <% }) %>
        </div>

```

```

    <!-- Passwortänderungseingabe -->
    <form class="change-password-form" action="/change-password" method="POST">
      <input class="change-password-input" id="newpassword" name="newpassword" place-
holder="Neues Passwort...">
      <input class="change-password-input" id="newpasswordcheck" name="newpassword-
check" placeholder="Neues Passwort erneut eingeben...">
      <button class="dashboard-buttons">Passwort ändern</button>
    </form>
  </div>
</main>

<!-- Footer -->
<%- include("includes/footer") %>
</body>
</html>

```

## 8.2.8 contact.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/contact.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- Kontaktdetails -->
      <div class="contact-div">
        <p class="contact-name"><%= admin.adminVisualName %></p>
        <p class="contact-email"><%= adminEmail.email %></p>
        <p class="maturaarbeit-link"><a class="maturaarbeit-link"
href="https://github.com/jerryvscode/maturaarbeit">Source Code und schriftliche Maturaar-
beit zu diesem Projekt (inkl. Quellen)</a>
      </div>
    </main>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

## 8.2.9 create-article.ejs

```

<!DOCTYPE html>
<html>

```

```

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Blog</title>
  <!-- Import von CSS-Dateien -->
  <link rel="stylesheet" href="/general.css">
  <link rel="stylesheet" href="/header.css">
  <link rel="stylesheet" href="/create-edit-article.css">
</head>

<body>
  <!-- Header -->
  <%- include("includes/header") %>

  <main>
    <div class="create-article-div">
      <!-- Errormeldungen -->
      <% errors.forEach(errors => { %>
        <p class="notice"><%= errors %></p>
      <% }) %>
      <!-- Artikeleingabe -->
      <form class="create-article-form" action="/create-article" method="POST" >
        <input class="title-input" name="title" id="title" placeholder="Titel...">
        <textarea class="article-input" name="article" id="article" placeholder="Arti-
kel..."></textarea>
        <button class="create-article-button">Veröffentlichen</button>
      </form>
    </div>
  </main>

  <!-- Footer -->
  <%- include("includes/footer") %>
</body>
</html>

```

## 8.2.10 dashboard-admin.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/dashboard.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- Begrüssung und Angabe aktuelle Emailadresse -->

```

```

<p class="dashboard-title">Dashboard von <b><%= user.username %></b></p>
<p class="dashboard-email"><%= email.email %></p>
<div class="dashboard-div">
  <br>
  <!-- Logoänderungsknopf -->
  <button class="dashboard-buttons" onclick="location.href='/change-logo.ejs';">Logo
ändern</button>
  <br>
  <!-- Echter Name des Admins ändern -->
  <form class="dashboard-forms" action="/admin-visual-name" method="POST">
    <input class="dashboard-input" id="adminVisualName" name="adminVisualName" place-
holder="Echter Vor- und Nachname...">
    <button class="dashboard-buttons">Aktualisieren</button>
  </form>
  <br>
  <!-- Emailadresseänderungs-Seite -->
  <button class="dashboard-buttons" onclick="location.href='/change-email';">Email
ändern</button>
  <!-- Passwortänderungs-Seite -->
  <button class="dashboard-buttons" onclick="location.href='/change-password';">Pass-
wort ändern</button>
  <br>
  <!-- Artikelerstellen-Seite -->
  <button class="dashboard-buttons" onclick="location.href='/create-article';">Arti-
kel erstellen</button>
  <!-- Artielwunschlisten-Seite -->
  <button class="dashboard-buttons" onclick="location.href='/articlewishes-
list.ejs';">Artikelwünsche</button>
  <br>
  <!-- Schreiberhinzufügen-Eingabe -->
  <form class="dashboard-forms" action="/add-writer" method="POST">
    <input class="dashboard-input" id="addwriter" name="addwriter" placeholder="Be-
nutzernamen...">
    <button class="dashboard-buttons">Schreiber hinzufügen</button>
  </form>
  <br>
  <!-- Schreiberentfernen-Eingabe -->
  <form class="dashboard-forms" action="/remove-writer" method="POST">
    <input class="dashboard-input" id="removewriter" name="removewriter" placehol-
der="Benutzernamen...">
    <button class="dashboard-buttons">Schreiber entfernen</button>
  </form>
  <br>
  <!-- Schreiberlisten-Seite -->
  <button class="dashboard-buttons" onclick="location.href='/writers.ejs';">Schrei-
ber-Liste</button>
  <br>
  <!-- Benutzerlisten-Seite -->
  <button class="dashboard-buttons" onclick="location.href='/users.ejs';">Benutzer-
liste</button>
  <br>
  <!-- Sicherheitsinformations-Seite -->
  <button class="dashboard-buttons" onclick="location.href='/security-info.ejs';">Si-
cherheitsinformation</button>
  <br>
  <!-- Logout-Knopf -->

```

```

        <button class="logout-button" onclick="location.href='/logout';">Logout</button>
    </div>
</main>

<!-- Footer -->
<%= include("includes/footer") %>
</body>
</html>

```

## 8.2.11 dashboard.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/dashboard.css">
  </head>

  <body>
    <!-- Header -->
    <%= include("includes/header") %>

    <main>
      <!-- Dashboardüberschrift -->
      <p class="dashboard-title">Dashboard von <b><%= user.username %></b></p>
      <p class="dashboard-email"><%= email.email %></p>

      <div class="dashboard-div">
        <!-- Email- und Passwortänderungsseiten -->
        <button class="dashboard-buttons" onclick="location.href='/change-email';">Email
ändern</button>
        <button class="dashboard-buttons" onclick="location.href='/change-password';">Pass-
wort ändern</button>

        <!-- Wenn Schreiber -> Artikelerstellen- und Artikelwunschlistenseite -->
        <% if (!isNotWriter) {%>
        <br>
        <button class="dashboard-buttons" onclick="location.href='/create-article';">Arti-
kel erstellen</button>
        <button class="dashboard-buttons" onclick="location.href='/articlewishes-
list.ejs';">Artikelwünsche</button>
        <%}%>
        <br>
        <!-- Logoutknopf -->
        <button class="logout-button" onclick="location.href='/logout';">Logout</button>
      </div>
    </main>

