

Dokumentation des Art-Gallery-Projekts

Gruppe 2

Modul: Webtechnologien

Gruppenmitglieder: Tim Fuchs, Arne Gutschick, Carlos Slaiwa, Tom Schröter, Kian van der Meer

Seminargruppen: IDP23, I23

Kursverantwortlicher: Peter Bernhardt

Sommersemester 2025

Berlin, den 22. Juni 2025

Inhaltsverzeichnis

1. Aufteilung der Use Cases	1
2. Softwaredesign	2
2.1 Genutzte Technologien	2
2.2 Architektur	2
2.3 Klassendiagramm	3
2.4 Sitemap	4
2.5 Verzeichnisstruktur	5
3. Datenbankdesign	6
3.1 Entity-Relationship-Diagramm	6
3.2 SQL-Abfrage mit besonderer Funktionalität	7
5. Erfahrungen und Probleme	8
6. Überblick über unsere Website	9
6.1 Home Page	9
6.2 Browse Results	9
6.3 Single Display	10
6.4 Registrierung	10

Abbildungsverzeichnis

Abbildung 1: UML Klassendiagramm	3
Abbildung 2: Sitemap	4
Abbildung 3: Verzeichnisstruktur	6
Abbildung 4: ER-Diagramm	6
Abbildung 5: Home Page	9
Abbildung 6: Browse Artworks Page	9
Abbildung 7: Display Single Artwork Page	10
Abbildung 8: Register Page	10

1. Aufteilung der Use Cases

Use Case	Verantwortlicher	Fortschritt	Probleme / Schwierigkeiten
01 - Site Design	Alle Gruppenmitglieder	Completed	Wechsel von Bootstrap 4 auf 5 erforderte komplette Neugestaltung der Formatierung
02 - Home Page	Tim Fuchs	Completed	Zufällige Bildauswahl aus der Datenbank führte zu fehlenden Bildern im Karussell
03 - Navigation	Tim Fuchs	Completed	
04 - About Us	Tom Schröter	Completed	
05 - Browse Artists	Arne Gutschick	Completed	
06 - Browse Artworks	Kian van der Meer	Completed	
07 - Browse Genres	Tom Schröter	Completed	
08 - Browse Subjects	Tom Schröter	Completed	
09 - Simple Search	Arne Gutschick	Completed	
10 - Search Results	Arne Gutschick	Completed	Dynamische Anzeige der Überschriften („Artist“ / „Artwork“) je nach Trefferanzahl
11 - Display Single Artist	Arne Gutschick	Completed	
12 - Display Single Artwork	Kian van der Meer	Completed	Modalgröße, Scrollverhalten und Bildanzeige
13 – Display Single Genre	Tim Fuchs	Completed	
14 – Display Single Subject	Tom Schröter	Completed	
15 – Missing Images	Arne Gutschick	Completed	
16 - Add A Review	Carlos Slaiwa	Completed	Darstellung der Bewertung mit Sternen
17 - Delete A Review	Carlos Slaiwa	Completed	
18 - Add To Favorites List	Kian van der Meer	Completed	Darstellung der Pop-Up-Messages
19 - View Favorites List	Kian van der Meer	Completed	
20 - Register User	Carlos Slaiwa	Completed	Prüfen der Zugriffsrechte, Korrekte Umsetzung des Sessionmanagements, Validierung von E-Mails mit Unicode-Zeichen
21 - Manage Users	Carlos Slaiwa	Completed	
22 - Login as User	Carlos Slaiwa	Completed	
23 - My Account	Tom Schröter	Completed	Inputvalidierung, Sicherheitsprüfungen beim Passwortwechsel
24 - Advanced Search	Tim Fuchs	Completed	Weiterleitung zu den Suchergebnissen
25 – Map to Museum	Tom Schröter	Completed	Einbindung der Karte in ein Bootstrap-Accordion

2. Softwaredesign

2.1 Genutzte Technologien

LAMP-Stack (Linux/Apache/MySQL/PHP):

- *XAMPP* als Entwicklungsumgebung, die den gesamten LAMP-Stack bereitstellt
- *Apache Webserver* für HTTP-Request-Handling
- *MySQL*-Datenbank für Datenpersistierung
- *PHP* als serverseitige Programmiersprache
- *PDO (PHP Data Objects)* für sichere Datenbankverbindungen

Frontend-Technologien:

- *HTML5* für die Website-Struktur
- *Bootstrap 5.3.7* als CSS-Framework für Design und UI-Komponenten
- *Vanilla JavaScript* für interaktive Funktionen (AJAX-Favoriten-Hinzufügen, Bootstrap Popup-Benachrichtigungen erzeugen)

2.2 Architektur

Der Code folgt einer abgewandelten MVC (Model-View-Controller) Architektur für eine saubere Trennung von Geschäftslogik, Präsentation und Datenverarbeitung.

