

3D Portfolio Gallery

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für IT - Medientechnik

Eingereicht von:

Lorenz Litzlbauer

Fabian Maar

Betreuer:

Patricia Engleitner

Natascha Rammelmüller

Leonding, April 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022

Lorenz Litzlbauer

Leonding, April 2022

Fabian Maar

Abstract [L]

3D Portfolio Gallery is a web application that was developed by Lorenz Litzlbauer and Fabian Maar as part of their diploma thesis. 3D Portfolio Gallery aims to enable designers to share their design portfolio with the world in an innovative way, all in a virtual three-dimensional space. The designers can use the 3D Portfolio Gallery to create a three-dimensional portfolio from their own media (film, photo or 3D data).



3D Portfolio Gallery is offered as an Single-Page-Application on the web and is accessible via a web link. 3D Portfolio Gallery offers a simple configuration process for this by selecting a gallery from several predefined floor plans. In a further step, the exhibits are placed either automatically or manually and provided with additional information.

Visitors can move freely through the three-dimensional web exhibition and view the exhibits with different types of interaction.

The following frameworks are used for implementation in the frontend: Angular for the Single-Page-Application and Three.js for the 3D display. Json-Web-Token is used in the frontend access control and user identification in the browser.

Zusammenfassung [L]

3D Portfolio Gallery ist eine Webapplikation, die von Lorenz Litzlbauer und Fabian Maar im Rahmen der Diplomarbeit entwickelt wurde. 3D Portfolio Gallery will Designer*innen ermöglichen, ihr Design-Portfolio auf eine innovative Art und Weise mit der Welt zu teilen und auf eine innovative Art und Weise in Form einer virtuellen dreidimensionalen Ausstellung mit der Welt zu teilen. Die Designer*innen können mithilfe von 3D Portfolio Gallery aus eigenen Medien (Film, Foto oder 3D-Daten) ein dreidimensionales Portfolio erstellen.



3D Portfolio Gallery wird als eine Single-Page-Application im Web angeboten und ist über einen Web-Link erreichbar. Das Tool unterstützt mit einem einfachen Konfigurationsprozess den User beim Erstellen der Ausstellung, indem aus mehreren vordefinierten Grundrissen eine Gallery ausgewählt wird. Anschließend können die Ausstellungsstücke entweder automatisch oder manuell platziert und mit Zusatzinformationen versehen werden.

Besucher der 3D-Gallery können sich in der dreidimensionalen Webausstellung frei bewegen und die Ausstellungsstücke mit verschiedenen Interaktionsarten anschauen.

Für die Umsetzung im Frontend werden folgende Frameworks verwendet: Angular für die Single-Page-Application und Three.js für die 3D-Darstellung. In der Frontend-Zugriffskontrolle und -Useridentifikation im Browser wird ein Json-Web-Token verwendet.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Ausgangslage [L]	1
1.2 Themenstellungen [L]	1
1.3 Zielsetzung [L]	2
1.4 Aufgabendifferenzierung [L]	2
2 Architektur	3
2.1 Client-Server-Architektur [L]	3
2.2 Technologie-Stack Client [L]	3
2.3 Technologiestack Backend [Team]	15
3 Werkzeuge	17
3.1 IDE	17
3.2 UI Prototyping Werkzeug [L]	17
3.3 3D Modellierung	19
3.4 Package Manager [L]	19
4 Planung	21
4.1 Ideenfindung [M]	21
4.2 User Stories [L]	22
5 Umsetzung	23
5.1 Design [L]	23
5.2 Umsetzung der Landingpage [L]	25
5.3 Web Gallery Prototyp [L]	35
5.4 Fertige Landingpage [L][M]	42
5.5 Userverwaltung [L]	44
5.6 Content Creation [L]	49
5.7 Interaktions-Seite [M]	55

5.8 Such-Seite [M]	67
6 Zusammenfassung [L]	70
6.1 Probleme und Lösungsstrategie	70
6.2 Zielerreichung [Team]	71
6.3 Erweiterungsvorschläge [Team]	72
7 Glossar	73
Literaturverzeichnis	VI
Abbildungsverzeichnis	XII
Tabellenverzeichnis	XIII
Quellcodeverzeichnis	XIV
Anhang	XVI

1 Einleitung

1.1 Ausgangslage [L]

Ausgangslage ist es, dass virtuelle Welten bzw. Realitäten sich derzeit einer großen Beliebtheit erfreuen. Viele Unternehmen wie *IKEA* oder *Audi* setzen auf virtuelle begehbarer Ausstellungsräume, um ihre Produkte interaktiv zu präsentieren. [1]

Wir wollten einen Prototypen einer webbasierten virtuellen Ausstellung für Designer*innen erstellen. In dieser Ausstellung sollen dreidimensionale Objekte, Fotos und Videos präsentiert werden können. Um unser Projekt weiter zu konkretisieren, überlegten wir uns die Anforderung, die sich an einen* eine Designer*in von digitalen 3D-Objekten bei der Präsentation von 3D-Objekten stellt.

Die virtuelle Präsentation in einer 3D-Portfolio-Gallery ermöglicht beispielhaft zusätzliche Funktionen:

- Die Objekte besitzen eine vordefinierte genormte Größe. Eine Verkleinerung oder Vergrößerung erzielt eine gesteigerte optische Wirkung oder eine verminderte optische Wirkung.
- Die Gestaltung und Ausstattung von virtuellen Ausstellungsräumen ist mit wenigen Mausklicken zu ändern.

1.2 Themenstellungen [L]

Die Diplomarbeit untersucht aktuelle Website-Technologien zur Darstellung von 3D-Modellen in Web-Browsern. Dem Anwender soll eine interaktive grafische Benutzeroberfläche zur Verwaltung und Gestaltung einer 3D-Gallery angeboten werden. Im Frontend ist es Ziel, eine möglichst ansprechende Designsprache durch die Anwendung beizubehalten. Daraus definieren sich folgende Anliegen:

- Auswahl und Entwicklung eines Designs für den Prototypen
- Bereitstellung einer Suchfunktion zum Auffinden von Ausstellungen

- Die Schaffung eines Konfigurationsmoduls für die Erstellung einer 3D-Gallerie. Die Konfiguration beinhaltet die Auswahl des Grundrisses der Gallery, die Platzierung der Objekte und den Upload von Multimediadaten.

1.3 Zielsetzung [L]

In der vorliegenden Diplomarbeit ist das Hauptaugenmerk auf die technische Umsetzung gerichtet. Ziel ist ein funktionierender Prototyp.

Ein funktionierender Prototyp erfordert die Erstellung eines Frontends für die Benutzerinteraktion, eines Backends zur Datenhaltung und weiters eine Benutzerverwaltung und Benutzerauthentifikation. Die aktuell verfügbaren Softwareframeworks werden nach den Kriterien Verfügbarkeit und Einsetzbarkeit und Stand der Technik zur Erstellung des Prototypens untersucht und ausgewählt. Vorgegeben ist eine Client-Server-Architektur, da das fertige Softwareprodukt über einen Web-Browser im Internet angeboten wird und die Daten dezentral abgespeichert werden.

1.4 Aufgabendifferenzierung [L]

Das Projekt wird von zwei Personen bearbeitet. Die Projektleitung liegt bei Lorenz Litzlbauer.

1.4.1 Lorenz [L]

Im Tätigkeitsbereich von Lorenz Litzlbauer wurde in der Softwareentwicklung das Usermanagement, das Tokenmanagement, die Content Creation, das UI-Design, die Platzierung von Objekten im Ausstellungsraum erstellt.

1.4.2 Fabian [M]

Fabian Maar war in der Softwareentwicklung verantwortlich für die Such-Unterseite, Besucheransicht der Ausstellung, die Interaktion mit den Ausstellungsstücken, die Navigation im dreidimensionalen Raum und das UI-Design.

2 Architektur

2.1 Client-Server-Architektur [L]

Die technische Umsetzung erfolgte im Frontend als SPA (Single Page Application) in einem Browser. Die Applikation wird auf einem Anwendungsserver gehostet und als HTML-Page geladen. Das Backend kümmert sich um die Datenspeicherung. Die Services am Backendserver werden als REST-Services angeboten.

2.1.1 Grundkriterien [M]

Die primären Kriterien welche an die Frontend-Technologien gestellt wurden, waren: eine einfache Integration, gute Dokumentation und ausreichende Funktionalitäten für die Webentwicklung mit 3D-Modellen besitzen. Ein weiteres Kriterium war, dass die Software eine übersichtliche Struktur besitzt und diese sich ebenfalls gut warten lässt.

2.2 Technologie-Stack Client [L]

Die Single-Page-Application verwendet Angular und Erweiterungen, die das Angular-Framework verwendet und bereitstellt.

Die folgenden Unterabschnitten setzen sich mit den Technologien, auf die Angular aufgebaut ist, auseinander und die Erweiterungen des Frontends.

2.2.1 Vorteile von Angular [M]

Folgende Aspekte sprechen für die Nutzung von Angular in diesem Projekt:

- Die Möglichkeit der komponentenbasiertes Programmierung um komplexe Benutzeroberflächen übersichtlich zu unterteilen (siehe 5.2.1)
- Die einfache und übersichtliche Einbindung von Libraries durch ngModules (siehe 2.2.1)

- Isolierung der Logik mit der Benutzeroberfläche durch die Model-View-Controller Architektur und die Model-View-ViewModel Architektur (siehe 2)
- Dependency Injection für die Verbindung von Front-End zum Back-End (siehe 5.2.3)

Angular [M]

Angular ist ein Framework für Webapplikationen, das auf der Programmiersprache Typescript basiert. Es ist eines der renommiertesten Frameworks zur Front-End-Entwicklung, wird als Open-Source-Software zur Verfügung gestellt und besitzt somit eine große Community. So zählte es 2022 zu den zweitbeliebtesten Front-End Framework [2] und wird von Milliardenkonzernen wie zum Beispiel Google oder PayPal verwendet. [3]



Abbildung 1: Angular Logo

Die große Stärke von Angular ist die Erstellung von Single-Page-Applikationen, da es eine komponentenbasierte Struktur besitzt. Das bedeutet, der Code ist wiederverwendbar und verkapselt. Komplexe Logiken werden auf ihre Grundelemente reduziert und beeinflussen sich nicht gegenseitig. [4]

Angular basiert auf dem Konzept des Model-View-Controller- (MVC) und Model-View-ViewModel (MVVM) Architecture Patterns. Diese Patterns werden verwendet, um die Logik von der Benutzeroberfläche zu trennen und komplexe Aufgaben einfacher zu bewältigen. Dies wird in der Fachsprache auch oft als *Separation of Concerns* bezeichnet. [5]

Architecture Patterns sind Muster, die verwendet werden, um

- eine Software-Anwendung zu strukturieren.
- die Redundanz von wiederholenden Code-Teile zu vermeiden.
- immer wiederkehrende Probleme durch eine einmalige Lösungen zu beheben.
- die Wartung und Testung zu erleichtern.
- Änderungen am Umfang und der Größe der Applikation leichter handhaben zu können. [6, 7, 8]

Model-View-Controller (MVC) Die *Separation of Concerns* wird bei diesem Pattern durch die Aufteilung der Applikation in Model, View und Controller realisiert. Dadurch ist es möglich, sich auf einen Aspekt der Implementierung zu fokussieren.

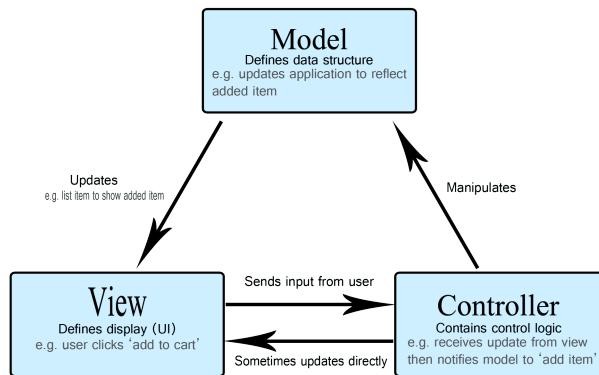


Abbildung 2: Die Model-View-Controller Pattern [6]

Welche Daten eine Applikation beinhalten soll, wird über das Model geregelt. Falls sich das Model ändert, werden die Benutzeroberfläche und manchmal auch der Controller entsprechend geändert. In Angular sind diese Modelle mit den Interfaces (siehe Interfaces 5.2.3) zu vergleichen.

View ist die Benutzeroberfläche des Programms. Der*die Benutzer*in kann mit dieser interagieren und diese verändern. Sie wird aus den Daten des Models erstellt und definiert. In Angular kann diese View mit dem HTML-Template einer Komponente verglichen werden.

Im Controller werden die Eingaben der Benutzer*innen verarbeitet und die betroffenen Model- und View-Komponenten beeinflusst und verändert. Auch ist es möglich zu bestimmen, welche Views verändert werden sollen um die Daten gegebenenfalls in unterschiedlichen Formaten anzuzeigen. Die TypeScript-Files in Angular sind vergleichbar mit diesen Controllern. [8]

Model-View-ViewModel (MVVM) Das MVVM basiert auf dem Konzept des MVC-Patterns und realisiert die *Separation of Concerns* durch die View, ViewModel und Model Komponenten.

View funktioniert hierbei ähnlich wie beim MVC-Pattern, jedoch ist es möglich, dass eine *View* auch Logik enthält, die Änderungen am Aussehen durchführt. Daher ist das

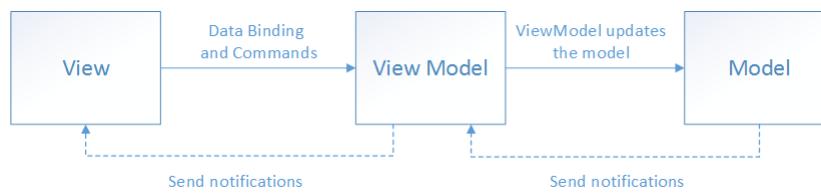


Abbildung 3: Die Model-View-ViewModel Pattern [7]

HTML-Template einer Angular-Komponente noch stärker vergleichbar zu dem Konzept einer View aus dem MVVM-Pattern als zu einer View aus dem MVC-Pattern.

Im ViewModel wird die Funktionalität der Benutzeroberfläche bestimmt. Dabei informiert das ViewModel die View über Änderungen im Model und liefert es mit Daten. Dieser Vorgang beschreibt das Konzept des Data-Bindings (siehe 5.2.1) in Angular.

Das Model funktioniert im Grunde gleich wie beim MVC-Pattern, nur wird hierbei nicht die View, sondern das ViewModel über Änderungen informiert. Somit ist das ViewModel das Mittelstück zwischen View und Model.

Angular nutzt hierbei eine Mischung aus beiden Architecture-Patterns. Diese Architektur wird durch das Konzept von Komponenten realisiert. In den nächsten Kapiteln, besonders im Kapitel Umsetzung der Landingpage 5.2 wird näher darauf eingegangen. Dabei werden die Parallelen der Patterns erkennbar. [7]

NgModules [M] *NgModules* erleichtern die Einbindung von Libraries. Da keine externen Dateien mühsam heruntergeladen und eingebunden werden müssen und jeder Import-Prozess durch wenige Zeilen Code erfolgt, wird Zeit beim Entwickeln gespart. Alle Importe werden in chronologischer Reihenfolge gelistet, wodurch ein guter Gesamtüberblick geliefert wird und schnell überflüssige Imports entfernt werden können. Die kontinuierliche Erweiterung und die Unterstützung von Third-Party-Libraries, die unter anderem für die 3D-Darstellung verwendet werden, wird ebenfalls von diesen ngModulen ermöglicht. Die Angular-Applikation wird hierbei organisiert und gestartet, in dem die Metadaten wie folgt abgespeichert werden:

- Declarations - In dieser Sektion werden alle zugehörigen Komponenten, Direktiven und Pipes deklariert.
- Providers - Sie initialisieren wie die Werte bei der Dependency Injection (siehe 5.2.3) abgerufen werden. [9]
- Imports - Hier werden alle Libraries und exportierte Modules importiert.

- Exports - Sie beinhalten alle zu exportierenden Module
- Bootstrap - Hierbei wird angegeben welche Komponente beim Anwendungsstart zuerst geladen wird

Die Verwendung von ngModules in der 3D-Gallerie-Applikation wird im Abschnitt Routing (siehe Routing 14) nochmals erklärt. [10] [11] [12]

Angular Schlüsseltechnologien [L]

Angular ist ein hoch anpassbares Framework. Bei der Verarbeitung von asynchronen Prozessen und Events kann Angular um die Library RxJS erweitert werden. Bei der Paketierung und beim Update von geändertem Code wird der Entwickler von der Library Webpack unterstützt.

RxJS [L] RxJS ist eine Implementierung von ReactiveX für die Programmiersprache Javascript.

ReactiveX ist eine Library für das Erstellen von asynchronen und eventbasierten Programmen, dafür benutzt es *observable sequences*. Ein beobachtetes Subjekt führt eine Liste von den Observern. Bei einer Veränderung im Subjekt werden die Beobachter nach der Reihenfolge der Liste über die Veränderung informiert (siehe 6.3). Die Library arbeitet mit dem *Observer-Design-Pattern* und fügt verschiedene neue Operatoren hinzu. Zusätzlich werden Low-Level-Funktionen wie Threading, Synchronisation und Thread-Sicherheit verarbeitet und keine Input-/Output-Prozesse blockiert. [13]



Abbildung 4: RxJS Logo

Webpack [L] Die Hauptfunktion von Webpack ist es, viele verschiedene Daten zu einem Paket für eine JavaScript Applikation zusammenzufassen. Angular benutzt Webpack, um TypeScript in JavaScript und SASS bzw. SCSS in CSS umzuwandeln. Beim Bauen des Projektes werden alle Module in ein einziges zusammengefasst. Bei der Entwicklung der Applikation wird durch Webpack *Live-Reloading* unterstützt. Dabei wird bei einer Änderung im Code die gesamte Applikation aktualisiert und neu gestartet. [14]



Abbildung 5: Webpack Logo

2.2.2 User-Interface-Frameworks (UI-Frameworks) [L]

Ein User-Interface-Framework ist eine Ansammlung von Web Komponenten wie beispielsweise einer Navigationsleiste, die sofort im Projekt nutzbar ist. [15]

Die Vorteile von UI-Frameworks sind:

- Sie vereinfachen und beschleunigen den Entwicklungsprozess. Typische Web-Komponenten Web-Komponenten sind vordefinierte, der Entwickler muss diese nicht mehr implementieren und erspart sich Zeit. [16]
- Sie achten bei den Komponenten und Styling-Features darauf, dass diese auch auf allen Internet-Browser angezeigt werden können. Besonders bei neuen CSS-Feature kann es sein, dass diese noch nicht von allen Browsern unterstützt werden. Ein UI-Framework benutzt in der Regel nur Funktionen, die auch alle Browser anzeigen können. [15]
- Der Code hat eine bessere Lesbarkeit. In einem UI-Framework gibt es gewisse Konventionen wie Klassennamen, die befolgt werden müssen. Dadurch wird der Code besser lesbar. [15]

Die Nachteile von UI-Frameworks sind:

- Die Ladezeit der Webseite erhöht sich; Schließlich müssen auch mehr Daten an den Browser geschickt werden.
- Webseiten, die das selbe UI-Framework verwenden, weisen visuelle Ähnlichkeiten auf, da die Web-Komponenten gleich implementiert wurden.

2.2.3 UI-Frameworks im Projekt [L]

Im Projekt werden die beiden UI-Frameworks, *Bootstrap* und *AngularMaterials* verwendet.

Angular Material [L]

Angular Material ist eine UI Framework und wird seit 2014 von Google entwickelt. Das Framework baut auf Angular auf und erweitert es mit eigenen Komponenten, Styleguides, Typography und vielem mehr. Die Design-Sprache orientiert sich dabei an der *Material Design Spezifikation*

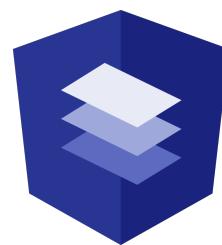


Abbildung 6: Angular Material UI Logo

von Google. Ein großer Fokus des Frameworks liegt in der Responsivität. [17, 18]

Angular Material bietet folgende Features:

- Erweiterbar: bei der Installation lassen sich viele Designanpassungen machen, zusätzlich lassen sich Styles mit dem globalen Stylesheet überschreiben. [17]
- Hochwertig: die Komponenten sind erprobt und wurden auf die Performanz und Verlässlichkeit getestet. [17]
- Reibungslos: Angular Material ist einzig und allen für Angular als UI-Library entwickelt worden.[17]

Angular Material bietet eine hohe Anzahl an sofort einsetzbaren Komponenten. Zu all diesen Komponenten gibt es auf der offiziellen Webseite eine ausführliche Dokumentation mit Beispielen. [17, 18]

Material Design Spezifikation [L] *Material Design* ist ein Styleguide der von der Firma Google entwickelt wurde. Er soll dabei helfen, qualitative hochwertige digitale Produkte für Android, iOS, Flutter und das Web zu gestalten.

Material Design befolgt für die Design-Richtung mehrere Prinzipien.

Material Design soll die reale Welt abbilden. Durch Typographie, Raster, Abstände, Farben und Bilder soll eine erkennbare Hierarchie entstehen. Die Komponenten dienen als interaktive Bauklötze. Komponenten haben alle ein Status-System, welches den Status der Komponente fokussiert, selektiert, aktiviert, fehlerhaft, schwebend, gedrückt und/oder deaktiviert anzeigt. Durch mehrere vordefinierte Themes ist es möglich einzelne Komponenten zu verändern und das Design so anzupassen, dass es zur Corporate Identity der*des Benutzer*in passt.[19]

Angular Material Komponente im Projekt [L] Im Projekt wurden hauptsächlich die Angular Material Komponenten *mat-icon*, *mat-chips*, *mat-stepper* und *mat-chard* verwendet (siehe Abbildung 7).

mat-icon wird dafür verwendet, die Icons auf der Single-Page-Application anzuzeigen. Angular Material besitzt eine Icon-Library. Wenn *mat-icon* verwendet wird, wird durch den Inhalt des *mat-icon*-Tags das Icon dynamisch aus der Icon-Library geladen.