    <!-- Footer -->
    <%= include("includes/footer") %>

```

```
</body>
</html>
```

## 8.2.12 data-protection.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/data-protection.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>
    <main>
      <!-- Datenschutzerklärung -->
      <div class="data-protection-text">
        <p><b>Datenschutzerklärung</b></p>
        <p>Diese Website wird verwaltet von <%= admin.adminVisualName %>.</p>
        <p>Es werden bei der Registrierung eingegebene Emailadresse, Benutzername und Passwort auf unbestimmte Zeit abgespeichert.</p>
        <p>Die Emailadresse wird verwendet, um Kontakt aufnehmen zu können mit Benutzer. Benutzername und Passwort werden verwendet, damit Benutzer ein Konto erstellen können.</p>
        <p>Ausserdem wird ein Cookie auf dem Browser des Benutzers abgespeichert, um ihn einzuloggen.</p>
        <p>Auch wurden verschiedene Externe Dienste, wie Node.js Frameworks oder ein Hostinganbieter, zur Erstellung und Veröffentlichung der Website verwendet. Es ist dem Inhaber nicht bekannt, ob diese Dienste auch Benutzerdaten sammeln.</p>
        <p>Bei Fragen, kontaktieren Sie den Inhaber über die Emailadresse: <%= adminEmail.email %></p>
      </div>
    </main>
  </body>
</html>
```

## 8.2.13 edit-article.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/create-edit-article.css">
  </head>
```

```

<body>
  <!-- Header -->
  <%= include("includes/header") %>

  <main>
    <div class="create-article-div">
      <!-- Errormeldungen -->
      <% errors.forEach(errors => { %>
        <p class="notice"><%= errors %></p>
      <% }) %>

      <!-- Titel- und Artikeleingabe -->
      <form class="create-article-form" action="/edit-article/<%= article.id %>" me-
thod="POST">
        <input class="title-input" value="<%= article.title %>" name="title" id="title"
placeholder="Titel...">
        <textarea class="article-input" name="article" id="article" placeholder="Arti-
kel..."><%= article.article %></textarea>
        <button class="create-article-button">Änderungen veröffentlichen</button>
      </form>
    </div>
  </main>

  <!-- Footer -->
  <%= include("includes/footer") %>
</body>
</html>

```

## 8.2.14 index.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" type="text/css" href="/general.css">
    <link rel="stylesheet" type="text/css" href="/header.css">
    <link rel="stylesheet" type="text/css" href="/index.css">
  </head>

  <body>
    <!-- Header -->
    <%= include("includes/header") %>

    <main>
      <!-- Sortierknöpfe -->
      <div class="order-div">
        <button class="order-buttons-active" onclick="location.href='/in-
dex.ejs';">Neuste</button>
        <button class="order-buttons" onclick="location.href='/oldest-in-
dex.ejs';">Älteste</button>

```

```

        <button class="order-buttons" onclick="location.href='/mostpopular-index.ejs';">Be-
liebteste</button>
    </div>

    <!-- Artikelübersicht -->
    <div class="articles">
        <% articles.forEach(article => { %>
            <div class="article-div" onclick="location.href='/article/<%= article.id %>';">
                
                <div class="article-text">
                    <p class="article-title"><%= article.title %></p>
                    <p class="article-description"><%= article.article.slice(0, 300) %>...</p>
                </div>
            </div>
        <% }) %>
    </div>
</main>

<!-- Footer -->
<%= include("includes/footer") %>
</body>
</html>

```

## 8.2.15 login.ejs

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Blog</title>
        <!-- Import von CSS-Dateien -->
        <link rel="stylesheet" href="/general.css">
        <link rel="stylesheet" href="/header.css">
        <link rel="stylesheet" href="/account.css">
    </head>

    <body>
        <!-- Header -->
        <%= include("includes/header") %>

        <main>
            <div class="login-div">
                <form action="/login" method="POST">
                    <!-- Errormeldungen -->
                    <% errors.forEach(error => { %>
                        <p class="notice"><%= errors %></p>
                    <% }) %>

                    <!-- Logineingabe -->
                    <div class="login-form">
                        <legend class="login-register-legend">Einloggen</legend>
                        <input class="account-input" id="username" name="username" placeholder="Benut-
zername...">

```

```

        <input class="account-input" id="password" name="password" type="password"
placeholder="Passwort..."></input>
        <button class="send-button">Einloggen</button>
    </div>
</form>
</div>
</main>

<!-- Footer -->
<%- include("includes/footer") %>
</body>
</html>

```

## 8.2.16 mostpopular-index.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" type="text/css" href="/general.css">
    <link rel="stylesheet" type="text/css" href="/header.css">
    <link rel="stylesheet" type="text/css" href="/index.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- Sortierknöpfe -->
      <div class="order-div">
        <button class="order-buttons" onclick="location.href='/index.ejs';">Neuste</button>
        <button class="order-buttons" onclick="location.href='/oldest-in-
dex.ejs';">Älteste</button>
        <button class="order-buttons-active" onclick="location.href='/mostpopular-in-
dex.ejs';">Beliebtteste</button>
      </div>

      <!-- Artikelübersicht -->
      <div class="articles">
        <% articles.forEach(article => { %>
          <div class="article-div" onclick="location.href='/article/<%= article.id %>';">
            
            <div class="article-text">
              <p class="article-title"><%= article.title %></p>
              <p class="article-description"><%= article.article.slice(0, 300) %>...</p>
            </div>
          </div>
        <% }) %>
      </div>
    </main>

```

```

<!-- Footer -->
<%- include("includes/footer") %>
</body>
</html>

```

## 8.2.17 oldest-index.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" type="text/css" href="/general.css">
    <link rel="stylesheet" type="text/css" href="/header.css">
    <link rel="stylesheet" type="text/css" href="/index.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- Sortierknöpfe -->
      <div class="order-div">
        <button class="order-buttons" onclick="location.href='/index.ejs';">Neuste</button>
        <button class="order-buttons-active" onclick="location.href='/oldest-in-
dex.ejs';">Älteste</button>
        <button class="order-buttons" onclick="location.href='/mostpopular-index.ejs';">Be-
liebteste</button>
      </div>