Controller-Schicht

- *FrontController*, der für gesamtes Routing zuständig ist
- *BaseController* als gemeinsame Basis für wiederkehrende Funktionalitäten, wie das Rendern von Views
- Spezialisierte Controller für verschiedene Bereiche (Artists, Artworks, Genres, Authentication, etc.)
- *ErrorController* für dynamische Fehlerbehandlung

Model-Schicht (vom typischen MVC abgewandelt)

- Keine dedizierten Model für jeden View, sondern „\$data“-Array, das mit notwendigen Daten gefüllt wird
- Entität-Repositories für strukturierten/vereinheitlichten Datenbankzugriff
- DTO (Data Transfer Objects) für typisierte Datenübertragung zwischen Schichten
- Dedizierte Klassen für Entitäten (Artist, Artwork, Customer, Review, etc.)

View-Schicht

- Template-basierte Views mit wiederverwendbaren Komponenten
- Layoutsystem mit *main.php* als Master-Layout (beinhaltet Navigationsleiste, Footer, Benachrichtigungen, etc.)
- Modulare UI-Komponenten (Navbar, Footer, Cards, etc.)

Zusätzlich

- Top-Level Fehlerbehandlung in „index.php“
- Environment Konfiguration mithilfe von „.env“-Datei
- Session-basierte Authentifizierung mit Rollen-Management (Admin/User)