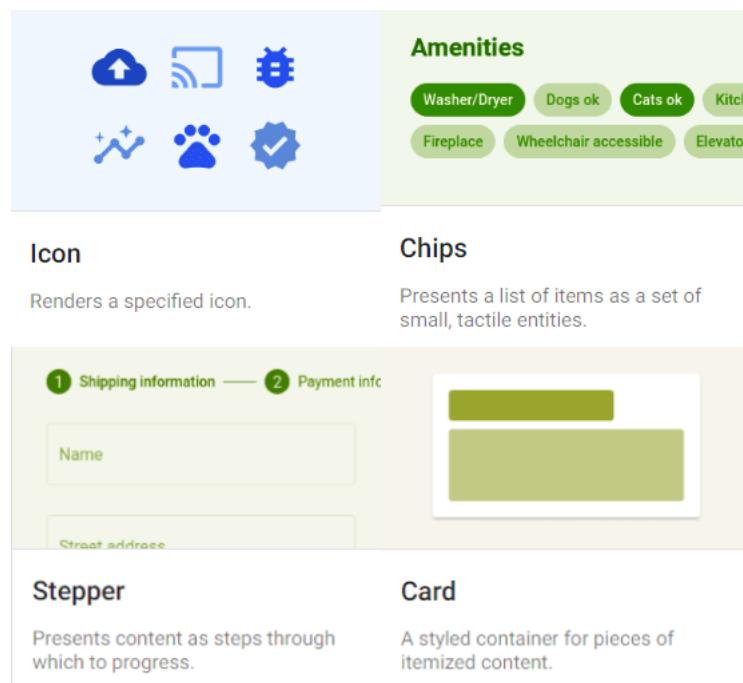


Abbildung 7: Angular Material: Komponenten die im Projekt verwendet wurden [20]

mat-chips werden für das Design der Ausstellungskriterien benutzt. Sie kommen in der Content-Creation und der Suchseite vor.

mat-stepper wird bei der Content-Creation verwendet, um die Wizard-Funktion umzusetzen.

mat-card wird für die Ausstellungsstückkarten auf der Single-Page-Application verwendet. Diese kommen auf der Profile-, Such- und Landing-Page vor.

Bootstrap [L]

Bootstrap ist ein kostenloses Open-Source-Frontend-Framework. Es fungiert als eine Art Bibliothek mit zahlreichem HTML sowie CSS Gestaltungsvorlagen. Auch die Integration von JavaScript ist durch Bootstrap möglich. Es bietet mehrere Komponenten an. Ziel von Bootstrap ist es den*der Entwickler*in das Umsetzen von interaktive und responsive Webseiten zu erleichtern. [16]

Bootstrap beinhaltet folgenden Features:

- Web-Komponenten, die sofort anwendbar sind [16]
- eine gute Dokumentation [16]
- Mobile Centered Design [16]



Abbildung 8: Bootstrap Logo

- hohe Benutzerfreundlichkeit [16]

SCSS [L]

SCSS ist eine Syntax- und Feature-Erweiterung von CSS,

und bietet folgende Vorteile:

- Variablen [21]
- Schleifen [21]
- Kalkulationen [21]
- Mixins [21]
- Imports [21]
- Verschachtelung [21]



Abbildung 9: Sass Logo

Besonders ausschlaggebend für die Auswahl von SCSS war das Prinzip der Verschachtelung. Durch die Verschachtelung kann an redundanten Selektoren gespart werden, was die Style-Definition besser lesbar macht. In den angeführten Beispielen (siehe SCSS-Code-Beispiel 1 und natives CSS-Code-Beispiel 2) werden die Schreibweisen von SCSS und CSS gegenübergestellt dargelegt. [21]

Listing 1: SCSS - Code Beispiel [21]

```

1  nav {
2      ul {
3          margin: 0;
4          padding: 0;
5          list-style: none;
6      }
7
8      li { display: inline-block; }
9
10     a {
11         display: block;
12         padding: 6px 12px;
13         text-decoration: none;
14     }
15 }
```

Listing 2: CSS - Code Beispiel [21]

```

1  nav ul {
2      margin: 0;
3      padding: 0;
4      list-style: none;
5  }
6  nav li {
7      display: inline-block;
8  }
9  nav a {
10     display: block;
11     padding: 6px 12px;
12     text-decoration: none;
13 }
```

Die Syntax- und Feature-Erweiterung haben uns dazu veranlasst, SCSS im Projekt einzusetzen.

2.2.4 3D Rendering [L]

Es gab verschiedene Auswahlkriterien für die benötigte 3D-Web-API, die ausschlaggebend für den Projekterfolg waren:

- Effizienzen der 3D Engine (Hardware Acceleration, RAM-Auslastung)
- Benutzerfreundlichkeit (Developer-Experience)
- Das Laden von 3D-Modellen aus 3D-Dateien und aus dem Internet
- Unterstützung von Video-Texturen

Three.js [L]

Three.js ist eine JavaScript Library für die Darstellung von 3D-Grafiken im Web. Für die 3D-Darstellung nutzt Three.js WebGL (mehr dazu im nächsten Abschnitt WebGL), ein Low-Level-Framework. WebGL hat eine hohe Komplexität. Three.js bietet eine Abstraktion von WebGL, um Anwendungen für 3D-Webanwendungen für Entwickler zugänglicher zu machen. Three.js regelt Systeme wie die 3D-Szene, Lichter, Schatten, Materialien, Texturen und 3D-Matrix-Rechnungen, deren Umsetzung mittels WebGL sehr aufwändig wäre. [22]

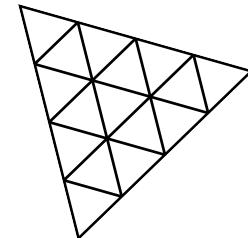


Abbildung 10: Three.js Logo

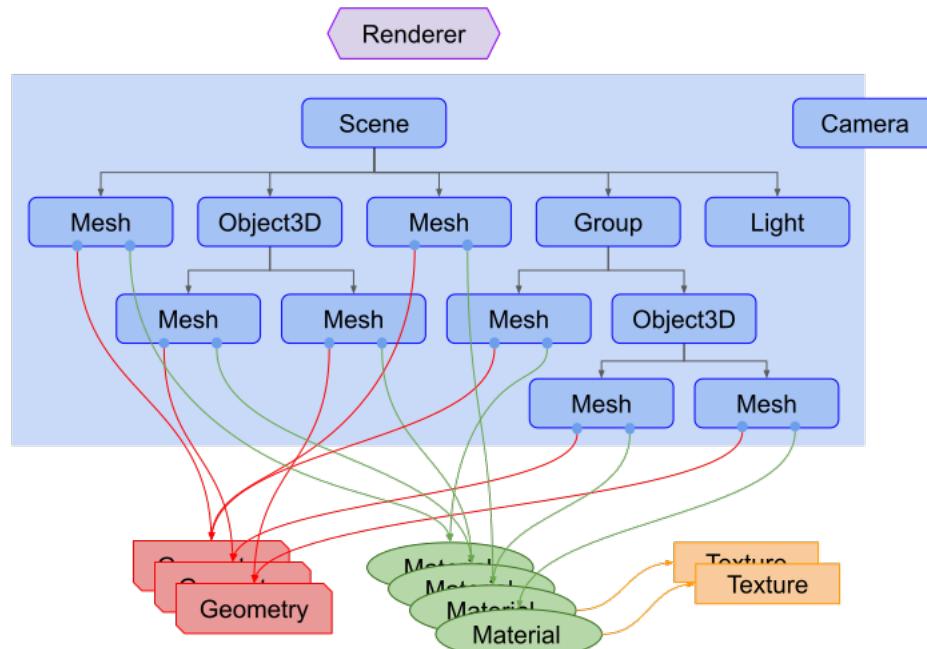


Abbildung 11: Die Struktur von Three.js [22]

In Three.js werden Geometrie, Objekte und Materialien verbunden, um ein 3D-Objekt zu erstellen. Die Abbildung 11 veranschaulicht den beispielhaften Aufbau eines solchen Objektes. In diesem Objekt werden Meshes, Gruppen, Lichter und 3D-Objekte in einer Baumdatenstruktur gesammelt. Die Daten über die Geometrie sowie Materialien der Meshes werden außerhalb der Szene gespeichert und die Meshes referenzieren im Renderprozess darauf. [22]

WebGL [L]

WebGL ist eine lizenzenfreie Low-Level-API, die dafür benutzt wird, 3D-Grafiken im Web darzustellen. Sie basiert auf OpenGL ES 2.0 und benutzt auch dieselbe Grafik-Programmiersprache, wie Open GLSL (OpenGL Shading Language). Eine Low-Level-API hat einen höheren Konfigurierungsgrad und lässt sich für spezielle Anwendungsfälle besser anpassen. Die API setzt ein viel tieferes Verständnis von dem Material (WebGL Shader-Programmierung, Matrixrechnungen usw.) voraus und bringt somit einen höheren Programmieraufwand mit sich. [23, 24]



Abbildung 12: WebGL Logo

Weitere Web Rendering APIs [L]

Es gibt viele weitere Technologien, die 3D-Grafiken im Web ermöglichen, wie beispielsweise Babylon.js, A-Frame, X3DOM und WebGL.

A-Frame [L] *A-Frame* wird von der *Mozilla Foundation* als OpenSource-Projekt entwickelt. Die 3D-Szene wird durch eine deklarative Sprache mit XML-Syntax definiert. Über die WebVR-API bietet die Libray auch die Möglichkeit, 3D-Szenen durch eine VR-Brille zu erfahren. [25]

A-Frame hat ähnliche Funktionen und Leistung im Vergleich zu Three.js. A-Frame sticht besonders bei der VR-Unterstützung hervor. Doch wird dieses Feature nicht im Projekt gebraucht. A-Frame baut auf Three.Js auf und erweitert es. Dadurch steigt die Komplexität, deswegen wird sich im Projekt gegen die Verwendung von A-Frame entschieden. [25]

Angular Three [L]

Angular Three ist ein Open Source Projekt von Matt DesLauriers. Es zielt darauf ab die Vorteile von Angular und Three.js zu kombinieren. Dabei verbindet es das Prinzip der Komponenten von Angular mit der 3D Darstellung von Three.js.

Listing 3: Angular Three - Komponentenbasiertes 3D Scenen in HTML [26]

```

1  <ngt-canvas>
2      <ngt-ambient-light intensity="0.5"></ngt-ambient-light>
3      <ngt-spot-light [position]="10" angle="0.15" penumbra="1"></ngt-spot-light>
4      <ngt-point-light [position]=-10></ngt-point-light>
5
6      <app-cube [position]=[1.2, 0, 0]></app-cube>
7      <app-cube [position]=[1.2, 0, 0]></app-cube>
8
9      <ngt-soba-orbit-controls></ngt-soba-orbit-controls>
10 </ngt-canvas>
```

Listing 4: Angular Three - App Cube [26]

```

1  <ngt-mesh
2      (beforeRender)="onCubeBeforeRender($event)"
3      (click)="active = !active"
4      (pointerover)="hovered = true"
5      (pointerout)="hovered = false"
6      [scale]="active ? 1.5 : 1"
7      [position]="position"
8  >
9      <ngt-box-geometry></ngt-box-geometry>
10     <ngt-mesh-standard-material [color]="hovered ? 'turquoise' :
11         'tomato'"></ngt-mesh-standard-material>
12 </ngt-mesh>
```

Angular Three ermöglicht eine schnelle und einfache Erstellung einer 3D-Szene. Die 3D-Abbildung kann durch nur wenige Zeilen Code 3 definiert werden. Die Business-Logik kann durch das komponentenbasierte Programmieren ausgelagert werden. Das wird durch das Codebeispiel App-Cube 4 verdeutlicht. [27]

Ein weiterer Vorteil von Angular Three ist die ausführliche Dokumentation mit Codebeispielen. [27]

Angular Three bietet viele Vorteile:

- hohe Benutzerfreundlichkeit
- ähnliche 3D-Leistung zu Three.js
- gute Integration mit Angular
- komponentenbasiert Programmieren

Wegen den zahlreichen Vorteilen bietet sich die Library für das Projekt an.

Mithilfe eines Prototypen (siehe fortlaufendes Prototyping) wurde getestet, ob Angular Three alle Features, die für das Projekt benötigt werden, unterstützt. Dabei wurde festgestellt, dass sich die Library Angular Three für das Projekt nicht eignet.

Trotz der vielen Features konnten einige Kriterien für die Projektumsetzung wie das dynamische Laden von Objekten, komplexe Szenen und Interaktions- und Bewegungsarten im Prototypen nicht erfüllt werden. So wurden beispielsweise 3D-Objekte, wenn sie mehrere Male von einer lokalen Datei geladen werden, aufgrund von Problemen bei der Implementierung Angular Three's Caching nicht mehr angezeigt. Außerdem funktionierten Angular-Three-Module wie die FristPersonControls nicht.

Weil Angular Three nicht die für das Projekt aufgestellten Anforderungen erfüllt hat, wurde sich dafür entschieden, rein auf Three.js ohne weitere Abhängigkeiten zu setzen.

Angular Thee - Abgebrochene Entwicklung Das GitHub-*Angular Three*-Repository wurde am 10. Februar 2023 archiviert. Der Source Code ist noch immer öffentlich zugänglich, doch gibt es keine Pläne, das Projekt weiterzuentwickeln. [28]

2.2.5 Sicherheit [L]

2.2.6 JWT - Json Web Token [L]

Der Json Web Token (JWT) ist ein offenerer Standard des Access-Token, der durch RFC 7519 definiert wird. JWT wird zur sicheren Übertragung von Informationen über ein JSON-Objekt benutzt, indem er den*die Benutzer*in authentifiziert. Die Authentifizierung passiert am Server. Dort wird nach einer Validierung erfolgt eine digitale Signatur des Token. Dies geschieht mittels Passwort oder einer Hashfunktion wie HMAC oder durch ein asymmetrisches Kryptosystem wie RSA. [29]

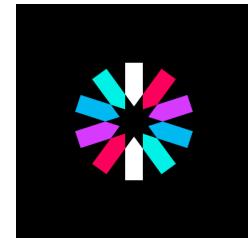


Abbildung 13: JWT Logo

2.3 Technologiestack Backend [Team]

Die Grundkriterien für die Wahl der Tools und Technologien hier waren, dass diese fortlaufend weiterentwickelt werden und zum derzeitigen Standpunkt ein ausreichendes Spektrum an Funktionalität für Mikroservice-Architekturen ermöglichen.[30] Ebenso sollen diese gute Dokumentationen und breite Communities enthalten.

2.3.1 Applikationserver [Team]

Als Applikationsserver wurde Quarkus ausgewählt. Quarkus zeichnen sich aus durch kurze Startzeiten und gute Performance in Hinsicht auf den Verbrauch des Arbeitsspeichers. Nach der Erstellung eines neuen Projekts wird standardmäßig eine Maven-Struktur mitgeliefert. Zusätzlich verfügt Quarkus eine Vielzahl von Extensions, welche durch Command-line-Befehle oder händisch zu Projekten hinzugefügt werden können. [31, 32] Quarkus besitzt die Fähigkeit im Hintergrund nach jeder Sourcecodeänderung die betroffenen Applikationen upzudaten und Unitest auszuführen. Die Services werden als REST-Services bereitgestellt. Quarkus besitzt eine einfache Konfigurationsmöglichkeit um REST-Services anzubinden.

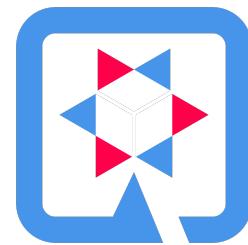


Abbildung 14: Quarkus Logo

2.3.2 Datenhaltung [Team]

PostgreSQL ist ein open source Managementsystem für relationale Datenbanken, welches seit 35 Jahren entwickelt wird. Hinter diesem System befindet sich eine große Community, welche weiterhin Features entwickelt. Die Entscheidung Eine gute Dokumentation und die vielen verschiedenen Anwendungsfälle [33]

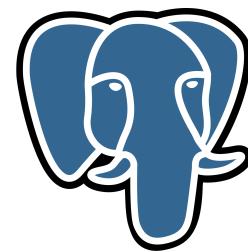


Abbildung 15: PostgreSQL Logo

3 Werkzeuge

3.1 IDE

3.1.1 IntelliJ IDEA [E]

IntelliJ IDEA ist eine IDE, welche von JetBrains entwickelt wurde. Diese ist ausgelegt für Java- und Kotlin-Projekte. Durch eingebaute Features erleichtert diese Entwicklungsumgebung das Programmieren für den*die Nutzer*in. Plugins ermöglichen es, Datenbankverbindungen und weiteres in der IDE zu konfigurieren, sodass dem*der Entwickler*in eine Übersicht von benötigten Informationen gegeben werden kann. [34]

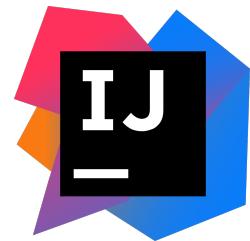


Abbildung 16: IntelliJ Logo

3.1.2 Webstorm [M]

Webstorm ist eine Entwicklungsumgebung vom Unternehmen JetBrains, die sich auf die Programmiersprache JavaScript spezialisiert hat. Sie wurde besonders für das Arbeiten mit Angular optimiert. Dies zeigt sich durch viele Features, die das Entwickeln von Angular-Projekten erleichtern. So macht es Webstorm möglich, mit nur wenigen Mausklicks eine neue Angular-Dependency oder Komponente zu erstellen. Auch wird die Entwicklungszeit durch intelligente Code-Vervollständigung, Code-Formatierung, einfache Navigation und viele weitere hilfreiche Features deutlich verkürzt. [35]



Abbildung 17: Webstorm Logo

3.2 UI Prototyping Werkzeug [L]

Es gab verschiedene Auswahlkriterien für das UI-Prototyping-Werkzeug:

- flache Lernkurve - Das Werkzeug sollte es ermöglichen, sofort in den Designprozess einzusteigen, ohne neue Konzepte und Arbeitsschritte lernen zu müssen.
- komponentenbasiert - Da Angular für die Umsetzung der Single-Page-Application gewählt wurde, sollte das Werkzeug auch komponentenbasiert sein.
- Kollaboration - Das Werkzeug sollte es ermöglichen, zusammen Design-Prototypen zu entwickeln und schnelles Feedback einzuholen.

Das Tool *Figma* erfüllte die Auswahlkriterien und wurde für dieses Projekt als UX/UI-Design-Tool gewählt.

3.2.1 Figma [L]

Figma ist ein Programm für die Erstellung und Testung von UI und UX Prototypen. Das Programm ist einsteigerfreundlich, denn es hat eine benutzerfreundliche Oberfläche und es gibt viele Tutorials auf der Webseite und von der Figma Community. Figma ist primär eine Web-Applikation, bietet aber auch eine Desktop-Version für macOS und Windows sowie eine mobile Version für iOS und Android an. Figma arbeitet mit mehreren Konzepten, um den Designprozess zu vereinfachen:

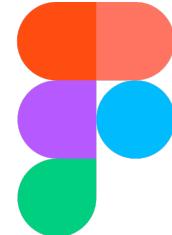


Abbildung 18: Figma Logo

Kollaboration In Figma können mehrere Personen gleichzeitig auf verschiedenen Geräten designen. Es gibt verschiedene Rollen, wie Zuschauer*in, Kritiker*in oder Mitarbeiter*in. Die Möglichkeiten können genutzt werden um einen*einer Kunden*in in den Designprozess zu involvieren und schon früh Feedback auf das Design bekommen zu können.

Plugins Figma hat eine große Community, gibt es ein Feature nicht, kann dieses von der Community im PluginStore als Plugin hinzugefügt werden. Ein Plugin ist eine Software-Erweiterung, welche die Fähigkeiten oder die Features eines Softwareprojektes erweitert.

Für die Design-Prototypen wurde das Plugin *Iconify* verwendet. Dadurch wird Figma durch eine weitere Oberfläche erweitert. Auf dieser Oberfläche kann nach Icons gesucht und diese Icons direkt im Design verwendet werden.

Assets Figma arbeitet mit dem Konzept der Assets. Bereits designte Komponenten können als Assets gespeichert und wiederverwendet werden. Figma legt einen großen Wert auf die Modularität. Beispielsweise können Farben und Pixelgrößen auch als Variablen gesichert werden. Bei einer Veränderung der Variablen wirkt sich diese sofort auf das Design aus.

3.3 3D Moddelierung

3.3.1 Cinema4D [M]

Cinema 4D ist ein professionelles Programm des Unternehmens Maxon zur 3D-Modellierung und Animation. Zum einen ist die Software leicht zu bedienen und zum anderen bietet Maxon eine Vielzahl an Tutorials und Anleitungen. Im Unterschied zu anderen 3D-Programmen wie Autodesk Maya, führt Cinema4D schnell zu einem anwendbaren Wissen. Aufgrund der intuitiven Benutzeroberfläche ist das Programm sowohl für Einsteiger als auch für Profis geeignet. [36]

Ein weiterer wichtiger Aspekt ist die Unterstützung des Exportes von GLTF-Files anzuführen. GLTF ist das Dateiformat, das genutzt wird, um ein 3D-Modell in der Three.js Szene zu laden. Da alternative 3D-Programme wie zum Beispiel Blender diese Exportmöglichkeit nicht anbieten, war Cinema 4D die gewählte Option für die 3D-Modellierung.



Abbildung 19: Cinema4D Logo

3.4 Package Manager [L]

3.4.1 NPM - Node Package Manager [L]

Der Node Package Manager ist ein Softwareverwaltungstool zum Downloaden, Aktualisieren, Veröffentlichen und Verwalten von OpenSource-Programmen in der NodeJs-Umgebung. [38] [39]

Funktionen	
Alle Funktionen	16
✓ 2D-Zeichnung	✓ 2D-Zeichnung
✓ 3D-Modellierung	✓ 3D-Modellierung
✓ Aktivitäts-Dashboard	✗ Aktivitäts-Dashboard
✓ Animation	✗ Animation
✓ Animationen und Übergänge	✓ Animationen und Übergänge
✓ Bild-Nachverfolgung	✗ Bild-Nachverfolgung
✓ Bildbearbeitung	✗ Bildbearbeitung
✓ Daten-Import / -Export	✗ Daten-Import / -Export
✓ Digital Asset Management	✗ Digital Asset Management
✓ Drag-and-Drop	✓ Drag-and-Drop
✓ Inhalt-Bibliothek	✓ Inhalt-Bibliothek
✓ Medienimport	✓ Medienimport
✓ Rendering	✓ Rendering
✓ Suche	✓ Suche
✗ Texteinblendung	✓ Texteinblendung
✓ Video-Inhalte	✓ Video-Inhalte
✗ Videobearbeitung	✓ Videobearbeitung
✓ Vorlagen	✗ Vorlagen

Abbildung 20: Cinema4D vs Maya [37]

4 Planung

4.1 Ideenfindung [M]

//TODO

Um dies genauer zu spezifizieren, erfolgte die Ideenfindung mithilfe der Anwendung einer passenden Kreativitätstechnik.