      <!-- Artikelübersicht -->
      <div class="articles">
        <% articles.forEach(article => { %>
          <div class="article-div" onclick="location.href='/article/<%= article.id %>';">
            
            <div class="article-text">
              <p class="article-title"><%= article.title %></p>
              <p class="article-description"><%= article.article.slice(0, 300) %>...</p>
            </div>
          </div>
        <% }) %>
      </div>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

## 8.2.18 register.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/account.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>
    <main>
      <div class="register-div">
        <form action="/register" method="POST">
          <!-- Errormeldungen -->
          <% errors.forEach(error => { %>
            <p class="notice"><%= errors %></p>
          <% }) %>
          <!-- Registrierungseingabe -->
          <div class="register-form">
            <legend class="login-register-legend">Registrieren</legend>
            <input class="account-input" id="email" name="email" placeholder="Email...">
            <input class="account-input" id="username" name="username" placeholder="Benut-
zername...">
            <input class="account-input" id="password" name="password" type="password"
placeholder="Passwort..."></input>
            <p class="accept-data-protection">Mit Klick auf "Senden" akzeptieren Sie die <a
class="data-protection-link" href="/data-protection.ejs">Datenschutzerklärung</a>.</p>
            <button class="send-button">Registrieren</button>
          </div>
        </form>
      </div>
    </main>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

## 8.2.19 security-info.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/dashboard.css">

```

```

</head>

<body>
  <!-- Header -->
  <%- include("includes/header") %>

  <main>
    <!-- Sicherheitsinformation -->
    <div class="security-info-div">
      <div>
        <p class="security-info">Diese Website hat einige Sicherheitslücken. Wenn Sie auf
        Nummer sicher gehen wollen, dann laden Sie ab und zu die Dateien "ourApp.db" und "/pic-
        tures/logo/logo.png" und alle Bilder aus dem Ordner "/pictures/articles/" vom Server herun-
        ter und speichern Sie sie ab.</p>
        <br>
        <p class="security-info">Falls die Website gehackt wird (erkennbar durch z.B.
        keinen Zugriff mehr auf Ihren Account, Blogartikel werden einfach gelöscht oder hinzuge-
        fügt, etc.), dann löschen Sie den gesamten Server und setzen ihn neu auf mit dem ursprüng-
        lichen Blogtemplate (wie beim ersten Mal aufsetzen) und löschen Sie die Dateien auf dem
        Server "ourApp.db-shm", "ourApp.db-wal", "ourApp.db" und "/pictures/logo/logo.png". An-
        schliessend laden Sie Ihre neusten heruntergeladenen Versionen der oben genannten Dateien
        am jeweils gleichen Ort, wo Sie sie heruntergeladen haben, wieder hoch.</p>
      </div>
    </div>
  </main>

  <!-- Footer -->
  <%- include("includes/footer") %>
</body>
</html>

```

## 8.2.20 single-article.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/single-article.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <div class="article-div">
        <!-- Artikelüberschrift -->
        <h1 class="article-title"><%= article.title %></h1>

```

```

        <p class="name-date">-- veröffentlicht von <%= article.username %> am <%= new
Date(article.createdDate).getDate() %>. <%= new Date(article.createdDate).getMonth() + 1
%>. <%= new Date(article.createdDate).getFullYear() %> --</p>
        <!-- Artikelinhalt -->
        <div class="article-text-div">
            <p class="article-text"><%- filterUserHTML(article,article) %></p>
        </div>
    </div>
    <br>
    <!-- Likeknopf -->
    <div class="like-button-div">
        <form action="/like-article/<%= article.id %>" method="POST">
            <button class="like-button"><%= likes.likes %> &#128077;</button>
        </form>
    </div>
    <br>
    <!-- Wenn Autor des Artikels oder Admin -> Artikelbearbeiten- und ArtikelLöschenknopf
-->
    <% if (isAuthor || userid === 1) { %>
        <div class="change-article-buttons-div-div">
            <div class="change-article-buttons-div">
                <button class="change-article-buttons" onclick="location.href='/edit-ar-
ticle/<%= article.id %>';">Artikel bearbeiten</button>
                <form action="/delete-article/<%= article.id %>" method="POST">
                    <!-- Löschen bestätigen -->
                    <button class="change-article-buttons" onclick="return confirm('Wollen Sie
diesen Artikel löschen?')">Artikel löschen</button>
                </form>
            </div>
        </div>
    <% } %>
</main>

<!-- Footer -->
<%- include("includes/footer") %>
</body>
</html>

```

## 8.2.21 upload-picture.ejs

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Blog</title>
        <!-- Import von CSS-Dateien -->
        <link rel="stylesheet" href="/general.css">
        <link rel="stylesheet" href="/header.css">
        <link rel="stylesheet" href="/create-edit-article.css">
    </head>

    <body>
        <!-- Header -->
        <%- include("includes/header") %>
    </body>
</html>

```

```

    <main>
      <div class="create-article-div">
        <!-- Bild hochladen -->
        <form action="/save-picture" method="POST" enctype="multipart/form-data">
          <legend class="picture-upload-legend">Titelbild mit Format 16:9 hochladen.</le-
gend>
          <input class="picture-input" type="file" name="picture" id="picture">
          <button class="create-article-button">Upload bestätigen</button>
        </form>
      </div>
    </main>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

