

3D Portfolio Gallery

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für IT - Medientechnik

Eingereicht von:

Ema Halilovic

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

S. Schwammal & S. Schwammal

Abstract

3D Portfolio Gallery is a web application developed by Lorenz Litzlbauer and Fabian Maar as part of the thesis. The 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. Designers can use 3D Portfolio Gallery to create a three-dimensional portfolio from their own media (film, photo or 3D data). The 3D Portfolio Gallery is offered as a SPA on the web and can be accessed via a web link. The 3D Portfolio Gallery provides a simple configuration process by selecting a gallery from several pre-defined layouts. In another step, the exhibition pieces are either automatically or manually placed and provided with additional information.



For the implementation in the front end, the following frameworks are used: Angular for the SPA (single-page application) and ThreeJs for the 3D presentation. The front-end access control and user identification in the browser uses JWT (JsonWebToken). The JWT is part of a standard described by the IETF III RFC7520. The backend uses Quarks as a server platform, Maven as a build system, and in the database access layer JPA (Java Persistence Architecture), Panach and Hibernate ORM.

Zusammenfassung

3D Portfolio Gallery ist eine Webapplikation, 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 das alles in einem virtuellen dreidimensionalen Raum. 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 SPA im Web angeboten und ist über einen Web-Link erreichbar. 3D Portfolio Gallery bietet dafür einen einfachen Konfigurationsprozess, indem aus mehreren vordefinierten Grundrissen eine Gallery ausgewählt wird. In einem weiteren Schritt werden die Ausstellungsstücke entweder automatisch oder manuell platziert und mit Zusatzinformationen versehen.



Für die Umsetzung im Frontend werden folgende Frameworks verwendet: Angular für die SPA (Single-Page-Applikation) und ThreeJs für die 3D-Darstellung. . In der Frontend-Zugriffskontrolle und -Useridentifikation im Browser wird JWT (JsonWebToken) verwendet. Das JWT ist Teil eines Standards, der durch die III RFC7519 IETF beschrieben wird. Das Backend verwendet als Serverplattform Quarks, als Buildsystem Maven und in der Datenbankzugriffsschicht JPA (Java Persistence Architecture), Panach und Hibernate ORM.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Ausgangslage	1
1.2 Untersuchung Anliegen der individuellen Themenstellungen	2
1.3 Zielsetzung	2
1.4 Aufgabendifferenzierung	2
2 Architektur	4
2.1 Client-Server-Architektur [L]	4
2.2 Technologie-Stack Client [L]	4
2.3 Technologiestack Backend	15
3 Werkzeuge	17
3.1 IDE	17
3.2 UI Prototyping Werkzeug	17
3.3 3D Modellierung	18
3.4 Package Manager [L]	20
4 Planung	21
4.1 Ideenfindung [M]	21
4.2 User-Stories	22
4.3 Evolutionäres Prototyping	25
4.4 Prototyp im Projekt	25
5 Umsetzung	27
5.1 Design [L]	27
5.2 Initialisierung der Landingpage [L]	33
5.3 Web Gallery Prototype	43
5.4 Fertige Landingpage	49
5.5 Userverwaltung [L]	50

5.6	Content Creation [L]	56
5.7	3D-Portfolio-Gallery Konfigurationstool [L]	58
5.8	Interaktions-Seite [M]	63
6	Zusammenfassung	75
6.1	Probleme und Lösungsstrategie	75
6.2	Zielerreichung	76
6.3	Erweiterungsvorschläge	76
7	Glossar	77
Literaturverzeichnis		VI
Abbildungsverzeichnis		XI
Tabellenverzeichnis		XII
Quellcodeverzeichnis		XIII
Anhang		XV

1 Einleitung

1.1 Ausgangslage

Ausgangslage ist die Idee für ein Eisenbahnmuseum eine Applikation zur Unterhaltung von Museumsbesuchern. Die Vorgaben für die Applikation waren wenig spezifisch. In weiteren Schritten wurde uns klar, dass wir eine ganze Ausstellung dreidimensionaler Objekte anbieten wollten. Um unser Projekt weiter zu konkretisieren, überlegten wir uns die Anforderung, die sich an einen Designer von digitalen 3D-Objekten bei der Präsentation von 3D-Objekten, stellen. Dass wir in unserem 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

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 Untersuchung Anliegen der individuellen Themenstellungen

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

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 Prototyps 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

Das Projekt wird von drei Personen bearbeitet. Die Projektleitung liegt bei Lorenz Litzlbauer. Das Projekt erhält Unterstützung aus den Fächern Design und 3D durch Frau Professor Engleitner und Frau Professor Rammelmüller.

1.4.1 Ema

Leider konnte Ema Halilovic das Projekt wegen persönlichen Gründen nicht vollen-den.

1.4.2 Lorenz

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

1.4.3 Fabian

Im Lead von Fabian Maar wurde in der Softwareentwicklung die Such-Unterseite, Besucheransicht der Ausstellung, die Interaktion mit den Ausstellungsstücken, die Navigation im dreidimensionalen Raum und das UI-Design erstellt.

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 Datenhaltung. Die Services am Backendserver werden als REST-Services angeboten.

2.2 Technologie-Stack Client [L]

Die SPA verwendet Angular und Erweiterungen, die das Angular-Framework verwendet und bereitstellt.

In den folgenden Unterabschnitten wird sich mit den Technologien, auf die Angular aufgebaut ist und die Erweiterungen des Frontends auseinandergesetzt.

2.2.1 GUI

Folgende Aspekte sprechen für die Nutzung Angular in diesem Projekt:

- komponentenbasiertes Programmieren; im Abschnitt 5.2.2 wird nochmals deutlich, wie dieser Aspekt genutzt wird.
- einfache und übersichtliche Einbindung von Libraries durch ngModules; wird behandelt in Kapitel NgModules 2.2.1
- Isolierung der Logik mit der Benutzeroberfläche durch die Model-View-Controller Architektur und die Model-View-Viewmodel Architektur; wird behandelt in Kapitel MVC und MVVM ??
- Dependency Injection für die Verbindung von Front-End zum Back-End; siehe Kapitel HTTP Services 5.2.4

Angular [M]

Angular ist ein Framework für Webapplikation, das auf der Programmiersprache TypeScript basiert. Es ist eines der renommiertesten Frameworks zur Front-End-Entwicklung und wird als Open-Source-Software zur Verfügung gestellt. So besitzt Angular eine große Community und wird von vielen Software-Entwicklern benutzt. So zählt es 2022 zu dem zweitbeliebtesten Front-End Framework [1] und wird von Milliardenkonzernen wie zum Beispiel Google oder PayPal verwendet. [2]

Die große Stärke von Angular ist die Erstellung von Single-Page-Webanwendung, 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. [3]

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. [4]

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
- um die Wartung und Testung zu erleichtern
- Änderungen am Umfang und der Größe der Applikation leichter handhaben zu können

[5, 6, 7]

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.

Welche Daten eine Applikation beinhalten soll, wird über das Model geregelt. Falls sich das Model ändert, werden die Benutzeroberfläche und manchmal auch der Controller

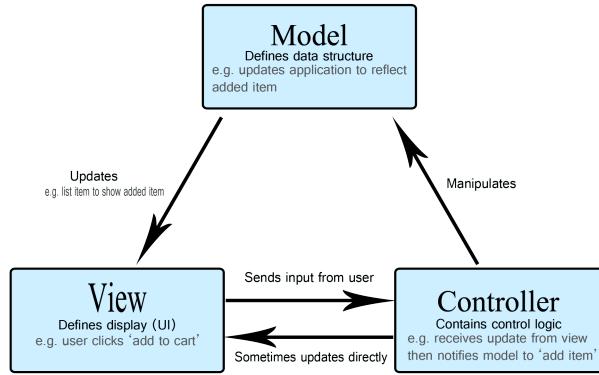


Abbildung 1: Die Model-View-Controller Pattern [5]

entsprechend geändert. In Angular sind diese Modelle mit den Interfaces (siehe Interfaces 5.2.4) zu vergleichen.

Die 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.

In dem 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 und die Daten gegebenenfalls in unterschiedliche Formate anzuzeigen. Die TypeScript-Files in Angular sind vergleichbar mit diesen Controllern. [7]

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

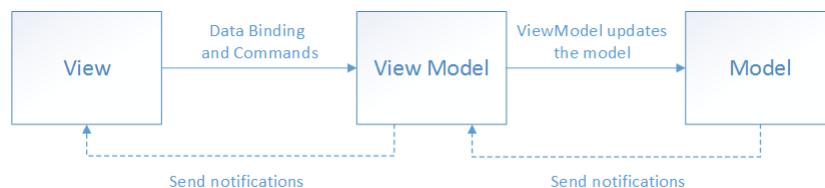


Abbildung 2: Die Model-View-ViewModel Pattern [6]

Die 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 HTML-Template einer Angular-Komponente noch ähnlicher 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 beliefert es mit Daten. Dieser Vorgang beschreibt das Konzept des Data-Bindings (siehe Data-Binding) 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. Sie wird durch das Konzept von Komponenten realisiert, auf welches im nächsten Abschnitt näher eingegangen wird. Dabei werden die Parallelen der Patterns erkennbar. [6]

NgModules [M] NgModules erleichtern die Einbindung von Libraries enorm. Da sich keine externen Files mühsam gedownloadet und eingebunden werden müssen und jeder Import-Prozess fast von selbst geschieht, wird enorm Zeit beim Entwickeln gespart. Außerdem sind alle Importe in chronologischer Reihenfolge gelistet, wodurch ein guter Gesamtüberblick geliefert wird und schnell überflüssige Imports entfernt werden können. Die ständige 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 Dependency Injection 5.2.4) abgerufen werden [8]
- 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

Wie die ngModules verwendet in der 3D-Gallerie-Applikation wurden, wird im Abschnitt Routing oder Landing Page aufsetzen //TODO Referenz nochmals erklärt. [9] [10] [11]

Angular Schlüsseltechnologien [L]

RxJS [L] RxJS ist eine Implementierung von ReactiveX für die Programmiersprache Javascript. Angular verwendet RxJS für reaktive Programmierart.

ReactiveX ist eine Library für das Erstellen von asynchronen und Event-basierenden Programmen, dafür benutzt es *observable sequences* (Ein beobachtetes Subjekt führt eine Liste von den Observern. Bei einer Veränderung im Subjekt werden die Beobachter nach dieser Liste der Reihe nach informiert. Siehe im Glossar 6.3 auf Seite 77). Die Library arbeitet so mit dem *Observer-Design-Pattern* und fügt verschiedene neue Operatoren hinzu. Zusätzlich werden "LowlevelFunktionen wie Threading, Synchronisation und Thread-Sicherheit verarbeitet und keine Input-/Output-Prozesse blockiert. [12]

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.

UserInterface-Frameworks (UI-Frameworks)

Ein UserInterface-Frameworks ist eine Ansammlung von Web Komponenten wie eine Navigationsleiste die sofort Nutzbar sind.

Die Vorteile von UI-Frameworks sind:

- vereinfacht und beschleunigt den Entwicklungsprozess; typische Web-Komponenten sind vordefinierte, der Entwickler muss diese nicht mehr implementieren und erspart sich Zeit.
- Cross-Browser Unterstützung; 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.

- 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.

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, schauen einander ähnlich aus; Da die Web-Komponenten gleich implementiert werden.

[13] [14]

2.2.2 UI-Frameworks im Projekt

Im Projekt wurde sich für zwei UI-Frameworks, Bootstrap und AngularMaterials, entschieden.

Angular Material

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, Typographie und vielem mehr. Die Design-Sprache orientiert sich dabei an der Material Design Spezifikation von Google. Ein großer Fokus des Frameworks ist die Responsiveness, Komponenten werden auf verschiedenen Auflösungen gut aussehen.

[15, 16]

Angular Material hat folgende Features: [15, 16]

- Erweiterbar; Bei der Installation lassen sich viele Designanpassungen machen, zusätzlich lassen sich Styles mit dem globalen Stylesheet überschreiben.
- Hochwertig; Die Komponenten sind erprobt und wurden auf die Performanz und Verlässlichkeit getestet.
- Reibungslos; Angular Material ist gut mit Angular integriert

Die Abbildung 3 zeigt einen Teil der Komponenten, die Angular Material anbietet.

Zu allen diesen Komponenten gibt es eine ausführliche Dokumentation mit Beispiele.

[15, 16]

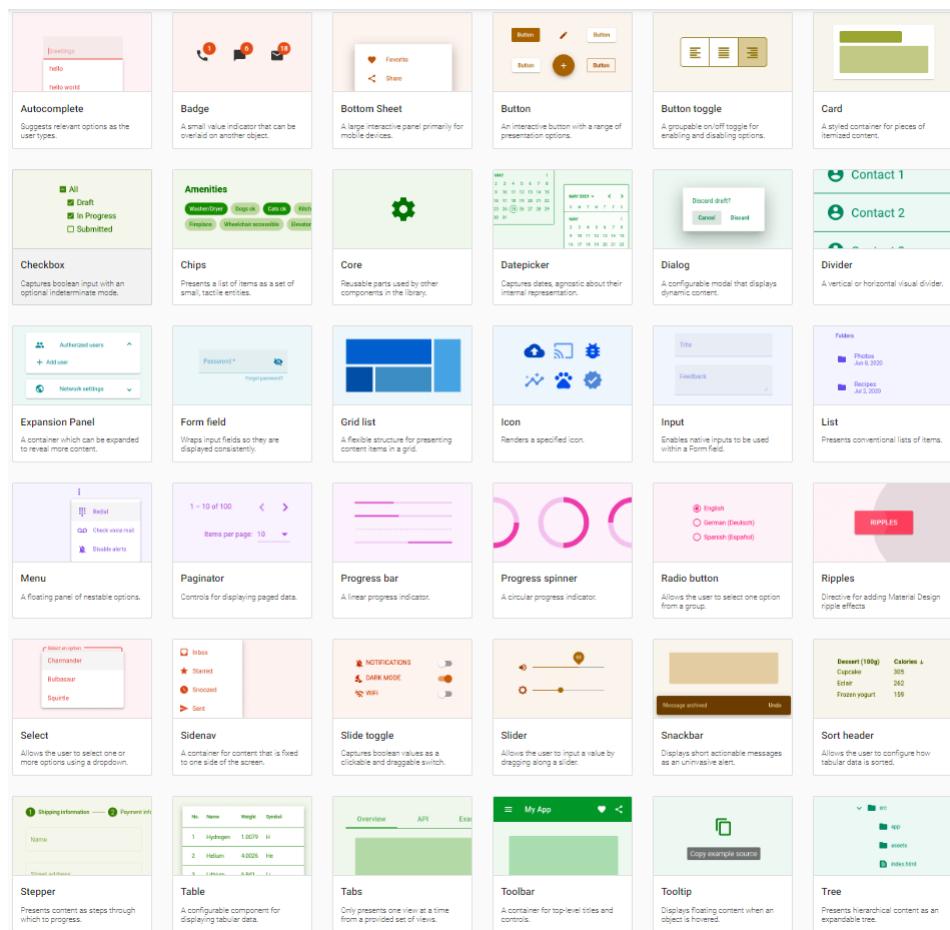


Abbildung 3: Angular Material: Komponente Überblick [17]

Material Design Spezifikation Material Design ist ein Designssystem, welches von Google erstellt wurde, um Teams zu helfen, qualitative hochwertige Digitale-Erfahrungen für Android, iOS, Flutter und das Web zu gestalten.

Material Design hat mehrere Prinzipien, die sie ihr Design beeinflussen lassen.