Kreativitätstechnik sind besonders hilfreich, wenn ein rationales Problemlösen nicht zielführend erscheint. Sie helfen Ziele, Lösungen und Risiken zu finden, evaluieren und festzulegen. Für die Diplomarbeit bot sich besonders die Kreativitätstechnik Brainstorming an, da durch sie viele verschiedene Ideen in kürzester Zeit gesammelt werden können. [40]

Beim Brainstormen findet jede*r Teilnehmer*in zu einem bestimmten Thema so viele Ideen wie möglich. Das Brainstormen erfolgt in 2 Phasen. Zuerst werden alle Ideen niedergeschrieben und anschließend die Beste herausgefiltert.

Beim Filterprozess waren uns folgende Kriterien wichtig:

- Quantität vor Qualität
- Freies Assoziieren ist erwünscht
- Keine Kritik, Korrektur oder Meinungsäußerung
- Möglichst viele Ideen
- Inspirieren lassen von anderen

[40]

4.1.1 Namensfindung [L]

Der Fokus bei der Namensfindung lag darauf, mittels des Namens der Diplomarbeit schon einen Eindruck auf die Funktionalitäten und den Umfang dieser zu geben.

Für die Namensfindung wurde die Kreativitätstechnik Brainstorming benutzt. Die Teammitglieder trafen sich und brachten Ideen ein.

Schlussendlich wurde sich für den Namen *3D Portfolio Gallery* entschieden.

Dass wir im Projektnamen das Wort Gallery benutzen, hat folgende Gründe:

- In einer Gallery werden Gegenstände oder Bilder oder künstlerische Installationen präsentiert.
- Eine Gallery ist nach Themen und oder Räumen aufgeteilt.
- Eine Gallery hat ein Orientierungssystem, um die Besucher*innen durch die Ausstellung zu leiten.
- die virtuell begehbar Räume und die darin befindlichen präsentierten Objekte setzen die Ausstellungsstücke in eine örtliche Beziehung zueinander

4.2 User Stories [L]

User Stories stellen konkrete Anforderungen an die Software. Sie werden user- und werden aus der Perspektive einer einer im Projekt involvierten Person, meistens des Users, beschrieben. [41]

Hier ein konkretes Beispiel, wie eine User Story gestaltet werden könnte: "*Als Designer möchte ich einen Login, um meine Ausstellungen speichern und im Nachhinein immer öffnen zu können.*"

Die User Stories wirken als Kommunikationsmittel zwischen Kunde und Entwickler und bieten bieten auch eine Möglichkeit, den Projektfortschritt messbarer zu machen. [41]

User Stories kommen vor allem in der agilen Softwareentwicklung vor. Dort besprechen der Kunde und der Product Owner eine Zielbestimmung. Anhand dieser Bestimmungen werden Funktionen festgelegt, die das Projekt erfüllen muss und diese mithilfe von User Stories formuliert. [41]

In weiterer Folge wird in dem Kapitel 5 die Umsetzung der User Stories beleuchtet.

5 Umsetzung

5.1 Design [L]

Im Designprozess wurden zwei Landingpages in Form von Mockups erstellt (siehe Abbildung 21). Dafür wurde das Tool Figma benutzt. Danach wurden die Designs einer Auswahlgruppe präsentiert. Die ausgewählte Landingpage soll das weitere Design der grafischen Benutzeroberfläche und der Komponenten der Single-Page-Application bestimmen.

Zur Auswahl stand ein edles Design, das in dunklen Farben und eckigen Formen auftritt. Das zweite Design ist ein freundliches Design mit bunten und hellen Farben und runden Formen.

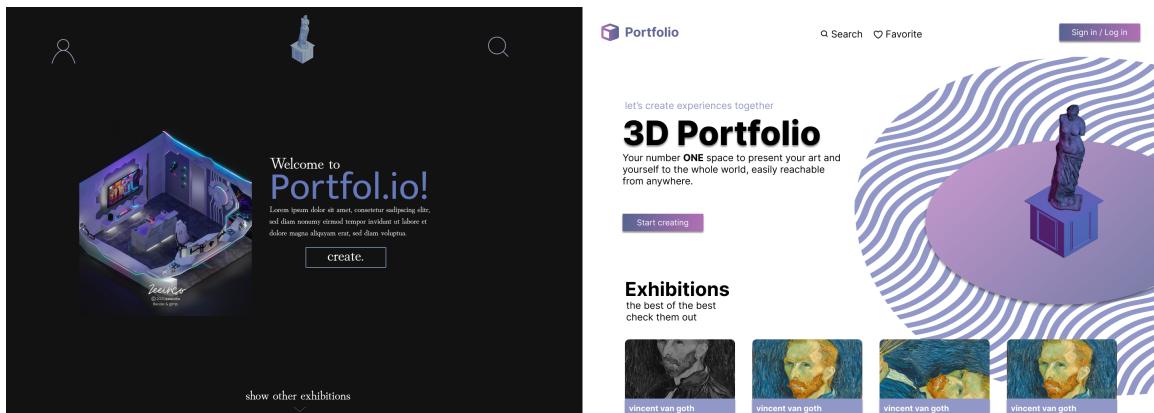


Abbildung 21: Landingpage Design Mockups Gegenüberstellung

Einer Auswahlgruppe wurde der Zweck der Prototypen erklärt und die beiden Designs präsentiert. In der Auswahlrunde erhielt das hellere Design das bessere Feedback und wurde als ausgewählt.

Danach wurde der helle Prototyp auf seine Charakteristiken wie Farben, Formen, Typografie analysiert und basierend darauf ein Styleguide für die grafische Benutzeroberfläche der Single-Page-Application angefertigt.

5.1.1 User Experience [L]

Ein gutes User-Experience-Design ist die Grundlage für eine erfolgreiche Positionierung am Markt und Kommunikation mit dem*der Benutzer*in.

Die User Experience bezieht sich dabei auf die Interaktion des Benutzers mit der Umwelt, aber auch mit dem angebotenen Service oder Produkt. In diesem Kontext hat Design mehrere Aufgaben. Zum Einen gilt es, die Interaktionsmöglichkeiten zu gestalten (UI-Design). Dies bedeutet visuelle Kommunikation über Zeichen und Symbole.

Weiters muss erkannt werden, was dem*der Benutzer*in wichtig ist und dieses dem Produkt zugewiesen werden, sodass es auch der*die Benutzer*in erkennen kann [42].

Anwendungen von UX im Projekt [L]

Schon beim Projektstart standen die Benutzer*innen im Mittelpunkt. Dazu wurden die in Kapitel 4.2 beschriebenen User-Stories aus Sicht eines Benutzers formuliert, um klar herausarbeiten zu können, welchen effektiven Nutzen unsere Web-Applikation dem*der Nutzer*in bringt.

Die Oberfläche wurde so gestaltet, damit sie leicht verständlich, nützlich und benutzerfreundlich ist. Dafür wurden die oben genannten Prototypen (siehe Abbildung 22) in Figma (siehe 3.2.1) gestaltet und durch Umfragen ein Favorit bestimmt.

5.1.2 User-Interface-Design [M]

//TODO

5.1.3 Logoentwicklung [M]

Ein wichtiger Teil der Design-Phase war die Entwicklung eines Logos. Hierbei sollte es darum gehen, ein Wiedererkennungsmerkmal zu schaffen, das sowohl im Gedächtnis bleibt, als auch das Produkt repräsentiert. Um dies zu gewährleisten sollte bei der Logoentwicklung auf folgende Faktoren geachtet werden:

- Flexibilität - geeignet für große und kleine Formate
- Einzigartig - von der Konkurrenz differenzierbar
- innovativer Gestaltungsansatz



Abbildung 22: OberflächenDesign: Prototypen in Figma

- Klarheit in der Formensprache

[43]

Zur Erstellung des Logos wurde *Adobe Illustrator* verwendet, ein Programm speziell für das Illustrieren von Vektorgrafiken. Für das Logo der Diplomarbeit wurde ein isometrischer Zeichenstil verwendet. Dabei verlaufen alle Fluchtenlinien parallel und dadurch erscheint das Design zum einen flach, als auch dreidimensional. Es soll den 3D-Raum symbolisieren, in dem sich ein Podest befindet. Farblich wurde auf eine Abstimmung mit dem Prototypen geachtet, um leichter eine Verbindung mit dem Produkt schaffen zu können. [44] [45]

5.2 Umsetzung der Landingpage [L]

Der erste Entwicklungsschritt für die vollständige Single-Page-Application beginnt mit der Initialisierung der Landingpage. Aus Sicht der User-Experience war es uns wichtig, dass der*die neue Nutzer*in sofort den Verwendungszweck des Applikation begreift und dazu animiert wird, bei der Applikation mitzumachen.

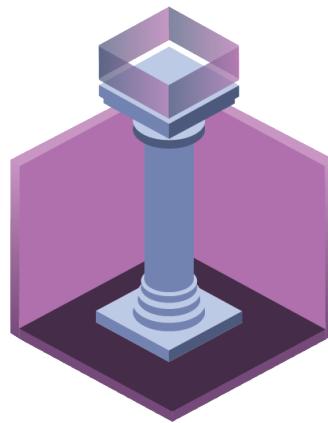


Abbildung 23: Logo der 3D-Portfolio-Gallery

Die Entwicklung startete damit, dass die benötigten Technologien Angular, Angular-Three, AngularMaterials und Bootstrap heruntergeladen wurden.

Dafür wurde der *NPM* Node-Package-Manager verwendet.

Angular installieren

Angular hat ein eigenes Tool, die Angular CLI (Command Line Interface), um Projekte zu erstellen und zu bearbeiten, sowie um Komponenten, Services und vordefinierte Codemodule hinzuzufügen.

Listing 5: Terminal - Angular aufsetzen, Installation der CLI, Configuration eines neuen Projektes, Starten des Projektes

```

1  npm install -g @angular/cli
2  ng new Gallery
3  ? Would you like to add Angular routing? Yes
4  ? Which stylesheet format would you like to use? SCSS      [
5    https://sass-lang.com/documentation/syntax#scss ]
5  cd Gallery
6  ng serve -o

```

In der ersten Codezeile wird das Angular-CLI-Tool global von NPM installiert. Danach wird mit dem Befehl *ng new* mit dem CLI-Tool ein neues Angular Projekt erstellt. Anschließend gibt es mehrere Konfigurationsauswahlmöglichkeiten. Für dieses Projekt wurde das Routing (siehe 5.2.2) aktiviert und als Stylesheet-Formatierung SCSS ausgewählt. Daraufhin wurde in das (Projekt-) Verzeichnis *Gallery* gewechselt und dort mit dem Befehl *ng serve* ein Webpack-Server gestartet, welcher den vorgenerierten Code von dem neu erstellen Angular-Projekt zeigt (siehe in Abbildung 24).

Globale oder lokale Module Bei einer lokalen Installation werden die installierten Module in einem Node-Modul-Computerordner lokal im Projekt abgespeichert, wäh-

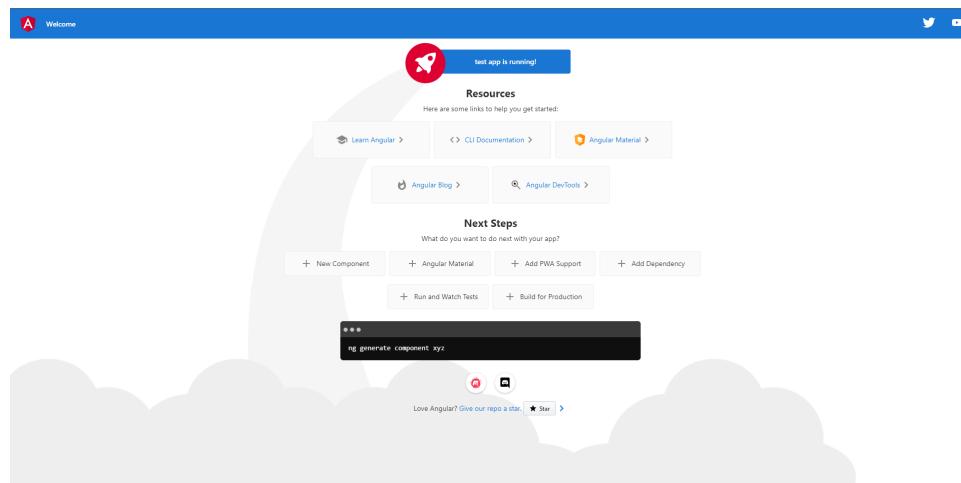


Abbildung 24: Angular: Automatisch generierte Start-Webseite

rend alle globalen Installationen in einem einzigen Computerordner abhängig vom Computersystem gespeichert werden.

Generell kann man bei NPM-Modulen zwischen lokalen und globalen Installationen unterscheiden. Grundsätzlich wird eine lokale Installation empfohlen, referenzieren mehrere Projekte auf ein globales Modul, kann es dazu kommen, dass bei einer Aktualisierung des globalen Modules verschiedene Projekte beschädigt werden. Das kommt daher, dass die darauf referenzierenden Projekte wegen ihrer Funktionalität anders auf die Veränderung des Modules reagieren. CLI-Module haben eine Berechtigung dafür, global installiert zu werden. Denn CLI-Module sind abgekapselte Softwarelösungen, auf die kein anderes Projekt referenziert. Sie können auch lokal installiert werden und mit dem Befehl *npx* nur im Projektordner ausgeführt werden.[46]

Installation der UI-Frameworks

Nach der Installation von Angular wurden die UI-Frameworks *Angular Material* und *Bootstrap* installiert.

Angular Material konnte mithilfe des Befehls 6 mittels der Angular CLI in das Angular-projekt eingebunden werden.

Listing 6: Terminal - Angular Material Installation

```
1      ng add @angular/material
```

Bootstrap wurde mithilfe des Befehls 7 von NPM installiert. Anschließend muss noch eine Verknüpfung zwischen Bootstrap und Angular hergestellt werden. Dafür musste in

der Angular-Konfigurationsdatei *angular.json* die Bootstrap Scss-Liberay eingebunden werden. Das wird in dem Code Beispiel 8 veranschaulicht.

Listing 7: Terminal - Bootstrap Installation

```
1      npm i bootstrap
```

Listing 8: angular.json - Bootstrap Angular Verknüpfung

```
1      {
2          ...
3          "projects": {
4              "Gallery": {
5                  ...
6                  "architect": {
7                      "build": {
8                          ...
9                          "options": {
10                             ...
11                             "styles": [
12                                 ...
13                                 "node_modules/bootstrap/scss/bootstrap.scss"
14                             ]
15                         }
16                     }
17                 }
18             },
19         ...
20     }
```

Installation von Three.js

Nach der Installation der UI-Libraries wurde Three.js installiert.

```
1      npm install three
2      npm i @types/three
```

5.2.1 Komponenten [M]

Der nächste Schritt der Entwicklung ist das Erstellen der Komponenten und das Festlegen ihrer Struktur. Eine Komponente kann ebenfalls über die Angular CLI erstellt werden:

Listing 9: Terminal - Component erstellen

```
1      ng generate component home-page
```

Allgemein

Angular Komponenten lassen sich mit einem eigenen HTML-Selektor (z.B. <my-component>) ansprechen und in die Benutzeroberfläche einbinden. Components besitzen viele Features, die den Entwicklungsprozess, Wartung des Codes und Fehlersuche erleichtern können:

- Trennung von Logik und Design
- Skalierbarkeit der Applikation
- Data-Binding
- Hierarchie
- Lesbarkeit/Übersichtlichkeit

Aufbau

Um die Logik strikt von der Benutzeroberfläche zu trennen, werden die Dateien und der Code innerhalb einer Komponente strukturiert und separiert. Eine Komponente besteht somit aus:

- einem HTML-Template, dass die Benutzeroberfläche darstellt
- einer TypeScript-Klasse, die die Logik und Funktionalitäten der Komponente beinhaltet
- einem Stylesheet, dass das Aussehen der Benutzeroberfläche beeinflusst
- einer TypeScript-Testklasse, um individuell Komponenten zu testen

[47]

Skalierung

Die Skalierung beschreibt, wie gut eine Applikation mit Erweiterungen und ihrem Wachstum umgeht. Dabei wird geschaut, wie leicht sich zusätzliche Änderungen und Features implementieren lassen oder wie sich die Performanz der Anwendung bei zunehmender Größe verhält. Angular unterstützt dies durch einige Features:

TypeScript hilft durch seine Code-Syntax, auch bei großen Applikationen schnell Fehler zu erkennen.

Durch die Angular CLI kann schnell und einfach ein Grundgerüst für die Entwicklung erstellt werden. Weiteres können Components, Services und andere Angular Funktionen in jedem Entwicklungsstadium problemlos hinzugefügt werden, ohne dass der bestehende Code beeinflusst wird (siehe 5.2).

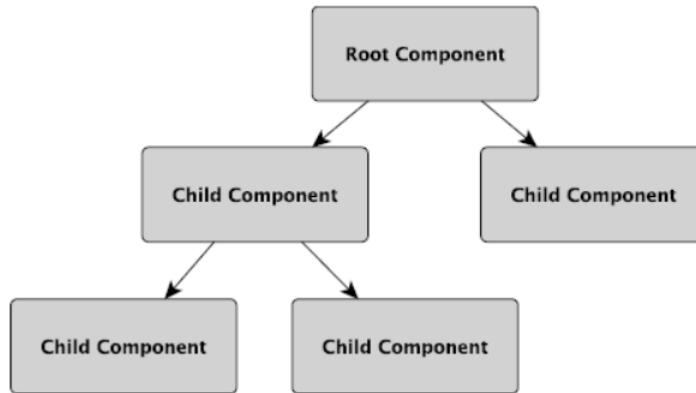


Abbildung 25: Component-Hierarchy [12]

Angular stellt zudem Libraries wie Angular Materials oder das Component Development Kit (CDK) zur Verfügung. Dadurch können problemlos vorgefertigte Angular Komponenten und Designs zu einer Applikation hinzugefügt werden.

[48]

Hierarchie

Die Hierarchie von Angular Komponenten lässt sich wie folgt in einer Baumstruktur abbilden:

Die Hauptkomponente ist der Root-Component. Von ihr aus lassen sich weitere Child-Components ineinander verschachteln. Diese Verschachtelung ist der Grund, warum die Anwendung in viele einzelne Teile ausgelagert werden kann. Außerdem können Komponenten durch das Data-Binding untereinander Daten austauschen.

[49] [12]

Data-Binding

Data-Binding ist eine Methode, Daten zwischen der Logik und der Benutzeroberfläche auszutauschen. Dabei bleibt die Website ohne Refresh immer aktuell. Das Binden von Daten kann direkt am DOM erfolgen, wofür es eine eigene Code-Syntax gibt. Beim Data-Binding wird zwischen verschiedenen Kategorien unterschieden [12] [50]:

Interpolation Bei der Interpolation werden Daten innerhalb einer Komponente von der Datenquelle in das HTML-Template eingefügt. Die Anzeige wird auch automatisch aktualisiert, wenn sich Daten im Hintergrund ändern. [50]

Listing 10: Beispiel für Interpolation in der 3D-Gallery

```
1   <div>
2     <p class="text-center py-2">{{exhibit.title}}</p>
3   </div>
```

Property-Bindings Beim Property-Binding werden Daten direkt an ein DOM-Element übermittelt und ausgewertet. Somit werden die Attribute, das Aussehen und die Funktionen der jeweiligen DOM-Elemente dynamisch beeinflusst und automatisch bei Datenänderung aktualisiert. [51]

Listing 11: Beispiel für Property-Bindings in der 3D-Gallery

```
1   <img [src]="user?.icon_url ?? 'assets/image/profile-photo.jpg'">
```

Event-Bindings Um auf Eingaben und Interaktionen von Benutzer*innen zu reagieren, werden Event-Bindings verwendet. Dabei werden die Daten vom HTML-Template zur TypeScript-Klasse übertragen und können dort genutzt werden. Sie sind also das Gegenstück zu den Property-Bindings. [52]

Listing 12: Beispiel für Event-Bindings in der 3D-Gallery

```
1   <button(click)="delete()">
2     <mat-icon>close</mat-icon>
3   </button>
```

Two-Way-Bindings Das To-Way-Binding benutzt beide Varianten, Property- und Event-Binding, um Daten auszutauschen. Hierbei ist es möglich, die Daten sowohl von der TypeScript-Klasse zum DOM-Element zu übertragen und umgekehrt. [53]

Listing 13: Beispiel für Two-Way-Bindings [53]

```
1   <app-sizer [(size)]="fontSizePx"></app-sizer>
```

In den folgenden Kapitel werden die Komponenten und ihre Funktionalitäten näher beschrieben.

5.2.2 Routing [M]

Routing ist das Konzept einer Single-Page-Applikation, wodurch die Seite niemals neu geladen werden muss und alle Daten beim Navigieren beibehalten werden können. Beim Routing werden dabei bestimmte Komponenten angezeigt, abhängig von der aktuellen URL. Dabei kann durch die Applikation navigiert werden, was durch den sogenannten Router realisiert wird. Um das Routing überhaupt verwenden zu können, müssen die Pfade zu den zugehörigen Components initialisiert werden. Dies geschieht in einem *ngModule*, wo alle initialisierten Routen über das RouterModule importiert werden. Um die Funktionalitäten und Routen für alle Components verwenden zu können, muss dieses RouterModule ebenfalls exportiert werden. Der Code zeigt dies anhand der 3D-Gallerie-Anwendung: [12]

Listing 14: Routing in der 3D-Gallery

```

1  const routes: Routes = [
2    {path: '', component: HomePageComponent},
3    {path: 'home', component: HomePageComponent},
4    {path: 'log-signin', component: LogSignInPageComponent},
5    {path: 'search', component: SearchPageComponent},
6    {path: 'profile', component: ProfilePageComponent},
7    {path: 'create', component: CreateExhibitionPageComponent},
8    {path: 'signup', component: SignupPageComponent},
9    {path: 'room/:id', component: RoomPageComponent},
10   {path: '**', redirectTo: 'home'}
11 ];
12 @NgModule({
13   declarations: [],
14   imports: [ CommonModule, RouterModule.forRoot(routes) ],
15   exports: [ RouterModule ]
16 })

```

Hierbei steht der Pfad ‘**’ für alle ungültigen URLs wodurch der*die Benutzer*in auf die Landingpage geleitet wird.