2.3 Klassendiagramm

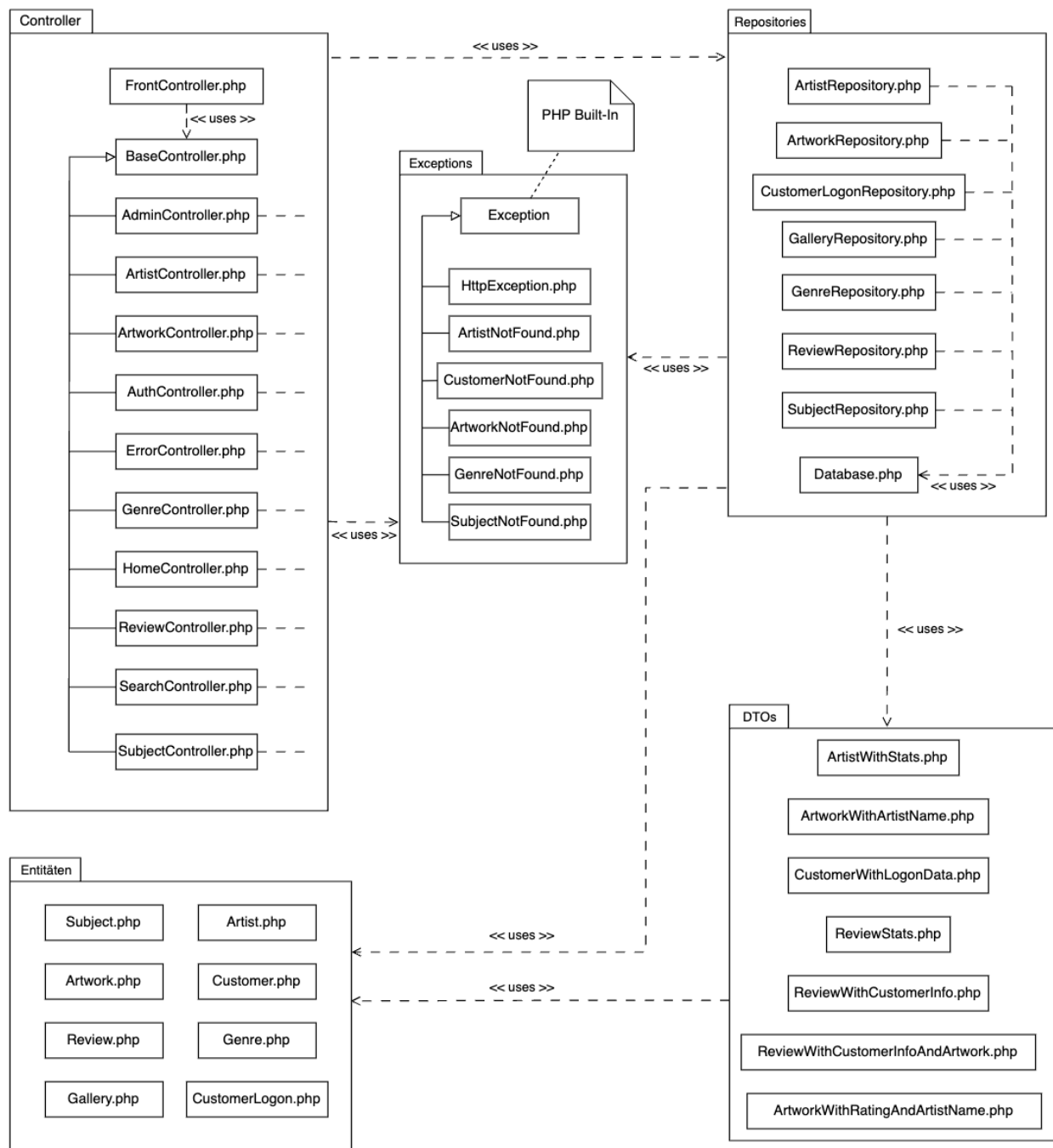


Abbildung 1: UML Klassendiagramm

2.4 Sitemap

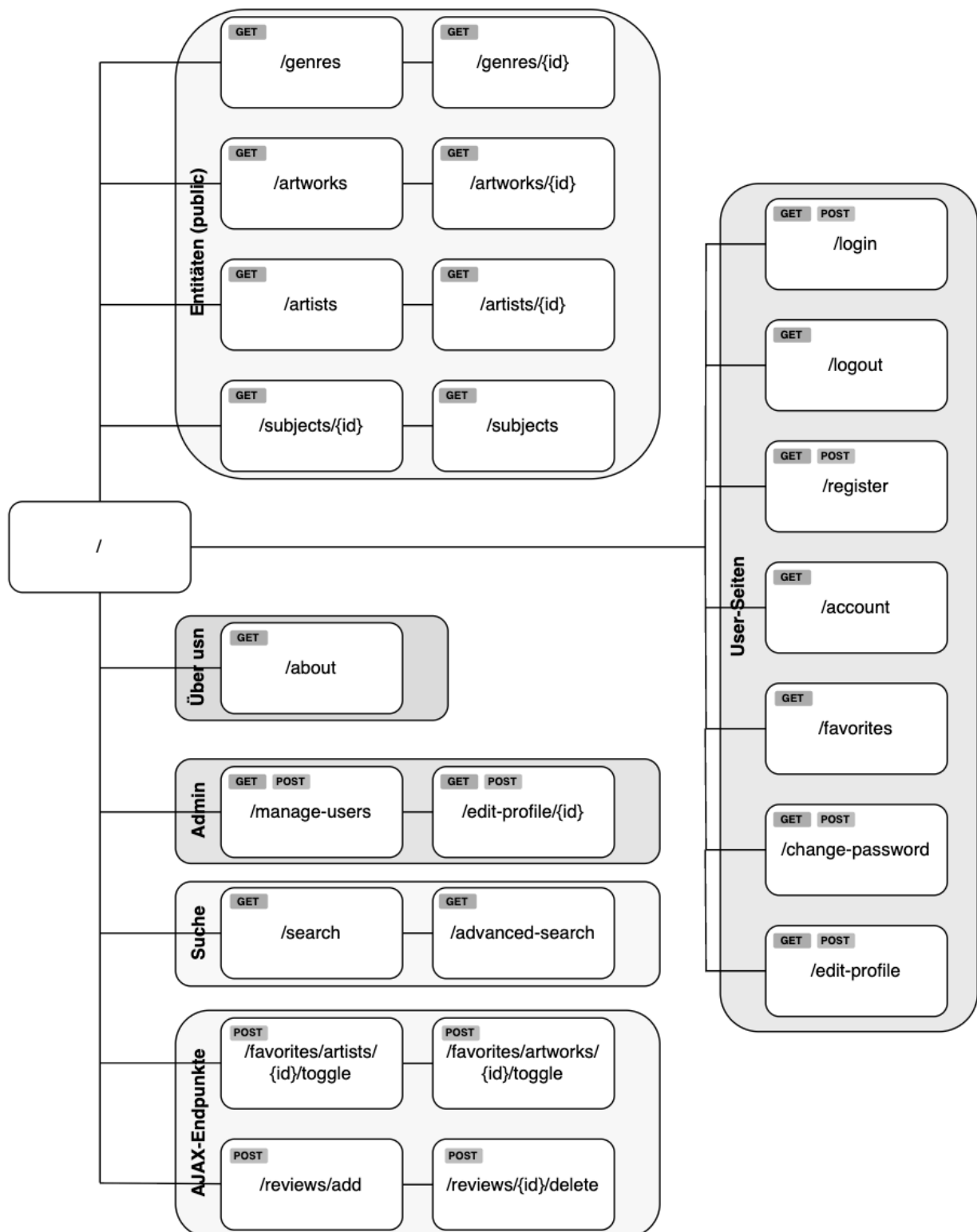


Abbildung 2: Sitemap

2.5 Verzeichnisstruktur

```
/
├── index.php                # Einstiegspunkt der Anwendung
├── Database.php             # Datenbankverbindungsklasse
├── env.php                  # Environment-Konfiguration (Datenbank-Credentials)
├── README.md                # Einfache Projekt Doku (Setup-Anleitung)

├── assets/                  # Statische Ressourcen (CSS, JS, Bootstrap, Bilder, Fonts, SVGs, Screenshots)

├── classes/                 # Entitäts-Klassen (Artist, Artwork, Customer, Review, Genre, Subject, etc.)

├── controllers/             # Controller-Schicht
│   ├── BaseController.php  # Beinhaltet grundlegende Funktionalitäten,
│   │                       # wie das Rendern von Views, Weiterleiten, ...
│   ├── AdminController.php # AdminController behandelt Admin-Funktionalitäten, wie Deaktivieren von Usern
│   ├── AuthController.php  # AuthController ist für Login/Register/Logout,
│   │                       # Account Ansicht, Profil Bearbeitung, Favoriten zuständig
│   ├── ErrorController.php # ErrorController behandelt Fehler und zeigt entsprechende Error Seiten an
│   ├── ArtworkController.php # Anzeigen von Allen und spezifischen Artists
│   └── ...

├── repositories/           # Datenabfrage-Schicht (Artist, Artwork, Customer, Review, etc.)
│   └── ...

├── dtos/                   # Data Transfer Objects
│   ├── ArtistWithStats.php # Künstler mit Statistik (Anzahl der Reviews)
│   ├── ArtworkWithArtistName.php # Kunstwerk mit Künstlernamen
│   ├── ArtworkWithRatingAndArtistName.php # Kunstwerk mit Bewertung und Künstlernamen
│   ├── CustomerWithLogonData.php # Kundendaten mit dessen Login-Daten (ohne Passwort)
│   ├── ReviewWithCustomerInfo.php # Bewertung mit Kundeninfo