Material Design soll die reale Welt abbilden. Text soll durch Typographie, Raster, Abstände, Farben und Bilder soll eine erkennbare Hierarchie entstehen. Komponenten sind interaktive Bauklötze für ein User Interface. Komponenten haben alle eine Status-System welches den Status der Komponente fokussiert, selektiert, aktiviert, fehlerhaft, schwebend, gedrückt und ausgeschalten anzeigt. Durch das Theming ist es möglich einzelne Komponenten zu verändern und das Design so anzupassen, dass es zur Corporate Identity der Benutzer*in passt.[18]

Bootstrap

Bootstrap ist ein kostenloses Open-Source-UI-Framework. Es ist nicht nur eine Ansammlung von CSS sondern auch von JavaScript und HTML code um schöne, interaktive und responsive Webseiten. [14]

Bootstrap hat folgenden Features: [14]

- Web-Komponenten, die sofort Anwendbar sind
- Eine gute Dokumentation
- Mobile Centered Design
- Hohe Benutzerfreundlichkeit

Scss

SCSS ist kein UI-Framework, sondern vielmehr eine Syntax- und Feature-Erweiterung von CSS, deswegen wurde es schlussendlich auch für dieses Projekt gewählt. SCSS hat viele Vorteile wie zum Beispiel Variablen, Schleifen, Mixins und Imports gegenüber CSS, aber besonders ausschlaggebend war das Prinzip der Verschachtelung. [19, Sass Guide]

Durch die Verschachtelung kann an redundanten Styles-Selektoren gespart werden, was die Style-Definition besser lesbarer macht. Im den Beispielen (siehe SCSS-Code-Beispiel 1 und natives CSS-Code-Beispiel 2) wird die gleiche Style-Definition von SCSS und CSS ausgedrückt, doch die Schreibweise unterscheidet sich. [19, Sass Guide]

[19, Sass Guide]

Listing 1: SCSS - Code Beispiel

```

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

```

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 }
```

Code Beispiele1 2 [19, Sass Guide]

2.2.3 3D Rendering [L]

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

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

ThreeJs [L]

ThreeJs ist eine JavaScript Library für die Darstellung von 3D-Grafiken im Web. Für die 3D-Darstellung nutzt ThreeJs WebGL (mehr dazu im nächsten Abschnitt ThreeJs Dependency), ein low-level Framework. WebGL hat eine hohe Komplexität. ThreeJs bietet eine Abstraktion zu WebGL,

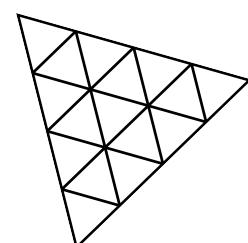


Abbildung 4: ThreeJs go

um Anwendungen für 3D-Webanwendungen für Entwickler zugänglicher zu machen. ThreeJs verarbeitet dabei viele Sachen wie die 3D-Szene, Lichter, Schatten, Materialien, Texturen und 3D-Matrix-Rechnungen, die mit WebGL sehr aufwändig wären umzusetzen.

In ThreeJs werden Geometrie, Objekte und Materialien verbunden, um ein 3D-Objekt zu erstellen. Dabei kann die Struktur einer Szene der Abbildung 5 ähneln. Dort kann beobachtet werden, dass die Scene das Root-Objekt ist. In der Scene werden Objekte wie Meshes, Gruppen, Lichter und 3D-Objekte in einer Baumdatenstruktur gesammelt. Die Daten über die Geometrie, Materialen der 3D-Meshes werden außerhalb der Szene gespeichert und die Meshes referenzieren darauf.

[20, ThreeJs fundamentals]

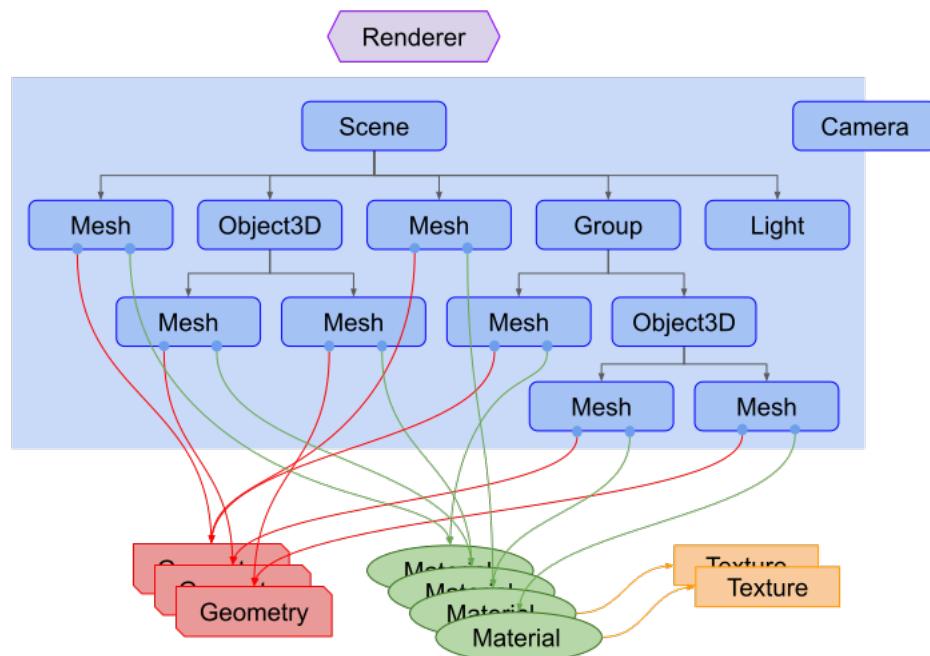


Abbildung 5: Die Struktur von ThreeJs [20]

Dependency

Um 3D-Darstellungen zu rendern, benutzt ThreeJs WebGL.

WebGL WebGL ist eine lizenzzfreie Low-Level-API, die dafür benutzt wird, 3D-Grafiken im Web darzustellen. Sie basiert auf OpenGL ES 2.0 und benutzt auch dieselbe shading language GLSL wie OpenGL. Eine Low-Level-API



Abbildung 6: WebGL Logo

hat einen höheren Konfigurierungsgrad und lässt sich für spezielle Anwendungsfälle besser anpassen. Der*Die Programmierer*in muss, aber auch ein viel tieferes Verständnis von dem Material haben (im Fall von WebGL Shader-Programmierung, Matrixrechnungen usw.) und hat in der Regel einen viel größeren Programmieranteil als wenn er eine High-Level-API benutzt. Eine High-Level-API ist besser für allgemeine Anwendungsfälle geeignet. [21, WebGL Getting Started] [22]

Alternativen 3D Web Apis

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

A-Frame 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. Bei der 3D-Darstellung setzt A-Frame auf ThreeJs. [23, A-Frame Wikipedia]

A-Frame hat ähnliche Funktionen und Leistung im Vergleich zu ThreeJs, da es ja schlussendlich darauf aufbaut. A-Frame sticht bei der VR support hervor (ThreeJs hätte auch eine Unterstützung dafür, diese ist aber schwieriger einzubinden), doch ist das kein unbedingt nötiges Feature. ThreeJs ist ja bereits eine Abstraktion von WebGL. Für das Projekt wird keine weiter Abstraktion gebraucht.

Angular Three

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

Listing 3: Angular Three - Komponentenbasiertes 3D Scenen in HTML

```

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

```

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>

```

Code Beispiele 3 4 [24]

Ein Vorteil von Angular Three ist, dass durch nur wenige Zeilen Code 3 eine 3D-Szene mit Lichtern und Orientierungsfunktionen erstellt werden kann und die Business-Logik, App-Cube 4 durch die Verwendung einer Komponente ausgelagert werden kann.

Ein weiterer Vorteil von Angular Three ist die ausführliche Dokumentation mit Codebeispielen. (link) Angular Three Dokumentation <https://angular-three.netlify.app/docs/getting-started/overview>

Wegen der vielen Vorteile hohe Benutzerfreundlichkeit, ähnliche 3D-Leistung zu ThreeJs (Angular Three basiert auf ThreeJs, welches selbst die WebGL-Renderengine benutzt) und wegen der Verbindung von Angular und ThreeJs Features bietet sich die Liberry für das Projekt an.

Mithilfe eines Prototypen (mehr dazu im Abschnitt fortlaufendes Prototyping) wurde getestet, ob Angular Three alle Features, die für das Projekt benötigt werden, unterstützen kann, die für das Projekt nötig sind. Dabei wurde festgestellt, dass sich die Library Angular Three für das Projekt nicht eignet. Trotz der vielen Features konnten einige Kriterien nicht erfüllt werden. So wurden beispielsweise 3D-Objekte nicht richtig geladen. ThreeJs Module funktionierten Teilweise nicht, wie die FristPersonControls.

Weil Angular Three nicht die für das Projekt aufgestellten Anforderungen erfüllt hat, wurde sich dafür entschieden, ThreeJs zu benutzen. ThreeJs hat ähnliche Konzepte und Logik wie Angular Three deswegen fiel dem Team der Umstieg darauf leicht. Diese 3D-Library wurde schlussendlich auch im Projekt verwendet.

2.3 Technologiestack Backend

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.[25] Ebenso sollen diese gute Dokumentationen und breite Communities enthalten. Schon vorhandene Praxiserfahrung stellte sich bei der finalen Auswahl als entscheidender Faktor dar.

2.3.1 Applikationserver

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. [26, 27] 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.

2.3.2 Datenhaltung

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 [28]

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. Plug-Ins ermöglichen es, Datenbankverbindungen und weiteres in der IDE zu konfigurieren, sodass dem*der Entwickler*in eine Übersicht von benötigten Informationen gegeben werden kann. [29]

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.

3.2 UI Prototyping Werkzeug

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 und eine mobile Version für iOS und Android an. Figma arbeitet mit mehreren Konzepten, um den Designprozess zu vereinfachen:

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.

Assets Figma arbeitet mit dem Konzept der Assets. Bereits designte Komponenten können als Assets gespeichert werden. Figma legt einen großen Wert auf die Modularität, es können Farben und Pixelgrößen als Variable gespeichert werden, wenn diese sich verändern, verändern sich gleichzeitig die darauf referenzierenden Assets.

Auswahl

Wegen dieser vielen Features und persönlichen Erfahrungen in privaten Projekten bot sich Figma für dieses Projekt als UX/UI-Design- Tool an.

3.3 3D Modellierung

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, gelingt es durch Cinema 4D eine geringe und effiziente Lernkurve zu erreichen. Aufgrund der intuitiven Benutzeroberfläche mit vielen Funktionen ist das Programm sowohl für Einsteiger als auch für Profis geeignet.
[30]

Funktionen	
Alle Funktionen	Alle Funktionen
2D-Zeichnung	2D-Zeichnung
3D-Modellierung	3D-Modellierung
Aktivitäts-Dashboard	X Aktivitäts-Dashboard
Animation	X Animation
Animationen und Übergänge	✓ Animationen und Übergänge
Bild-Nachverfolgung	X Bild-Nachverfolgung
Bildbearbeitung	X Bildbearbeitung
Daten-Import / -Export	X Daten-Import / -Export
Digital Asset Management	X 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	X Vorlagen

Abbildung 7: Cinema4D vs Maya [31]

Ein weiterer wichtiger Aspekt ist, dass Cinema 4D den Export von GLTF-Files unterstützt. 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.

Modellierung der 3D-Räume in Cinema 4D

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 erstellt, der die Höhe und Breite besitzt, die eine Wand haben soll. Da Cinema 4D die Modelle in Zentimeter misst, können diese in realitätsnahen Proportionen modelliert werden. Um den 2D-Grundriss nun als dreidimensionale Wände abzubilden, 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.

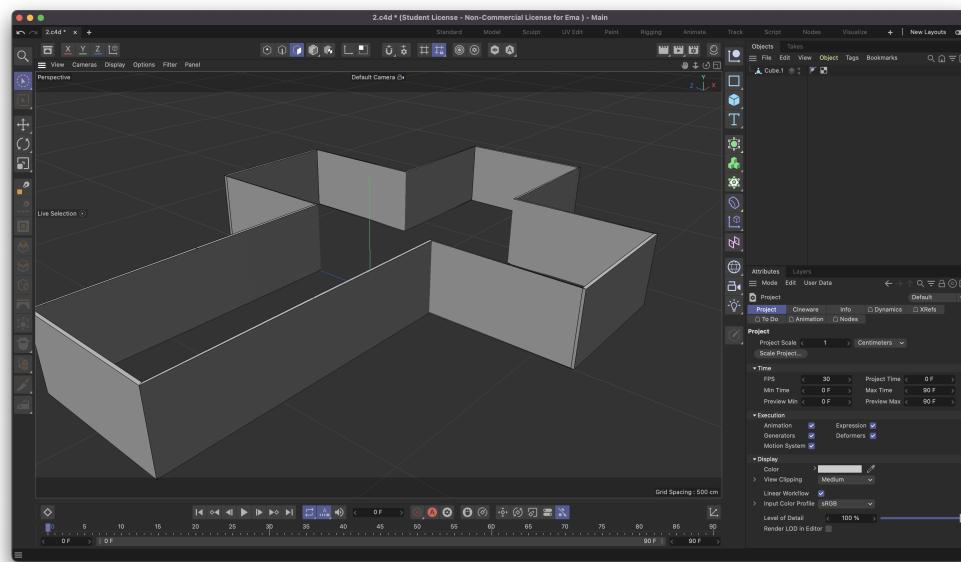


Abbildung 8: Modellierung des Raumes in Cinema4D

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. [32] [33]

4 Planung

4.1 Ideenfindung [M]

Die Ideenfindung ist der Start eines Projektes und bildet das Fundament, auf dem das Projekt entsteht. Die Idee für die Entstehung eines Projektes kann viele Motive haben, so ist es wichtig, als Projektteam diese zu erkennen und als Motivation zu nehmen. Während oder nach dem Prozess der Ideenfindung wird schon bald klar, welche Projektwürdigkeit das Projekt besitzt. Somit kann schon frühzeitig entschieden werden, ob es sinnvoll erscheint, das Projekt zu realisieren oder ob nochmals eine neue Ideenfindung vonnöten ist. Auch bei dieser Diplomarbeit wurde der Prozess der Ideenfindung mehrmals wiederholt, wodurch die Wichtigkeit und die Methodik der Findung von Ideen besonders bemerkbar wurde. Die Ideenfindung erfolgt meist mithilfe der Anwendung einer passenden Kreativitätstechnik.

Kreativitätstechniken sind vielseitig einzusetzen, werden aber häufig am Anfang eines Projektes angewandt. Sie 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.

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. Aufgrund der Expertise der Betreuungslehrerinnen im Bereich 3D-Modellierung und aus eigenem Interesse, konnte sich relativ schnell auf das Thema *eine Software mit 3D-Integration* geeinigt werden. Schlussendlich konnte sich für eine Idee entschieden werden, nachdem das Brainstormen nach folgenden Kriterien erfolgt hat:

- Quantität vor Qualität
- Freies Assoziieren und Fantasieren sind erwünscht
- Keine Kritik, Korrektur oder Meinungsäußerung

- Möglichst viele Ideen
- Inspirieren lassen von anderen

[34]

4.2 User-Stories

User Stories stellen Anforderungen an die Software. Sie wird user- und aufgabenorientiert formuliert und erzählt von der ersten Perspektive einer im Projekt involvierten Person, meistens des Users.

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 eine messbare Metrik zum Testen des Projektfortschritts.

User Stories kommen oft in der agilen Softwareentwicklung vor. Dort besprechen der Kunde und der Product Owner eine Zielbestimmung. Anhand dieser Bestimmungen werden Funktionen bestimmt, die das Projekt erfüllen muss und diese mithilfe von User Stories formuliert. [35]

Auch für dieses Projekt wurden User Stories formuliert. In diesem Kapiteln (Umsetzung) wird darüber geschrieben, wie diese erfüllt worden sind.