Ebenfalls wird das Routing genutzt um das Navigieren durch die Navbar (siehe fertige Landingpage 5.4.1) zu ermöglichen. Hierbei wird ein deklarierter Pfad direkt mit einem HTML-Element, über das Attribut *routerLink*, verbunden. Falls sich der*die Benutzer*in auf dem momentanen Pfad des *routerLinks* befindet, wird das Attribut *routerLinkActive* aktiv. Dadurch kann dem*der Benutzer*in ein bestimmtes visuelles Feedback geliefert werden. (siehe Code 15)

Listing 15: Routing über einen routerLink

```

1      <a class="navbar-brand"
2        routerLink="/home"
3          routerLinkActive="link-activated"><mat-icon>home</mat-icon>Home</a>
4      <a class="navbar-brand"
5        routerLink="/search"
6          routerLinkActive="link-activated"><mat-icon>search</mat-icon>Search</a>
7      <a class="navbar-brand" *ngIf="this.auth.isLoggedIn()"
8        routerLink="/profile"
9          routerLinkActive="link-activated"><mat-icon>person</mat-icon>Profile</a>

```

Routingparameter

Um Routen dynamisch zu deklarieren, werden sogenannte Routenparameter benötigt. Hierbei beginnt der dynamische Teil einer Route mit einem Doppelpunkt.

Listing 16: Routingparameter in der 3D-Gallery

```
1 const routes: Routes = [
2   {path: 'room/:id', component: RoomPageComponent},
3 ];
```

Um den aktuellen Zustand der URL auszulesen, muss der Router im Konstruktor injiziert werden. Anschließend kann der Parameter wie folgt ausgelesen und als Wert einer Variable zugewiesen werden:

- Dies geschieht entweder über die Snapshot-Methode, die den momentanen Zustand der URL ausliest

Listing 17: Snapshot der URL abfragen

```
1 this.id = this.route.snapshot.paramMap.get('id');
```

- oder, wie auch im Projekt der 3D-Gallery umgesetzt, über das Observable-Pattern auf den Router subscriben, um auf ständige Änderungen im Pfad zu reagieren:

Listing 18: Die URL subscriben

```
1 this.sub = this.route.params.subscribe(params => {
2   this.id = +params['id'];
3 })
```

[12]

5.2.3 Angular Services [M]

Um eine Schnittstelle zwischen dem Backend und dem Frontend herzustellen, werden *Angular Services* verwendet.

Allgemein

In Angular entspricht ein Service einer TypeScript-Klasse, die verwendet wird, um Logik auszulagern, die von der gesamten Applikation verwendet werden kann. Services werden meist dann erstellt, wenn eine einfache Logik oft von verschiedenen Komponenten benötigt wird. Um die globale Verwendung zu realisieren, wird das Konzept der *Dependency Injection* verwendet. [12] [54]

Dependency Injection

In Angular ist es möglich, einen Service in eine beliebige Komponente zu injizieren. Das bedeutet, ein Service wird anhand von `@Injectable()` definiert und kann dadurch von anderen Komponenten verwendet werden [12]:

Listing 19: Eine Klasse Injectable machen

```

1  @Injectable({
2      providedIn: 'root'
3  })
4  export class GalleryService{
5      ..
6  }
7

```

Die Codezeile "providedIn: 'root'"orgt dafür, dass sich Services entweder nur in spezifische Komponenten injizieren lassen oder sich wie hier auf root-level, also überall, injizieren lassen. [12]

Ein Service wird zur Verwendung im Konstruktor einer Komponente initialisiert (Constructor Injection):

Listing 20: Constructor Injection

```
1     constructor(private gs: GalleryService) { }
```

Anschließend können die Methoden und Daten eines Service ganz normal verwendet werden.

HTTP Module

Um mit dem Backend über das HTTP-Protokoll kommunizieren zu können, wird eine HTTP-API (siehe 6.3) namens `HttpClient` verwendet. Zunächst muss das `HttpClientModule` im `ngModule` importiert werden, um den `HttpClient` als Dependency zu injizieren. Injiziert wird er über den Konstruktor in einem Service. Um eine Transaktion am Frontend durchzuführen, werden Observables verwendet. Dies ermöglicht den einzelnen Komponenten, die einen Service mit HTTP-Abfragen injiziert haben, diese Abfragen mittels einem `Subscribe` zu nutzen. Ein `Subscribe` ist ein Operator von `RxJS` (siehe 2.2.1). Eine HTTP-Abfrage mit dem `HttpClient` wird wie folgt aufgerufen [12] [55]:

Listing 21: HttpClient Abfragen

```

1     URL = "http://localhost:8080/api/"
2
3     getAllRooms(): Observable<Room []>{
4         return this.httpClient.get<Room []>(`${this.URL}rooms/allRoomPositions`);
5     }

```

Wie anhand des oberen Code-Beispiels erkennbar ist, wird hier das von uns erstellte Room-Interface benutzt.

Interface [M]

Interfaces werden verwendet, um ein Objekt typsicher zu strukturieren. Dabei wird, um Daten des Servers korrekt erhalten zu können, eine bestimmte Entität der Datenbank verglichen und durch die richtige Benennung und der korrekte Datentyp im Frontend abgebildet. Üblicherweise wird ein solches Datenobjekt exportiert, um es in der ganzen Anwendung zu verwenden. Folgendes Beispiel demonstriert dieses Konzept anhand des Exhibit-Interfaces [12]:

Listing 22: Das Datenmodell eines Ausstellungsstückes

```

1  export class Exhibit{
2      id : number;
3      url : string;
4      data_type : string;
5      title : string;
6      description : string;
7      alignment: string | undefined
8      position: Position | undefined
9      scale: number | undefined
10
11
12      constructor(id: number, model_url: string, data_type: string, title: string,
13          desc: string, alignment: string | undefined, position: Position | undefined,
14          scale: number | undefined) {
15          this.id = id;
16          this.url = model_url;
17          this.data_type = data_type;
18          this.title = title;
19          this.description = desc;
20          this.alignment = alignment;
21          this.position = position
22          this.scale = scale
23      }
24  }
```

5.3 Web Gallery Prototyp [L]

Unser Web-Gallery ist mithilfe eines evolutionären Prototypen (siehe 6.3) entwickelt worden.

Anfangs wurde dabei die Machbarkeit des Projekts mit einer Umsetzbarkeitsanalyse, bei der der Prototyp eine große Rolle spielte, getestet. Im Prototyp wurde überprüft, ob die Technologien, auf die der Prototyp aufbaute, auch die Anforderungen erfüllen konnten.

Da die finale Entscheidung für Three.js schon früh in dem Entwicklungsprozess gemacht wurde, entstand mit der Änderung der 3D-Web-API im Prototypen nur wenig zusätzlicher Programmieraufwand.

In diesem Abschnitt wird erklärt, auf welche Weise die 3D-Ausstellung in der Webapplikation realisiert wurde.

Modellierung der 3D-Räume in Cinema 4D [M]

Bevor mit der Implementierung des Ausstellungsraumes in der Webapplikation begonnen werden kann, muss das 3D-Modell des Raumes zunächst in Cinema4D modelliert werden. Zu Beginn wird der Grundriss des Raumes gezeichnet. Dies erfolgt durch das Spline-Werkzeug. Hierbei werden 2D-Linien im dreidimensionalen Raum erstellt. Anschließend wird ein Würfel generiert, der die Höhe und Breite der Wand besitzt. Da Cinema 4D die Modelle in Zentimeter misst, können diese in realitätsnahen Proportionen modelliert werden. Um den 2D-Grundriss nun als dreidimensionale Wände darzustellen, werden die gezeichneten Splines mit der Form des Würfels als Sweep extrudiert. Das bedeutet, eine 2D-Form wird als schlauchförmiges 3D-Objekt mit den Werten des Würfels erstellt. Nach diesem Vorgang kann schlussendlich der Boden des Raumes hinzugefügt werden. Dazu wird eine Platte in Form des Grundrisses erstellt.

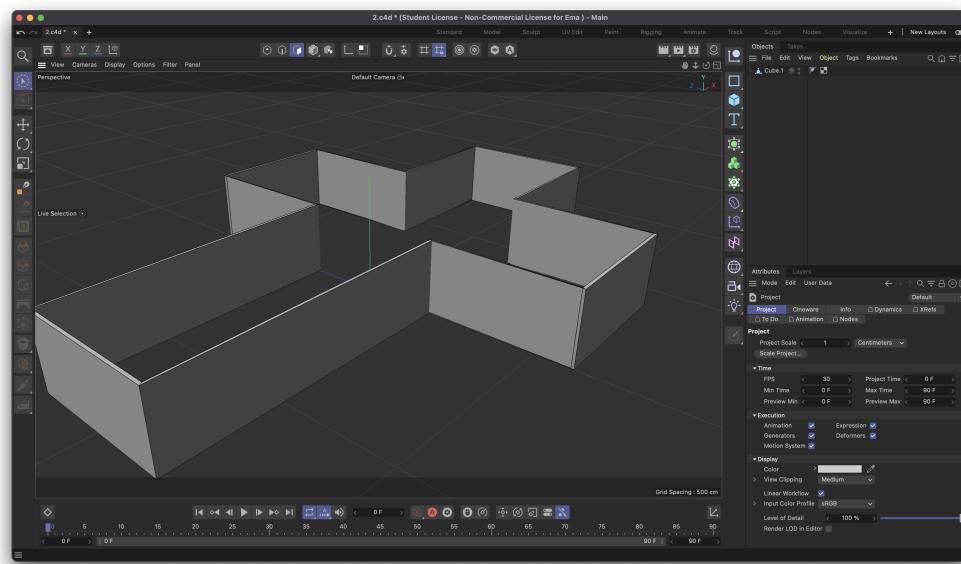


Abbildung 26: Modellierung des Raumes in Cinema4D

5.3.1 Rendering [M]

Um 3D-Modelle, wie den Ausstellungsraum anzeigen zu können, wird der hochkomplexe Prozess des Renderings benötigt. Dabei wird das 3D-Modell zu einem realistischen 2D-Bild gerendert. [56] [57]

Der Prozess des Renderns

Hinter dem Renderprozess stecken verschiedene automatische Prozesse die aus vorliegenden Werten und Daten ein fertiges 3D-Bild berechnen. Die dahinterstehenden Render-Vorgaben beeinflussen die Qualität und die Geschwindigkeit des Prozesses unterschiedlich - beispielsweise:

- die Komplexität des 3D-Objektes und die gewünschte Realitätsnähe
- die Art des Renderns
- die Lichtquellen und die entstehenden Schatten (siehe 5.3.2)
- die Kameraposition und der Renderbereich (Clipping)
- der Anti-Aliasing Prozess
- die Art des Render-Algorithmus
- weitere atmosphärische Effekte

[57]

3D-Modelle

3D-Modelle besitzen verschiedene Eigenschaften, wie zum Beispiel Textur, Farbe oder die Fähigkeit Licht zu reflektieren. All dies muss beim Rendern berücksichtigt werden. Die 3D-Daten des Modells werden ermittelt und dabei in Pixelinformationen umgewandelt und durch die Ermittlung der Positionierungskoordinaten korrekt platziert.[57]

Arten des Renderns

Beim Rendern wird zwischen folgenden Arten unterscheiden:

- Offline-Rendering - Das Offline-Rendering rendert über einen längeren Zeitraum in hoher und realistischer Qualität. Diese Art kommt zum Beispiel bei Filmen zum Einsatz. [58]

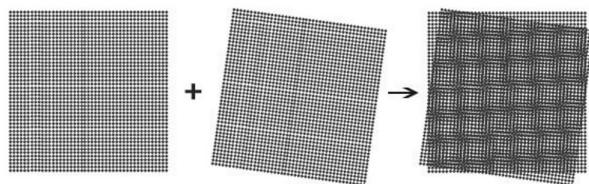


Abbildung 27: Der Moire-Effekt visualisiert [59]

- Echtzeit-Rendering - Beim Echtzeit-Rendering wird ein 3D-Modell in kürzester Zeit gerendert. Diese Art kommt zum Beispiel bei Videospielen oder in der aktuellen 3D-Gallery zum Einsatz. Hierbei werden meist 25-30 Bilder pro Sekunde gerendert, um Bewegungen flüssig wahrzunehmen. [58]

Clipping

Beim Clipping werden Ebenen zur Begrenzung des Bereichs, den es zu rendern gilt, erstellt. Diese Ebenen werden durch die Weltkoordinaten positioniert. Die seitlichen sowie unteren und oberen Clipping-Ebenen werden durch die Kameraposition und die Fensterposition definiert. Auch gibt es das Far- und Near-Clipping, wodurch die Tiefe des 3D-Modells begrenzt wird. Gleichzeitig werden unerwünschte Objekte, die im Hintergrund liegen, nicht gerendert. [57]

Anti-Aliasing

Beim Anti-Aliasing Prozess wird versucht, den auftretenden Aliasing-Effekt zu reduzieren und zu entfernen. Dieser tritt auf, wenn die Pixel zu grob für feine Strukturen und Muster sind. Der Effekt ist besonders stark, wenn diese Muster nicht senkrecht sind. Dies wirkt sich unmittelbar auf das 3D-Modell aus: es entstehen Zacken an den Polygon-Kanten, oder es treten unerwünschte Effekte wie beispielsweise der Moiré Effekt (siehe Abbildung 27) auf. [57]

Verschiedene Render-Algorithmen [M]

Rasterization

Über den Bildschirm wird ein Raster aus Polygon-Formen gespannt. Dabei werden meistens Dreiecke verwendet, da diese am einfachsten zu berechnen sind. In den Ecken der Dreiecke, den sogenannten Scheitelpunkten, sind die Informationen enthalten, die zur Darstellung eines 3D-Objektes in 2D benötigt werden. Dieser Prozess wird für das Echtzeit-Rendering genutzt. [61]

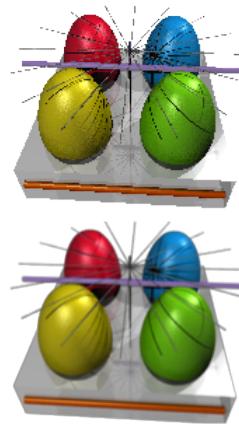


Abbildung 28: Rendering mit und ohne Anti-Aliasing [60]

Wire-Frame

Bei einem Wire-Frame wird ein 3D-Modell erstellt, in dem der Algorithmus versucht, lineare Merkmale wie Kanten oder Konturlinien zu verfolgen. Dabei werden auch verdeckte oder nicht sichtbare Teile inkludiert. [57]

Visible Line

Dieser Vorgang funktioniert ähnlich wie ein Wire-Frame, nur dass hierbei nur Linien gerendert werden, die tatsächlich von der Kamera sichtbar sind. Dieser sichtbare Bereich wird auch Visible Surface genannt. [57]

Raycasting

Beim *Raycasting* wird ermittelt, welches 3D-Objekt zuerst von einem sogenannten *Ray* getroffen wird. Ein Ray ist ein Strahl, der von der Position der Kamera ausgeht. Für jedes Pixel wird ein *Ray* durch den dreidimensionalen Raum geschickt. Dadurch können die Schnittpunkte der Objekte und der Rays ermittelt werden. Dabei werden alle Objekte mit einem sichtbaren Schnittpunkt angezeigt. Die Fläche normal dazu bestimmt die Schattierung. Dieser Prozess wird in 5.7.6 näher erläutert. [57]

Raytracing

Der *Raytracing* Prozess funktioniert ähnlich wie der Prozess des *Raycastings*. Zusätzlich werden hierbei jedoch auch verschiedene Oberflächen und Lichtquellen berücksichtigt. Dies funktioniert, indem ein *Ray* vom Algorithmus mitverfolgt wird. Dabei ist es möglich, zu erkennen ob ein *Ray*:

- durch eine lichtdurchlässige Oberfläche durchdringt
- von einem reflektierenden Objekt reflektiert wird
- oder ob ein Objekt von einer Lichtquelle getroffen wird und einen Schatten wirft

[61]

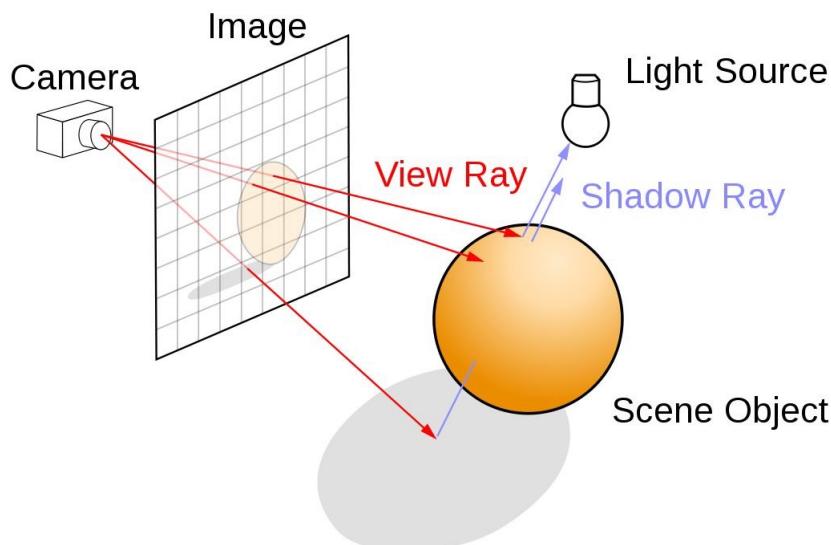


Abbildung 29: Vorgang des Raytracing [61]

Rendering in Three.js

Für das Rendern der 3D-Gallery wird der in Three.js inkludierte Renderer von WebGL (siehe ThreeJS 2.2.4) Referenz verwendet. Um das gerenderte 3D-Objekt in das DOM (siehe DOM 6.3) zu integrieren, wird das HTML-Element <canvas> verwendet. [62]

Listing 23: Canvas-Element in HTML

```
1   <canvas #threeCanvas style="width: 100%; height: 100%"  
        (window:resize)="onResize($event)"></canvas>
```

Dabei wird dieses DOM-Element als Angular-View-Child initialisiert. Dadurch können Änderungen am DOM erkannt und das betroffene Element neu definiert werden. [63]

Listing 24: Canvas als View-Child initialisieren

```
1   @ViewChild('threeCanvas') threeCanvas!: ElementRef;
```

Anschließend wird ein neuer WebGL-Renderer angelegt, dem der Canvas zugewiesen wird. Falls noch kein Canvas besteht, wird dieser automatisch neu erstellt. [62]

Listing 25: WebGLRenderer anlegen

```

1      this.renderer = new THREE.WebGLRenderer({
2          canvas: this.threeCanvas.nativeElement
3      });

```

Um die Render-Funktion des Renderers als Echtzeit-Rendering anzuwenden, wird eine Render-Schleife benutzt. Darin ist zum einen die Render-Funktion mit der Szene und der Kamera, die gerendert werden soll. In einer Szene befinden sich alle 3D-Objekte. Zum anderen befindet sich die Funktion requestAnimationFrame in der Schleife, wodurch die Szene jedes Mal neu gerendert wird, wenn der Bildschirm neu geladen wird. Dies geschieht üblicherweise 60-mal in der Sekunde. [64]

Listing 26: Animations-Schleife

```

1      animate = () => {
2          requestAnimationFrame(this.animate);
3          this.renderer.render(this.scene, this.camera!)
4      }

```

5.3.2 Lichtsetzung [M]

Um die Szene zu aufzuhellen müssen Lichtquellen angelegt werden. Dabei können ebenfalls unterschiedliche Optionen konfiguriert werden, um die Szene so realistisch wie möglich aussehen zu lassen. Das Licht wird jedoch nicht nur von der Lichtquelle beeinflusst, sondern auch von dem Objekt, das beleuchtet wird. Material und Oberfläche eines Objektes reagieren unterschiedlich auf den Einfall des Lichtes, wodurch zum Beispiel die Lichtreflektion stärker ausfällt. Dieses Konzept wird auch Global Illumination genannt. Three.js bietet eine Vielzahl von Lichtarten und Quellen:

- Das Standard-Licht, *Light* genannt, ist das Grundgerüst der anderen Lichttypen. Es kann die Farbe und Intensität des Lichtes individuell geändert werden. [65]
- Die *LightProbe* ist eine alternative Methode, Licht zu erzeugen. Dabei wird nicht direkt von einer Quelle das Licht ausgestrahlt, sondern es speichert die Lichtinformation ab. Dabei wird erst beim Rendern der Lichteinfall durch die Daten der LightProbe ermittelt. [66]
- Das *AmbientLight* leuchtet die gesamte Szene gleichmäßig aus, kann dabei jedoch keine Schatten werfen. [67]
- Das *DirectionalLight* wirft Licht parallel in eine bestimmte Richtung. Da es unendlich weit weg erscheint, wird es oft als Tages- oder Sonnenlicht verwendet. [68]

- Das *HemisphereLight* wird über der Szene positioniert und besitzt eine Himmel- und Bodenfarbe mit Verlauf. Diese Lichtquelle kann keine Schatten werfen. [69]
- Das *PointLight* wirft Licht von einem Punkt in alle Richtungen und simuliert den Lichtfall einer Glühbirne. Hierbei kann ebenfalls die Reichweite des Lichtes und ein Dimmeffekt bestimmt werden. [70]
- Das *RectAreaLight* strahlt Licht in Form eines Rechtecks aus. Damit können zum Beispiel Fenster simuliert werden. [71]
- Das *SpotLight* wirft Licht in Form eines Kegels. [72]

Das Point Light wurde in der 3D-Ausstellung wie folgt angewandt:

Listing 27: Lichtsetzung in der 3D-Ausstellung

```

1      const bulbGeometry = new THREE.SphereGeometry(.02, 16, 8);
2      const bulbLight = new THREE.PointLight( 0xffffffff, 3, 1000, 2 );
3      const bulbMat = new THREE.MeshStandardMaterial( {
4          emissive: 0xfffffff,
5          emissiveIntensity: 1,
6          color: 0x000000
7      });
8      bulbLight.add( new THREE.Mesh( bulbGeometry, bulbMat ) );
9      bulbLight.position.set( 0, 100, 0 );
10     bulbLight.castShadow = true;
11     this.scene.add( bulbLight );

```

Zunächst wurde ein Objekt angelegt, welches das Licht ausstrahlen soll. Anschließend wird die Lichtquelle und deren Material initialisiert. Um Schatten zu werfen, wird das Attribut `.castShadow` auf `true` gesetzt.

5.4 Fertige Landingpage [L][M]

Die Landingpage (siehe Abbildung 30) repräsentiert das Projekt, denn es ist das erste, was die jeweiligen Benutzer*innen von der Webanwendung zu sehen bekommt. Es war wichtig, ein modernes Design dafür zu entwickeln.