## 8.2.22 users.ejs

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/dashboard.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- Benutzerliste -->
      <div class="users-list-div">
        <% users.forEach(users => { %>
          <p class="users-list-email"><%= users.email %></p>
          <p class="users-list-username"><%= users.username %></p>
          <hr>
          <br>
        <% }) %>
      </div>
    </main>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>

```

### 8.2.23 writers.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
    <!-- Import von CSS-Dateien -->
    <link rel="stylesheet" href="/general.css">
    <link rel="stylesheet" href="/header.css">
    <link rel="stylesheet" href="/dashboard.css">
  </head>

  <body>
    <!-- Header -->
    <%- include("includes/header") %>

    <main>
      <!-- SchreiberListe -->
      <p class="writerlist-title">Schreiber</b></p>
      <div class="dashboard-div">
        <% writers.forEach(writer => { %>
          <p class="writer-list"><%= writer.writer %></p>
        <% }) %>
      </div>
    </main>

    <!-- Footer -->
    <%- include("includes/footer") %>
  </body>
</html>
```

## 8.3 Sonstige Dateien

### 8.3.1 .env

```
JWTSECRET=lösfdöljfdfeifh
```

### 8.3.2 ourApp.db, ourApp.db-smh und ourApp.db-wal

Diese Dateien kann man nicht in einem Texteditor anzeigen, um herauszufinden, was diese Datenbank speichert braucht man eine spezielle App, ich verwende «DB Browser (SQLite)».

Auf <https://github.com/jerryvscodematurationarbeit> kann man die «ourApp.db»-Datei herunterladen, die man dann mit der besagten App öffnen kann.

### 8.3.3 package.json

Die einzige selbstgeschriebene Zeile Code in dieser Datei ist die orange, der Rest wurde automatisch erstellt.

```
{
  qq
```

```

"name": "webapp",
"version": "1.0.0",
"description": "",
"main": "index.js",
"scripts": {
  "dev": "nodemon server",
  "test": "echo \\\"Error: no test specified\\\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC",
"dependencies": {
  "bcrypt": "^6.0.0",
  "better-sqlite3": "^12.4.1",
  "cookie-parser": "^1.4.7",
  "dotenv": "^17.2.3",
  "ejs": "^3.1.10",
  "express": "^5.1.0",
  "jsonwebtoken": "^9.0.2",
  "marked": "^16.4.0",
  "multer": "^2.0.2",
  "nodemon": "^3.1.10",
  "sanitize-html": "^2.17.0"
}
}

```

### 8.3.4 server.js

```

// NPM Package

require("dotenv").config()
const fork = require('child_process').fork
const multer = require("multer")
const jwt = require("jsonwebtoken")
const marked = require("marked")
const sanitizeHTML = require("sanitize-html")
const bcrypt = require("bcrypt")
const cookieParser = require("cookie-parser")
const express = require("express")
const db = require("better-sqlite3")("ourApp.db")
const ejs = require("ejs")
db.pragma("journal_mode = WAL")

// SQLite Setup

const createTables = db.transaction(() => {
  // Benutzertabelle erstellen
  db.prepare(
    `
    CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      email STRING NOT NULL,
      username STRING NOT NULL UNIQUE,

```

```

    password STRING NOT NULL
  )
  `
).run()

// Artikeltablelle erstellen
db.prepare(
  CREATE TABLE IF NOT EXISTS articles (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    createdAt TEXT,
    title STRING NOT NULL,
    article TEXT NOT NULL,
    likes INTEGER,
    authorid INTEGER,
    FOREIGN KEY (authorid) REFERENCES users (id)
  )
  `).run()

// Artikelwunschtabelle erstellen
db.prepare(
  `
  CREATE TABLE IF NOT EXISTS articlewishes (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    email STRING,
    articlewish STRING NOT NULL
  )
  `
).run()

// Schreibertabelle erstellen
db.prepare(
  `
  CREATE TABLE IF NOT EXISTS writers (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    writer STRING NOT NULL
  )
  `
).run()

// Sicherungstabelle für den echten und sichtbaren Namen des Admins erstellen
db.prepare(
  `
  CREATE TABLE IF NOT EXISTS adminVisualName (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    adminVisualName STRING NOT NULL
  )
  `
).run()

// Für jeden Benutzer automatisch eine Tabelle erstellen
const usersStatement = db.prepare("SELECT username FROM users ORDER BY id ASC")
const users = usersStatement.all()

users.forEach(users => {
  db.prepare(
    `

```

```

    CREATE TABLE IF NOT EXISTS ${users.username} (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      likedArticles STRING NOT NULL
    )
    `
  ).run()
})
})

createTables()

// NPM Packete initialisieren

const app = express()

app.set("view engine", "ejs")
app.use(express.urlencoded({extended: false}))
app.use(express.static("public"))
app.use(express.static("pictures"))
app.use(cookieParser())

// Middleware

app.use(function (req, res, next) {
  // Markdownfunktion einrichten
  res.locals.filterUserHTML = function (content) {
    return sanitizeHTML(marked.parse(content), {
      allowedTags: ["p", "br", "ul", "li", "ol", "strong", "bold", "i", "em", "h1", "h2",
"h3", "h4", "h5", "h6"],
      allowedAttributes: {}
    })
  }

  // Cookie überprüfen
  res.locals.errors = []

  try {
    const decoded = jwt.verify(req.cookies.ourSimpleApp, process.env.JWTSECRET)
    req.user = decoded
  } catch (err) {
    req.user = false
  }

  res.locals.user = req.user
  console.log(req.user)

  next()
})

// Funktionen

// Sicherstellung, dass Benutzer Admin ist
function mustBeAdmin(req, res, next) {
  if (req.user.userid == "1") {