4.2.1 Projekt User Stories

Die für das Projekt formulierten User Stories sind folgenden:

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 erstmalige*r Besucher*in der Webseite möchte ich die benötigten Informationen über die Funktionen der Applikation verständlich erkennen können. Auswahlkriterien:
 - Auf der Landingpage befinden sich:

- Textstellen und Grafiken, die unser Projekt und die Funktionalitäten davon erklären
 - Einen Call-to-Action(CTA)-Button, der den User*in dazu einlädt, seine eigene Ausstellung zu erstellen. Beim Betätigen wird der Benutzer, falls er schon eingeloggt ist, weitergeleitet zum Editor, sonst zur Login Seite (hier gibt es die Möglichkeit, sich auch einen Account zu erstellen)
3. Als Besucher*in möchte ich eine Suchseite haben, um Designer*innen und deren Ausstellungen finden zu können. Auswahlkriterien:
 - CTA-Search Field
 - Eine Unterseite, welche durch den CTA-Button aufgerufen wird und unterschiedliche Optionen zur Verfügung stellt und das Suchergebnis ansprechend darstellt
 4. Als Besucher*in der Webseite, will ich beim Suchen filtern können, um für sich die relevantesten Ergebnisse zu bekommen. Auswahlkriterien:
 - filtern über ...
 - Tags
 - Favoriten
 - Erstellungsdatum
 5. Als Benutzer*in möchte ich meinen Ausstellungen Tags zuordnen können, damit diese leichter gefunden werden können. Auswahlkriterien:
 - Tags werden beim erstellen der Ausstellung aus einem vordefinierten Tag-Pool ausgewählt.
 6. Als User*in möchte ich mich auf verschiedene Arten in der Ausstellung bewegen können. Auswahlkriterien:
 - Per Slideshow (über den Vorwärts/Rückwärts-Pfeil zu nächstem Ausstellungsstück springen)
 - Per Touch / Click (Google Maps Street View, NavigationMesh in Three.js <https://github.com/donmccurdy/three-pathfinding>)
 7. Als User*in möchte ich auf das Ausstellungsstück klicken können, um weitere Informationen (z.B.: Titel, Künstler, Jahr, ...) zu erhalten. Auswahlkriterien:
 - größere Ansicht des Ausstellungsstücks wird vergrößert angezeigt
 - Nähere Details zum Ausstellungsstück, wie Titel, Künstler, Jahr, usw.
 8. Als User*in will ich eine Profil-Unterseite haben, auf der ich userrelevante Informationen angezeigt bekomme. Auswahlkriterien:
 - userrelevante Informationen, die angezeigt werden, sind:

- Name
 - Profilfoto
 - Erstellte Ausstellungen
 - Der*Die User*in kann dort seine Ausstellungen löschen.
9. Als User*in will ich bei der Erstellung einer Ausstellung zwischen verschiedenen Templates wählen können, um diese auf einfache Weise zu individualisieren. Auswahlkriterien:
- Es gibt 2 Templates am Anfang, in denen die Wandfarbe, der Boden und die Podeste für die Dateien vorgegeben sind.
 - Der*Die User*in kann keine Templates erstellen.
 - Der*Die User*in kann manuell die Podeste für jede Datei zwischen 5 vorgefertigten auswählen und für den Boden + Wand ein Pattern hochladen
 - Bei den Templates kann man aber manuell einzelne Aspekte verstellen (siehe Punkt oben)
10. Als User will ich meine Daten auf den Server laden, um diese jederzeit innerhalb einer Ausstellung platzieren und integrieren zu können. Auswahlkriterien:
- Unterstützte Dateien (Bilder-, Audio-, Video-, 3D-Dateien)
 - Uploadmöglichkeiten:
 - Drag and Drop
 - Im Ordner auswählen.
 - Dateigrößenlimit: 50MB
 - Dateityp Limit: nur Standardformate (Bilder: JPG, PNG, etc.; Audio: WAV, MP3, etc.; ...)
 - Fehlermeldung bei falschem Upload
11. 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. Auswahlkriterien:
- Falls dies nicht möglich ist, soll eine Fehlermeldung angezeigt werden
12. Als User*in möchte ich die Reihenfolge und Platzierung meiner Werke innerhalb der Ausstellung adaptieren können. Auswahlkriterien:
- Die Werke sollen zwischen vordefinierten Plätzen tauschbar sein.
13. Als User*in will ich andere Rechte haben, je nachdem ich angemeldet bin oder nicht. Auswahlkriterien:
- Wenn ein*e User*in angemeldet ist, kann er*sie:
 - Ausstellungen ansehen

- Ausstellungen erstellen/löschen
 - Ausstellungen favorisieren
- Wenn ein ein*e User*in nicht angemeldet ist, kann er*sie:
 - Ausstellungen ansehen
 - keine Ausstellungen erstellen
 - keine Ausstellungen favorisieren
14. Als User*in möchte ich meine Ausstellung abspeichern und löschen können. Auswahlkriterien:
- 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.

4.3 Evolutionäres Prototyping

Das evolutionäre Prototyping ist eine Art des Prototyping, dabei wird der Prototyp über die Dauer des gesamten Projektes weiterentwickelt, bis daraus das fertige Softwareprojekt wird.

In der Softwareentwicklung kommt es zu großen Problemen. Der Entwicklungsprozess dauert lang und die Kundschaft sieht erst am Ende des Werdegangs das funktionierende Produkt, deshalb kann es dazu kommen, dass Fehler und Missverständnisse aus der Spezifikationsphase erst zu diesem Zeitpunkt erkannt werden. Zu diesem Zeitpunkt ist jede Veränderung am Projekt sehr kostenaufwendig. Eine Lösungsstrategie für dieses Problem ist die Anwendung von Prototypen. Prototypen sind Annäherungen oder funktionelle Modelle von Systemteilen des fertigen Produkts. Sie werden im Vergleich zu dem Produkt mit weniger Aufwand produziert und das Klientel kann sich vorab ein Bild machen und Missverständnisse kommen durch die vermehrte Absprache mit der Kundschaft weniger auf und kann mit wenig Aufwand gelöst werden. [36]

4.4 Prototyp im Projekt

Im Projekt wurden mehrere Prototyparten verwendet, primär aber der evolutionäre Prototyp.

Anfangs wurde so die Machbarkeit des Projekts mit einer Umsetzbarkeitsanalyse, bei der der Prototyp eine große Rolle spielte, getestet. Im Prototyp (siehe Seite 14 Angular

Three) wurde überprüft, ob die Technologien auf die das Projekt aufbaut auch die Anforderungen (siehe Seite 22 Projekt User Stories) erfüllen konnten. Dadurch konnten die Erkenntnisse gemacht werden, dass Angular Three nicht geeignete für das Projekt war und eine anderen 3D-Web-API ThreeJs benutzt werden. Da dieses Resultat schon früh in dem Entwicklungsprozess gemacht wurde, kam mit der Änderung der 3D-Web-Apis im nur wenig Programmieraufwand zusätzlicher Aufwand.

5 Umsetzung

5.1 Design [L]

Im Designprozess wurden zwei Landingpages in Form von Mockups erstellt (siehe Abbildung 9), dafür wurde das Tool Figma benutzt. Die auszuwählende Landingpage soll das weitere Design des GUIs bestimmen. Zur Auswahl stand ein wertiges Design, das in dunklen Farben und eckigen Formaten auftritt. Das zweite Design ist ein fröhliches Design in bunten und hellen Farben und runden Formen. Der Auswahlgruppe wurde der Zweck des Prototyps erklärt und die beiden Designs präsentiert. In der Auswahlrunde erhielt das fröhliche Design das bessere Feedback und wurde ausgewählt.

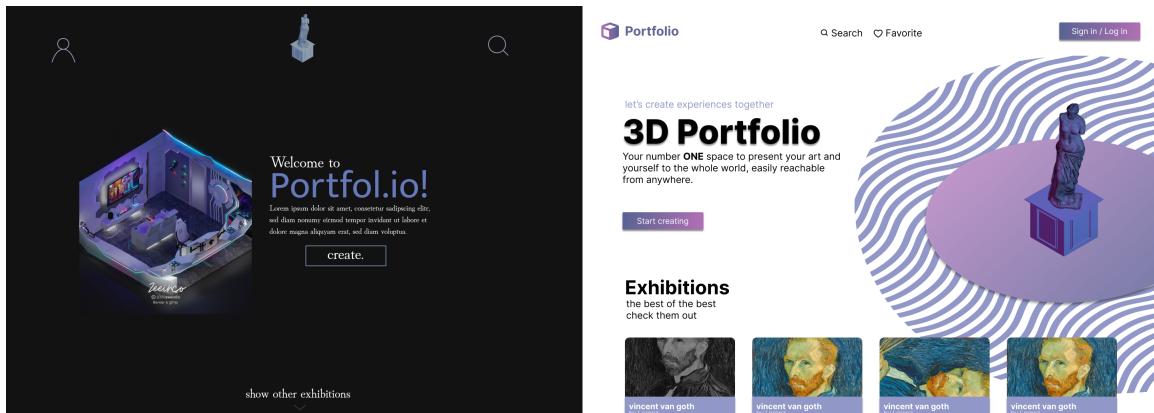


Abbildung 9: Landingpage Design Mockups Gegenüberstellung

5.1.1 Userexperience [L]

In keinem Softwareprojekt darf nicht der*die Benutzer*innen vergessen werden. Feature und Design müssen immer mit dem Gedanken entwickelt werden, wie hilft das dem*der Kunden*in oder der Zielgruppe weiter.

In den folgenden Kapiteln wird das Thema User Experience und wie dieses Thema im Projekt umgesetzt wurde, behandelt.

UX Design (Userexperience)

Ein gutes Userexperience-Design ist die Grundlage für eine erfolgreiche Positionierung und Kommunikation mit dem*der Benutzer*in. UX 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 Bedeutungen. Erstens gibt es den Punkt der Gestaltung der Interaktionen, hierzu gehört der Begriff Interface Design (UI Design), er bedeutet visuelle Kommunikation über Zeichen und Symbole. Zweitens gibt es Design als den Designprozess. Im Designprozess 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 [37].

Anwendungen von UX im Projekt

Schon beim Projektstart lagen die Benutzer*innen im Mittelpunkt. User Stories waren aus der Sicht eines Benutzers formuliert und das Projekt startete mit der Frage, welchen Nutzen kann das Projekt einem*einer Nutzer*in geben.

Im Userinterface-Design wurde Figma (siehe Kapitel Technologien Figma) benutzt. Die Oberfläche wurde so designt, damit sie leicht verständlich, nützlich und benutzerfreundlich ist. Dafür wurden viele Prototypen (siehe Abbildung 10) in Figma gestallten und durch Umfragen die besten bestimmt und dementsprechende Anpassungen am Design gemacht.

5.1.2 Corporate Design [M]

Das Corporate Design unterstützt die Corporate Identity und deren Ziele. Die Corporate Identity bezeichnet das äußere Erscheinungsbild eines Unternehmens . Hierbei sollen Bestandteile eines Unternehmens nach außen hin einheitlich, unverwechselbar und positiv wirken und mit der Gesamtstrategie stimmig sein. Zum Beispiel werden Mitarbeiter, Abteilungen und Produkte sowie ihre Beziehungen zur Firma nach gewissen Normen strategisch gestaltet und geregelt. Dabei lässt sich die Corporate Identity in 3 Teilbereiche gliedern:

- Corporate Behaviour
- Corporate Communication
- Corporate Design

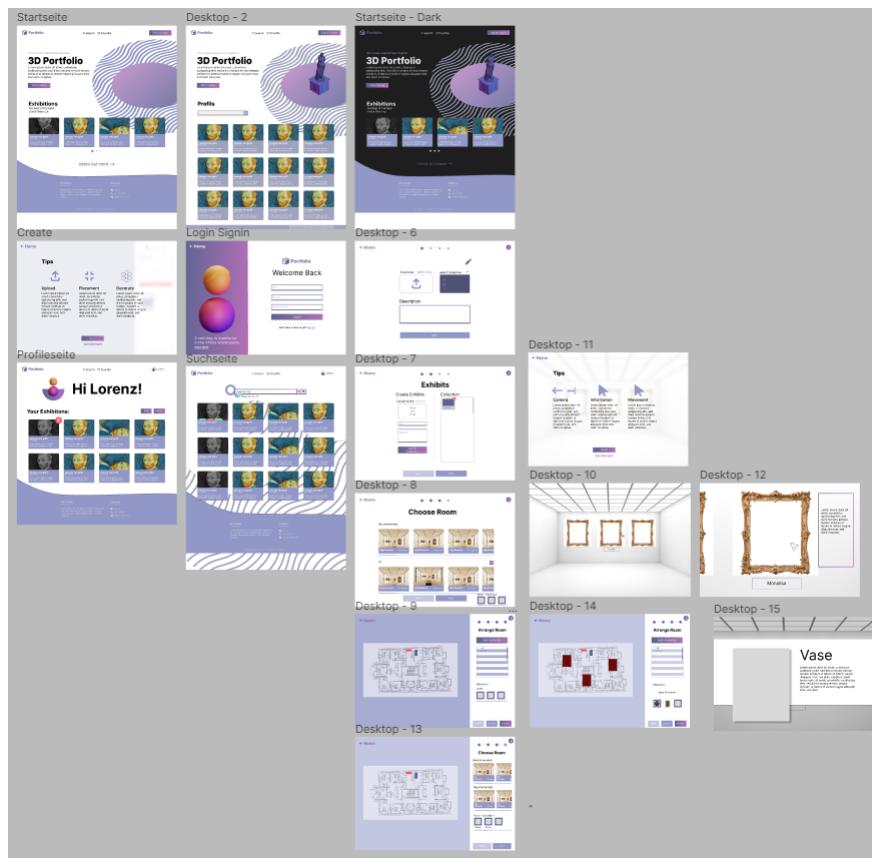


Abbildung 10: OberflächenDesign: Prototypen in Figma

[38]

Bei der Diplomarbeit kam vor allem das Corporate Design zum Einsatz. Hierbei wird vor allem das äußere Erscheinungsbild des Produkts als Einheit repräsentiert. Dabei werden zum Beispiel Hausfarbe, Logo, Gestaltungsraster und weitere Design-Elemente aufeinander abgestimmt. Dies war auch fester Bestandteil der Design-Phase. [39]

Der erste Schritt des Designs war es, zwei unterschiedliche Konzepte der ersten Webseiten-Elemente zu erstellen. Anschließend wurden diese verglichen und sich für eines entschieden. Dieses erste Grundkonzept wurde das Grundgerüst für den späteren Verlauf der Designarbeit.