Durch eine logische Hierarchie soll der*die Benutzer*in gezielt durch die Anwendung geleitet werden. Zuerst gibt es eine Einleitung, um das Projekt zu erklären, danach erscheint ein großer Button, der dazu einlädt, das Produkt zu verwenden. Er verlinkt zu der Anmeldung. Danach kommt eine Auswahl aus den neuesten erstellten Ausstellungen. Damit kann sich der*die Benutzer*in einen Eindruck der möglichen Endergebnis machen.

Um zwischen den verschiedenen Unterseiten der Webseite zu wechseln, wurde am oberen Bildschirmrand zusätzlich eine Navigationsleiste implementiert. Diese sogenannte



Abbildung 30: Landing Page

Navbar besitzt je nach Unterseite verschiedene Designs. Diese ermöglicht ein angenehmes Navigieren auf jeder Seite (siehe 14). Auch der Footer muss auf jeder Unterseite passend angezeigt werden. Um das Design von Navbar und Footer auf jeder Seite individuell zu gestalten, wird jeweils ein Service in die entsprechende Komponente injiziert. Dadurch kann die Konfiguration am Design mit wenig Programmcode und Redundanz vorgenommen werden.

5.4.1 Erledigte User-Stories [L]

Bei der Fertigstellung der Landingpage wurden folgende User-Stories vollendet:

- Als erstmalige*r Besucher*in der Webseite möchte ich die benötigten Informationen über die Funktionen der Applikation verständlich erkennen können. Dazu erwarte ich mir:
 - Textstellen und Grafiken, die unser Projekt und die Funktionalitäten davon erklären.
 - einen Call-To-Action (CTA) Button, welcher den*die User*in dazu einlädt, seine eigene Ausstellung zu erstellen. Beim Betätigen wird der Benutzer bei bestehender Anmeldung zum Editor weitergeleitet, und ansonsten auf die Login-Seite verwiesen.

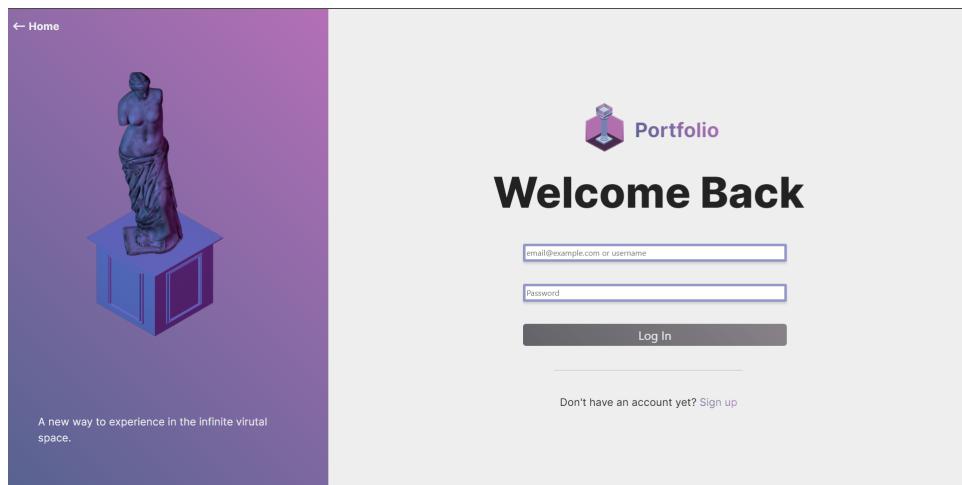


Abbildung 31: Projekt: Login Page

5.5 Userverwaltung [L]

Im Projekt wurde eine Userverwaltung eingebaut. Der*die User*in kann sich auf der Weboberfläche anmelden und registrieren. Je nach Registrierungsstatus (bestehende Anmeldung oder nicht) stehen mehr oder weniger Funktionalitäten zur Verfügung. Ein*e Besucher*in, die nicht angemeldet ist, kann alle Ausstellungen ansehen und auf der Suchunterseite nach Ausstellungen suchen. Ein*e registrierte*er User*in kann zusätzlich auch seine*ihr Profilseite sehen und Ausstellungen selbst erstellen und veröffentlichen. Die Navigationsleiste wurde so programmiert, dass sie dem*der User*in nur die Unterseiten zeigt, die diese durch ihren Registrierungsstatus auch besuchen kann.

Im folgendem Abschnitt 'Sign-, Log-In Funktionalitäten' wird die Implementation des Userverwaltung und die damit verbundenen Features erklärt.

5.5.1 Implementation im Frontend [L]

Nachdem der Json-Web-Token im Backend implementiert wurde, galt es diesen auch auf dem Frontend einzubauen. Der Json-Web-Token dient zur Authentifizierung des*der User*in und zum rollenbasierenden Schutz der API.

Sign-In und Login

Da der Json-Web-Token als Beweis dafür dient, dass der*die User*in authentifiziert ist, muss sich der*die Benutzer vor dem Erhalten des Token dafür erst auf der Weboberfläche registrieren oder anmelden.

Für das User-Interface ist eine Anmelde- und Registrierungsfunktionalität implementiert worden (siehe Abbildung 31). Die Anmelde- und Registrierungsfelder wurden mithilfe von reaktiven Formularen und Validatoren (siehe 5.5.1) umgesetzt. Die Validatoren bestimmen, dass ein Passwort mindestens 8 Zeichen besitzen muss, um akzeptiert zu werden.

Nachdem vom User eine valide Eingabe getätigt wurde, aktiviert sich der Anmelde- bzw. Registrierungsbutton und lässt sich betätigen.

Bei der Aktivierung ändert sich die Farbe des Buttons von Grau auf einen bunten blauen pinken Farbverlauf, um ein visuelles Feedback an den*die Benutzer*in zu senden. Nachdem der*die User*in die Eingabe bestätigt hat, indem sie*er die Schaltfläche drückt, werden die Anmelddaten über das HTTPS-Protokoll mithilfe des HTTP-Moduls (siehe 5.2.3) an den Server gesendet. Nach einer erfolgreichen Anmeldung bzw. Registrierung wird an die Weboberfläche eine Antwort geschickt, in der der Json-Web-Token enthalten ist. Danach wird der*die User*in auf die Profilseite weitergeleitet. Bei einem gescheiterten Anmelde- bzw. Registrierungsversuch wird der*die User*in durch eine Fehlermeldung informiert.

Nach der erfolgreichen Anmeldung bzw. Registrierung wird vom Server ein Json-Web-Token ausgestellt. Bei einem HTTP-Request vom Client zum Server kann der Json-Web-Token im Header des Requests platziert werden. Der Server kann diesen auslesen und so feststellen, ob ein*e Benutzer*in auch wirklich authentifiziert ist.

Nun gibt es aber mehrere Nachteile, die aus diesem Prozess entstehen. Der*Die User*in muss bei jedem neuen Besuch der Seite sowie bei einem Reload neu anmelden. Im Code muss bei jedem HTTP-Request (also jeder Kommunikation mit dem Server) der Json-Web-Token manuell in dem Header platziert werden. Dadurch entsteht im Code eine hohe Redundanz. Im folgenden Kapitel "Token-Verwaltung"(siehe 5.5.1) und "HTTP-Interceptoren"(siehe 5.5.1) wird beschrieben, wie mit diesen Problemen umgegangen wurde.

Formulare und Validation In Webanwendungen gibt es zahlreiche Formulare, die als eine der primären Kommunikationsschnittstellen zum Besucher*in dienen. Es gibt dafür die nativen HTML-Inputelemente, aber zu einer guten User-Experience gehört auch ein visuelles Feedback für den*die Benutzer*in bei Bearbeitung eines Formulars. Der Entwicklungsaufwand wächst allerdings mit der Anzahl der im Formular eingefügten Features.

Angular bietet verschiedene Implementierungsarten, um die visuelle Komponenten umzusetzen und den Entwicklungsaufwand zu minimieren. Bei diesen Ansätzen können mehrere Eingabedaten zentral ausgewertet und weiterverarbeitet werden und der Status des Formulars bei Änderungen und Fehlern visuell dargestellt werden. [12, Bookmonkey - 12 Formularverarbeitung und Validierung: Iteration IV]

Bei den Ansätzen unterscheidet man zwischen den reactive Forms und den Template-Driven-Forms.

Token-Verwaltung

Um das Problem, dass beim neu Laden der Webseite der Anmeldestatus bzw. der Json-Web-Token verloren geht, zu lösen, wird der Json-Web-Token im Locale-Storage des Browsers gespeichert. Im Codeausschnitt (siehe Abbildung 28) ist ein Teil des Authentifizierungsdienstes des Projektes zu sehen. Mit der Funktion *setSaveJWT* wird der ausgestellte Json-Web-Token ausgelesen und die darin enthaltenen Informationen, der Name und die ID des Benutzers, die ID des Tokens und das Ablaufdatum des Tokens im Locale-Storage (siehe im Absatz WebStorage API 5.5.1) gespeichert, damit die Daten auch nach dem neu Laden der Webseite erhalten bleiben. Die Funktion *isLoggedIn* gibt nach dem Aufruf den Status mit, ob eine Person angemeldet ist oder nicht. Dafür benutzt die Funktion das im Json-Web-Token enthaltene Ablaufdatum und vergleicht es mit dem aktuellen Datum. Zuletzt gibt es noch die *logout* Funktion, diese löscht die im Locale-Storage enthaltenen Json-Web-Token-spezifischen Daten.

Listing 28: auth.service.ts - Json-Web-Token und Localstorage

```

1  export class AuthService {
2      ...
3      setSaveJWT(value: any) {
4          let decodedJWTPayload = JSON.parse(atob(value.split('.')[1]))
5          localStorage.setItem("user", decodedJWTPayload.sub)
6          localStorage.setItem('id_token', value)
7          localStorage.setItem('expires_at', decodedJWTPayload.exp)
8          localStorage.setItem('user_id', decodedJWTPayload.userid)
9      }
10
11     isLoggedIn(): boolean {
12         if (localStorage.getItem('id_token') &&
13             localStorage.getItem('expires_at')) {
14             let temp = new Date().getTime()
15             const exp = Number(localStorage.getItem('expires_at'))
16             return temp < exp
17         } else {
18             return false;
19         }
20     }
21
22     logout() {
23         localStorage.removeItem("user")
24         localStorage.removeItem('id_token')
25         localStorage.removeItem('expires_at')
26         localStorage.removeItem('user_id')
27     }

```

```

27     ...
28 }

```

WebStorage API Das *WebStorage API* bietet verschiedene Möglichkeiten, Daten per Schlüssel-Werte-Paare im Web zu persistieren. Persistenz ist die Fähigkeit, Daten in einem nicht flüchtigen Speicher zu speichern, um so den Datenverlust beim Neustart des Systems zu verhindern. Das WebStorage API ist nicht Teil des DOM, sondern der globalen Web-Variable `window`. Die zwei Arten, Daten mittels der WebStorage API zu persistieren, sind der Locale-Storage und der Session-Storage. [73] [74]

Session-Storage Daten im Session-Storage werden je nach ihrem Ursprung getrennt aufbewahrt. Sie werden für die Zeit der Webseitensession gespeichert, wenn der Browser oder der Tab geschlossen wird, werden die Informationen gelöscht. [74]

Local-Storage Daten werden im Session-Storagepersistiert und bleiben somit nach dem Schließen des Browsers oder des Tabs erhalten. Nur mittels JavaScript oder durch ein Löschen des Webbrowsers-Caches können die Daten gelöscht werden. [74]

Allgemeine Informationen Beide Speichermethoden benutzen keine Server, um die Daten zu speichern, sondern den Cache des Webbrowsers. Das Speicherlimit hängt vom Webbrowser ab, doch beträgt es mindesten 5 MB und es können nur Zeichenketten gespeichert werden. [74]

HTTP-Interceptoren

HTTP-Interceptoren funktionieren in Angular als Zwischenschicht zwischen ausgehenden HTTP-Abfragen und eingehenden HTTP-Antworten und können diese manipulieren. HTTP-Interceptoren werden auf jeden Request angewendet. Es ist somit das perfekte Werkzeug, um in allen HTTP-Requests den JWT in den Header zu verpacken. [12, 10.3 Interceptoren: HTTP-Requests abfangen und transformieren]

Wird ein Request an den Server gesendet, wird die Verarbeitung vom Interceptor unterbrochen. Es wird überprüft, ob der Json-Web-Token verfügbar ist. Wenn das zutrifft, wird der ursprüngliche Request geklont und in der Authentifizierungsspalte des Headers wird die Json-Web-Token-Id hinzugefügt. Danach wird der Request wieder zum Server weitergeleitet (siehe Code 29).

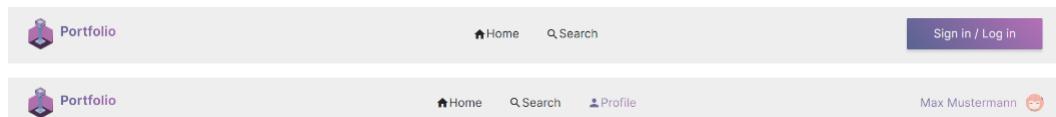


Abbildung 32: Navbar: authentifiziert vs noch nicht authentifiziert

Listing 29: auth.interceptor.ts - add JWT to Request Header

```

1  import { Injectable } from '@angular/core';
2  import {
3    HttpRequest,
4    HttpHandler,
5    HttpEvent,
6    HttpInterceptor
7  } from '@angular/common/http';
8  import { Observable } from 'rxjs';
9
10 @Injectable()
11 export class AuthInterceptor implements HttpInterceptor {
12
13   constructor() {}
14
15   intercept(request: HttpRequest<unknown>, next: HttpHandler):
16     Observable<HttpEvent<unknown>> {
17
18     const idToken = localStorage.getItem('id_token')
19     if (idToken) {
20       const cloned = request.clone(
21         {
22           headers: request.headers.set('Authorization', 'Bearer ' + concat(idToken))
23         }
24       )
25
26       return next.handle(cloned)
27     }
28
29     return next.handle(request);
30   }
31 }
```

Routing- und Navigations-Einschränkung

Zum Start dieses Entwicklungsschrittes war der Interceptor und die JWT Verwaltung fertig. Doch das Projekt brauchte eine besseres visuelles Feedback System um dem*der Kunden*in den Anmeldestatus zu zeigen.

Deswegen und um zu verhindern das Benutzer*in auf Unterseiten sind auf denen sie wegen ihres Registrationsstatus (noch nicht authentifiziert) noch nichts machen können, wurde je nach Anmeldestatus andere Routen und Informationen auf der Navigationsbar gezeigt (siehe Abbildung 32). Dafür wurde die Navigationsleistenlogik und die Navigationsleistenservice angepasst. Zusätzlich wurde AuthGuard verwendet um Routen zu sperren auf, die der*die Benutzer*in wegen dem Registrationsstatus noch keinen Zugriff hat.

5.5.2 Erledigte User-Stories [L]

Im Entwicklungsprozess des Usermanagement-System wurden folgende User-Stories vollendet:

1. Als Designer möchte ich einen Login, um meine Ausstellungen speichern und im Nachhinein immer öffnen zu können. Akzeptanzkriterien:
 - Man kann sich neu registrieren
 - Registrierung mittels Username und Passwort
 - Das Passwort muss überprüft werden beim Erstellen und mind. 8 Zeichen enthalten
2. Als User*in will ich andere Rechte haben, je nachdem ob ich angemeldet bin oder nicht. Akzeptanzkriterien: Wenn ein*e User*in angemeldet ist, kann er*sie:
 - Ausstellungen ansehen
 - Ausstellungen erstellen/löschenWenn ein ein*e User*in nicht angemeldet ist, kann er*sie:
 - Ausstellungen ansehen
 - keine Ausstellungen erstellen
 - keine Ausstellungen favorisieren

5.6 Content Creation [L]

Die Inhaltserstellungs-Funktion ist die Kernfunktion des Projekts. Die Content Creation definiert ein 3D-Portfolio. Ziel ist ein einfacher und intuitiver Konfigurationsprozess für ein 3D-Portfolios.

5.6.1 Datenstruktur

Die 3D-Ausstellungen lassen sich durch Daten definieren. Die Daten werden bei der Content Creation vom*n Benutzer*in konfiguriert.

Kategorie

Die Datenstruktur des Protfolios lässt sich logisch in vier Kategorien (siehe Abbildung 33) teilen. Die Kategorien sind:

- Metadaten - Die Metadaten sind beschreibende Daten des Portfolio, dazu gehören Name, zugehörige Kategorien, eine Beschreibung und ein Thumbnail.

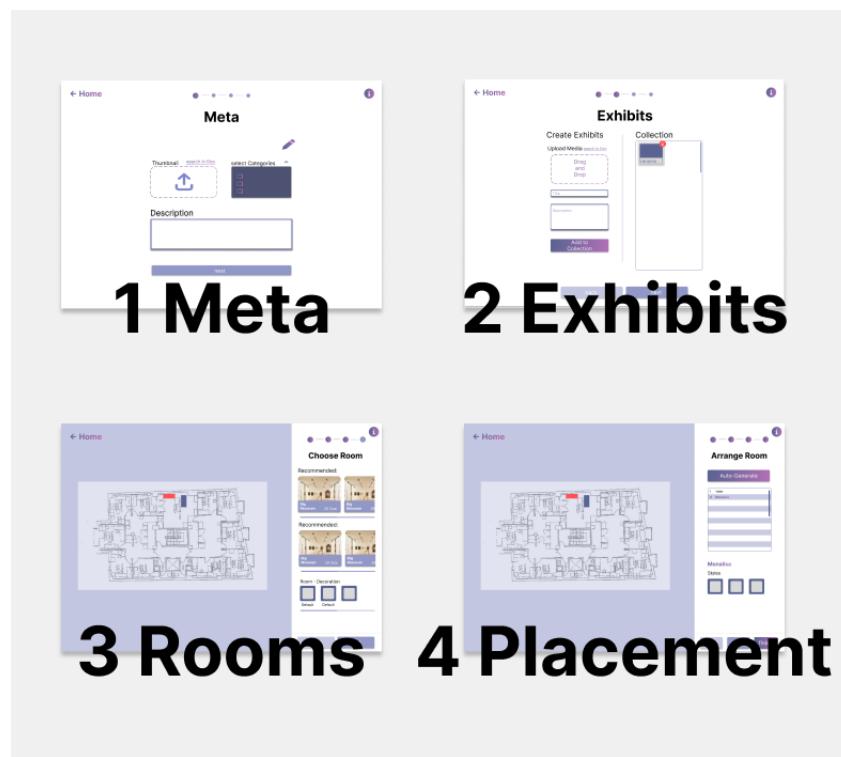


Abbildung 33: Inhaltserstellungstool

- Ausstellungsstücke - Die Ausstellungsstücke werden in dem 3D-Portfolio präsentiert. Die Ausstellungsstücke, Film-, Foto- und 3D Dateien, werden vom* von Benutzer*in hochgeladen und mit Zusatzinformation wie Namen und Beschreibung ergänzt.
- Raumdaten - Raumdaten bestehen aus virtuellen 3D-Daten des Raumes und weiteren Konfigurationsdaten wie den Positionen, an denen Ausstellungsstücke im Raum platziert werden können. Der Grundriss des Raumes kann jede beliebige Form besitzen.
- Ausstellungsplatzierungsdaten - Mit diesen Daten werden die Ausstellungsstücke mit dem virtuellen Raum verbunden. Außerdem ist in den Ausstellungsplatzierungsdaten die Dimension eines Ausstellungsstückes enthalten. Jedes Position besitzt eine definierbare Sockelart.

Ausprägungen In der Single-Page-Application werden die vier Kategorien durch Klassen (siehe Abbildung 34) definiert.

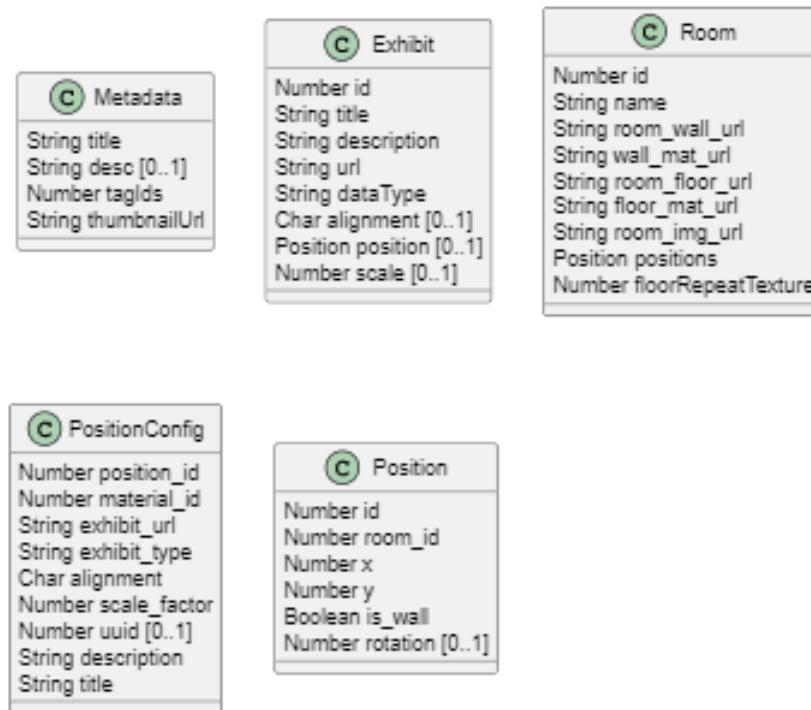


Abbildung 34: Inhaltserstellung Datenklassen

5.6.2 GUI

Profilepage

Auf der Profileseite (siehe Abbildung 35) sieht der*die Benutzer*in userspezifische Informationen wie den Profilnamen, das Profilfoto und die erstellten Ausstellungen. Zusätzlich kann der*die Benutzer*in hier neue Ausstellungen anlegen und bereits erstellte Ausstellungen löschen.

Add Button Der Content-Creation-Prozess beginnt damit, dass der Add-Button auf der Profileseite gedrückt wird. Danach wird der*die Benutzer*in zu dem Content-Creation-Tool auf der Single-Page-Application weitergeleitet. Dort startet er*sie den Konfigurationsprozess mit dem Metadaten-Formular.

Formulare

Die Datensammlung wurde auf vier Formulare, die jeweils einer der vier Datenkategorien (siehe 5.6.1) sammeln, aufgeteilt. Das hat das Ziel, dass der*die Benutzer*in sich im Konfigurationsprozess nur mit jeweils einer Portfolio-Daten-Kategorie beschäftigen muss und nicht überfordert wird. Dabei wurde sich am *Wizard-UI-Pattern* (siehe 5.6.2) orientiert.