```

```

    return next()
  }
  return res.redirect("/")
}

// Sicherstellung, dass Benutzer Schreiber ist
function mustBeWriter(req, res, next) {
  const searchWriter = db.prepare("SELECT writer FROM writers WHERE writer = ?")
  const writer = searchWriter.get(req.user.username)

  if (!writer && !(req.user.userid == "1")) {
    return res.redirect("/")
  } else {
    next()
  }
}

// Sicherstellung, dass Benutzer eingeloggt ist
function mustBeLoggedIn(req, res, next) {
  const loggedInStatement = db.prepare(`SELECT username FROM users WHERE id = ?`)
  const loggedIn = loggedInStatement.get(req.user.userid)
  const loggedInCheckFalse = !loggedIn

  if (loggedInCheckFalse) {
    return res.redirect("/account.ejs")
  } else {
    return next()
  }
}

// Überprüfung bei Artikelerstellung

// Kopiert aus Back-End-Tutorial von...
function sharedArticleValidation(req) {
  const errors = []

  if(typeof req.body.title !== "string") req.body.title = ""
  if(typeof req.body.article !== "string") req.body.article = ""

  // Schliessen von Sicherheitslücken
  req.body.title = sanitizeHTML(req.body.title.trim(), {allowedTags: [], allowedAttributes: {}})
  req.body.article = sanitizeHTML(req.body.article.trim(), {allowedTags: [], allowedAttributes: {}})

  // Vorgabenprüfung
  if (!req.body.title) errors.push("Titelfeld muss ausgefüllt sein.")
  if (!req.body.article) errors.push("Artikelfeld muss ausgefüllt sein.")
  if (req.body.title.length >= 30) errors.push("Titel darf nicht länger als 30 Zeichen sein.")
  return errors
}

// Bildhochladefunktion
let whichPicture = ""
function logoPicture(req, res, next) {

```

uu

```

    whichPicture = "logo"
    next()
  }
  function thumbnailPicture(req, res, next) {
    whichPicture = "thumbnail"
    next()
  }
}

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    if (whichPicture === "logo") {
      cb(null, 'pictures/logo ');
    }
    if (whichPicture === 'thumbnail') {
      cb(null, 'pictures/articles ');
    }
  },
  filename: function (req, file, cb) {
    if (whichPicture === "logo") {
      cb(null, 'logo.jpg ');
    }
    if (whichPicture === 'thumbnail') {
      cb(null, `${realArticle.id}.jpg ');
    }
  },
});
const upload = multer({storage});

// GET requests

// Hauptseite (neuster Artikel zuoberst)
app.get("/", (req, res) => {
  // Artikel auslesen
  const articlesStatement = db.prepare("SELECT * FROM articles ORDER BY createdAt DESC")
  const articles = articlesStatement.all()
  res.render("index", { articles })
})

// Hauptseite mit anderem GET-Request (neuster Artikel zuoberst)
app.get("/index.ejs", (req, res) => {
  // Artikel auslesen
  const articlesStatement = db.prepare("SELECT * FROM articles ORDER BY createdAt DESC")
  const articles = articlesStatement.all()

  res.render("index", { articles })
})

// Hauptseite (ältester Artikel zuoberst)
app.get("/oldest-index.ejs", (req, res) => {
  // Artikel auslesen
  const articlesStatement = db.prepare("SELECT * FROM articles ORDER BY createdAt ASC")
  const articles = articlesStatement.all()

  res.render("oldest-index", { articles })
})

```

```

})

// Hauptseite (beliebtester Artikel zuoberst)
app.get("/mostpopular-index.ejs", (req, res) => {
  // Artikel auslesen
  const articlesStatement = db.prepare("SELECT * FROM articles ORDER BY likes DESC")
  const articles = articlesStatement.all()

  res.render("mostpopular-index", { articles })
})

// Artikelwunschseite
app.get("/articlewish.ejs", (req, res) => {
  res.render("articlewish")
})

// Kontaktseite
app.get("/contact.ejs", (req, res) => {
  // Auslesen des echten Adminnamens
  searchAdmin = db.prepare("SELECT adminVisualName FROM adminVisualName WHERE id = 1")
  admin = searchAdmin.get()

  // Auslesen der Emailadresse des Admins
  searchAdminEmail = db.prepare("SELECT email FROM users WHERE Rowid = 1")
  adminEmail = searchAdminEmail.get()

  res.render("contact", {admin, adminEmail})
})

// Datenschutzseite
app.get("/data-protection.ejs", (req, res) => {
  // Echten Adminnamen auslesen
  searchAdmin = db.prepare("SELECT adminVisualName FROM adminVisualName WHERE id = 1")
  admin = searchAdmin.get()

  // Adminemail auslesen
  searchAdminEmail = db.prepare("SELECT email FROM users WHERE Rowid = 1")
  adminEmail = searchAdminEmail.get()

  res.render("data-protection", {admin, adminEmail})
})

// Accountseite
app.get("/account.ejs", (req, res) => {
  res.render("account")
})

// Loginseite
app.get("/login.ejs", (req, res) => {
  res.render("login")
})

// Registrierungsseite
app.get("/register.ejs", (req, res) => {
  res.render("register")
})

```

```

}))

// Dashboardseite
app.get("/dashboard.ejs", (req, res) => {
  // Admin weiterleiten auf Admin-Dashboard
  if (req.user.userid == "1") {
    return res.redirect("dashboard-admin.ejs")
  }

  // Überprüfen, ob Benutzer Schreiber ist
  const searchWriter = db.prepare("SELECT writer FROM writers WHERE writer = ?")
  const writer = searchWriter.get(req.user.username)
  const isNotWriter = !writer

  // Email des Benutzer auslesen
  const emailStatement = db.prepare("SELECT email FROM users WHERE id = ?")
  const email = emailStatement.get(req.user.userid)

  res.render("dashboard", {isNotWriter, email})
})

// Emailändernseite
app.get("/change-email", (req, res) => {
  res.render("change-email")
})

// Passwortändernseite
app.get("/change-password", (req, res) => {
  res.render("change-password")
})

// Logout
app.get("/logout", (req, res) => {
  // Cookie löschen
  res.clearCookie("ourSimpleApp")

  res.redirect("/account.ejs")
})

// Artikelerstellenseite
app.get("/create-article", mustBeWriter, (req, res) => {
  res.render("create-article")
})

// Artikelwunschlistenseite
app.get("/articlewishes-list.ejs", mustBeWriter, (req, res) => {
  // Artikelwünsche auslesen
  const articlewishStatement = db.prepare("SELECT * FROM articlewishes ORDER BY id ASC")
  const articlewishes = articlewishStatement.all()

  res.render("articlewishes-list", { articlewishes })
})