5.1.3 Webseiten Design - Entwicklung [L]

In den vorherigen Kapiteln (Userexperience und Corporate Design) wurde bereits besprochen, wie das Konzept für das Webseiten-Design aussehen sollte und designt wurde. Eine kurze Zusammenfassung: Erst wurde das Corporate Design festgelegt und darauf aufbauend wurden die einzelnen Seiten der Webseite, um die Anforderungen

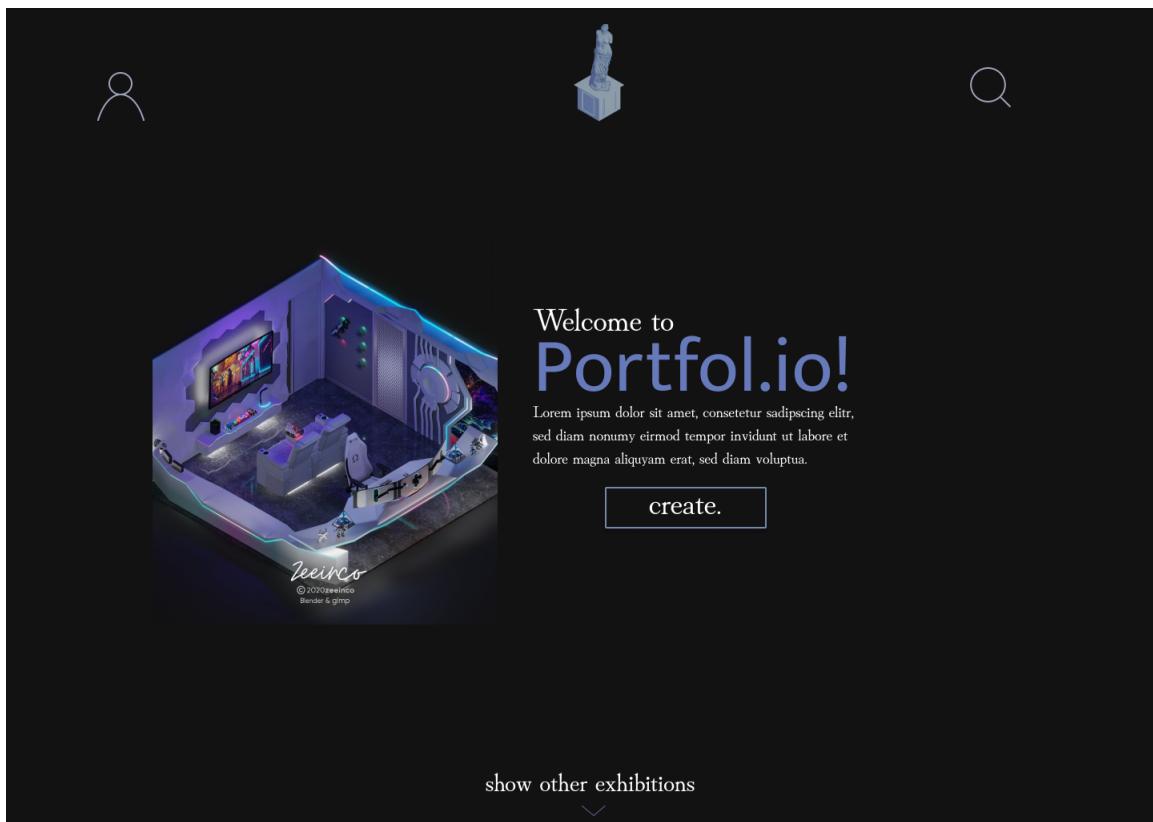


Abbildung 11: Design Konzept 1 Page 1

(siehe Seite 22 Projekt User Stories) zu erfüllen, designt, dafür wurde das Tool Figma verwendet.

Danach wurde das *Framework* Angular verwendet, um aus den einzelnen Webseiten-Seiten-Angaben eine echte Webseite zu entwickeln. Dabei wurden Userinterface-Frameworks (oder auch Design-Frameworks/Libraries) verwendet, um den Entwicklungsprozess zu beschleunigen.

5.1.4 Logoentwicklung [M]

Ein Schritt 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. Dabei muss auf folgende Faktoren geachtet werden:

- Wiedererkennbarkeit
- Flexibilität - geeignet für große und kleine Formate
- Einzigartig
- Innovation

Featured Exhibitions:

Vincent van Gogh
by Fabian

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Vintage van Gogh
by Fabian

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Vincent van Goth
by Fabian

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Gogh van Vincent
by Fabian

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

3D Portfolio

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Contacts

Austria
+43 1234567
info@example.com

Abbildung 12: Design Konzept 1 Page 2



Abbildung 13: Design Konzept 2

- Klarheit

[40]

Zur Erstellung des Logos wurde Adobe Illustrator verwendet, welches ein Programm speziell für das Illustrieren von Vektorgrafiken ist. Für das Logo der Diplomarbeit wurde ein isometrischer Zeichenstil verwendet. Dabei verlaufen alle Fluchlinien parallel und dadurch erscheint das Design zum einen flach, als auch dreidimensional. Es soll den 3D-Raum symbolisieren, in dem sich ein Podest befindet. Auch ist es in den Farben der Website gehalten, um leichter eine Verbindung mit dem Produkt schaffen zu können.

[41] [42]

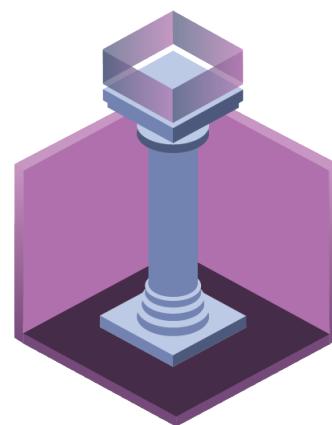


Abbildung 14: Logo der 3D-Portfolio-Gallery

5.2 Initialisierung der Landingpage [L]

Der erste Entwicklungsschritt für die vollständige Single-Page-Application beginnt mit der Initialisierung der Landingpage. Die Landingpage ist die Startseite der Webseite. Es ist das Erste, was der*die neue Nutzer*in sieht. Daher muss die Webseite alle benötigten Informationen über die Funktionen der Applikation verständlich erkennbar machen.

5.2.1 Aufsetzen der Landingpage [L]

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

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

Vorbereitung

Erstmal musste NodeJs installiert werden. Dafür wird aber zuvor noch NodeJs benötigt. NodeJs ist eine JavaScript Laufzeitumgebung. NodeJs kann von der eigenen Webseite nodejs.org mit dem Installer für alle Betriebssysteme installiert werden. Im Projekt wurde sich für eine LTS - Version(long term support) entschieden, weil diese am längsten von den Entwicklern am längsten unterstützt werden und dadurch beständiger sind. Während dem Installationsprozess von NodeJs kann durch eine Auswahl auch NPM installiert werden.

Angular installieren

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

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. Danach wird es mehrere Konfigurationsauswahlmöglichkeiten geben. Für dieses Projekt wurden Routing aktiviert und als Stylesheet-Formatierung Scss ausgewählt. Danach wurde in das (Projekt-) Verzeichnis Gallery gewechselt und dort mit dem Befehl serve der ein Webpack-Server gestartet, welcher den vorgenerierten Code von dem neu erstellen Angular-Projekt zeigt (siehe in Abbildung 15).

Globale oder Lokale Module Bei einer lokalen Installation werden die installierten Module in einem node-module Computerordner lokal im Projekt abgespeichert, während alle globalen Installationen in einem einzigen Computerordner, abhängig vom Computersystem gespeichert werden. Generell kann man bei NPM-Module zwischen lokalen und globalen Installationen unterscheiden. In der Regel ist eine lokale Installation besser, denn referenzieren mehrere Projekte auf ein globales Modul, kann es dazu kommen, dass bei einer Aktualisierung des globalen Modules verschiedene Projekte, sei

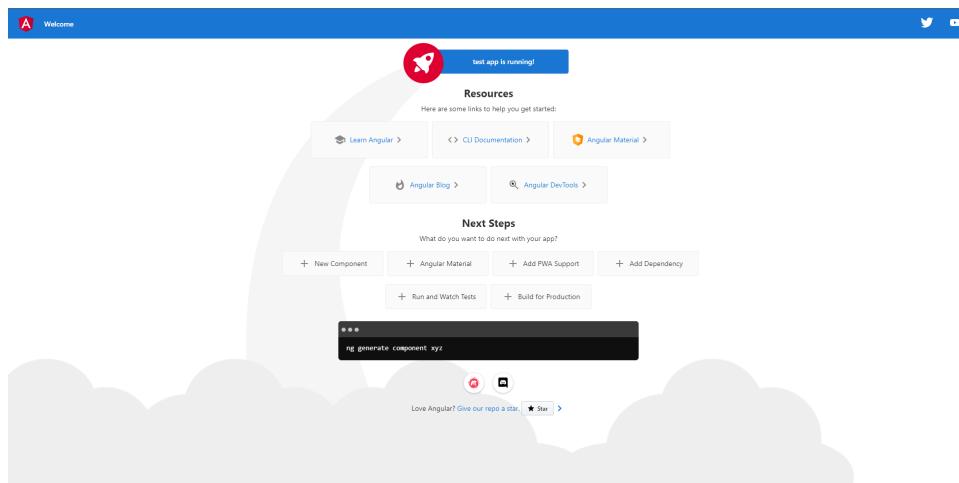


Abbildung 15: Angular: Automatisch generierte Start-Webseite

es wegen veralteten Funktionen oder neuer Logik, darauf anders reagieren und es zu Problemen bei diesen Projekten kommt, da nichts im Projekt darauf referenzieren muss. CLI-Module können aber auch lokal installiert werden und mit dem Befehl npx nur im Projektordner ausgeführt werden. [43]

Installation der UI-Framework

Nach der Installation von Angular wurden die UI-Frameworks installiert. Angular Material konnte mithilfe des Befehls 6 mittels der Angular CLI in das Angularprojekt eingebunden werden.

Listing 6: Terminal - Angular Material Installation

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

Bootstrap konnte mithilfe des Befehls 7 und von NPM installiert werden. Bootstrap wurde zwar dem Projekt hinzugefügt, Angular weiß aber davon noch nichts. Deswegen musste in der Anuglar-Konfigurationsdatei *angular.json* die Bootstrap Scss-Liberay eingebunden werden. Das wird in diesem Code 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
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
314
315
315
316
317
317
318
319
319
320
321
322
323
323
324
325
325
326
327
327
328
329
329
330
331
331
332
333
333
334
335
335
336
337
337
338
339
339
340
341
341
342
343
343
344
345
345
346
347
347
348
349
349
350
351
351
352
353
353
354
355
355
356
357
357
358
359
359
360
361
361
362
363
363
364
365
365
366
367
367
368
369
369
370
371
371
372
373
373
374
375
375
376
377
377
378
379
379
380
381
381
382
383
383
384
385
385
386
387
387
388
389
389
390
391
391
392
393
393
394
395
395
396
397
397
398
399
399
400
401
401
402
403
403
404
405
405
406
407
407
408
409
409
410
411
411
412
413
413
414
415
415
416
417
417
418
419
419
420
421
421
422
423
423
424
425
425
426
427
427
428
429
429
430
431
431
432
433
433
434
435
435
436
437
437
438
439
439
440
441
441
442
443
443
444
445
445
446
447
447
448
449
449
450
451
451
452
453
453
454
455
455
456
457
457
458
459
459
460
461
461
462
463
463
464
465
465
466
467
467
468
469
469
470
471
471
472
473
473
474
475
475
476
477
477
478
479
479
480
481
481
482
483
483
484
485
485
486
487
487
488
489
489
490
491
491
492
493
493
494
495
495
496
497
497
498
499
499
500
501
501
502
503
503
504
505
505
506
507
507
508
509
509
510
511
511
512
513
513
514
515
515
516
517
517
518
519
519
520
521
521
522
523
523
524
525
525
526
527
527
528
529
529
530
531
531
532
533
533
534
535
535
536
537
537
538
539
539
540
541
541
542
543
543
544
545
545
546
547
547
548
549
549
550
551
551
552
553
553
554
555
555
556
557
557
558
559
559
560
561
561
562
563
563
564
565
565
566
567
567
568
569
569
570
571
571
572
573
573
574
575
575
576
577
577
578
579
579
580
581
581
582
583
583
584
585
585
586
587
587
588
589
589
590
591
591
592
593
593
594
595
595
596
597
597
598
599
599
600
601
601
602
603
603
604
605
605
606
607
607
608
609
609
610
611
611
612
613
613
614
615
615
616
617
617
618
619
619
620
621
621
622
623
623
624
625
625
626
627
627
628
629
629
630
631
631
632
633
633
634
635
635
636
637
637
638
639
639
640
641
641
642
643
643
644
645
645
646
647
647
648
649
649
650
651
651
652
653
653
654
655
655
656
657
657
658
659
659
660
661
661
662
663
663
664
665
665
666
667
667
668
669
669
670
671
671
672
673
673
674
675
675
676
677
677
678
679
679
680
681
681
682
683
683
684
685
685
686
687
687
688
689
689
690
691
691
692
693
693
694
695
695
696
697
697
698
699
699
700
701
701
702
703
703
704
705
705
706
707
707
708
709
709
710
711
711
712
713
713
714
715
715
716
717
717
718
719
719
720
721
721
722
723
723
724
725
725
726
727
727
728
729
729
730
731
731
732
733
733
734
735
735
736
737
737
738
739
739
740
741
741
742
743
743
744
745
745
746
747
747
748
749
749
750
751
751
752
753
753
754
755
755
756
757
757
758
759
759
760
761
761
762
763
763
764
765
765
766
767
767
768
769
769
770
771
771
772
773
773
774
775
775
776
777
777
778
779
779
780
781
781
782
783
783
784
785
785
786
787
787
788
789
789
790
791
791
792
793
793
794
795
795
796
797
797
798
799
799
800
801
801
802
803
803
804
805
805
806
807
807
808
809
809
810
811
811
812
813
813
814
815
815
816
817
817
818
819
819
820
821
821
822
823
823
824
825
825
826
827
827
828
829
829
830
831
831
832
833
833
834
835
835
836
837
837
838
839
839
840
841
841
842
843
843
844
845
845
846
847
847
848
849
849
850
851
851
852
853
853
854
855
855
856
857
857
858
859
859
860
861
861
862
863
863
864
865
865
866
867
867
868
869
869
870
871
871
872
873
873
874
875
875
876
877
877
878
879
879
880
881
881
882
883
883
884
885
885
886
887
887
888
889
889
890
891
891
892
893
893
894
895
895
896
897
897
898
899
899
900
901
901
902
903
903
904
905
905
906
907
907
908
909
909
910
911
911
912
913
913
914
915
915
916
917
917
918
919
919
920
921
921
922
923
923
924
925
925
926
927
927
928
929
929
930
931
931
932
933
933
934
935
935
936
937
937
938
939
939
940
941
941
942
943
943
944
945
945
946
947
947
948
949
949
950
951
951
952
953
953
954
955
955
956
957
957
958
959
959
960
961
961
962
963
963
964
965
965
966
967
967
968
969
969
970
971
971
972
973
973
974
975
975
976
977
977
978
979
979
980
981
981
982
983
983
984
985
985
986
987
987
988
989
989
990
991
991
992
993
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1
```

```

9          "options": {
10            ...
11            "styles": [
12              ...
13                "node_modules/bootstrap/scss/bootstrap.scss"
14            ]
15          }
16        }
17      }
18    },
19  ...
20 }
21 ...

```

Installation von Three Js

Nach der Installation der UI-Libraries wurde ThreeJs installiert.

Mit dem ersten Befehl wird AngularJs eine JavaScript Libray durch NPM installiert und durch den zweiten Befehl wurde ein Typelibary für ThreeJs installiert um mit Typescript ThreeJs bearbeiten zu können.