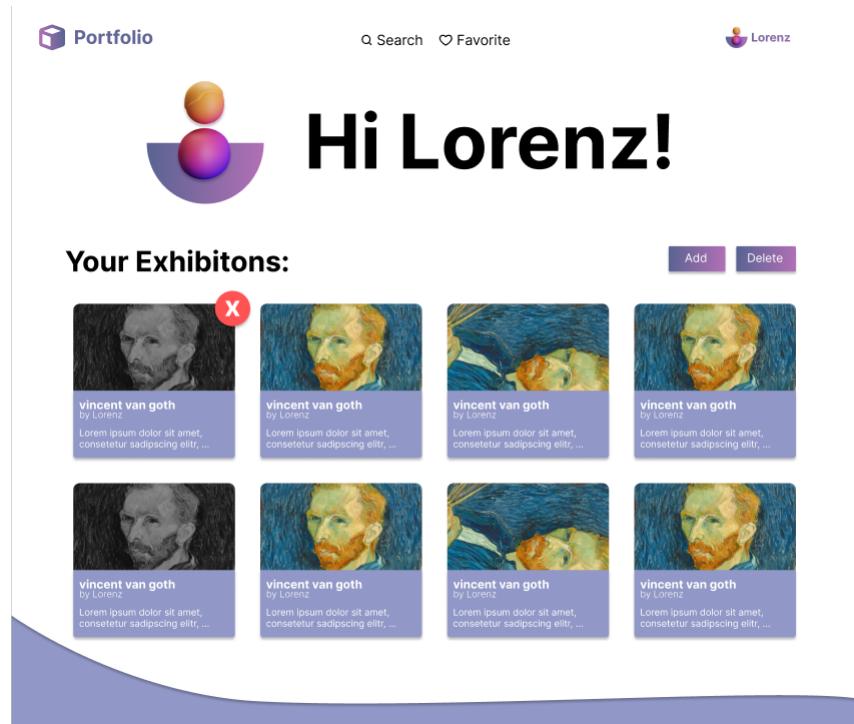


Abbildung 35: Profilepage

Metadaten-Formulare Im Metadaten-Formular werden die Metadaten der Ausstellung gesammelt.

Dabei wurden für die Sammlung der Namen, Beschreibungen und Kategorien der Ausstellung die reaktiven Formulare von Angular und für das Vorschaubild der Ausstellung die eigene Upload-Komponente benutzt. (siehe 5.6.3).

Exhibit-Formular Im Exhibit-Formular lädt der*die Benutzer*in die gewünschten, die eigenen digitalen Ausstellungsstück hoch.

Diese können nur hochgeladen werden, wenn ihre Datengröße kleiner als 50 MB ist und ihr Filetyp unterstützt wird. Es werden für 3D-Dateien GLTF-Files, für Video-Dateien MP4 und für Photo-Dateien JPG und PNG unterstützt (siehe 5.6.3)

Raum-Formular Im Raumformular wählt der*die User*in zwischen den vordefinierten Grundrissen aus.

In einer 3D-Ansicht neben den Auswahlmöglichkeiten sieht man eine Vorschau, wie der ausgewählte Raum ohne Ausstellungstücke aussieht.

Abhängig von der Anzahl der hochgeladenen Ausstellungsstücke werden dem*der User*in passende Räume vorgeschlagen.

Ausstellungs-Platzierungs-Formular Im Ausstellungs-Platzierungs-Formular werden die hochgeladenen Ausstellungstücke mit dem Grundriss der Ausstellung verbunden. In einer 3D-Ansicht neben den Auswahlmöglichkeiten sieht man eine Vorschau, wie der Raum mit den Ausstellungstücke aussieht.

Jeder Grundriss bietet dabei individuelle, im Vorfeld definierte, Positionen. An diesen können Ausstellungsobjekte platziert werden. Dies kann manuell einzeln oder bei allem automatisiert durchgeführt werden.

Je nach Datentyp des Ausstellungsstückes wird an der Position entweder ein Sockel oder ein Bilderrahmen mit dem Ausstellungstück erschaffen. Der Typ des Sockels und des Bilderrahmens können geändert werden. Die Größe eines 3D-Ausstellungsstückes wird normalisiert, damit alle Ausstellungstücke gleich groß sind. Dem*der User*in wird die Möglichkeit gegeben, diese Größe noch zu ändern.

Userinterface-Struktur / Wizard Design Pattern [L] Im Projekt wurde das Wizard-UI-Pattern benutzt, um den Konfigurationsprozess des Portfolios einfacher zu gestalten.

Wizard Begriffserklärung [L] Der Begriff 'Wizard' in der Softwareentwicklung leitet sich von Systemadministratoren, welche bei komplizierten Installationsprozessen halfen oder ganz übernahmen, ab. Mit der Zeit änderte sich der Begriff zu Software-Assistenz-Applikationen, welche dem*der User*in dabei helfen, die Software zu konfigurieren. [75, Ursprung des Begriffs Wizard]

Wizard-UI-Pattern [L] Im Web-Bereich gibt es zwei Arten Dateneintragungen zu verarbeiten: Formulare und Wizards. Ein *Wizard* ist eine eigene kleine Applikation, die den*die Benutzer*in durch eine Folge von Formularen führt und gegebenenfalls beim Ausfüllen unterstützt. Je nach Eingabe können Wizards die Formulare passend adaptieren.[76, Wizards: Definition and Design Recommendations]

Umsetzung des Wizards [L] Für die Umsetzung des Wizards wurde die Angular Material Komponente *mat-stepper* benutzt. Sie bietet eine solide Basis für den Wizard. Content kann sequentiell dargestellt werden, während dem*der User*in der aktuelle Status des Wizards durch eine Navigationsleiste angezeigt wird (siehe Abbildung 36). [77]

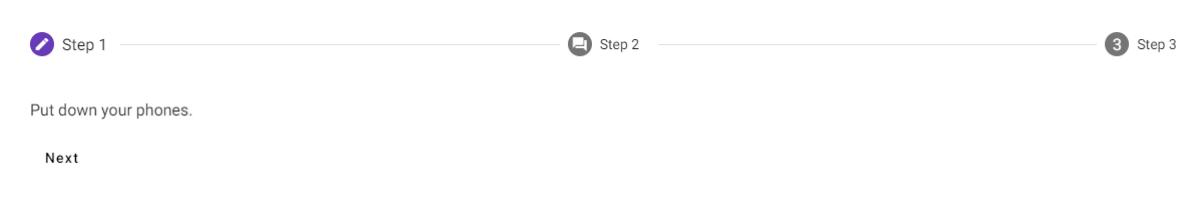


Abbildung 36: Angular Material: horizontaler Stepper [77]

5.6.3 Upload

Der Upload-Prozess wurde mit dem Gedanken entwickelt, hoch modular und anpassungsfähig zu sein.

Bei der Komponente sind akzeptable Filetypen und Filegrößen einstellbar. Werden die gesetzten Anforderungen an Filetypen und Filegrößen von den hochzuladenden Daten nicht erfüllt, so wird der Upload abgebrochen und die Komponente gibt eine Fehlermeldung. Bei einem erfolgreichen Hochladeprozess bekommt die Komponente eine URL vom Server. Unter der URL ist die hochgeladene Datei erreichbar.

5.6.4 Erledigte User-Stories [L]

In dem Entwicklungsprozess des Content-Creation-Tools wurden folgende User-Stories vollendet:

1. Als User*in will ich eine Profil-Unterseite haben, auf der ich userrelevante Informationen angezeigt bekomme. Dazu gehören der Name, das Profilfoto, und meine erstellten Ausstellungen. Diese möchte ich auch löschen können
2. Als User*in will ich bei der Erstellung einer Ausstellung zwischen verschiedenen Templates wählen können, um diese auf einfache Weise zu individualisieren - beispielsweise durch die Änderung des Podestes.
3. Als User*in will ich meine Daten auf den Server laden, um diese jederzeit innerhalb einer Ausstellung platzieren und integrieren zu können. Dazu möchte ich Bilder-, Video-, 3D-Dateien mit einer Größe von maximal 50 MB hochladen können. Bei falschem Upload benötige ich eine Fehlermeldung zur Information.
4. Als User*in will ich, dass meine Daten automatisch als Ausstellungsstücke in der Ausstellung angeordnet werden, damit die Nutzung für persönliche Bereiche unkompliziert möglich ist. Falls dies nicht möglich ist, soll eine Fehlermeldung angezeigt werden

5. Als User*in möchte ich die Reihenfolge und Platzierung meiner Werke innerhalb der Ausstellung adaptieren können. Dazu soll die Wahl zwischen vordefinierten Plätzen möglich sein.
6. Als User*in möchte ich meine Ausstellung abspeichern und löschen können.
 - Die Daten im Bezug auf die Ausstellungen werden auf dem Server gespeichert.
 - Bevor die Ausstellung gelöscht wird, soll ein Warnhinweis angezeigt werden, welcher noch bestätigt werden muss.
7. Als Benutzer*in möchte ich meinen Ausstellungen Tags zuordnen können, damit diese leichter gefunden werden können. Dazu soll ein vordefinierter Tag-Pool zur Verfügung stehen, aus welchem ich auswählen kann.

5.7 Interaktions-Seite [M]

Bei der Implementierung der Interaktions-Seite war uns wichtig, eine möglichst benutzerfreundliche Besucheransicht der vorhandenen Ausstellungen zu erstellen.

5.7.1 Verbindung der Interaktions-Seite mit der Landing-Page

Wenn ein*e Benutzer*in auf eine gewünschte Ausstellung klickt, muss sichergestellt werden, dass er auch zum richtigen Inhalt weitergeleitet wird. Dabei hilft das Routing mit Parametern (siehe 5.2.2). Jede Ausstellung besitzt eine eigene ID. Diese wird beim Auswählen einer Ausstellung in die URL geschrieben, um zu identifizieren zu können, welche Ausstellung im Moment besucht wird. Anhand dieser Informationen werden die Ausstellungsstücke vom Server abgefragt und mit der Logik, die im Kapitel Konfiguration der Ausstellung 5.6 beschrieben wird, richtig in den Raum geladen und platziert.

5.7.2 Bewegung im dreidimensionalen Raum

Um sich im Ausstellungsraum bewegen zu können, ist es nötig, den Besuchern*innen eine entsprechende Fortbewegungsmöglichkeit zu bieten. Three.js implementiert diese Funktion mittels Controls. Dabei ändert sich die Position und Rotation der Kamera durch Benutzereingaben. Die wesentlichsten Controls in Three.js sind

- *DragControls* - Der*die Nutzer*in kann sich mittels *DragNDrop* Interaktionen fortbewegen [78]

- *FirstPersonControls* - Der*die Nutzer*in bewegt sich wie in einem Videospiel mit den Tasten W, A, S und D fort. Dabei ist es möglich, das Blickfeld über die Maus zu steuern. [79]
- *OrbitControls* - Der*die Nutzer*in kann mittels linker Maustaste um ein bestimmtes Ziel kreisen. Durch die rechte Maustaste kann dieses Ziel geändert werden. [80]
- *TransformControls* - Den Nutzer*innen ist es möglich ein Objekt, ähnlich wie bei 3D-Modellierungs-Programme, durch Anfasser zu positionieren, skalieren und rotieren. [81]
- *PointerLockControls* - Diese Art von Controls funktioniert ähnlich wie *FirstPersonControls*. Hierbei werden ebenfalls die Tasten W, A, S, D benutzt, um sich fortzubewegen. Dabei wird die Maus verwendet, um sich umzuschauen. Jedoch kann die Maussteuerung durch einen *Pointer-Lock* gesperrt werden.[82]

Die beste Wahl für die Fortbewegung in der Ansicht der Erstellung eines Ausstellungsraumes, waren die *OrbitControls*. Sie eignen sich besonders, da die Benutzer*innen einen guten und schnellen Überblick über den gesamten Raum erhalten. Die Controls werden initialisiert, indem die Kamera angegeben wird, die gesteuert werden soll und das HTML-Element, welches für die Event-Listener (sieh Event-Listener 6.3) benutzt wird.

Listing 30: OrbitControls initialisieren

```
1   this.controls = new OrbitControls(this.camera, this.renderer.domElement)
```

Für die Bewegung der Besucher*innen in einem Ausstellungsraum kamen die *FirstPersonControls* und die *PointerLockControls* in Frage. Auch wenn sie im Grunde sehr ähnlich sind, besitzen sie verschiedene Vor- und Nachteile. Daher wurden im Entwicklungsprozess beide Varianten getestet und im Kontext des Projektes bewertet.

Vor- und Nachteile der FirstPersonControls

Die *FirstPersonControls* bieten eine einfache Einbindung in das Programm. Die Logik des Bewegens muss nicht erst programmiert werden, sondern ist bereits im Programm in den Controls vollständig implementiert. Dabei ist es direkt möglich, sich entweder durch die Tasten W, A, S, D, den Pfeiltasten oder durch Linksklick und Rechtsklick fortzusetzen. Außerdem bieten diese Controls mehr Konfigurationsmöglichkeiten als die

PointerLockControls. Jedoch funktioniert die Maussteuerung nicht gleich, wie sie bei First Person Spiele und Anwendungen üblich ist. Die Kamera dreht sich automatisch in die Richtung weiter, in welche die Maus zeigt. Dabei wird die Rotationsgeschwindigkeit von der Position der Maus bestimmt. Je weiter sich die Maus am Bildschirmrand befindet, desto schneller dreht sich die Kamera. An dieses Gefühl müssen sich Nutzer*innen oft erst gewöhnen.

Implementierung der FirstPersonControls

Zu Beginn werden neue Controls initialisiert. Dazu wird sowohl die Kamera, als auch das passende HTML Element angegeben, welches für den Event-Listener (siehe Event-Listener 6.3) benutzt wird.

Listing 31: FirstPersonControls initialisieren

```
1     this.controls = new FirstPersonControls(this.camera, this.renderer.domElement)
```

Anschließend werden die Controls wie gewünscht konfiguriert. Spezifische Konfigurationsmöglichkeiten sind:

- *activeLook* - bestimmt ob die Maussteuerung aktiviert sein soll oder nicht
- *autoForward* - bestimmt, ob sich die Kamera automatisch vorwärts bewegen soll oder nicht. Diese Einstellung ist bei den Controls der 3D-Ausstellung deaktiviert.
- *constrainVertical*, *verticalMin* und *verticalMax* - grenzen das vertikale Blickfeld ein
- *heightMin*, *heightMax* und *heightSpeed* - werden benutzt um die Bewegungsgeschwindigkeit im Zusammenhang mit der Kamerahöhe zu ändern
- *lookVertical* - bestimmt, ob es möglich ist sich vertikal umzuschauen. Diese Einstellung ist bei den Controls der 3D-Ausstellung deaktiviert.
- *lookSpeed* - gibt an mit welcher Geschwindigkeit sich der*die Benutzer*in umschauen kann
- *movementSpeed* - gibt an mit welcher Geschwindigkeit sich der*die Benutzer*in fortbewegen kann

Vor- und Nachteile der PointerLockControls

Die *PointerLockControls* bieten, im Gegensatz zu den *FirstPersonControls*, eine realistische und gewohnte Maussteuerung. Dabei stoppt die Kamera jedes Mal an der aktuellen Position, wenn der*die Benutzer*in nicht mit der Maus interagiert. Zudem können die Controls schnell durch ein Attribut gesperrt werden. Im entsperrten Zustand ist es jedoch nicht möglich mit der Maus zu interagieren. Das bedeutet, Benutzer*innen können sie sich entweder im Ausstellungsraum fortbewegen, oder mit einem Ausstellungsstück interagieren. Beides gleichzeitig ist nicht möglich. Der gesperrte Zustand wird oft auch auf unabsichtliche Weise aktiviert - beispielsweise durch das Drücken der Escape-Taste, das Wechseln vom Browser in ein anderes Programm oder beim Wechseln des Browser-Tabs.

Implementierung der PointerLockControls

Zu Beginn werden neue Controls ähnlich den *FirstPersonControls* initialisiert, indem die zu verwendende Kamera und ein HTML-Element zugewiesen werden.

Listing 32: PointerLockControls initialisieren

```
1   this.controlsVisitor = new PointerLockControls(this.camera,
      this.renderer.domElement)
```

Um sich fortbewegen zu können, muss nun jedoch die Logik dafür eigenständig implementiert werden:

Listing 33: Logik der PointerLockControls

```
1   this.onKeyDown = (event: KeyboardEvent) => {
2     switch (event.code) {
3       case 'KeyW':
4         this.controlsVisitor?.moveForward(2)
5         break
6       case 'KeyA':
7         this.controlsVisitor?.moveRight(-2)
8         break
9       case 'KeyS':
10        this.controlsVisitor?.moveForward(-2)
11        break
12       case 'KeyD':
13         this.controlsVisitor?.moveRight(2)
14         break
15     }
16   }
17   document.addEventListener('keydown', this.onKeyDown, false)
```

Um die Maussteuerung zu aktivieren müssen die Controls entsperrt werden:

Listing 34: Controls entsperren

```
1   this.controlsVisitor.lock()
```

Auch kann zusätzlich die Geschwindigkeit geändert werden, die angibt, wie schnell sich ein Benutzer im Raum umschauen kann. Zusätzlich kann der horizontale Kamerawinkel konfiguriert werden.

Fazit

Da die FirstPersonControls mehr Konfigurations und Interaktionsmöglichkeiten mit den Ausstellungsstücken bieten, wurde für die Fortbewegung als Besucher*in einer Ausstellung diese Art von Controls verwendet.[82]

5.7.3 ThreeJS Clock

Um die im vorherigen Kapitel erwähnten Controls zu updaten, werden sie in der Animations-Schleife (siehe Rendering in ThreeJs 26) wie folgt aufgerufen:

Listing 35: Controls updaten

```

1  animate = () => {
2      //..
3      this.controls?.update(this.clock.getDelta())
4      //..
5 }
```

Dabei überprüft die Update-Funktion, welche aktuellen Benutzereingaben betätigt worden sind, um gleichzeitig und flüssig, diese am Bildschirm des*der Benutzers*in anzuzeigen. Dies funktioniert, indem die Position der Kamera zu einem gewissen Zeitpunkt abgefragt wird. Um den aktuellen Zeitpunkt zu erhalten, wird ein Clock-Objekt verwendet. Dieses Clock-Objekt besitzt die Methode *getDelta()*, um die Sekunden zu erhalten, die vergangen sind, seit die Clock gestartet wurde oder die Methode *getDelta()* aufgerufen wurde. Außerdem startet die Methode *getDelta()* die Uhr, falls sie dies noch nicht bereits getan hat. Dadurch können jegliche Updates, die unter anderem durch den*der Benutzer*in, an der Szene vorgenommen worden sind, identifiziert und ausgewertet werden.

[83]

5.7.4 Border-Collision

Um den Benutzern*innen ein möglichst realistisches Gefühl für die Ausstellung zu geben, ist es wichtig, den Ausstellungsraum authentisch zu gestalten. Daher ist der Raum durch Wände abgegrenzt, wodurch es nicht mehr möglich ist, sich über den Raum

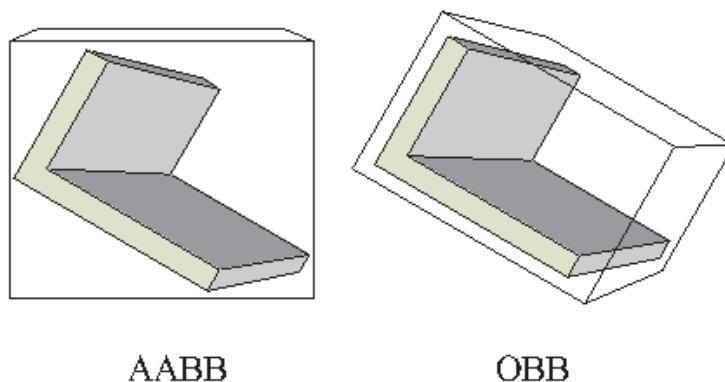


Abbildung 37: Der Unterschied zwischen AABB und OBB [84]

hinaus zu bewegen. Um eine Kollision mit der Wand zu berechnen, gibt es durch die Three.js Bibliothek einige Möglichkeiten.

5.7.5 Bounding Box

Der erste Ansatz der Umsetzung war das Erstellen einer Bounding Box. Dabei kann zwischen zwei verschiedenen Arten unterschieden werden.

- Axis Aligned Bounding Box (AABB)

Zum einen werden Axis Aligned Bounding Box verwendet, um eine Box rund um das 3D-Objekt zu erstellen, die sich nicht an die Rotation des Objekts anpasst.

- Oriented Bounding Box (OBB)

Die Oriented Bounding Box weist eine ähnliche Funktionsweise auf, unterscheidet sich aber darin, dass sie sich an die Achsen des Objekts anpasst

Dies stellte uns in der Entwicklung jedoch vor ein Problem. Da sich die Bounding Boxen jedoch über den ganzen 3D-Raum erstrecken, wird eine Kollision direkt berechnet, nachdem der*die Benutzer*in den Raum betritt. Da sich die Bounding Boxen nicht an jede einzelne Wand anpassen konnten, musste ein anderer Lösungsweg gefunden werden.

Zweiter Ansatz

Um eine Veränderung der Position des*der Benutzers*in festzustellen, muss die Veränderung der Kameraposition evaluiert werden. Bei der Initialisierung der Kamera wird eine Kopie erstellt. In der Animate-Funktion wird eine Veränderung der Kamera überprüft, indem die Positionen der Kamera mit der Kopie verglichen werden. Die Kamera nimmt

dabei immer eine neue Position ein, wenn sich der*die Benutzer*in im Raum bewegt, während die Kopie dabei die alte Position der Kamera einnimmt. 36 Jedes Mal wenn eine Veränderung geschieht, wird überprüft, ob die Kamera mit der Wand kollidiert. Dies geschieht, indem ein Raycast mit den Positionen der Kamera und der Kopie initialisiert wird ??.

Listing 36: Methode zum vergleichen der Position der Kameras

```

1  animate = () => {
2    //..
3    this.controls?.update(this.clock.getDelta())
4    if (this.mode != "create"){
5      var cameraChanged = this.compareCameras(this.camera!, this.camera2!)
6      if (cameraChanged){
7        this.detectCollision()
8      }
9    }
10   //..
11 }
12
13 }
14
15 compareCameras(camera: PerspectiveCamera, camera2: PerspectiveCamera): boolean{
16   if(camera.position.x == camera2.position.x && camera.position.y ==
17     camera2.position.y){
18     return false
19   }else{
20     return true
21   }
22 }
```

Far und Near

Die Attribute Far und Near werden dafür verwendet, um die Objekte, die im Ray liegen, einzugrenzen. Dies wird durch die Methode des Clippings realisiert (siehe Clipping 5.3.1). Dabei können die Werte nicht negativ sein und der Far-Wert muss größer als der Near-Wert sein. Um die Kollision erst direkt am Ursprungsort, im Falle der Kamera, des Rays zu berechnen, wird der Far-Wert auf 100 gesetzt.