│   ├── ReviewWithCustomerInfoAndArtwork.php # Bewertung mit Kunde und Kunstwerk
│   └── ReviewWithStats.php # Review Statistiken (Durchschnitt, Anzahl) für ein Kunstwerk

├── views/                  # View-Schicht
│   ├── layouts/
│   │   └── main.php        # Master-Layout-Template
│   ├── home/
│   │   ├── index.php       # Hauptseite
│   │   └── about.php        # Über uns
│   ├── artists/
│   │   ├── index.php        # Künstler Übersicht
│   │   └── show.php         # Einzelner Künstler
│   ├── artworks/           # Kunstwerk-Views (index.php, show.php)
│   ├── genres/             # Genre-Views (index.php, show.php)
│   ├── subjects/           # Subject-Views (index.php, show.php)
│   ├── auth/
│   │   ├── login.php        # Login-Seite
│   │   ├── register.php     # Registrierungs-Seite
│   │   ├── account.php      # Account-Übersicht
│   │   ├── edit-profile.php  # Profil bearbeiten
│   │   ├── change-password.php # Passwort ändern
│   │   └── favorites.php     # Favoriten-Seite
│   ├── search/
│   │   ├── index.php        # Einfache Suche
│   │   └── advanced.php      # Erweiterte Suche
│   ├── admin/
│   │   └── manage-users.php  # Benutzerverwaltung
│   └── errors/
│       └── error.php         # Allgemeine (dynamische) Fehlerseite
```

components/	# Wiederverwendbare UI-Komponenten
├─ add-to-favorites-button.php	# Favoriten-Button-Komponente (z.B. auf Kunstwerkseite, Künstlerseite)
├─ footer.php	# Footer
├─ head.php	# HTML-Head
├─ most-reviewed-artists.php	# Meist-bewertete Künstler (für Startseite)
├─ navbar.php	# Navigationsleiste
└─ ...	
router/	
├─ FrontController.php	# Haupt-Router (leitet auf spezifische Controller weiter)
exceptions/	
├─ HttpException.php	# Eigene HTTP-Fehler-Klasse
sql/	# Datenbankskripte
├─ db_setup.sql	# Datenbanksetup (isAdmin Spalte, Admin User)
└─ hash_passwords.php	# Passwörter von existierenden Nutzern hashen