```

1   npm install three
2   npm i @types/three

```

5.2.2 Components [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 sind die Bausteine für eine Angular-Anwendung. Mit ihnen lässt sich eine komplexe Benutzeroberfläche in mehrere unterschiedlich große Elemente unterteilen. Sie 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
- ein Stylesheet, dass das Aussehen der Benutzeroberfläche beeinflusst
- eine TypeScript-Testklasse, um individuell Komponenten zu testen

[44]

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 hilft beim Skalieren durch einige Features:

TypeScript hilft durch Code-Syntax, wie unter anderem Optionals, auch bei großen Applikationen schnell Fehler zu erkennen.

Durch die Angular CLI kann durch wenige Zeilen Befehle, zum einen ein Grundgerüst für die Entwicklung erstellt werden. Andererseits können Components, Services und andere Angular Funktionen in jedem Entwicklungsstadium problemlos hinzugefügt werden, ohne dass der bestehende Code beeinflusst wird. siehe Kapitel 5.2.1

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.

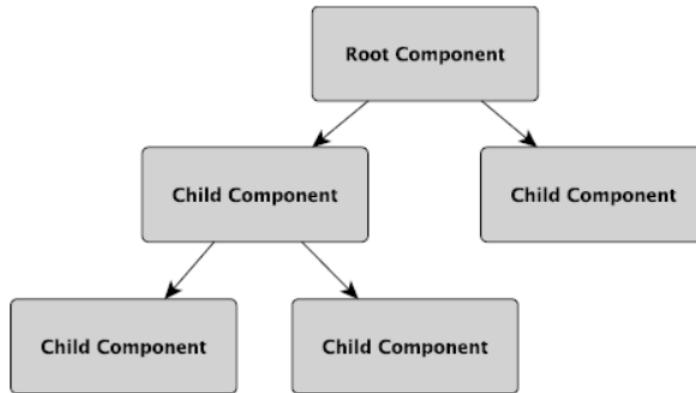


Abbildung 16: Component-Hierarchy [11]

Hierarchie

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

Die Hauptkomponente ist in jedem Falle 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.

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 [11] [45]:

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.

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, Aussehen und Funktionen der jeweiligen DOM-Elemente dynamisch beeinflusst und automatisch bei Datenänderung aktualisiert. [46]

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. [47]

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. [48]

Listing 13: Beispiel für Two-Way-Bindings [48]

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

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

5.2.3 Routing [M]

Beim Routing werden bestimmte Components angezeigt, abhängig von der aktuellen URL. Dabei kann durch die Applikation durch navigiert werden, was durch den sogenannten Router realisiert wird. Routing ist das Konzept einer Single-Page-Applikation, wodurch die Seite niemals neu geladen werden muss und alle Daten beim Navigieren beibehalten werden können. 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:

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 //TODO Referenz) 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:

- 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 es in der 3D-Gallerie Anwendung gelöst ist, 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      })
```

[11]

5.2.4 Angular Services [M]

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

Allgemein

Ein Service in Angular ist eine 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. [11] [49]

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 dadurch kann dieser von anderen Komponenten verwendet werden [11]:

Listing 19: Eine Klasse Injectable machen

1

```

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

```

Auch wird ein Provider mit angegeben. Dieser sorgt dafür, dass sich Services entweder nur in spezifische Komponenten injizieren lassen oder sich wie hier auf root-level, also überall injizieren lassen. [11]

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 API 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. Eine HTTP-Abfrage mit dem HTTP-Client wird wie folgt aufgerufen [11] [50]:

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 Room-Interface benutzt.

Interface [M]

Interfaces werden verwendet, um typsicher ein Objekt zu strukturieren. Dabei wird, um Daten des Servers korrekt erhalten zu können, eine bestimmte Entität der Datenbank

verglichen und durch die richtige Benennung und 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 [11]:

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 |
14          undefined, 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 Prototype

5.3.1 Zusammenfassung [M]

In diesem Abschnitt wird erklärt, auf welche Weise die 3D-Ausstellung in der Webapplikation realisiert wurde. Dabei wurde das Konzept des evolutionären Prototypings (siehe Prototyping 4.3) für die Entwicklung verwendet.

5.3.2 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. [51] [52]

Der Prozess des Renderings

Hinter dem Rendering stecken verschiedene Algorithmen, die ein solches 3D Modell anzeigen lassen. Verschiedene Werte und Daten werden zur Berechnung benötigt, welche die Qualität und die Geschwindigkeit des Prozesses unterschiedlich beeinflussen:

- das 3D-Objekt und wie realistisch es dargestellt werden soll
- die Art des Renderns
- die Lichtquellen und die entstehenden Schatten; wird Behandel im Kapitel Lichtsetzung 5.3.4
- die Kameraposition und Renderbereich (Clipping)
- der Anti-aliasing Prozess
- die Art des Render-Algorithmuses
- weitere atmosphärische Effekte

[52]

3D-Modelle

3D-Modelle besitzen verschiedene Eigenschaften, wie zum Beispiel ihre Textur, Farbe oder ihre Fähigkeit Licht zu reflektieren, die beim Rendern berücksichtigt werden müssen. Die 3D-Daten des Modells werden ermittelt und dabei in Pixelinformationen umgewandelt. Diese werden durch die Ermittlung der Koordinaten des Objektes an bestimmten Positionen abgebildet. [52]

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.
- Echtzeit-Rendering - Beim Echtzeit-Rendering wird ein 3D-Modell in kürzester Zeit gerendert. Diese Art kommt zum Beispiel bei Videospielen oder im 3D-Gallery zum Einsatz. Hierbei werden meist 24 Bilder pro Sekunde gerendert, um Bewegungen flüssig wahrzunehmen.

[53]

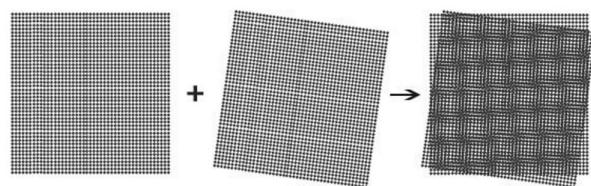


Abbildung 17: Der Moire-Effekt visualisiert [54]

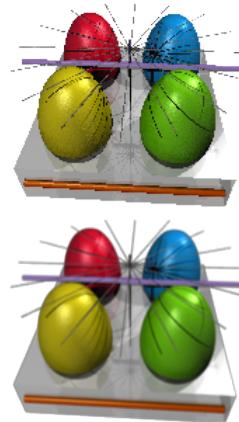


Abbildung 18: Rendering mit und ohne Anti-Aliasing [55]

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 die unteren und oberen Clipping-Ebenen werden durch die Kameraposition und die Fensterposition definiert. Auch gibt es das Far- und Near-Clipping, wodurch zum einen die Tiefe des 3D-Modells begrenzt wird. Zum anderen werden unerwünschte Objekte, die im Hintergrund liegen, ebenfalls nicht gerendert. [52]

Anti-Aliasing

Beim Anti-Aliasing Prozess wird versucht, den auftretenden Aliasing-Effekt zu reduzieren und zu entfernen. Der Aliasing-Effekt tritt auf, wenn die Pixel zu grob für feine Strukturen und Muster sind. Der Effekt ist besonders stark, wenn diese Muster nicht senkrecht sind. Dieser Aliasing-Effekt wirkt sich auf das 3-Modell aus, in dem Zacken an Polygon-Kanten entstehen, ein Moiré-Effekt auftritt (siehe Abbildung 17) Verweis oder generell unerwünschte Muster auftreten.[52]

Verschiedene Render-Algorithmen [M]

Rasterization

Über den Bildschirm wird ein Raster aus meistens Dreiecken gespannt, da diese am einfachsten zu berechnen sind. In den Ecken der Dreiecke, auch genannt Scheitelpunkte, sind die Informationen enthalten, die zur Darstellung eines 3D-Objektes in 2D benötigt werden. Dieser Prozess wird für Echtzeit-Rendering genutzt. Er ist zwar rechenintensiv, jedoch nicht so intensiv wie beim Ray-Tracing. [56]

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. [52]

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. [52]

Raycasting

Beim Raycasting wird ermittelt, welches 3D-Objekt zuerst von einem 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 und die Fläche normal dazu bestimmt die Schattierung. Dieser Prozess wird im Kapitel 5.8.6 näher erklärt und angewandt. [52]

Ray-Tracing

Der Ray-Tracing Prozess funktioniert ähnlich wie der Prozess des Raycastings. Beim Ray-Tracing werden 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

[56]

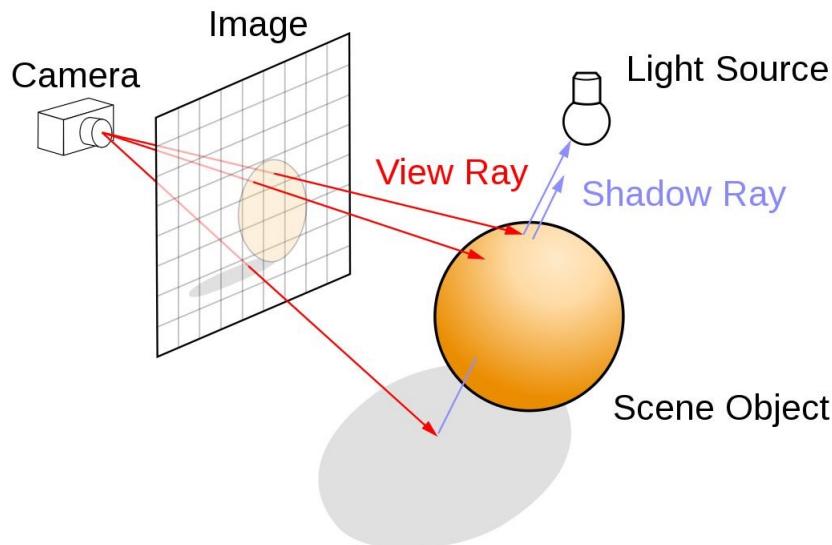


Abbildung 19: Vorgang des Ray-Tracing [56]

Rendering in Three.js

Für das Rendern der 3D-Gallery wird der in Three.js inkludierte Renderer von WebGL (siehe ThreeJS 2.2.3) Referenz verwendet. Um das gerenderte 3D-Objekt in das DOM (siehe DOM 6.3) zu integrieren, wird das HTML-Element <canvas> verwendet. [57]

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. [58]

Listing 24: Canvas als View-Child initialisieren

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

Anschließend wird ein neuer WebGLRenderer angelegt, dem der Canvas zugewiesen wird. Falls noch kein Canvas besteht, wird automatisch ein neuer erstellt. [57]

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. [59]

Listing 26: Animations-Schleife

```

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

```

5.3.3 Resizing [L]

<https://www.omnicalculator.com/other/resolution-scale> <https://discourse.threejs.org/t/render-half-size-then-upscale/13228>

5.3.4 Lichtsetzung

Das Licht ist das 2. Wichtig, denn ohne würden die gerenderten Objekte im Dunklen stehen. Um dies zu ändern, wird eine Lichtquelle installiert, um einen bestimmten Bereich der Szene zu erhellen. 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. [60]
- 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. [61]

- Das AmbientLight belichtet die gesamte Szene gleich, kann dabei jedoch keine Schatten werfen. [62]
- Das DirectionalLight wirft Licht parallel in eine bestimmte Richtung. Da es unendlich weit weg erscheint, wird es oft als Tages- oder Sonnenlicht verwendet. [63]
- Das HemisphereLight wird über der Szene positioniert und besitzt eine Himmel- und Bodenfarbe mit Verlauf. Diese Lichtquelle kann keine Schatten werfen [64]
- Das PointLight wirft Licht von einem Punkt in alle Richtungen. Es soll eine Glühbirne simulieren. Hierbei kann ebenfalls die Reichweite des Lichtes und ein Dimmefekt bestimmt werden. [65]
- Das RectAreaLight strahlt Licht in Form eines Rechtecks aus. Damit können zum Beispiel Fenster simuliert werden [66]
- Das SpotLight wirft Licht in Form eines Kegel [67]

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( 0xffffff, 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

Die Landingpage (siehe Abbildung 20) repräsentiert das Projekt, denn es ist das erste, was ein neuer Benutzer*in von der Webanwendung zu sehen bekommt. Es war wichtig, ein modernes Design dafür zu entwickeln.

Wichtige Punkte, die bei der Landingpage beachtet wurden, waren die Hierarchie der Elemente. Zuerst gab es eine Einleitung, um das Projekt zu erklären, danach kam



Abbildung 20: Landing Page

ein großer Knopf, der dazu einlädt, das Produkt zu verwenden. Er verlinkt zu der Anmeldung. Danach kam eine Selektion aus den neuesten erstellten Ausstellungen, damit kann sich der User gleich ein Bild vom Projekt machen.

Um zwischen den verschiedenen Unterseiten der Webseite zu wechseln, wurde am oberen Bildschirmrand zusätzlich eine Navigationsleiste implementiert. Diese sogenannte Navbar besitzt je nach Unterseite verschiedene Designs. Dies ermöglicht ein angenehmes Navigieren auf jeder Seite (Siehe Routing 14). Ebenfalls muss auch der Footer auf jeder Unterseite passend angezeigt werden. Um das Design der Navbar und Footer auf jeder Seite individuell zu gestalten, wird ein Service jeweils in die entsprechende Komponente injiziert. Dadurch können Konfigurationen am Design durch wenig Code und Redundanz vorgenommen werden.

5.5 Userverwaltung [L]

Im Projekt wurde eine Userverwaltung eingebaut. Die User können sich auf der Weboberfläche anmelden und registrieren. Je nach Registrationsstatus (auf der Webseiten angemeldet oder nicht) stehen ihnen mehr Funktionalitäten zu Verfügung. Ein*e Besucher*in, die nicht angemeldet ist, kann alle Ausstellungen ansehen und auf der Suchunterseite nach Ausstellungen suchen. Ein*e requistiert*er User*in könnte das

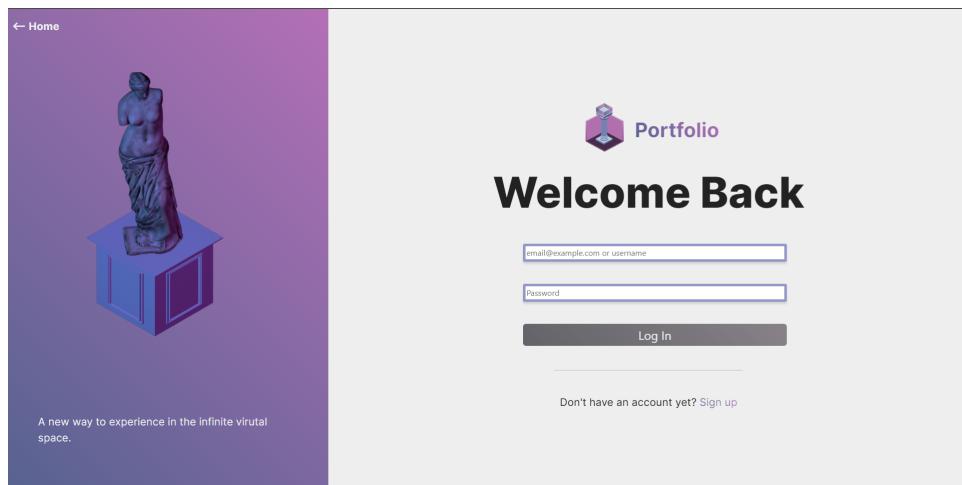


Abbildung 21: Projekt: Login Page

auch, zusätzlich können sie auch ihre Profileseite 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 Registrationsstatus auch besuchen kann.

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

5.5.1 JWT - Json Web Token [H]

5.5.2 Implementation im Frontend [L]

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

Sign- und Log-In

Da der JWT 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 Userinterface wurden Anmelde- und Registrierungs-Funktionalitäten implementiert (siehe Abbildung der Anmeldeseite 21). Die Anmelde- und Registrationsformulare wurden mithilfe von reaktiven Formularen und Validatoren (siehe Absatz 5.5.2) umgesetzt. Nachdem eine valide Eingabe von dem*der User*in gemacht wurde, aktiviert sich der Anmelde- bzw. Registrierungsknopf und lässt sich drücken. Die Aktivierung des

Knopfes wird durch die Farbveränderung von Grau auf Bunt sichtbar gemacht. Nachdem der*die User*in die Eingabe bestätigt hatte, indem sie*er den Knopf drückte, werden die Anmeldedaten über das HTTPS-Protokoll mithilfe des HTTP-Moduls (siehe Abschnitt TODO referzieren fabians http-module is not net commitet) an den Server gesendet. Nach einer erfolgreichen Anmeldung / Registration wird an die Weboberfläche eine Antwort geschickt, in der der JWT Token enthalten ist. Danach wird der*die User*in auf die Profilseite weitergeleitet. Bei einer gescheiterten Anmelde- / Registrierungsversuch wird der*die User*in durch eine Fehlermeldung darüber informiert.