```

```

// Admin-Dashboardseite
app.get("/dashboard-admin.ejs", mustBeAdmin, (req, res) => {
  // Benutzernamen auslesen
  const userLookUp = db.prepare("SELECT * FROM users WHERE id = ?")
  const user = userLookUp.get(req.user.userid)

  // Email des Benutzers auslesen
  const emailStatement = db.prepare("SELECT email FROM users WHERE id = ?")
  const email = emailStatement.get(req.user.userid)

  res.render("dashboard-admin", {email, user})
})

// Logoändernseite
app.get("/change-logo.ejs", (req, res) => {
  res.render("change-logo")
})

// Schreiberlistenseite
app.get("/writers.ejs", mustBeAdmin, (req, res) => {
  // Schreiber auslesen
  const writerStatement = db.prepare("SELECT * FROM writers ORDER BY id ASC")
  const writers = writerStatement.all()

  res.render("writers", {writers})
})

// Benutzerlistenseite
app.get("/users.ejs", mustBeAdmin, (req, res) => {
  // Benutzer auslesen
  const usersStatement = db.prepare("SELECT * FROM users ORDER BY id ASC")
  const users = usersStatement.all()

  res.render("users", {users})
})

// Sicherheitsinformationsseite
app.get("/security-info.ejs", (req, res) => {
  res.render("security-info")
})

// Artikel öffnen
app.get("/article/:id", (req, res) => {
  // Artikel in Datenbank suchen
  const articleStatement = db.prepare("SELECT articles.*, users.username FROM articles INNER JOIN users ON articles.authorid = users.id WHERE articles.id = ?")
  const article = articleStatement.get(req.params.id)

  // Wenn Artikel nicht existiert, Weiterleitung zur Hauptseite
  if (!article) {
    return res.redirect("/")
  }

  // Überprüfung, ob Benutzer Autor des Artikels ist

```

yy

```

const isAuthor = article.authorid === req.user.userid
const userid = req.user.userid

// Likeanzahl auslesen
const likesStatement = db.prepare(`SELECT likes FROM articles WHERE id = ?`)
const likes = likesStatement.get(req.params.id)

res.render("single-article", {article, isAuthor, userid, likes})
})

// "Artikel bearbeiten"-Seite
app.get("/edit-article/:id", mustBeWriter, (req, res) => {
  // Artikel auslesen
  const statement = db.prepare("SELECT * FROM articles WHERE id = ?")
  const article = statement.get(req.params.id)

  // Errorvermeidung bei manueller Abrufung eines Artikels über Suchbar
  if (!article) {
    return res.redirect("/")
  }

  // Überprüfung, ob Benutzer Autor (oder Admin) des Artikels ist
  if (article.authorid !== req.user.userid && req.user.userid !== 1) {
    return res.redirect("/")
  }

  res.render("edit-article", { article: article })
})

// POST requests

// Artikelwunsch eingereicht
app.post("/articlewish-input", (req, res) => {
  let errors = []

  // Wenn Emailangabe weggelassen wurde, Platzhalter einsetzen
  if (!req.body.email) {
    req.body.email = "-"
  }
  // Errormeldung, wenn Artikelwunsch weggelassen wurde
  if (!req.body.articlewish) {
    errors = ["Artikelwunschfeld muss ausgefüllt sein."]
    return res.render("articlewish.ejs", {errors})
  } else {
    // Artikelwunsch abspeichern
    const saveArticlewish = db.prepare("INSERT INTO articlewishes (email, articlewish) VA-
LUES (?, ?)")

    saveArticlewish.run(req.body.email, req.body.articlewish)
  }

  res.redirect("/articlewish.ejs")
})

```

```

// Registriert
app.post("/register", (req, res) => {
  // Errors
  const errors = []
  // Prüfung, ob alle Felder ausgefüllt sind
  if (req.body.email.trim() == "") {errors.push("Bitte Emailadresse eingeben."); return res.render("register", {errors})}
  if (req.body.username.trim() == "") {errors.push("Bitte Benutzername eingeben."); return res.render("register", {errors})}
  if (req.body.password.trim() == "") {errors.push("Bitte Passwort eingeben."); return res.render("register", {errors})}

  // Sicherheitslücke schliessen und potentielle Errormeldungen vermeiden
  if (typeof req.body.email !== "string") {req.body.email = ""; errors.push("Die Emailadresse muss Textformat haben."); return res.render("register", {errors})}
  if (typeof req.body.username !== "string") {req.body.username = ""; errors.push("Der Benutzername muss Textformat haben."); return res.render("register", {errors})}
  if (typeof req.body.password !== "string") {req.body.password = ""; errors.push("Das Passwort muss Textformat haben."); return res.render("register", {errors})}

  // Versehentliche Leerschläge am Anfang oder Ende entfernen
  req.body.username = req.body.username.trim()
  req.body.email = req.body.email.trim()

  // Email-, Benutzernamen und Passwortprüfung
  if (req.body.username.length == 1 || req.body.username.length == 2) {errors.push("Der Benutzername muss mindestens 3 Zeichen lang sein."); return res.render("register", {errors})}
  if (req.body.username.length > 20) {errors.push("Der Benutzername darf nicht länger als 20 Zeichen sein."); return res.render("register", {errors})}
  if (!req.body.username.match(/^[a-zA-Z0-9]+$/)) {errors.push("Nur Buchstaben und Zahlen sind erlaubt im Benutzernamen."); return res.render("register", {errors})}
  if (req.body.password.length == 1 && req.body.password.length == 2 && req.body.password.length == 3 && req.body.password.length == 4) {errors.push("Das Passwort muss mindestens 5 Zeichen lang sein."); return res.render("register", {errors})}

  // Versuchen, neuen Benutzer abzuspeichern -> falls Benutzername schon existiert -> Error
  -> springt zu catch {}
  try {
    // Passwort verschlüsseln
    const salt = bcrypt.genSaltSync(10)
    req.body.password = bcrypt.hashSync(req.body.password, salt)

    // Neuer Benutzer in Datenbank abspeichern
    const ourStatement = db.prepare("INSERT INTO users (email, username, password) VALUES (?, ?, ?)")
    const result = ourStatement.run(req.body.email, req.body.username, req.body.password)

    // Hinzugefügter Benutzer auslesen
    const lookupStatement = db.prepare("SELECT * FROM users WHERE ROWID = ?")
    const ourUser = lookupStatement.get(result.lastInsertRowid)

    // Server neustarten, um Datenbank zu aktualisieren
    let server = fork('server')
    server.on('close', (code) => {
      console.log("Restarted")
    })
  }
}