Abbildung 3: Verzeichnisstruktur

3. Datenbankdesign

3.1 Entity-Relationship-Diagramm



Abbildung 4: ER-Diagramm

3.2 SQL-Abfrage mit besonderer Funktionalität

Da Nutzer bei der erweiterten Suche verschiedene Suchkriterien angeben können, beispielsweise die Nationalität eines Künstlers oder das Genre eines Kunstwerks, muss die zugrundeliegende SQL-Abfrage dynamisch aufgebaut werden, um optionale Filterkriterien zu berücksichtigen. Die Abfrage beginnt mit WHERE 1=1. Dieser Platzhalter dient dazu, das einfache Anfügen weiterer Bedingungen zu ermöglichen und sicherzustellen, dass bei nicht angegebenen Suchkriterien alle Zeilen ausgegeben werden. Abhängig von der Benutzereingabe werden dann Bedingungen zu Name, Nationalität sowie einem Geburtsjahresbereich ergänzt. Parameter, die nicht ausgefüllt wurden, werden so bewusst nicht in der SQL-Abfrage integriert. Gemäß des *Search Results* Use Cases werden die ausgewählten Künstler zum Schluss alphabetisch nach Nachnamen sortiert.

Die Abfrage für die erweiterte Suche nach Kunstwerken folgt demselben Prinzip.

```
// Initialize SQL query with WHERE 1=1 to simplify appending conditions
$sql = "SELECT * FROM artists WHERE 1=1";
$params = [];

// Append name condition if provided, filtering by first and last name
if (!empty($name)) {
    $nameParts = preg_split('/\s+/', trim($name));
    $i = 0;
    foreach ($nameParts as $part) {
        $sql .= " AND (FirstName LIKE :namePart{$i} OR LastName LIKE
:namePart{$i})";
        $params["namePart{$i}"] = '%' . $part . '%';
        $i++;
    }
}

// Append nationality condition if provided
if (!empty($nationality)) {
    $sql .= " AND Nationality = :nationality";
    $params['nationality'] = $nationality;
}

// Append start year range condition if provided, NULL is allowed
if (!empty($startYear)) {
    $sql .= " AND (YearOfBirth >= :startYear OR YearOfBirth IS NULL)";
    $params['startYear'] = (int) $startYear;
}

// Append end year range condition if provided, NULL is allowed
if (!empty($endYear)) {
    $sql .= " AND (YearOfBirth <= :endYear OR YearOfBirth IS NULL)";
    $params['endYear'] = (int) $endYear;
}

// Ordering appended at the end
$sql .= " ORDER BY LastName " . ($sortDesc ? "DESC" : "ASC");
```

4. Arbeitsverteilung im Team

Um eine effiziente Projektumsetzung sicherzustellen, lag unser Hauptziel bei der Arbeitsverteilung darin, die Aufgaben gleichmäßig zu verteilen und möglichst wenig Abhängigkeiten untereinander zu kreieren.

Während das grundlegende Seitenlayout und Design einheitlich von allen Gruppenmitgliedern in den einzelnen Use Cases umgesetzt wurde, übernahm jedes Teammitglied jeweils spezifische Funktionsbereiche. Tim war vor allem für die Startseite und die Navigation zuständig. Tom entwickelte unter anderem mehrere Browse-Ansichten (Genres, Subjects) sowie die Benutzerkontoverwaltung. Arne war für die Suchfunktionalitäten, darunter die einfache Suche und die Darstellung der Suchergebnisse, und das Anzeigen der Künstler verantwortlich. Kian konzentrierte sich auf die Anzeige der Kunstwerke sowie die Favoritenfunktionen. Carlos war primär für Benutzerfunktionen wie das Hinzufügen und Löschen von Reviews, die Registrierung und den Login zuständig.