Nach der erfolgreichen Anmeldung / Registration wird vom Server ein JWT ausgestellt. Bei einem HTTP-Request vom Client zum Server kann der JWT 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 Unbequemlichkeiten, die aus diesem Prozess resultieren. Der*Die User*in muss sich jedes Mal, bei jedem neuen Webseitbesuch und beim neuen Laden der Webseite, neu anmelden und der*die Entwickler*innen müssen bei jedem HTTP-Request (also jeder Kommunikation mit dem Server) manuell den JWT in den Header platzieren.

Formulare und Validation In Webanwendungen gibt es viele Formulare, weil sie dort als eine der primären Kommunikationsschnittstellen zum Besucher*in dienen. Natürlich gibt es dafür den nativen Inputelement von HTML, aber zu einem Formular mit guter Usererfahrung gehört auch ein visuelles Feedback für den*die Benutzer*in, zusätzlich wächst der Entwicklungsaufwand exponentiell mit der Featureanzahl des Formulars.

Angular bietet verschiedene Implementierungsarten, um die visuelle Komponente 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. [11, 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 der Anmeldung zu lösen, wurde der JWT im LocalStorage des Browsers gespeichert. Im Codeausschnitt (siehe 28) ist ein Teil des Authentifizierungsdienstes des Projektes zu sehen. Mit der Funktion `setSaveJWT` wird der ausgestellte JWT ausgelesen und die darin enthaltenen Informationen, der Name und die Id des Benutzers, die Id des Tokens und das Ablaufdatum des Tokens im LokalenStorage (siehe im Absatz WebStorage API 5.5.2) gespeichert, damit die Daten auch nach dem Neuladen 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 JWT enthaltene Ablaufdatum und vergleicht es mit dem aktuellen Datum. Zuletzt gibt es noch die `logout` Funktion, diese löscht die im LocalStorage enthaltenen JWT-spezifischen Daten.

Listing 28: auth.service.ts - JWT 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   logout() {
22     localStorage.removeItem("user")
23     localStorage.removeItem('id_token')
24     localStorage.removeItem('expires_at')
25     localStorage.removeItem('user_id')
26   }
27   ...
28 }
```

WebStorage API Die 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. Die 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 `LocalStorage` und der `SessionStorage`. [68] [69]

SessionStorage Daten im SessionStorage werden je nach ihrem Ursprung getrennt aufbewahrt. Sie werden für die Zeit der Webseitensession gespeichert, wenn der Browser oder der Tab geschlossen wird, sind die Informationen auch fort. [69]

LocalStorage Daten werden wie im SessionStorage auchpersistiert, doch auch wenn der Browser geschlossen wird bleiben die Daten erhalten. Daten können nur durch JavaScript oder das Löschen des Webbrowsers-Caches gelöscht werden. [69]

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 meistens 5 MB und es können nur Zeichenketten gespeichert werden. [69]

HTTP-Interceptoren

HTTP-Interceptoren funktionieren in Angular als Zwischenschicht zwischen den ausgehenden HTTP-Abfragen und den eingehenden HTTP-Antworten und können diese umwandeln. Das Einsatzgebiet von HTTP-Interceptoren ist bei globalen Änderungen und Funktionen, die an allen HTTP-Requests durchgeführt werden sollen. Es war somit das perfekte Werkzeug, um in allen HTTP-Requests den JWT in den Header zu verpacken. [11, 10.3 Interceptoren: HTTP-Requests abfangen und transformieren]

Wenn ein Request an den Server geht, wird dieser unterbrochen. Es wird überprüft, ob der JWT verfügbar ist. Wenn das zutrifft, wird der ursprüngliche Request geklont, in der Authentifizierungsspalte des Headers wird die JWT-Id hinzugefügt, danach wird der Request wieder zum Server weitergeleitet (siehe Code 29).

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) {
      const cloned = request.clone()

```

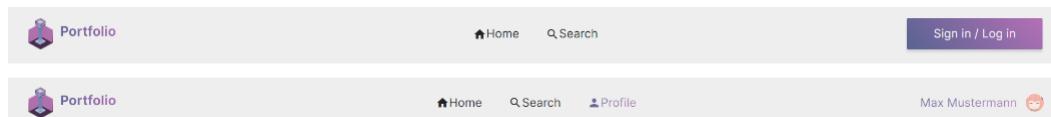


Abbildung 22: Navbar: authentifiziert vs noch nicht authentifiziert

```

20      {
21        headers: request.headers.set('Authorization', 'Bearer ' + concat(idToken))
22      }
23    )
24
25    return next.handle(cloned)
26  }
27
28  return next.handle(request);
29}
30

```

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 zu den Anmeldestatus zu zeigen.

Deswegen und um zu verhindern das Benutzer*in auf Unterseiten sind auf denen sie wegen ihres Requistratonstatus (noch nicht authentifiziert) noch nichts machen können, wurde je nach Anmeldestatus andere Routen und Informationen auf der Navigationsbar gezeigt (siehe Abbildung 22). Dafür wurde die Navigationsleistenlogik und die Navigationsleistenservice angepasst. Zusätzlich wurde AuthGuard verwendet um Routen zu sperren auf, die der*die Benutzer*in wegen dem Requistratonstatus noch keinen Zugriff hat.

5.5.3 Profilepage

Auf der Profileseite (siehe Abbildung 23) sieht der*die Benutzer*in userspezifische Informationen wie den Profilnamen, das Profilfoto, die erstellten Exhibition. Zusätzlich kann der*die Benutzer*in hier neue Ausstellungen anlegen und bereits erstellte Ausstellungen löschen.

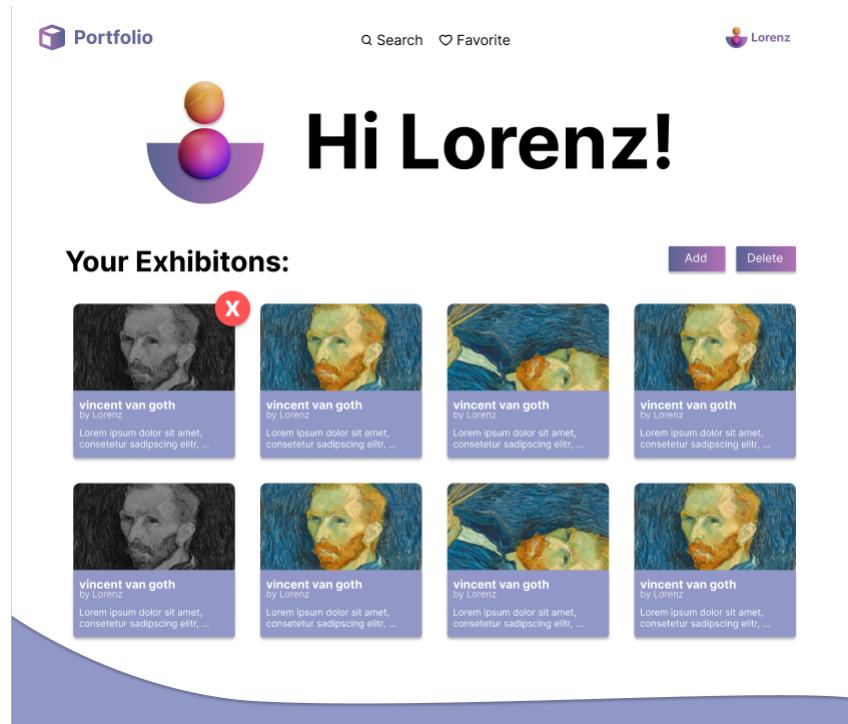


Abbildung 23: Profilepage

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

Das 3D-Gallery-Portfolio lässt 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 27) teilen. Die Kategorien sind:

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

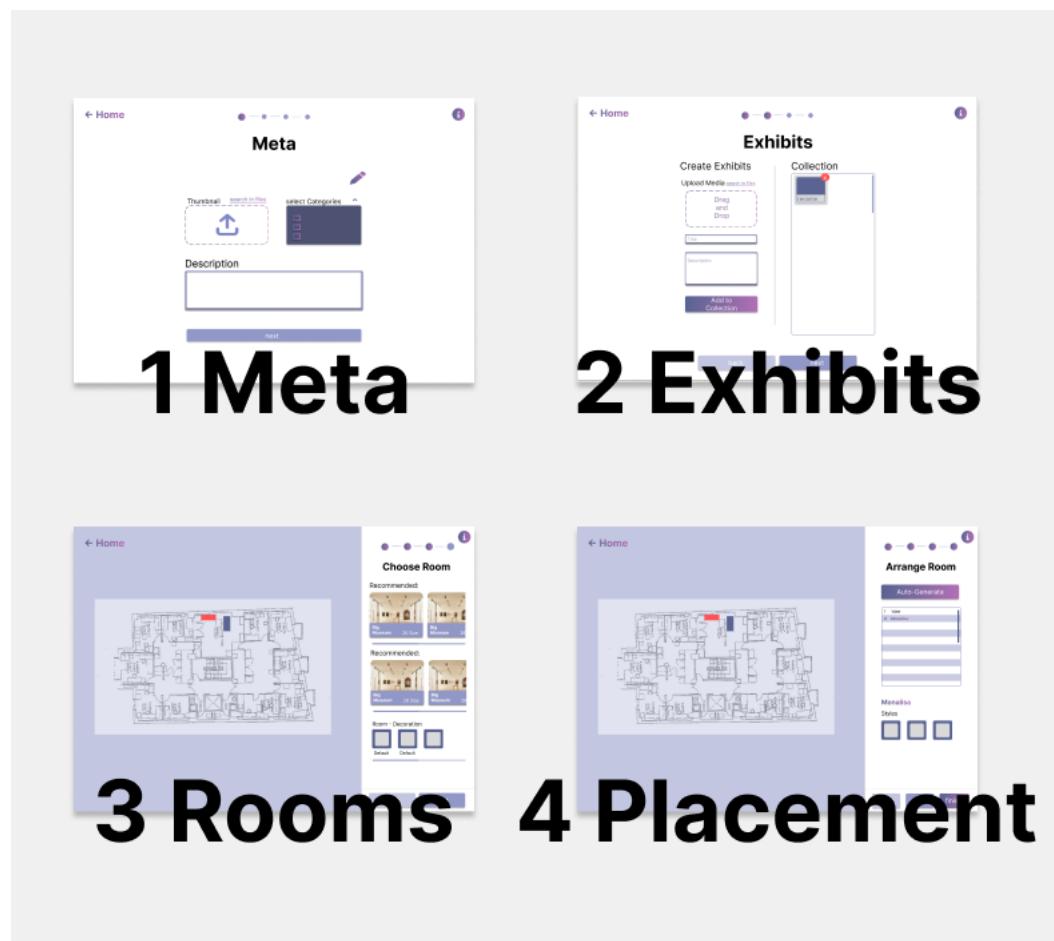


Abbildung 24: Inhaltserstellungstool

- 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 SPA werden die vier Kategorien durch Klassen (siehe Abbildung 29) definiert.

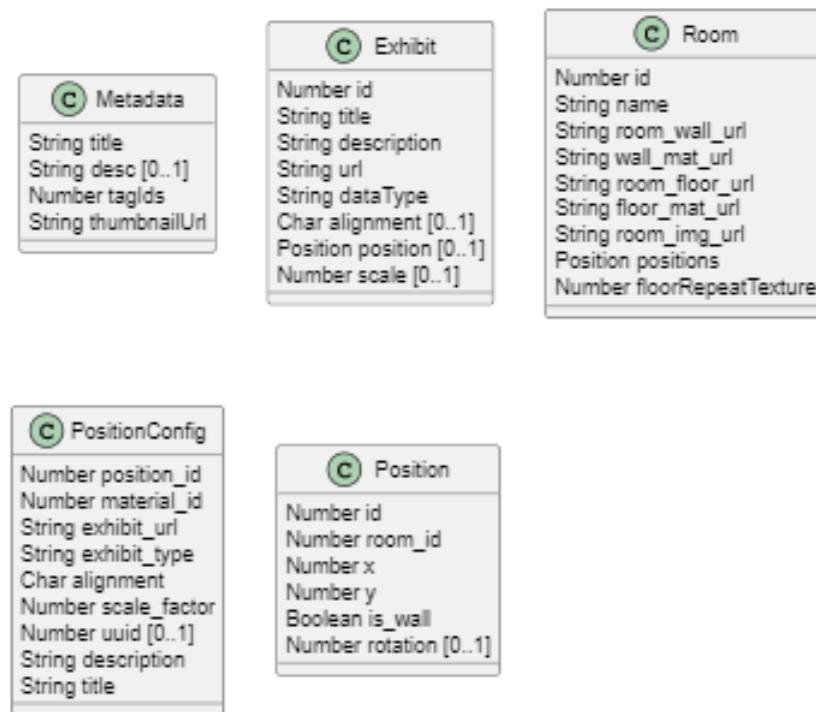


Abbildung 25: Inhaltserstellung Datenklassen

5.6.2 GUI

Auf der Weboberfläche beginnt der Prozess der Content Creation damit, dass auf der Profilseite der 3D-Gallery-Portfolio Hinzufügen Knopf (siehe Abbildung 26)

5.7 3D-Portfolio-Gallery Konfigurationstool [L]

Die Inhaltserstellung-Funktion ist die Kernfunktion des Projekts. Die Content Creation definiert ein 3D-Portfolio. Ziel ist ein einfacher und intuitiver Konfigurationsprozess für ein 3D-Portfolios. Im Entwicklungsprozess wurden viele Entscheidungen getroffen, die auf das ganze Projekt Einfluss hatten. Z.B. wurde eine Art der Datenrepräsentation der 3D-Portfolios und die Darstellungsweise der Daten in der 3D-View entwickelt.

In diesem Kapitel wird die Softwareentwicklung und der Prozess der Inhaltserstellung erklärt.