```

aaa

```

});

// Benutzer einloggen -> Cookie geben
const ourTokenValue = jwt.sign({exp: Math.floor(Date.now() / 1000) + 60 * 60 * 24, skyColor: "blue", userid: ourUser.id, username: ourUser.username}, process.env.JWTSECRET)

res.cookie("ourSimpleApp", ourTokenValue, {
  httpOnly: true,
  secure: true,
  sameSite: "strict",
  maxAge: 1000 * 60 * 60 * 24
})

res.redirect("/dashboard.ejs")
} catch {
  errors.push("Dieser Benutzername existiert schon."); return res.render("register", {errors});
}
})

// Loginbutton gedrückt
app.post("/login", (req, res) => {
  const errors = []

  // Prüfung, ob alle Felder ausgefüllt sind
  if (req.body.username.trim() == "") {errors.push("Bitte Benutzername eingeben."); return res.render("login", {errors});}
  if (req.body.password.trim() == "") {errors.push("Bitte Passwort eingeben."); return res.render("login", {errors});}

  // Sicherheitslücke schliessen und potentielle Errormeldungen vermeiden
  if (typeof req.body.username !== "string") {req.body.username = ""; errors.push("Der Benutzername muss Textformat haben."); return res.render("login", {errors});}
  if (typeof req.body.password !== "string") {req.body.password = ""; errors.push("Das Passwort muss Textformat haben."); return res.render("login", {errors});}

  // Eingegebenen Benutzernamen in Datenbank suchen
  const userInQuestionStatement = db.prepare("Select * FROM users WHERE username = ?")
  const userInQuestion = userInQuestionStatement.get(req.body.username)

  // Vorhandensein des Benutzernamens prüfen
  if (!userInQuestion) {
    errors.push("Invalidier Benutzername")
    return res.render("login", {errors})
  }

  // Richtigkeit des Passwort prüfen
  const matchOrNot = bcrypt.compareSync(req.body.password, userInQuestion.password)
  if (!matchOrNot) {
    errors.push("Invalides Passwort")
    return res.render("login", {errors})
  }

  // Benutzer einloggen -> Cookie geben

```

```

const ourTokenValue = jwt.sign({exp: Math.floor(Date.now() / 1000) + 60 * 60 * 24, skyColor: "blue", userid: userInQuestion.id, username: userInQuestion.username}, process.env.JWTSECRET)

res.cookie("ourSimpleApp", ourTokenValue, {
  httpOnly: true,
  secure: false,
  sameSite: "strict",
  maxAge: 1000 * 60 * 60 * 24
})

res.redirect("/")
})

// Emailadresse geändert
app.post("/change-email", (req, res) => {
  const errors = []

  // Versehentliche Leerschläge am Anfang oder am Ende entfernen
  req.body.newemail = req.body.newemail.trim()
  // Prüfung der neuen Emailadresse
  if (typeof req.body.newemail !== "string") {req.body.email = ""; errors.push("Die Emailadresse muss Textformat haben."); return res.render("change-email", {errors})}
  else if (!req.body.newemail) {errors.push("Es muss eine Emailadresse eingegeben werden."); return res.render("change-email", {errors})}

  // Alte Emailadresse durch Neue ersetzen in der Datenbank
  const changeEmailStatement = db.prepare("UPDATE users SET email = ? WHERE id = ?")
  changeEmailStatement.run(req.body.newemail, req.user.userid)

  // Server neu Starten, um Datenbank zu aktualisieren
  let server = fork('server')
  server.on('close', (code) => {
    console.log("Restarted")
  });

  return res.redirect("/dashboard.ejs")
})

// Passwort geändert
app.post("/change-password", (req, res) => {
  const errors = []

  // Prüfung des neuen Passworts
  if (req.body.newpassword !== req.body.newpasswordcheck) {
    errors.push("Passwörter stimmen nicht überein.")
    return res.render("change-password", {errors})
  } else if (req.body.newpassword == "") {
    errors.push("Es muss ein Passwort eingegeben werden.")
    return res.render("change-password", {errors})
  } else {
    // Neues Passwort verschlüsseln
    const salt = bcrypt.genSaltSync(10)
    req.body.newpassword = bcrypt.hashSync(req.body.newpassword, salt)
    // Altes Passwort durch Neues ersetzen

```

```

    const changeStatement = db.prepare("UPDATE users SET password = ? WHERE id = ?")
    changeStatement.run(req.body.newpassword, req.user.userid)

    return res.redirect("dashboard-admin.ejs")
  }
})

// Artikel geliket
app.post("/like-article/:id", mustBeLoggedIn, (req, res) => {
  // Auslesen, ob der Benutzer Artikel schon geliket hat
  const alreadyLikedStatement = db.prepare(`SELECT likedArticles FROM ${req.user.username}
WHERE likedArticles = ?`)
  const alreadyLiked = alreadyLikedStatement.get(req.params.id)
  const alreadyLikedCheckFalse = !alreadyLiked

  if (alreadyLikedCheckFalse) {
    // Like hinzufügen
    const addLike = db.prepare(`UPDATE articles SET likes = likes + 1 WHERE id = ?`)
    addLike.run(req.params.id)

    // Abspeichern, dass dieser Benutzer Artikel geliket hat
    saveLikingUser = db.prepare(`INSERT INTO ${req.user.username} (likedArticles) VALUES
(?)`)
    saveLikingUser.run(req.params.id)
  } else {
    // Like entfernen
    const removeLike = db.prepare(`UPDATE articles SET likes = likes - 1 WHERE id = ?`)
    removeLike.run(req.params.id)

    // Abspeichern, dass dieser Benutzer Artikel nicht mehr geliket hat
    removeLikingUser = db.prepare(`DELETE FROM ${req.user.username} WHERE likedArticles =
?`)
    removeLikingUser.run(req.params.id)
  }

  res.redirect(`/article/${req.params.id}`)
})

// Artikel veröffentlicht

// Artikelid zwischenspeichern
let realArticle = ""

app.post("/create-article", mustBeWriter, (req, res) => {
  // Prüfung des Artikels
  const errors = sharedArticleValidation(req)

  // Wenn Prüfung nicht erfüllt -> Error
  if (errors.length) {
    return res.render("create-article", { errors })
  }

  // Artikel in Datenbank speichern
  const ourStatement = db.prepare("INSERT INTO articles (title, article, likes, authorid,
createdDate) VALUES (?, ?, 0, ?, ?)")