Nachdem der Raycast (siehe 5.7.6) korrekt initialisiert und angewandt wurde, muss bei Berührung mit der Wand richtig damit umgegangen werden. Dabei wird die Bewegung des Benutzers gestoppt. Um diese wieder zu starten, muss sich der*die Benutzer*in weg von der Wand begeben. Überprüft wird dies nach jeder Benutzer*inneneingabe mit einem Event-Listener. Falls nach dieser keine Berührung mehr mit einer Wand besteht, wird die Bewegung fortgesetzt und der*die Besucher*in kann sich wieder frei im Raum bewegen.

Listing 37: Methode zum ermitteln einer Kollision mit der Wand

```

1 detectCollision(){
2   this.collisionRaycaster.set(this.camera!.position,
3     this.camera2!.position.normalize())
4   this.collisionRaycaster.far = 100
```

```

5      const intersects =
6          this.collisionRaycaster.intersectObjects(this.scene.children)
7      if(intersects.length > 0){
8          this.clock.stop()
9      }

```

5.7.6 Interaktion

Als Besucher*in einer Ausstellung soll es möglich sein, benutzerfreundlich mit den Ausstellungsstücken zu interagieren. Dazu zählen die Markierung des ausgewählten Ausstellungsstückes und die dazugehörige Detailansicht.

Raycasting

Raycasting ist eine der wichtigsten Komponenten, um mit einem 3D-Modell interagieren zu können. Wie das Konzept des Raycastings funktioniert, wurde bereits im Kapitel Rendering erklärt (siehe Rendering 5.3.1) Referenz. Three.js besitzt einen eigenen, individuellen Raycaster. Er wird vor allem dafür genutzt, um herauszufinden, auf welchem 3D-Objekt sich der Mauszeiger aktuell befindet. Es gibt mehrere Möglichkeiten einen Raycaster zu erstellen. Die erste Option stellt die Initialisierung über einen Konstruktor dar. Dabei wird angegeben, von welchen Koordinaten der Ray ausgeht und in welche Richtung er geschickt wird. Alternativ kann ein Raycaster direkt von einer schon existierenden Kamera einen Ray wegschicken. Dafür wird die Methode `.setFromCamera` aufgerufen, wo die Koordinaten des Mauszeigers und die zugehörige Kamera angegeben werden. Alle vom Ray getroffenen 3D-Objekte werden Intersections genannt. Diese 3D-Objekte können durch die Methode `intersectsObject` ausgelesen und in ein Array und gespeichert werden. [85]

Hover-Effekt

Um zu erkennen, welches Ausstellungsstück momentan vom Mauszeiger berührt wird, wird ein oben angesprochener Raycaster verwendet. Zu Beginn werden die Koordinaten des Mauszeigers in einem zweidimensionalen Vektor gespeichert. Ein Event-Listener wird hierbei jedes Mal aufgerufen, wenn sich der Mauszeiger bewegt. Darin können dann die kalkulierten Positionen des Mauszeigers dem Vektor zugewiesen werden.

Listing 38: Aktuelle Koordinaten des Mauszeigers einem 2D-Vektor zuweisen

```

1
2     pointer = new THREE.Vector2()
3

```

```

4     window.addEventListener( 'pointermove', (event: PointerEvent) => {
5         // calculate pointer position in normalized device coordinates
6         // (-1 to +1) for both components
7         this.pointer.x = ( event.clientX / window.innerWidth ) * 2 - 1
8         this.pointer.y = - ( event.clientY / window.innerHeight ) * 2 + 1
9         this.hoverExhibit()
10    });

```

Anschließend wird ein neuer Raycaster mit dem zweidimensionalen Vektor und der bereits existierenden Kamera erstellt.

Listing 39: Neuen Raycaster anlegen

```

1
2     this.raycaster.setFromCamera(this.pointer, this.camera! )

```

Daraufhin können alle 3D-Objekte die vom Ray getroffen wurden, in einem Array gespeichert werden.

Listing 40: Intersected Objects auslesen

```

1
2     const intersects=this.raycaster.intersectObjects(this.scene.children)

```

Um die Ausstellungsstücke farblich zu umranden, müssen sie zunächst ausgelesen werden. Dabei iteriert eine For-Schleife durch alle Ausstellungsstücke. Um zu erkennen, ob ein Intersect ein Ausstellungsstück ist, werden die jeweiligen IDs miteinander verglichen. Um dem 3D-Objekt einen Bloom-Effekt, also einen farbigen Schein als Umrandung, hinzuzufügen, wird das Konzept des Post-Processings verwendet.

Listing 41: Intersects als Ausstellungsstück identifizieren

```

1
2     for (let value of values) {
3
4
5     if (value.uuid != null) {
6       if (value.uuid == intersects[0].object.parent?.parent?.parent?.uuid || value.uuid ==
7           intersects[0].object.uuid) {
8
9         const object = this.scene.getObjectByProperty('uuid', value.uuid);
10        const renderPass = new RenderPass( this.scene, this.camera! );
11        this.composer = new EffectComposer(this.renderer!)
12        this.composer.addPass( renderPass );
13
14        const outlinePass = new OutlinePass( new THREE.Vector2( window.innerWidth,
15          window.innerHeight ), this.scene, this.camera! );
16        this.composer.addPass( outlinePass );
17
18        this.addSelectedObjects(object!);
19
20        outlinePass.selectedObjects = this.selectedObjects;
21      }
22    }
23  }

```

Um den Effekt wieder zu entfernen, sobald ein anderes Ausstellungsstück mittels Maus berührt wird, wird ein Array verwendet, das das alte Objekt herausgibt und das Neue hinzufügt.

Listing 42: Hover-effekt wieder entfernen

```

1
2  addSelectedObjects( object: Object3D ){
3      this.selectedObjects = [];
4      this.selectedObjects.push(object)
5  }

```

Postprocessing

Post-Processing ist eine Render Methode, die für zahlreiche 3D-Effekte, wie beispielsweise Anti-Aliasing, Tiefenschärfe und andere 3D-Effekte, genutzt wird. Dabei wird zuerst die Szene mit den gewünschten 3D-Objekten gerendert und im Speicher der Grafikkarte gesichert. Anschließend werden beim Post-Processing verschiedene Filter angewandt, diese gerendert und dem*der Benutzer*in angezeigt. In Three.js wird dafür ein *EffectComposer* verwendet. Dieser enthält ein Array von allen Effekten, die angewandt werden und dann mittels WebGL-Renderer in einer bestimmten Reihenfolge gerendert werden. Für die visuelle Anzeige sind Post-Processing-Passes zuständig. Zu Beginn wird ein RenderPass angelegt, der die zugehörige Kamera und Szene rendert. Anschließend können verschiedene Effekt-Passes, wie zum Beispiel der OutlinePass oder der GlitchPass, hinzugefügt werden. [86]

Detailansicht

Um beim Klicken auf ein Ausstellungsstück zur gewünschten Detailansicht zu kommen, wird ebenfalls ein Raycaster benutzt. Die Logik funktioniert ähnlich dem Hover-Effekt. Die Koordinaten des Mauszeigers werden diesmal jedoch bei jedem Mausklick aktualisiert. Auch hier wird eine For-Schleife genutzt, um das Array der Ausstellungsstücke durchzuiterieren. Dadurch kann identifiziert werden, welches Ausstellungsstück von dem*der Benutzer*in angeklickt wurde. Dabei werden die Informationen des Ausstellungsstückes an ein Dialogfenster geschickt, welches als Detailansicht fungiert.

Listing 43: identifizieren des geklickten Ausstellungsstückes

```

1
2  for (let value of values) {
3      if (value.uuid == intersects[0].object.parent?.parent?.parent?.uuid) {
4          if (value.uuid != null) {
5              const object = this.scene.getObjectByProperty('uuid', value.uuid);
6              this.objectDescription = value.description
7              this.objectTitle = value.title
8              this.objectUrl = value.exhibit_url
9              this.openDialog('1000ms', '300ms')
10         }
11     }
12 }

```

Geöffnet wird das Dialogfenster über eine eigene *openDialog-Funktion*. Dabei wird eine neue Instanz von einem Dialogfenster angelegt, welches Angular-Materials verwendet. Dabei wird angegeben, welche Größe und Breite das Fenster haben soll. Auch wird die Dauer der Animation des Öffnen und Schließen bestimmt. Um die Daten des Ausstellungsstücks auch im Dialogfenster zur Verfügung zu stellen, werden diese beim Initialisieren angegeben.

Listing 44: initialisieren des Dialogfensters

```

1  openDialog(enterAnimationDuration: string, Fertige Fertige
2      LandingLandingexitAnimationDuration: string): void {
3      const dialogRef = this.dialog.open(ExhibitDialog, {
4          maxWidth: '100vw',
5          maxHeight: '50vh',
6          width: '100%',
7          height: '100%',
8          data: {description: this.objectDescription, title: this.objectTitle,
9                 objectUrl: this.objectUrl},
10            enterAnimationDuration,
11            exitAnimationDuration,
12        });
13    //...
14 }
```

Wenn sich der*die Benutzer*in in der Detailansicht befindet, soll die Animation im Hintergrund gestoppt werden. Dies erfolgt durch die Methode *cancelAnimationFrame*, wodurch die Animations-Schleife angehalten wird. Um den aktuellen Zeitpunkt zu erhalten, zu dem die Animation gestoppt wird, wird die *requestAnimationFrame* Methode verwendet. Diese wird in der Animations-Schleife aufgerufen und dabei wird der aktuelle Frame der Animation zurück geliefert. Um sich nicht mehr weiter in der Ausstellung fortbewegen zu können, wird die Clock (siehe Clock 5.7.3) gestoppt, die für die Three.js Controls benötigt wird. Nachdem der*die Benutzer*in das Dialogfenster wieder schließt, wird die Animations-Schleife und die Clock wieder fortgesetzt.

Ein Problem ist, dass der Raycast nicht automatisch deaktiviert wird, wenn man sich in der Detailansicht befindet. Das bedeutet, dass es dem*der Benutzer*in weiterhin möglich ist, im Hintergrund weitere Instanzen des Dialogfensters zu öffnen. Dies führt zu einer Duplikierung der Detailansicht. Daher wird anhand einer Boolean-Variablen festgestellt, ob das Dialogfenster bereits geöffnet ist, um den Raycast zu deaktivieren.

Listing 45: Öffnen und Schließen des Dialogfensters

```

1  openDialog(enterAnimationDuration: string, exitAnimationDuration: string): void {
2      //..
3      this.dialogOpen = true
4      cancelAnimationFrame(this.animationid!)
5      this.clock.stop()
6
7      dialogRef.afterClosed().subscribe(r => {
8          this.dialogOpen = false
9          this.animate()
10         this.clock.start()
```

```

11      })
12

```

Das Dialogfenster ist eine eigene Angular-Komponente. Im Konstruktor der Komponente werden die Daten des Ausstellungsstückes injiziert, um es anschließend in einer detaillierten Ansicht zu rendern. Dabei werden eigens für die Detailansicht ein neuer Canvas angelegt, eine neue 3D-Szene gerendert und mittels Animations-Schleife neue Orbit-Controls (siehe Controls 5.7.2) initialisiert. Dadurch kann das 3D-Objekt frei betrachtet werden. Falls beim Konfigurieren der Ausstellung, dem Ausstellungsstück, eine Beschreibung mitgegeben wurde, wird diese in diesem Dialogfenster angezeigt.

Listing 46: Constructor-Injection in der Dialog Komponente

```

1   constructor(@Inject(MAT_DIALOG_DATA) public data: any, public dialogRef:
  MatDialogRef<ExhibitDialog>) {}

```

Navigation

Benutzer*innen können ebenfalls direkt in der Detailansicht zwischen den einzelnen Ausstellungsstücken wechseln. Dadurch müssen Benutzer*innen die Detailansicht nicht ständig verlassen, wodurch Zeit und Benutzereingaben eingespart werden können. Außerdem kann dadurch eine Ausstellung, ähnlich wie bei dem Programm PowerPoint, gut für Präsentationszwecke genutzt werden.

5.7.7 Erledigte User-Stories [L]

In dem Entwicklungsprozess der Interaktionsmethoden wurden folgende User-Stories vollendet:

1. Als User*in möchte ich mich auf verschiedene Arten in der Ausstellung bewegen können. Dies soll sowohl per Slideshow (mittels Vorwärts/Rückwärts Pfeilen) oder auch mittels Touch oder Click möglich sein - inspiriert von der Google Maps Street View. o. Ä.
2. Als User*in möchte ich auf das Ausstellungsstück klicken können, um weitere Informationen (z.B.: Titel, Künstler, Jahr, ...) zu erhalten. Dabei soll die Ansicht des Ausstellungsstücks vergrößert werden.

5.8 Such-Seite [M]

Auf der Such-Seite können Benutzer*innen nach bestimmten Ausstellungen suchen. Dabei kann nach dem Titel einer Ausstellung, nach dem Ersteller einer Ausstellung oder nach bestimmten Kategorien gefiltert werden. Bei der Suche handelt es sich um eine Vorschlagssuche, auch Typeahead-Suche genannt. Dabei werden schon während der Benutzereingabe mögliche Suchergebnisse angezeigt.

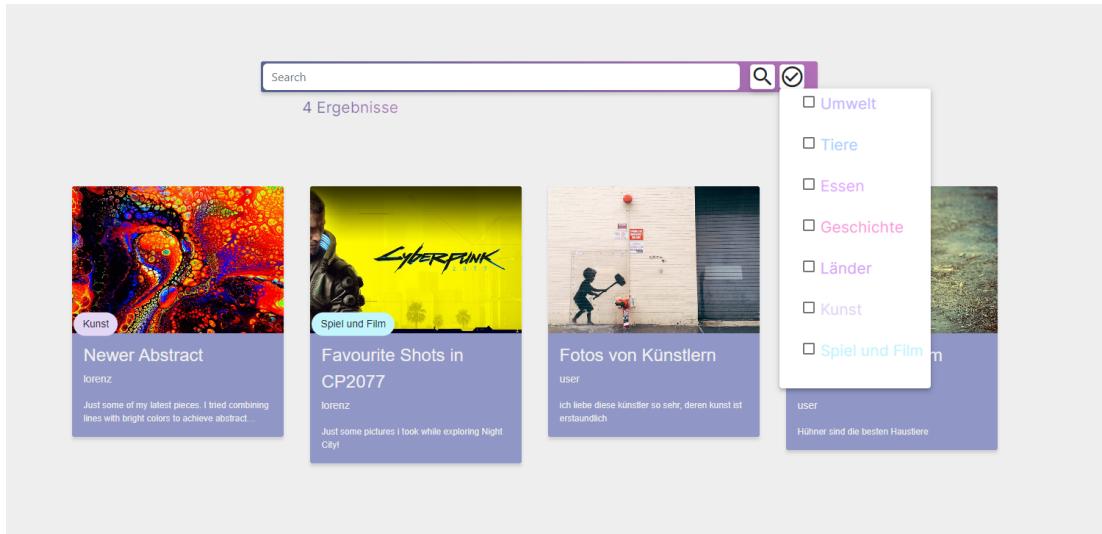


Abbildung 38: Search-Page

Falls der*die Nutzer*in noch keine Suche gestartet hat, werden ihm alle Ausstellungen angezeigt. Diese werden über die Datenbank und den HTTP-Service (siehe Angular Services 5.2.3) in die Benutzeroberfläche geladen.

Der*Die Benutzer*in kann in einem Eingabefeld einen beliebigen Suchbegriff eingeben. Nach jedem Buchstaben wird über einen Event-Listener (sieh Event-Listener 6.3) diese Eingabe überprüft.

Listing 47: Eingabefeld

```

1
2  <input type="search" class="form-control rounded" placeholder="Search" #input
   (keyup)="keyUp$.next(input.value)">

```

Anschließend wird die Logik der Suche implementiert. Um mehrere Operatoren in die Suchfunktion zu integrieren, wird eine Pipe benötigt. Anschließend wird ein Filter angewendet, um erst dann ein Suchergebnis zu liefern, wenn der*die Benutzer*in mindestens 3 Buchstaben in das Suchfeld eingegeben hat. Damit nicht nach jeder Tastaturabgabe eine Anfrage an die Datenbank geschickt wird, wird der Operator *debounceTime* angewandt. Dabei wird erst dann eine Anfrage gesendet, nachdem

der*die Benutzer*in eine gewisse Zeit nichts mehr in das Eingabefeld geschrieben hat. Durch die Methode `distinctUntilChanged` werden außerdem keine unnötigen Anfragen an den Server gesendet, falls sich der Suchbegriff nicht geändert hat. Da hierbei eine Subscription in einer Subscription vorhanden ist, wird ohne einen Flattening-Operator wieder der gleiche eingegebene Suchbegriff zurückgeliefert und keine Suchergebnisse. Daher wird der Operator `switchMap` verwendet, der zusätzlich alle Anfragen an den Server abbricht, sobald sich die Sucheingabe, durch Einwirken des*der Benutzers*in, ändert. Somit können die vom Server gesendeten Suchergebnisse direkt in ein Array von Ausstellungsstücken gespeichert werden. Dabei werden, wie bei einer Vorschlagssuche, die gefilterten Ausstellungen dem Benutzer angezeigt und bei Veränderung des Suchbegriffs aktualisiert.

Listing 48: Die Such-Pipe mit den Filter-Operatoren

```

1      this.keyUp$.pipe(
2          filter(term => term.length >= 3),
3          debounceTime(500),
4          distinctUntilChanged(),
5          switchMap(searchTerm => this.galleryService.getAllSearch(searchTerm)),
6          subscribe(exhibitions => this.searchResults = exhibitions)
7

```

5.8.1 Filtern mittels Kategorien

Zusätzlich zu der Suche über das Eingabefeld, kann der*die Benutzer*innen Ausstellungen zusätzlich filtern. Eine Ausstellung kann bei ihrer Konfiguration, keiner, einer oder mehrerer Kategorien zugeordnet werden. Alle Kategorien werden in einem Menü aufgelistet und können durch eine Checkbox für das Filtern ausgewählt werden. Beim Auswählen einer Kategorie wird diese einem Array aus Kategorien hinzugefügt. Falls der*die Benutzer*in eine Kategorie wieder abwählt, wird diese vom Array entfernt.

Listing 49: Auswählen und Abwählen der Kategorien

```

1      addCategory(id: number){
2          if(!this.selectedCategories.find(c => c == id)){
3              this.selectedCategories.push(id)
4          }else{
5              for( var i = 0; i < this.selectedCategories.length; i++){
6                  if ( this.selectedCategories[i] === id) {
7                      this.selectedCategories.splice(i, 1);
8                  }
9              }
10         }
11     }

```

Beim Schließen des Menüs wird automatisch nach den ausgewählten Kategorien gefiltert.

Listing 50: Filter von Kategorien anwenden

```

1
2     onMenuClose(){
3         this.filter_icon = "filter_alt";
4         let searchString = "";
5         for(let i = 0; i < this.selectedCategories.length; i++){
6             searchString += this.selectedCategories[i] + ","
7         }
8         if(this.selectedCategories.length > 0){
9             this.galleryService.getExhibitonByIds(searchString).subscribe(e => {
10                 this.searchResults = e
11             })
12         }else{
13             this.galleryService.getAllExhibitions().subscribe(res =>
14                 this.searchResults = res);
15         }
}

```

5.8.2 Erledigte User-Stories [L]

In dem Entwicklungsprozess der Suchseite wurden folgende User-Stories vollendet:

- Als Besucher*in möchte ich eine Suchseite haben, um Designer*innen und deren Ausstellungen finden zu können. Ich will, dass die Suchseite durch einen CTA-Button aufrufbar ist. Außerdem sollen auf der Suchseite unterschiedliche Optionen zur Suche sein. Das Suchergebnis soll ansprechend dargestellt werden.
- Als Besucher*in der Webseite will ich beim Suchen filtern können, um für mich die relevantesten Ergebnisse zu bekommen. Die Filtermöglichkeiten sollen Tags, Favoriten und das Erstellungsdatum beinhalten.

6 Zusammenfassung [L]

In diesem Kapitel wird das Projekt noch einmal revue passiert, die Probleme und ihre Lösungen beschrieben, die Untersuchung der Anliegen abgeschlossen und mehrere Erweiterungsvorschläge gegeben für die Weiterentwicklung des Projektes.

6.1 Probleme und Lösungsstrategie

6.1.1 Angular und Design Frameworks [L]

Die erste Herausforderung war, dass Design von Angular-Material-Komponente anzupassen.

Das Angular Webframework kapselt eigene Komponenten ab, damit sie sich nicht gegenseitig mit ihren Stylingbefehlen beeinflussen. Angular verwendet für die Einkapselung eine virtuellen Dom oder auch eine, nativen vom Browser unterstützen, ShadowDOM-API. [87]

Es gibt Situationen, in denen übergeordnete Komponenten das Styling untergeordneter ändern müssen. Besonders bei Angular Materials, einem UI- und Komponenten-Framework von Angular, kommt es oft dazu, dass die Angular-Material-Komponente meistens aus vielen kleinen Komponenten aufgebaut wird. Wenn nun der Style angepasst werden muss, damit die Material Komponente mehr zu der Corporate Identity des Produktes passt, ist das schwierig umsetzen.

Um dieses Problem zu lösen, gibt es zwei Möglichkeiten - `::ng-deep` und das globale Stylesheet.

`::ng-deep` [L]

`::ng-deep` durchbricht den virtuellen DOM und ShadowDOM und erlaubt es in Kombination mit weiteren Styleselektoren von der Eltern-Komponente auf alle Kinder-Komponenten zuzugreifen. (siehe Code Beispiel 51)

Listing 51: Parent.component.scss - Changing Styling in Child Components by using `:ngdeep`

```

1  ::ng-deep app-child-component{
2      -- change styling of child component
3      background-color: pink;
4  }

```

`::ng-deep` funktioniert, indem es die Einkapselung der Komponente ausschaltet und jeden Selektor (in Kombination mit `::ng-deep`) zu einem globalen Style befördert. Das kann, jedoch zu unvorhersehbaren Fehlern und Problemen führen. Empfehlenswert ist es, wenn man einen globalen Wert verändern will, das auch im globalen Stylesheet zu machen. Wenn der*die Entwickler*in Zugriff auf die Komponente hat und sie ändern kann, gibt es keinen Grund `::ng-deep` zu verwenden. Anpassungen können direkt in der Komponente gemacht, oder durch Input eine Interaktionsmöglichkeit hinzugefügt werden. Bei externen Komponenten wie Angular Material muss wohl eine Style-Überschreibung mittels `::ng-deep` oder dem globalen Stylesheet erfolgen, weil der*die Entwickler*in keinen Zugriff auf die Komponente hat. [88, 89]

6.1.2 Schnittstellen [Team]

Ein großes Problem bei der Entwicklung war die späte Fertigstellung des Servers. Dadurch konnten Funktionen der Single-Page-Application nicht mit realen Daten getestet werden. Der Lösungsversuch war es, Mockup-Daten zu verwenden und damit die Daten des Servers zu simulieren.