Durch das Aufteilen der Use Cases in weitestgehend voneinander unabhängige Funktionsbereiche konnten wir unseren Zeitplan individuell gestalten, doppelte Implementierungen vermeiden und Konflikte beim Zusammentragen der Ergebnisse weitgehend verhindern. Für das allgemeine Design der Website entschieden wir uns gemeinsam am Anfang des Projekts, sodass sich im weiteren Verlauf alle bei der Umsetzung ihrer einzelnen Seiten an den bereits bestehenden Komponenten orientieren konnten.

5. Erfahrungen und Probleme

Während der Projektarbeit konnten wir wertvolle Erfahrungen im Umgang mit *php* und *Bootstrap* sammeln. Die eigenständige Umsetzung der geforderten Funktionalitäten und das Zusammenarbeiten in einem größeren Team ohne enge Supervision durch Lehrkräfte bzw. Projektleiter hat unser Zeitmanagement, unsere Teamkommunikation und unsere Eigeninitiative gefördert. Dabei haben wir gelernt, Aufgaben sinnvoll zu verteilen, Abhängigkeiten frühzeitig zu erkennen und gemeinsam eigene Standards und Regeln für Design, Codequalität und den allgemeinen Umgang im Projekt miteinander festzulegen. Insbesondere in Bezug auf das Zeitmanagement haben wir die Erfahrung gemacht, dass das Setzen von Meilensteinen und Deadlines ein wichtiger Aspekt ist, um die Motivation im Team hochzuhalten und den Fortschritt des Projekts kontinuierlich voranzutreiben.

Ein zentrales Problem im Projektverlauf war, dass wir zu Beginn kein einheitliches Software-Design abgestimmt und klar kommuniziert hatten. Dadurch war unsere Projektstruktur von Anfang an nicht optimal, was uns jedoch erst nach einer erheblichen Anzahl an implementierten Use Cases auffiel. Die Einführung des MVC-Designprinzips gegen Ende des Projekts machte es demnach notwendig, bereits implementierte Funktionen in eine neue Struktur zu überführen. Auch das Erstellen eines skalierbaren Routing-Systems stellte sich unter diesen Bedingungen als schwierig heraus, da viele Komponenten nachträglich angepasst werden mussten. Außerdem hatten wir uns dazu entschieden, Bootstrap 4 zu benutzen. Dies führte jedoch zu Problemen bei der Implementierung bestimmter Elemente ohne die Nutzung von JavaScript, bspw. der Navigationsleiste und dem Karussell auf der Home Page. Aus diesem Grund mussten wir kurzfristig auf Bootstrap 5 wechseln und dabei jegliche bereits vorhandene Textformatierungen mit der neuen Version kompatibilisieren. Diese Umstellungen waren zwar zeitaufwendig, erwiesen sich im Nachhinein jedoch als wichtig, um die Wartbarkeit und Erweiterbarkeit der Anwendung sicherzustellen.

Andere typische Probleme wie inkonsistente Formatierungen und fehlerbehafteten Code, konnten wir durch regelmäßige Code Reviews und eine offene, teamorientierte Kommunikation größtenteils vermeiden.

Den Zeitmangel gegen Ende des Projekts konnten wir gut durch unsere frühzeitige Initiative kompensieren.