```

```

const result = ourStatement.run(req.body.title, req.body.article, req.user.userid, new
Date().toISOString())

// Hinzugefügter Artikel auslesen
const getArticleStatement = db.prepare("SELECT * FROM articles WHERE ROWID = ?")
realArticle = getArticleStatement.get(result.lastInsertRowid)

res.render("upload-picture")
})

// Titelbild hochgeladen
app.post("/save-picture", mustBeWriter, thumbnailPicture, upload.single("picture") (req,
res) => {
  // Weiterleitung zum gerade erstellten Artikel
  return res.redirect(`/article/${realArticle.id}`)
})

// Artikel bearbeitet und veröffentlicht

app.post("/edit-article/:id", mustBeWriter, (req, res) => {
  // Artikel in Datenbank suchen
  const statement = db.prepare("SELECT * FROM articles WHERE id = ?")
  const article = statement.get(req.params.id)

  // Wenn Artikel nicht vorhanden -> Weiterleitung zur Hauptseite
  if (!article) {
    return res.redirect("/")
  }

  // Prüfung, ob Benutzer Autor dieses Artikels oder Admin ist
  if (article.authorid !== req.user.userid && req.user.userid !== 1) {
    return res.redirect("/")
  }

  // Prüfung des Artikels
  const errors = sharedArticleValidation(req)

  // Wenn Prüfung nicht erfüllt -> Error
  if (errors.length) {
    return res.render("edit-article", { errors })
  }

  // Alter Artikel durch Neuen ersetzen
  const updateStatement = db.prepare("UPDATE articles SET title = ?, article = ? WHERE id =
?")
  updateStatement.run(req.body.title, req.body.article, req.params.id)

  res.redirect(`/article/${req.params.id}`)
})

// Artikel gelöscht

app.post("/delete-article/:id", mustBeWriter, (req, res) => {
  // Artikel in Datenbank suchen
  const statement = db.prepare("SELECT * FROM articles WHERE id = ?")
  const article = statement.get(req.params.id)

```

eee

```

// Wenn Artikel nicht vorhanden -> Weiterleitung zur Hauptseite
if (!article) {
    return res.redirect("/")
}

// Prüfung, ob Benutzer Autor dieses Artikels oder Admin ist
if (article.authorid !== req.user.userid && req.user.userid !== 1) {
    return res.redirect("/")
}

// Artikel aus Datenbank löschen
const deleteStatement = db.prepare("DELETE FROM articles WHERE id = ?")
deleteStatement.run(req.params.id)

res.redirect("/")
})

// Logo hochgeladen
app.post("/upload-logo", mustBeWriter, logoPicture, upload.single("logo"), (req, res) => {
    return res.redirect("/")
})

// Echter Adminname geändert
app.post("/admin-visual-name", mustBeAdmin, (req, res) => {
    // Alter Name durch Neuen ersetzen
    const adminVisualNameStatement = db.prepare("UPDATE adminVisualName SET adminVisualName = ? WHERE id = 1")
    adminVisualNameStatement.run(req.body.adminVisualName)

    res.redirect("/dashboard-admin.ejs")
})

// Schreiber hinzugefügt
app.post("/add-writer", mustBeAdmin, (req, res) => {
    // Wenn Feld leer -> Seite lädt neu
    if (!req.body.addwriter) {
        return res.redirect("dashboard-admin.ejs")
    } else {
        // Schreiber in Datenbank abspeichern
        const ourStatement = db.prepare("INSERT INTO writers (writer) VALUES (?)")
        ourStatement.run(req.body.addwriter)

        res.redirect("dashboard-admin.ejs")
    }
})

// Schreiber entfernt
app.post("/remove-writer", mustBeAdmin, (req, res) => {
    // Schreiber aus Datenbank entfernen
    const deleteStatement = db.prepare("DELETE FROM writers WHERE writer = ?")
    deleteStatement.run(req.body.removewriter)

    res.redirect("dashboard-admin.ejs")
})

```

```
// Server starten  
app.listen(3000)
```



# **Maturitätsarbeit 2025 Ehrlichkeitserklärung**

Name: Bisang

Vorname: Jeremias

Klasse: 4bW

Titel der Arbeit: Wie man einen Blog programmiert

Hiermit erkläre ich, dass ich die vorliegende Arbeit nach den üblichen Gepflogenheiten des wissenschaftlichen Arbeitens verfasst habe, d.h. im Besonderen:

- Ich habe diese Arbeit selbständig verfasst.
- Alle Hilfsmittel (inklusive KI-Tools), die ich verwendet habe, sind angegeben.
- Alle wörtlichen und sinngemässen Übernahmen aus anderen Werken sind als solche gekennzeichnet.
- Personen, die einen wesentlichen Beitrag zu dieser Arbeit geleistet haben (Betreuer/-in ausgenommen), habe ich ebenfalls erwähnt.

## **Zutreffendes bitte ankreuzen**

- ☐ Ich stelle meine Arbeit zu Demonstrationszwecken der Mediothek der KBW zur Verfügung.
- ☐ Meine Arbeit darf nicht zu Demonstrationszwecken verwendet werden.

Datum \_\_\_\_\_ Unterschrift \_\_\_\_\_