Zusammenfassend kann gesagt werden, dass dies keine gute Lösung darstellte. Arbeitsschritte mussten mehrere Male erledigt werden, wodurch sich der Aufwand erhöhte. Besser wäre, eine bessere Kommunikation mit der Backend-Entwicklung aufrechtzuerhalten und das Backend und Frontend schnellstmöglich zu verbinden.

6.2 Zielerreichung [Team]

Die im Projekt angestrebten Ziele, Untersuchungskriterien, User Stories und Meilensteine wurden erfüllt.

Das Endergebnis bietet Besucher*innen der Webseite eine Plattform, auf der sie sich und ihre Kunst präsentieren und sich informativ weiterbilden können. Das Produkt besteht aus insgesamt drei Kernbereichen: Der Webseite, der Ausstellung und dem Content Creation Tool.

Auf der Startseite des Web Applikation wird dem*der Benutzer*in das Projekt vorgestellt und zum Mitmachen angeregt. Des Weiteren wurde ein User Management System erstellt, bei dem sich Benutzer*innen registrieren und anmelden können. Dabei wurde auf die Sicherheit der Benutzer*innen geachtet. Außerdem ist es möglich, nach beliebigen Ausstellungen und Kategorien zu suchen, wobei ebenfalls die neuesten Ausstellungen empfohlen werden.

Das Content-Creation Tool ermöglicht es dem*der Benutzer*in einfach eine Ausstellung zu kreieren. Der*Die Benutzer*in lädt seine*ihrre Ausstellungsobjekte einfach hoch, wählt einen vordefinierten virtuellen Raum aus und konfiguriert diesen. Danach kann die Ausstellung auf der Webseite veröffentlicht werden.

In der Ausstellung werden Ausstellungsobjekte vom Server dynamisch geladen und aufgebaut. Für den*die Benutzer*in wurden verschiedene Bewegungsarten und Interaktionsmethoden innerhalb des Ausstellungsraumes implementiert.

6.3 Erweiterungsvorschläge [Team]

Das Projekt 3D-Gallery-Portfolio soll weiterentwickelt werden. Es ist auf Github öffentlich verfügbar, wodurch die Open-Source-Community es aktualisieren und erweitern kann.

7 Glossar

Framework [M]

Ein Framework soll dem*der Programmierer*in helfen neue Applikationen zu schaffen. Es bietet somit ein Grundgerüst, auf dass sich die Software aufbauen lässt.

Open-Source [L]

Bei einer Open Source Applikation ist der Code öffentlich zugänglich. Der Open Source Status sagt aber nichts über die Lizenzierung der Applikation aus. [90]

Single-Page-Application (SPA) [M]

Eine Single-Page-Webanwendung interagiert mit dem*der User*in, wodurch sich einzelne Komponenten der Website dynamisch verändern. //TODO

Design-Pattern [L]

Im Programmieren gibt es Probleme, die immer wieder auftreten. Diese Probleme wurden schon einmal sehr effizient gelöst und es besteht kein Nutzen sie immer wieder zu lösen. Design Patterns sind Strategien und Implementierungsvorgaben für die Lösung dieser Probleme. Sie beschleunigen den Entwicklungsprozess und machen ihn sicher, weil die Implementierungsvorgaben schon vorgegeben sind und diese auf einen hohen Grad der Sicherheit getestet wurden. [91]

Observer design pattern Ein konkretes Beispiel für Design Patterns ist das *Observer pattern*. Es wird dafür verwendet, Eins-zu-Eins-Beziehungen zwischen zwei oder mehreren Objekten zu erstellen. Ziel ist, dass Objekte (oder auch das Subject) bei allen Veränderungen diese schnell und leicht an das Objekt(oder auch den Observer) weitergeben. Das Subject führt eine Liste von Observern. Sobald Änderungen auftreten, liest das Subject die Liste aus und informiert alle eingetragenen Observer. Wenn ein Objekt ein Observer werden will, muss es sich in die Liste des zu observierenden Objekt

einschreiben. Ohne diese Funktionalität müsste der Observer in einem regelmäßigen Intervall das Subject abfragen, ob es eine Veränderung gab, die die Rechenzeit und Leistung verlängert würden. [92]

Event-Listener [M]

Event-Listener sind Methoden, die beliebige Logik enthalten und aufgerufen werden, wenn ein bestimmtes Event eintritt. Events können zum Beispiel Interaktion von Benutzer*innen über die Maus sein.

Third-Party [M]

Sind Funktionen, Services, Libraries etc., die von Drittanbietern bereitgestellt werden, also nicht direkt zum eigentlichen Produkt gehören.

API [L]

Ein API (oder auch Applikations Programmierungs Schnittstelle)

<https://www.redhat.com/de/topics/api/what-are-application-programming-interfaces>

DOM [M]

//TODO

Shadow-DOM

Direktiven [M]

//TODO

Pipes [M]

//TODO

Optionals [M]

//TODO

Mockup

//TODO

Evolutionäres Prototyping [L]

Beim evolutionären Prototyping wird der Prototyp über die Dauer des gesamten Projektes weiterentwickelt. Dies soll vermeiden, dass Fehler und Missverständnisse aus der Spezifikationsphase eines Projektes erst zu einem späten Projektstand entdeckt werden. [93]

Literaturverzeichnis

- [1] P. Hammer, „Virtual Reality: Die Erschaffung neuer Welten,” letzter Zugriff am 28.03.2023. Online verfügbar: <https://www.zukunftsinstut.de/artikel/virtual-reality-die-erschaffung-neuer-welten/>
- [2] A. Ivanos, „AngularEvidence,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://stackdiary.com/front-end-frameworks/>
- [3] A. P. Sofiya Merenchy, „AngularEvidence2,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://clockwise.software/blog/best-angular-applications/>
- [4] Angular Team, „What is Angular,” letzter Zugriff am 20.02.2023. Online verfügbar: <https://angular.io/guide/what-is-angular>
- [5] R. Gravelle, „AngularArchitecturePattern,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://www.htmlgoodies.com/javascript/the-model-view-viewmodel-pattern-and-angular-development/>
- [6] M. Team, „MVCmdn,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
- [7] ——, „MVVM,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm%0A>
- [8] ——, „MVC,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview>
- [9] Angular Team, „AngularProviders,” letzter Zugriff am 11.03.2023. Online verfügbar: [AngularProviders](#)
- [10] ——, „Angular ngModules,” letzter Zugriff am 20.02.2023. Online verfügbar: <https://angular.io/api/core/NgModule>
- [11] ——, „AngularNgModulesAPI,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://angular.io/guide/ngmodule-api>
- [12] D. K. Johannes Hoppe, Ferdinand Malcher, „AngularBuch, - Grundlagen, fortgeschrittene Themen und Best Practices,” 2020, letzter Zugriff am 10.03.2023.
- [13] ReactiveX Team, „Documentation Introduction for ReactiveX,” letzter Zugriff am 24.02.2023. Online verfügbar: <https://reactivex.io/intro.html>
- [14] Webpack Team, „Documentation for Webpack,” letzter Zugriff am 24.02.2023. Online verfügbar: <https://webpack.js.org/concepts/>
- [15] E. Team, „What Is A CSS Framework?” letzter Zugriff am 5.03.2023. Online verfügbar: <https://elementor.com/resources/glossary/what-is-a-css-framework/>
- [16] T. Team, „Best CSS Frameworks in 2022,” letzter Zugriff am 5.03.2023. Online verfügbar: https://dev.to/theme_selection/best-css-frameworks-in-2020-1jjh

- [17] S. J. . J. Team, „Angular Material Tutorial,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://www.javatpoint.com/angular-material>
- [18] E. Ighosewe, „What Is Angular Material,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://upstackhq.com/blog/software-development/what-is-angular-material>
- [19] G. Team, „Material Design: Introduction,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://m2.material.io/design/introduction>
- [20] Angular Team, „Angular Material - components,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://material.angular.io/components/categories>
- [21] Sass Team, „Sass Basics,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://sass-lang.com/guide>
- [22] ThreeJs Team, „Fundamentals of ThreeJs,” letzter Zugriff am 26.02.2023. Online verfügbar: <https://threejs.org/manual/#en/fundamentals>
- [23] Khronos Foundation, „Webgl - Getting Started - Wiki,” letzter Zugriff am 26.02.2023. Online verfügbar: https://www.khronos.org/webgl/wiki/Getting_Started
- [24] Reza Baradaran Gazorisangi, „What is the difference between a high-level and low-level Java API?” letzter Zugriff am 12.03.2023. Online verfügbar: <https://stackoverflow.com/questions/30897001/what-is-the-difference-between-a-high-level-and-low-level-java-api>
- [25] „A-Frame Wikipedia,” letzter Zugriff am 1.03.2023. Online verfügbar: [https://de.wikipedia.org/wiki/A-Frame_\(Framework\)](https://de.wikipedia.org/wiki/A-Frame_(Framework))
- [26] Angular Three Team, „Angular Three: Our first scene,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular-three.netlify.app/docs/first-scene>
- [27] A. T. Team, „Angular Three Documentation,” letzter Zugriff am 7.02.2023. Online verfügbar: <https://angular-three.netlify.app/>
- [28] ——, „Angular Three GitHub Repo,” letzter Zugriff am 29.03.2023. Online verfügbar: <https://github.com/nartc/angular-three>
- [29] auth0 Team, „Get Started with JSON Web Tokens,” letzter Zugriff am 22.03.2023. Online verfügbar: <https://auth0.com/learn/json-web-tokens>
- [30] Nishanil, „Microservices architecture, Microsoft Learn,” letzter Zugriff am 08.03.2023. Online verfügbar: <https://learn.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/microservices-architecture>
- [31] Quarkus, „What is Quarkus?” letzter Zugriff am 08.03.2023. Online verfügbar: <https://quarkus.io/about/>
- [32] ——, „Creating Your First Application,” letzter Zugriff am 08.03.2023. Online verfügbar: <https://quarkus.io/guides/getting-started>
- [33] T. P. G. D. Group, „PostgreSQL: About,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://www.postgresql.org/about/>
- [34] JetBrains, „IntelliJ IDEA - the Leading Java and Kotlin IDE,” letzter Zugriff am 08.03.2023. Online verfügbar: <https://www.jetbrains.com/idea/>

- [35] J. Team, „Webstorm,” letzter Zugriff am 21.02.2023. Online verfügbar: <https://www.jetbrains.com/de-de/webstorm/>
- [36] M. Team, „Cinema4D,” letzter Zugriff am 22.02.2023. Online verfügbar: <https://www.maxon.net/de/cinema-4d>
- [37] G. Team, „C4DvsMaya,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://www.getapp.de/compare/2036947/2051550/cinema-4d/vs/maya>
- [38] w3schools Team, „What is npm?” letzter Zugriff am 6.03.2023. Online verfügbar: https://www.w3schools.com/whatis/whatis_npm.asp
- [39] npm Team, „About npm,” letzter Zugriff am 6.03.2023. Online verfügbar: <https://docs.npmjs.com/about-npm>
- [40] I. S.-M. F. Schwab, „Ideenfindung,” in *Systemplanung und Projektentwicklung*, Wien, 2013, ch. 2, letzter Zugriff am 1.03.2023.
- [41] ——, „Agile Vorgehensmodelle,” in *Systemplanung und Projektentwicklung*, 2013, ch. 6.4.
- [42] P. D. K. Kuenen, „User Experience Design,” letzter Zugriff am 3.03.2023. Online verfügbar: <https://wirtschaftslexikon.gabler.de/definition/user-experience-design-100263/version-368987>
- [43] S. Kuppelwieser, „LogoKriterien,” letzter Zugriff am 20.03.2023. Online verfügbar: <https://www.sonja-kuppelwieser.com/logo-kriterien/>
- [44] A. . S. Team, „IsometricStyle,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://www.aleksundshantu.com/wiki/isometrischer-stil/>
- [45] P. D. F.-R. Esch, „Logo,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://wirtschaftslexikon.gabler.de/definition/logo-39571>
- [46] F. Copes, „When is a package best installed globally? Why?” letzter Zugriff am 6.03.2023. Online verfügbar: <https://flaviocopes.com/npm-packages-local-global/>
- [47] Angular Team, „AngularComponentOverview,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/component-overview>
- [48] ——, „AngularScaling,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/scaling>
- [49] ——, „AngularComponentsHierarche,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://angular.io/guide/architecture-components>
- [50] ——, „BindingSyntax,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/binding-syntax>
- [51] ——, „AngularPropertyBinding,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/property-binding>
- [52] ——, „AngularEventBinding,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/event-binding>
- [53] ——, „AngularTwoWayBinding,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/two-way-binding>
- [54] ——, „AngularArchitectureService,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/architecture-services>

- [55] ——, „AngularHTTPClient,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/http>
- [56] A. Team, „AdobeRendering,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://www.adobe.com/de/products/substance3d/discover/3d-rendering.html#:~:text=Beim3D-Renderingwirdaus,Modellsein2D-Bildgerendert>.
- [57] Norman I. Badler, Andrew S. Glassner, „Rendering3DModels,” letzter Zugriff am 11.03.2023. Online verfügbar: http://gamma.cs.unc.edu/courses/graphics-s09/LECTURES/3DModels_SurveyPaper.pdf
- [58] M. D. Team, „RenderArten,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://magic-3d.de/was-sind-3d-renderings/>
- [59] P. Team, „MoireEffekt,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://www.printer-care.de/de/drucker-ratgeber/moire-effekt>
- [60] J. Herzog, „AntiAliasing,” letzter Zugriff am 11.03.2023. Online verfügbar: [https://de.wikipedia.org/wiki/Antialiasing_\(Computergrafik\)#/media/File:EasterEgg_anti-aliasing.png](https://de.wikipedia.org/wiki/Antialiasing_(Computergrafik)#/media/File:EasterEgg_anti-aliasing.png)
- [61] N. Team, „RayTracingRasterization,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://developer.nvidia.com/discover/ray-tracing>
- [62] T. Team, „ThreejsWebGLRenderer,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/#api/en/renderers/WebGLRenderer>
- [63] Angular Team, „AngularViewChild,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/api/core/ViewChild>
- [64] T. Team, „ThreejsCreateAScene,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=scene#manual/en/introduction/Creating-a-scene>
- [65] ——, „StandardLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/index.html#/api/en/renderers/WebGLRenderer.physicallyCorrectLights>
- [66] ——, „LightProbe,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/LightProbe>
- [67] ——, „AmbientLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/AmbientLight>
- [68] ——, „DirectionalLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/DirectionalLight>
- [69] ——, „HemisphereLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/HemisphereLight>
- [70] ——, „PointLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/PointLight>
- [71] ——, „ReactAreaLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/RectAreaLight>
- [72] ——, „SpotLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/SpotLight>

- [73] Wiki Contributors, „Persistenz,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://de.wiktionary.org/wiki/Persistenz>
- [74] MDN Contributors, „Web Storage API,” letzter Zugriff am 11.03.2023. Online verfügbar: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- [75] Hugo, „Origin of the term "wizardin computing,"“ letzter Zugriff am 18.03.2023. Online verfügbar: <https://english.stackexchange.com/questions/65728/origin-of-the-term-wizard-in-computing>
- [76] R. Budiu, „Wizards: Definition and Design Recommendations,” letzter Zugriff am 18.03.2023. Online verfügbar: <https://www.nngroup.com/articles/wizards/>
- [77] A. M. Team, „Angular Material Documentation - Stepper Overview,” letzter Zugriff am 18.03.2023. Online verfügbar: <https://material.angular.io/components/stepper/overview>
- [78] T. Team, „DragControls,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=Control#examples/en/controls/DragControls>
- [79] ——, „FirstPersonControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/index.html#examples/en/controls/FirstPersonControls>
- [80] ——, „OrbitControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/#examples/en/controls/OrbitControls>
- [81] ——, „ThreejsTransformControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/?q=controls#examples/en/controls/TransformControls>
- [82] ——, „PointerLockControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/?q=controls#examples/en/controls/PointerLockControls>
- [83] ——, „ThreeJsClock,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/#api/en/core/Clock%0A>
- [84] M. Angelini, „AABBAndOBPPicture,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://devdept.zendesk.com/hc/en-us/articles/360011559320-How-Collision-Detection-works-v2020->
- [85] T. Team, „ThreeJSRaycaster,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/#api/en/core/Raycaster>
- [86] ——, „PostProcessing,” letzter Zugriff am 20.03.2023. Online verfügbar: <https://threejs.org/docs/#manual/en/introduction/How-to-use-post-processing>
- [87] Angular Team, „Angular: View encapsulation,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular.io/guide/view-encapsulation>
- [88] ——, „Angular: Component styles - (deprecated) /deep/,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular.io/guide/component-styles#deprecated-deep--and-ng-deep>
- [89] Joshua Colvin, „Understanding Angular ::ng-deep,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://www.joshuacolvin.net/understanding-ng-deep/>
- [90] R. H. Team, „Was ist Open Source?” letzter Zugriff am 29.03.2023. Online verfügbar: <https://www.redhat.com/de/topics/open-source/what-is-open-source>

- [91] I. Team, „Design Patterns – schneller und sicherer programmieren,” letzter Zugriff am 4.03.2023. Online verfügbar: <https://www.ionos.at/digitalguide/websites/webentwicklung/was-sind-design-patterns/>
- [92] ——, „Observer pattern: what does the observer design pattern do?” letzter Zugriff am 4.03.2023. Online verfügbar: <https://www.ionos.at/digitalguide/websites/webentwicklung/was-sind-design-patterns/>
- [93] I. S.-M. F. Schwab, „Prototyping,” in *Systemplanung und Projektentwicklung*, Wien, 2013, ch. 6.1.3.

Abbildungsverzeichnis

1	Angular Logo	4
2	Die Model-View-Controller Pattern [6]	5
3	Die Model-View-ViewModel Pattern [7]	6
4	RxJS Logo	7
5	Webpack Logo	7
6	Angular Material UI Logo	8
7	Angular Material: Komponenten die im Projekt verwendet wurden [20]	10
8	Bootstrap Logo	10
9	Sass Logo	11
10	Three.js Logo	12
11	Die Struktur von Three.js [22]	12
12	WebGL Logo	13
13	JWT Logo	15
14	Quarkus Logo	16
15	PostgreSQL Logo	16
16	IntelliJ Logo	17
17	Webstorm Logo	17
18	Figma Logo	18
19	Cinema4D Logo	19
20	Cinema4D vs Maya [37]	20
21	Landingpage Design Mockups Gegenüberstellung	23
22	OberflächenDesign: Prototypen in Figma	25
23	Logo der 3D-Portfolio-Gallery	26
24	Angular: Automatisch generierte Start-Webseite	27
25	Component-Hierarchy [12]	30
26	Modellierung des Raumes in Cinema4D	36
27	Der Moire-Effekt visualisiert [59]	38
28	Rendering mit und ohne Anti-Aliasing [60]	39
29	Vorgang des Raytracing [61]	40
30	Landing Page	43
31	Projekt: Login Page	44
32	Navbar: authentifiziert vs noch nicht authentifiziert	48
33	Inhaltserstellungstool	50
34	Inhaltserstellung Datenklassen	51
35	Profilepage	52
36	Angular Material: horizontaler Stepper [77]	54
37	Der Unterschied zwischen AABB und OBB [84]	60
38	Search-Page	67

Tabellenverzeichnis

Quellcodeverzeichnis

1	SCSS - Code Beispiel [21]	11
2	CSS - Code Beispiel [21]	11
3	Angular Three - Komponentenbasiertes 3D Scenen in HTML [26]	14
4	Angular Three - App Cube [26]	14
5	Terminalm - Angular aufsetzen, Installation der CLI, Configuration eines neuen Projektes, Starten des Projektes	26
6	Terminal - Angular Material Installation	27
7	Terminal - Bootstrap Installation	28
8	angular.json - Bootstrap Angular Verknüpfung	28
9	Terminal - Component erstellen	28
10	Beispiel für Interpolation in der 3D-Gallery	31
11	Beispiel für Property-Bindings in der 3D-Gallery	31
12	Beispiel für Event-Bindings in der 3D-Gallery	31
13	Beispiel für Two-Way-Bindings [53]	31
14	Routing in der 3D-Gallery	32
15	Routing über einen routerLink	32
16	Routingparamter in der 3D-Gallery	33
17	Snapshot der URL abfragen	33
18	Die URL subscriben	33
19	Eine Klasse Injectable machen	34
20	Constructor Injection	34
21	HttpClient Abfragen	34
22	Das Datenmodell eines Ausstellungsstückes	35
23	Canvas-Element in HTML	40
24	Canvas als View-Child initialisieren	40
25	WebGlRenderer anlegen	41
26	Animations-Schleife	41
27	Lichtsetzung in der 3D-Ausstellung	42
28	auth.service.ts - Json-Web-Token und Localstorage	46
29	auth.interceptor.ts - add JWT to Request Header	48
30	OrbitControls initialisieren	56
31	FirstPersonControls initialisieren	57
32	PointerLockControls initialisieren	58
33	Logik der PointerLockControls	58
34	Controls entsperren	58
35	Controls updaten	59
36	Methode zum vergleichen der Position der Kameras	61
37	Methode zum ermitteln einer Kollision mit der Wand	61
38	Aktuelle Koordinaten des Mauszeigers einem 2D-Vektor zuweisen	62
39	Neuen Raycaster anlegen	63
40	Intersected Objects auslesen	63
41	Intersects als Ausstellungsstück identifizieren	63
42	Hover-effekt wieder entfernen	64

43	identifizieren des geklickten Ausstellungsstückes	64
44	initialisieren des Dialogfensters	65
45	Öffnen und Schließen des Dialogfensters	65
46	Constructor-Injection in der Dialog Komponente	66
47	Eingabefeld	67
48	Die Such-Pipe mit den Filter-Operatoren	68
49	Auswählen und Abwählen der Kategorien	68
50	Filter von Kategorien anwenden	68
51	Parent.component.scss - Changing Styling in Child Componentes by using :ngdeep	70

Anhang