5.7.1 Überblick [L]

Das Inhaltserstellungs-Tool besteht aus insgesamt vier Bereichen (siehe Abbildung 27), mit diesen werden Daten gesammelt, um das Portfolio zu erstellen und vielen

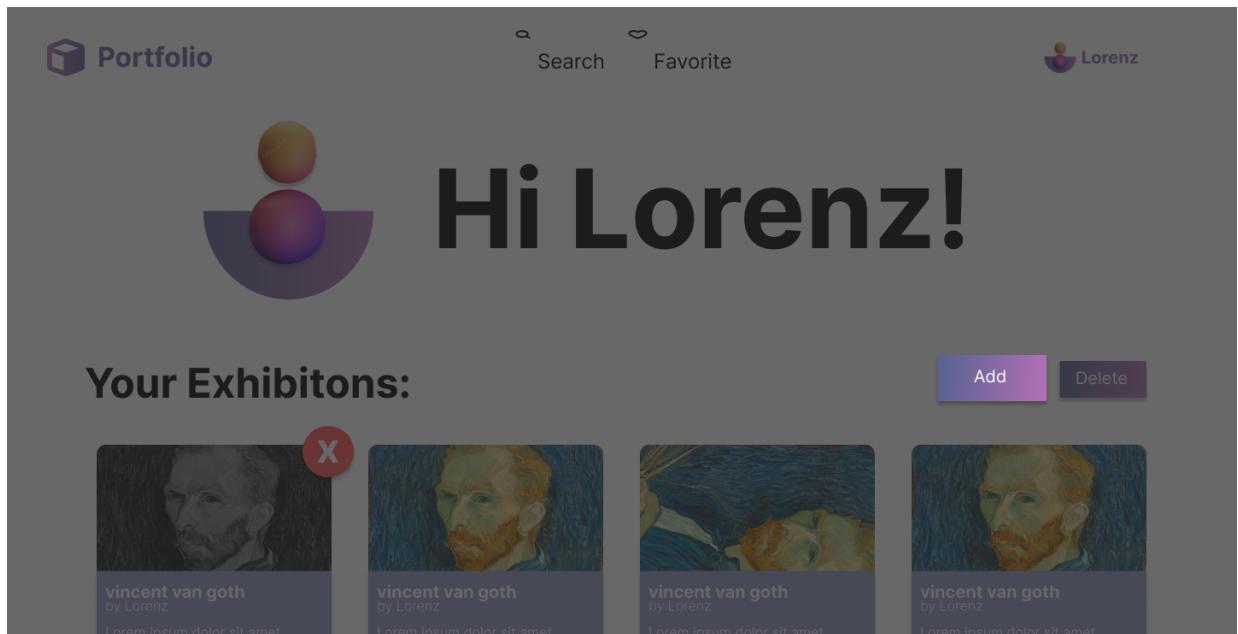


Abbildung 26: Profileseite: 3D-Gallery-Portfolio Hinzufügen Knopf

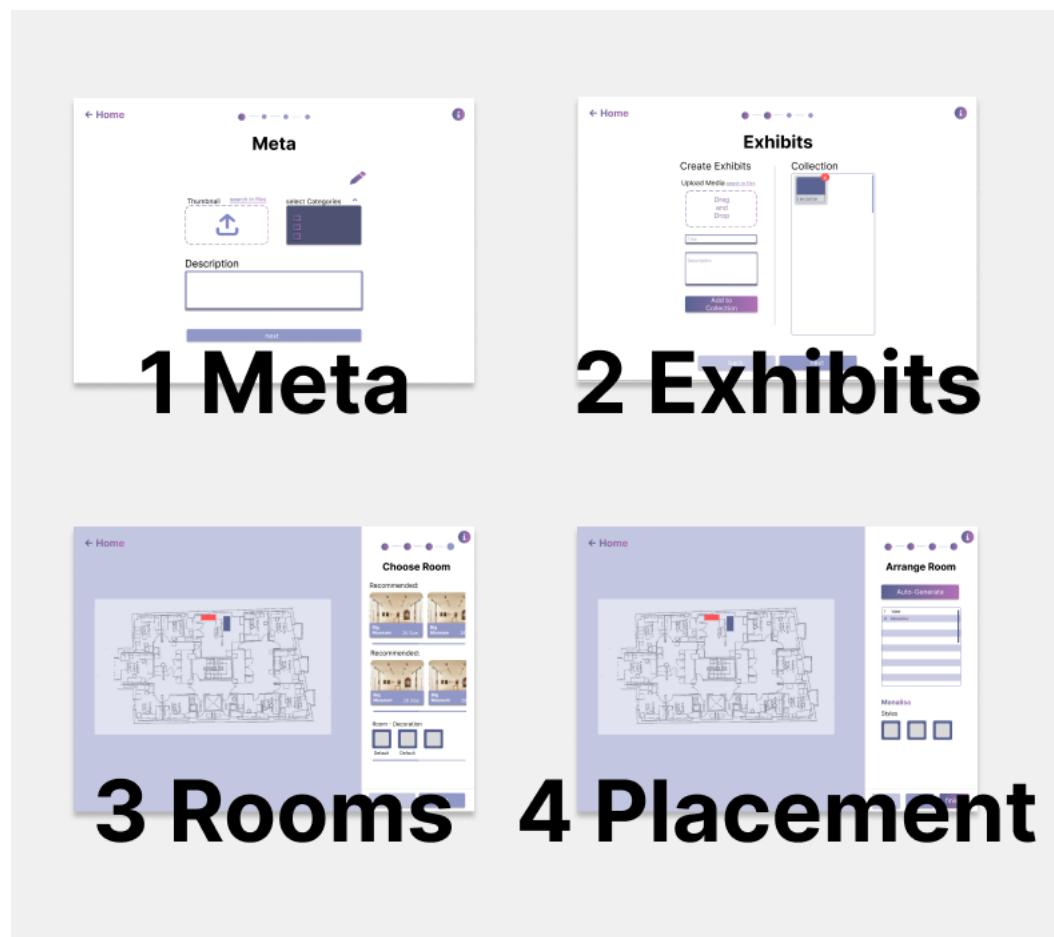


Abbildung 27: Inhaltserstellungstool

Untersystemen (bestehend aus Komponenten, Klassen und Services siehe Unterkapitel 5.7.4 auf Seite 63) , die beim Erstellungsprozess helfen.

Diese Unterseiten wurden aufgeteilt, damit 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* (mehr dazu im Unterkapitel 5.7.2 auf der Seite 60) orientiert.

5.7.2 Userinterface-Struktur / Wizard Design Pattern [L]

Im Projekt wurde das Wizard-UI-Pattern benutzt, um den Konfigurationsprozess des Portfolios einfacher zu gestalten. Dafür wurden die Konfigurationsdaten nach 4 Kategorien aufgeteilt und für jede eine eigene Unterseite (siehe Abbildung 27) entworfen.

Die Kategorien sind:

- Metadaten - Die Metadaten sind beschreibende Daten für das Portfolio, dazu gehören Name, zugehörige Kategorien, eine Beschreibung und ein Thumbnail.
- Ausstellungsstück - Die Daten des Ausstellungsstücks bestehen aus den vom User auf den Server geladenen Kunstwerken und Zusatzinformation wie Namen und Beschreibung.
- Raumdaten - Raumdaten bestehen aus den virtuellen 3D-Daten des Raumes und weiteren Konfigurationsdaten wie den Positionen, an denen Ausstellungsstücke im Raum platziert werden können.
- Ausstellungsplatzierungsdaten - Mit diesen Daten werden die Daten zu den Ausstellungsstücken und dem virtuellen Raum logisch verbunden.

Begriffserklärung [L]

Der Begriff "Wizard" in der Softwareentwicklung, kommt ursprünglich von Systemadministratoren, welche bei komplizierten Installationsprozessen halfen oder ganz übernahmen. Mit der Zeit änderte sich der Begriff zu Software-Assistenten-Programmen, welche dem*der User*in dabei unterstützten, die Software zu konfigurieren. [70, Ursprung des Begriffs Wizard]

Wizard-UI-Pattern [L]

Im Web-Bereich gibt es in der Regel zwei Arten Dateneintragungen zu verarbeiten.

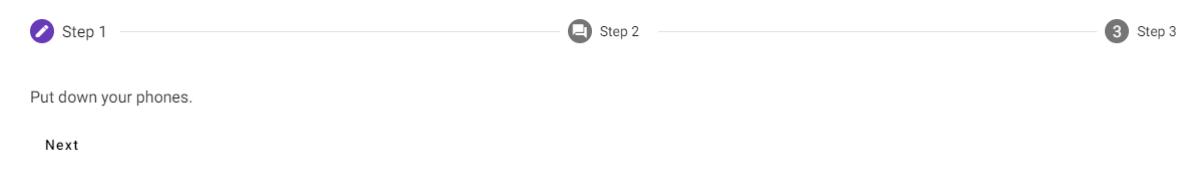


Abbildung 28: Angular Material: horizontaler Stepper [72]

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 den Eingaben können Wizards die Reihenfolge oder die Formulare ändern.

TODO: Bei Bedarf könnte hier mehr geschrieben werden. [71, Wizards: Definition and Design Recommendations]

Implementation [L]

Für die Implementation 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 28). [72]

5.7.3 4 Formularunterseiten [L]

Nachdem die Angular Material Komponente Mat-Stepper für die Wizard-Funktionalität gewählt wurde, war der nächste Schritt, die vier Formularunterseiten des Wizards (siehe Abbildung 27) zu gestalten.

In diesem Unterkapitel werden die Funktionalitäten und die Probleme bzw. Lösungen dieser Unterseiten beschrieben.

Kommunikationsschnittstelle [L]

Die Unterseiten sammeln Daten für die Erstellung des Portfolios. Die Kommunikations-schnittstellen der Unterseiten helfen dabei, dass die Unterseiten miteinander kommunizieren können. Dafür wurde das *Observer-Design-Pattern* (siehe im Glossar 6.3 auf Seite 77) gewählt.

In einer Dienstklasse, die durch Dependency Injection (siehe Unterkapitel 5.2.4 auf Seite 41) mit allen Unterseiten verbunden ist, befinden sich Observablen, die mit den

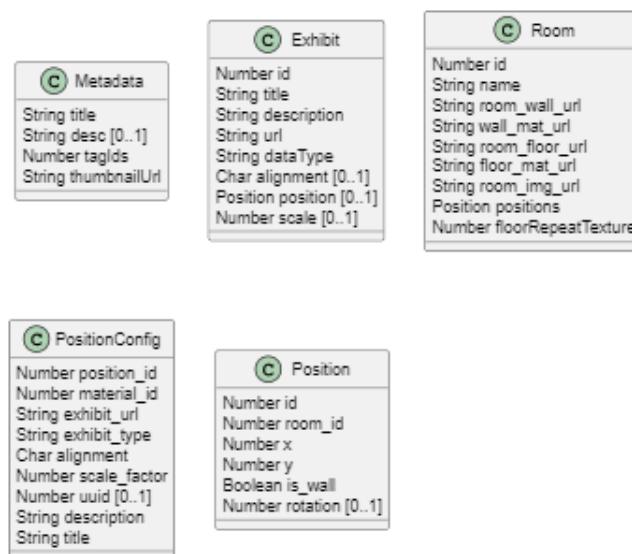


Abbildung 29: Inhaltserstellung Datenklassen

Eingabefeldern der Unterseiten verbunden sind. Wenn auf den Unterseiten eine neue Dateneingabe geschieht, wird die Veränderung an die Dienstklasse weitergegeben. Die Unterseiten haben jeweils eine Datenklasse nach den Kategorien des Portfolios (siehe Abbildung 29), in der die Daten gespeichert werden.

In der Dienstklasse werden Veränderungen durch den SessionStorage (siehe Unterkapitel 5.5.2 auf Seite 53) gespeichert und an den Wizard Mat-Stepper weitergeleitet, um dort den Fortschritt des Konfigurationsprozesses anzuzeigen.

Durch die Speicherung im SessionStorage kann der Konfigurationsprozess selbst bei einem Abbruch in der Zukunft nahtlos fortgesetzt werden.

Metadaten - Unterseite [L]

Die Metadaten-Unterseite (siehe Abbildung 30) wurde erstellt, um Metadaten wie den Namen, die Kategorien, das Vorschaubild und eine kurze Beschreibung zu der Ausstellung zu sammeln.

Alphanumerische Daten wie der Name und die Beschreibung wurden mit Angular's reaktiven Formularen (siehe Kapitel 5.5.2 auf Seite 52) gesammelt.

Das Vorschaubild wird hochgeladen mit der Dateien-Hochlade-Komponente (siehe Kapitel 5.7.4 auf Seite 63). Auf der Webapplikation wird die URL gespeichert, unter der die hochgeladene Datei zur Verfügung steht.

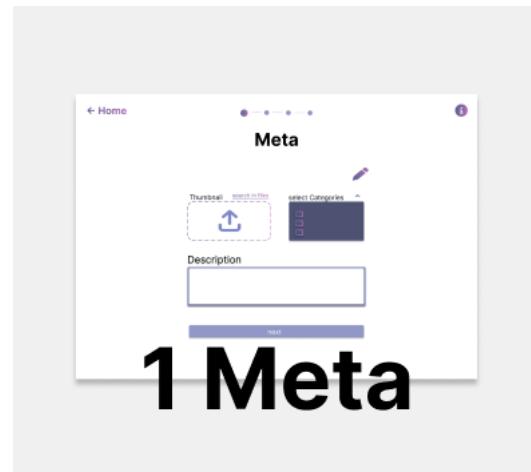


Abbildung 30: Metadaten Unterseite

Ausstellungsstücke - Raum

5.7.4 Unterstützende Services und Klassen

Dateien-Hochlade-Komponente [L]

TODO: Image vom der Komponente

Die Dateien-Hochlade-Komponente wurde mit dem Gedanken entwickelt, hoch modular und anpassungsfähig zu sein. Denn die Hochlade-Funktion wird öfters im Projekt gebraucht.

Bei der Komponente sind akzeptable Filetypen und Filegrößen einzustellen, werden die gesetzten Anforderungen an Filetypen und Filegrößen von den hochzuladenden Daten nicht erfüllt, so werden diese auch nicht hochgeladen 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.

Funktionalitäten und Implementierung [L]

Hochlade Funktionalitäten des Servers [E]

Portfolio-Veröffentlichung-System [L]

5.8 Interaktions-Seite [M]

Bei der Implementierung der Interaktions-Seite, ist es darum gegangen, eine möglichst benutzerfreundliche Besucheransicht der vorhandenen Ausstellungen zu erstellen.

5.8.1 Verbindung der Interaktions-Seite mit der Landing-Page

Vorerst muss darauf geachtet werden, dass wenn der*die Benutzer*in eine Ausstellung, die er*sie besuchen möchte, auswählt, auch auf die richtige weitergeleitet wird. Dabei hilft das Routing mit Parametern (siehe Routingparameter 5.2.3). Jede Ausstellung besitzt eine eigene ID. Diese ID 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 5.7 beschrieben wird, richtig in den Raum geladen und platziert.

5.8.2 Bewegung

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. Im Endeffekt 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 [73]
- 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. [74]
- OrbitControls - Der*die Nutzer*in kann mittels linker Maustaste um ein bestimmtes Ziel kreisen. Durch die rechte Maustaste kann dieses Ziel geändert werden. [75]
- TransformControls - Den Nutzer*innen ist es möglich ein Objekt, ähnlich wie bei 3D-Modellierungs-Programme, durch Anfasser zu positionieren, skalieren und rotieren. [76]
- 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.[77]

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 sich für die Bessere entschieden:

Vor- und Nachteile der FirstPersonControls

Die FirstPersonControls bieten eine einfache Einbindung in das Programm. Das bedeutet, die Logik des Bewegens muss nicht erst ausprogrammiert werden. 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 einige Konfigurationsmöglichkeiten mehr 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 hinzeigt. Dabei wird die Rotationsgeschwindigkeit von der Position der Maus bestimmt. Je weiter sich die Maus am Bildschirmrand befindet, desto schneller dreht sich die Kamera. Dies mag für viele Benutzer*innen vorerst ein ungewohntes Gefühl sein.

Implementierung der FirstPersonControls

Zu Beginn werden neue Controls 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 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

[74]

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 schnellstens durch ein Attribut ge- und entsperrt werden. Jedoch ist es nicht möglich, im entsperrten Zustand mit der Maus zu interagieren. Das bedeutet, Benutzer*innen können sie sich entweder im Ausstellungsraum fortbewegen oder mit einem Ausstellungsstück interagieren, jedoch nicht beides gleichzeitig. Außerdem kommt man viel zu einfach in den gesperrten Zustand, zum Beispiel durch das Drücken der Escape-Taste, das Wechseln vom Browser in ein anderes Programm oder beim Wechseln des Browser-Tabs.