6. Überblick über unsere Website

6.1 Home Page

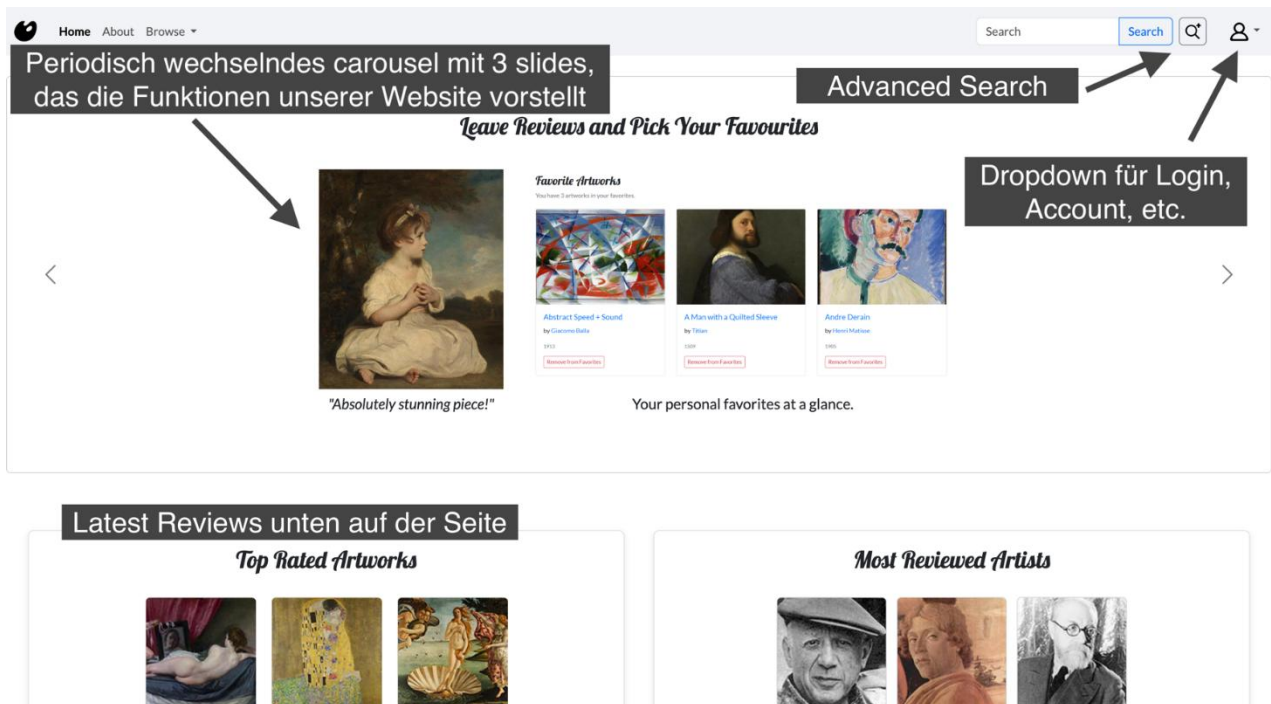


Abbildung 5: Home Page

6.2 Browse Results

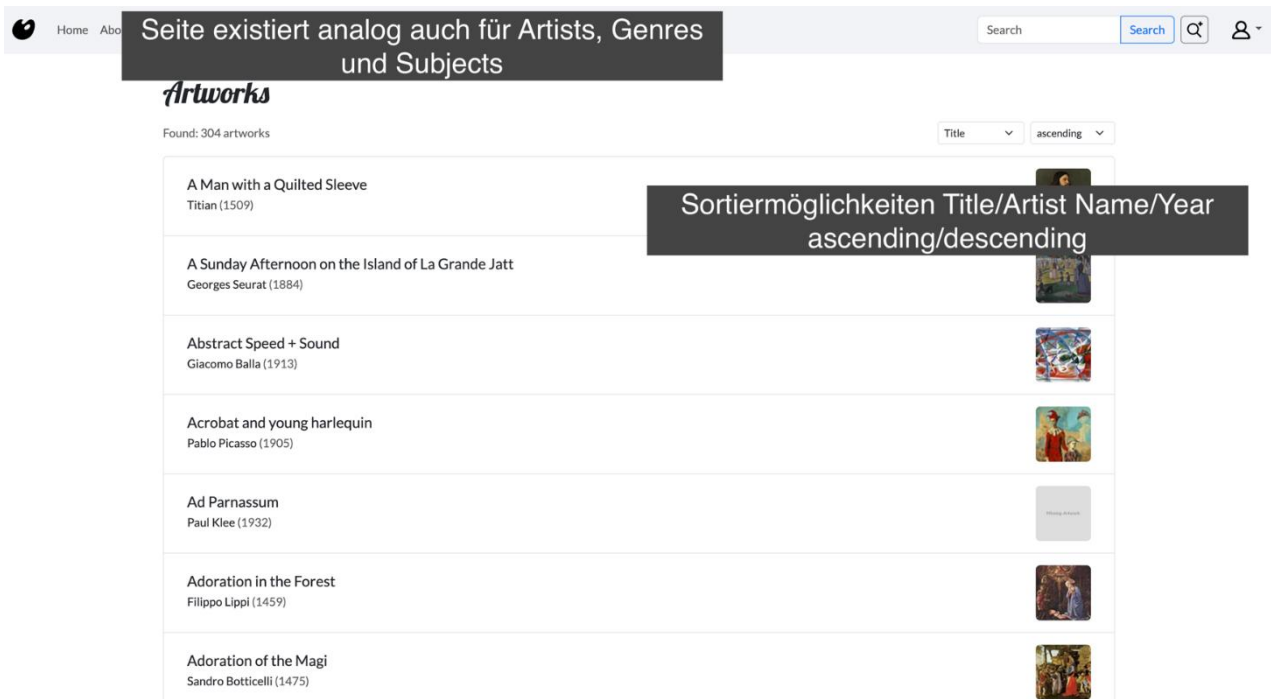



Abbildung 6: Browse Artworks Page

6.3 Single Display

Seite existiert analog auch für Artists, Genres und Subjects

Adoration of the Magi



by **Sandro Botticelli**

No reviews yet

Add to Favorites

Artwork Details		More Info
Year:	1475	
Medium:	Tempera on panel	
Dimensions:	111 × 134 cm	
Genres:	Renaissance	
Subjects:	People, Religion	
Google Arts:	View on Google Arts & Culture	

Description

The *Adoration of the Magi* is a painting by the Italian Renaissance master Sandro Botticelli, dating from 1475 or 1476. It is housed in the Uffizi of Florence. Botticelli was commissioned to paint at least seven versions of The Adoration of the Magi. The Adoration of the Magi theme was popular in the Renaissance Florence. The work was commissioned by Gaspare di Zanobi del Lama, a banker of humble origins and dubious morality connected to the House of Medici, for his chapel in the church of Santa Maria Novella (now destroyed). In the scene are present numerous characters among which are several members of the Medici family: Cosimo de' Medici (the Magus kneeling in front of the Virgin, described by Vasari as "the finest of all that are now extant for its life and vigour"), his sons Piero (the second Magus kneeling in the centre with the red mantle) and Giovanni (the third Magus), and his grandsons Giuliano and Lorenzo. The three Medici portrayed as Magi were all dead at the time the picture was painted, and Florence was effectively ruled by Lorenzo.

HTML-Formatierung der Beschreibung aus der Datenbank

Gallery

Abbildung 7: Display Single Artwork Page

6.4 Registrierung

HomeAboutBrowse

Search

Search

Q

Person

Register

Pflichtfelder mit * markiert

First Name

Last Name*

First Name

Last Name

Address*

City*

Address

City

Region

Country*

Postal

Region

Country

Postal Code

Phone

Email*

Phone

Email

Username*

Username

Passwortwiederholung zur Vermeidung von Tippfehlern

Repeat Password*

Repeat Password

Your password must be at least 6 characters, contain an uppercase letter, a digit, and a special character.

Register

weitgehende Frontend- und Backend-Checks der Eingaben

© 2025 Programmers Having Pizza GmbH – About Us

Abbildung 8: Register Page

Anhang

Selbstständigkeitserklärung

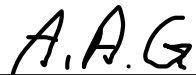
Hiermit erklären wir, dass wir wörtlich oder sinngemäß aus anderen Werken – dazu gehören auch Internetquellen – übernommene Inhalte als solche kenntlich gemacht und die entsprechenden Quellen angegeben habe. Wir willigen ein, dass unsere Arbeit mittels einer Software auf Plagiate überprüft werden kann. Uns ist bekannt, dass es sich bei der Abgabe eines Plagiats um ein schweres akademisches Fehlverhalten handelt und dass Täuschungen nach der für uns gültigen Studien- und Prüfungsordnung geahndet werden. Die vorliegende Arbeit haben wir selbstständig und ohne jede unerlaubte Hilfe konzipiert und angefertigt und keine anderen als die erlaubten und im Moodle-Kurs angegebenen Quellen und Hilfsmittel benutzt, sowie eigene Textbausteine keiner anderen Person zur Verfügung gestellt. Uns ist bewusst, dass wir Autoren der vorliegenden Arbeit sind und volle Verantwortung für den Text tragen. Zusätzlich versichern wir, dass wir IT-gestützte oder auf künstlicher Intelligenz (KI) basierende Schreibwerkzeuge nur in Absprache mit der Betreuungsperson verwendet haben. Dabei stand unsere eigene geistige Leistung im Vordergrund, und wir haben jederzeit den Prozess steuernd gearbeitet. Sofern die zuständigen Prüfenden bis zum Zeitpunkt der Ausgabe der Aufgabenstellung konkrete KI-gestützte Schreibwerkzeuge ausdrücklich als nicht anzeigepflichtig benennen, müssen diese nicht aufgeführt werden. Wir willigen ein, dass unsere Arbeit mittels einer Software auf KI-Textbausteine überprüft werden kann.



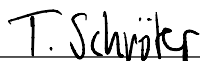
Carlos Slaiwa



Tim Fuchs



Arne Gutschick



Tom Schröter



Kian van der Meer