Implementierung 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 schnellstens durch ein Attribut ge- und entsperrt werden. Jedoch ist es nicht möglich, im entsperrten Zustand mit der Maus zu interagieren. Das bedeutet,

Benutzer*innen können sie sich entweder im Ausstellungsraum fortbewegen oder mit einem Ausstellungsstück interagieren, jedoch nicht beides gleichzeitig. Außerdem kommt man viel zu einfach in den gesperrten Zustand, zum Beispiel durch das Drücken der Escape-Taste, das Wechseln vom Browser in ein anderes Programm oder beim Wechseln des Browser-Tabs.

Zu Beginn werden neue Controls gleich wie bei 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 anders wie bei den FirstPersonControls, die Logik dafür 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 außerdem 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.

[77]

5.8.3 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.

[78]

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

5.8.5 Bounding Box

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

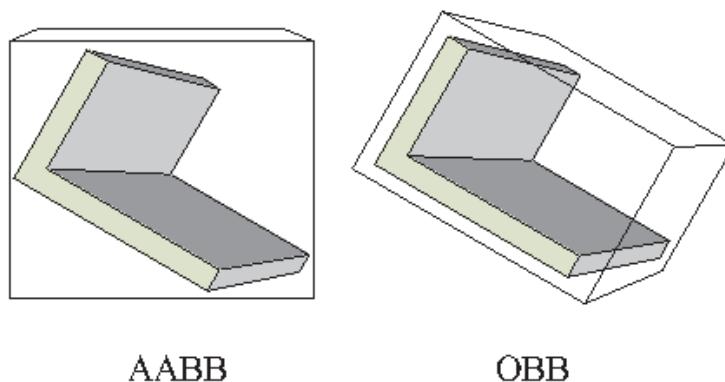


Abbildung 31: Der Unterschied zwischen AABB und OBB [79]

- 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 funktioniert im Endeffekt gleich, unterscheidet sich aber darin, dass sie sich an die Achsen des Objekts anpasst

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. 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.

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 5.3.2). 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 Raycast 5.8.6) Referenz korrekt initialisiert und angewandt wurde, muss bei einer Berührung mit der Wand nur noch richtig damit umgegangen werden. Dabei wird die Bewegung des Benutzers gestoppt. Um diese auch 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.

5.8.6 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 5.3.2) Referenz. Three.js besitzt seinen ganz eigenen 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. Zum einen kann er normal über einen Constructor initialisiert werden. Dabei wird angegeben, von welchen Koordinaten der Ray ausgeht und in welche Richtung er geschickt wird. Andererseits 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.

Hover-Effekt

Um zu erkennen, welches Ausstellungsstück momentan vom Mauszeiger gehovert wird, wird ein solcher 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 36: Aktuelle Koordinaten des Mauszeigers einem 2D-Vektor zuweisen

```

1      pointer = new THREE.Vector2()
2
3      window.addEventListener( 'pointermove', (event: PointerEvent) => {
4          // calculate pointer position in normalized device coordinates
5          // (-1 to +1) for both components
6          this.pointer.x = ( event.clientX / window.innerWidth ) * 2 - 1
7          this.pointer.y = - ( event.clientY / window.innerHeight ) * 2 + 1
8          this.hoverExhibit()
9      });
10

```

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

Listing 37: 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 38: 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 durch. 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 39: 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?.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! );

```

```

10      this.composer = new EffectComposer(this.renderer!)
11      this.composer.addPass( renderPass );
12
13      const outlinePass = new OutlinePass( new THREE.Vector2( window.innerWidth,
14          window.innerHeight ), this.scene, this.camera! );
15      this.composer.addPass( outlinePass );
16
17      this.addSelectedObjects(object!);
18
19
20      outlinePass.selectedObjects = this.selectedObjects;
21  }
22 }
23 }
```

Um den Effekt wieder zu entfernen, da ein anderes Ausstellungsstück gehovert wurde, wird ein Array verwendet, das das alte Objekt herausgibt und das neue hinzufügt.

Listing 40: Hover-effekt wieder entfernen

```

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

Postprocessing

Post-Processing ist eine Render-Methode, die zum Beispiel für Anti-Aliasing, Tiefenschärfe und andere 3D-Effekte benutzt wird. Dabei wird zuerst die Szene mit den gewünschten 3D-Objekten gerendert und im Speicher der Grafikkarte zwischengespeichert. 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. [80]

Detailansicht

Um beim Klicken auf ein Ausstellungsstück zur gewünschten Detailansicht zu kommen, wird ebenfalls ein Raycaster benutzt. Die Logik funktioniert fast gleich wie beim Hover-Effekt. Jedoch werden die Koordinaten des Mauszeigers nur bei jedem Mausklick aktualisiert. Gleich wie beim Hover-Effekt, wird das Array der Ausstellungsstücke mit einer For-Schleife durchiteriert. 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 41: identifizieren des geklickten Ausstellungsstückes

```

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

Geöffnet wird das Dialogfenster über eine eigene openDialog-Funktion. Dabei wird eine neue Instanz von einem Dialogfenster angelegt, welches Angular-Material verwendet. Dabei wird angegeben, welche Größe und Breite das Fenster haben soll. Auch wird festgelegt, wie schnell die Animation zum Öffnen und Schließen abläuft. Um die Daten des Ausstellungsstücks auch im Dialogfenster zur Verfügung zu stellen, werden diese beim initialisieren angegeben.

Listing 42: initialisieren des Dialogfensters

```

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

Wenn sich der*die Benutzer*in in der Detailansicht befindet, soll die Animation im Hintergrund gestoppt werden. Dies erfolgt durch die Methode cancelAnimationFrame //TODO kursiv, 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 //TODO Referenz) 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.

Listing 43: 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
8     dialogRef.afterClosed().subscribe(r => {
9       this.dialogOpen = false
10      this.animate()
11      this.clock.start()
12    })
13  }

```

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 //TODO Referenz) 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 44: Constructor-Injection in der Dialog Komponente

```

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

```

6 Zusammenfassung

In diesem Projekt machte das Diplomarbeitsteam große Erfahrungen und Fortschritte im Bereich Team- und Zeitmanagement, Programmierung, Prototyping und wissenschaftliches Arbeiten. In dem Kapitel Zusammenfassung 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

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. [81]

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 dabei 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`

`::ng-deep` durchbricht den virtuellen DOM (oder ShadowDOM) und erlaubt es in Kombination mit weiteren Styleselektoren von der Eltern-Komponente auf alle Kinder-Komponenten zuzugreifen. (siehe im Code Beispiel 45)

Listing 45: Parent.component.scss - Changing Styling in Child Componentes 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, aber zu unvorhersehbaren Fehlern und Problemen führen. Deswegen wird von Seitens Angular geraten es nicht zu verwenden und die Funktion als veraltet definiert. Viel besser 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, weil Anpassungen direkt in der Komponente gemacht werden können oder durch Input eine Interaktionsmöglichkeit hinzuzufügen. Bei Komponenten von dritten Seiten 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. [82] [83]

6.2 Zielerreichung

6.3 Erweiterungsvorschläge

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.

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. Design Patterns beschleunigen den Programmierungseffekt und machen ihn sicher, weil die Implementierungsvorgaben schon vorgegeben sind und diese auf einen hohen Grad der Sicherheit getestet wurden. [84]

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. Das Ziel ist es, dass Objekte (oder auch das Subject) bei allen Veränderungen diese an das andere Objekt (oder auch den Observer) weitergeben und das so schnell und leicht wie möglich. Die Idee ist, dass das Subject eine Liste von Observern führt, wenn sich etwas ändert, 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. [85]

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

Literaturverzeichnis

- [1] A. Ivanos, „AngularEvidence,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://stackdiary.com/front-end-frameworks/>
- [2] A. P. Sofiya Merenchy, „AngularEvidence2,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://clockwise.software/blog/best-angular-applications/>
- [3] Angular Team, „What is Angular,” letzter Zugriff am 20.02.2023. Online verfügbar: <https://angular.io/guide/what-is-angular>
- [4] R. Gravelle, „AngularArchitecturePattern,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://www.htmlgoodies.com/javascript/the-model-view-viewmodel-pattern-and-angular-development/>
- [5] M. Team, „MVCmdn,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
- [6] ——, „MVVM,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm%0A>
- [7] ——, „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>
- [8] Angular Team, „AngularProviders,” letzter Zugriff am 11.03.2023. Online verfügbar: [AngularProviders](#)
- [9] ——, „Angular ngModules,” letzter Zugriff am 20.02.2023. Online verfügbar: <https://angular.io/api/core/NgModule>
- [10] ——, „AngularNgModulesAPI,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://angular.io/guide/ngmodule-api>
- [11] D. K. Johannes Hoppe, Ferdinand Malcher, „AngularBuch, - Grundlagen, fortgeschrittene Themen und Best Practices,” 2020, letzter Zugriff am 10.03.2023.
- [12] ReactiveX Team, „Documentation Introduction for ReactiveX,” letzter Zugriff am 24.02.2023. Online verfügbar: <https://reactivex.io/intro.html>
- [13] 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/>
- [14] 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
- [15] S. J. . J. Team, „Angular Material Tutorial,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://www.javatpoint.com/angular-material>
- [16] E. Ighosewe, „What Is Angular Material,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://upstackhq.com/blog/software-development/what-is-angular-material>

- [17] Angular Team, „Angular Material - components,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://material.angular.io/components/categories>
- [18] G. Team, „Material Design: Introduction,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://m2.material.io/design/introduction>
- [19] Sass Team, „Sass Basics,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://sass-lang.com/guide>
- [20] ThreeJs Team, „Fundamentals of ThreeJs,” letzter Zugriff am 26.02.2023. Online verfügbar: <https://threejs.org/manual/#en/fundamentals>
- [21] Khronos Foundation, „Webgl - Getting Started - Wiki,” letzter Zugriff am 26.02.2023. Online verfügbar: https://www.khronos.org/webgl/wiki/Getting_Started
- [22] 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>
- [23] „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))
- [24] Angular Three Team, „Angular Three: Our first scene,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular-three.netlify.app/docs/first-scene>
- [25] 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>
- [26] Quarkus, „What is Quarkus?” letzter Zugriff am 08.03.2023. Online verfügbar: <https://quarkus.io/about/>
- [27] ——, „Creating Your First Application,” letzter Zugriff am 08.03.2023. Online verfügbar: <https://quarkus.io/guides/getting-started>
- [28] T. P. G. D. Group, „PostgreSQL: About,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://www.postgresql.org/about/>
- [29] JetBrains, „IntelliJ IDEA - the Leading Java and Kotlin IDE,” letzter Zugriff am 08.03.2023. Online verfügbar: <https://www.jetbrains.com/idea/>
- [30] M. Team, „Cinema4D,” letzter Zugriff am 22.02.2023. Online verfügbar: <https://www.maxon.net/de/cinema-4d>
- [31] G. Team, „C4DvsMaya,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://www.getapp.de/compare/2036947/2051550/cinema-4d/vs/maya>
- [32] w3schools Team, „What is npm?” letzter Zugriff am 6.03.2023. Online verfügbar: https://www.w3schools.com/whatis/whatis_npm.asp
- [33] npm Team, „About npm,” letzter Zugriff am 6.03.2023. Online verfügbar: <https://docs.npmjs.com/about-npm>
- [34] I. S.-M. F. Schwab, „Ideenfindung,” in *Systemplanung und Projektentwicklung*, Wien, 2013, ch. 2, letzter Zugriff am 1.03.2023.

- [35] ——, „Agile Vorgehensmodelle,” in *Systemplanung und Projektentwicklung*, 2013, ch. 6.4.
- [36] ——, „Prototyping,” in *Systemplanung und Projektentwicklung*, Wien, 2013, ch. 6.1.3.
- [37] 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>
- [38] P. D. F.-R. Esch, „Corporate Identity,” letzter Zugriff am 3.03.2023. Online verfügbar: <https://wirtschaftslexikon.gabler.de/definition/corporate-identity-31786>
- [39] ——, „Corporate Design,” letzter Zugriff am 3.03.2023. Online verfügbar: <https://wirtschaftslexikon.gabler.de/definition/corporate-design-30453>
- [40] S. Kuppelwieser, „LogoKriterien.” Online verfügbar: <https://www.sonja-kuppelwieser.com/logo-kriterien/>
- [41] A. . S. Team, „IsometricStyle,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://www.aleksundshantu.com/wiki/isometrischer-stil/>
- [42] P. D. F.-R. Esch, „Logo,” letzter Zugriff am 10.03.2023. Online verfügbar: <https://wirtschaftslexikon.gabler.de/definition/logo-39571>
- [43] 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/>
- [44] Angular Team, „AngularComponentOverview,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/component-overview>
- [45] ——, „BindingSyntax,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/binding-syntax>
- [46] ——, „AngularPropertyBinding,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/property-binding>
- [47] ——, „AngularEventBinding,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/event-binding>
- [48] ——, „AngularTwoWayBinding,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/two-way-binding>
- [49] ——, „AngularArchitectureService,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/architecture-services>
- [50] ——, „AngularHTTPClient,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/guide/http>
- [51] A. Team, „AdobeRendering,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://www.adobe.com/de/products/substance3d/discover/3d-rendering.html#:~:text=Beim 3D-Rendering wird aus, Modell sein 2D-Bild generiert>.
- [52] 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
- [53] M. D. Team, „RenderArten,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://magic-3d.de/was-sind-3d-renderings/>

- [54] P. Team, „MoireEffekt,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://www.printer-care.de/de/drucker-ratgeber/moire-effekt>
- [55] J. Herzog, „AntiAliasing,” letzter Zugriff am 11.03.2023. Online verfügbar: [https://de.wikipedia.org/wiki/Antialiasing_\(Computergrafik\)#/media/Datei:EasterEgg_anti-aliasing.png](https://de.wikipedia.org/wiki/Antialiasing_(Computergrafik)#/media/Datei:EasterEgg_anti-aliasing.png)
- [56] N. Team, „RayTracingRasterization,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://developer.nvidia.com/discover/ray-tracing>
- [57] T. Team, „ThreejsWebGLRenderer,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/#api/en/renderers/WebGLRenderer>
- [58] Angular Team, „AngularViewChild,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://angular.io/api/core/ViewChild>
- [59] T. Team, „ThreejsCreateAScene,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=scene#manual/en/introduction/Creating-a-scene>
- [60] ——, „StandardLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/index.html#api/en/renderers/WebGLRenderer.physicallyCorrectLights>
- [61] ——, „LightProbe,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/LightProbe>
- [62] ——, „AmbientLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/AmbientLight>
- [63] ——, „DirectionalLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/DirectionalLight>
- [64] ——, „HemisphereLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/HemisphereLight>
- [65] ——, „PointLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/PointLight>
- [66] ——, „ReactAreaLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/RectAreaLight>
- [67] ——, „SpotLight,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/SpotLight>
- [68] Wiki Contributors, „Persistenz,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://de.wiktionary.org/wiki/Persistenz>
- [69] 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
- [70] Hugo, „Origin of the term "wizard in computing,"“ letzter Zugriff am 18.03.2023. Online verfügbar: <https://english.stackexchange.com/questions/65728/origin-of-the-term-wizard-in-computing>
- [71] R. Budiu, „Wizards: Definition and Design Recommendations,” letzter Zugriff am 18.03.2023. Online verfügbar: <https://www.nngroup.com/articles/wizards/>

- [72] A. M. Team, „Angular Material Documentation - Stepper Overview,” letzter Zugriff am 18.03.2023. Online verfügbar: <https://material.angular.io/components/stepper/overview>
- [73] T. Team, „DragControls,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://threejs.org/docs/?q=Control#examples/en/controls/DragControls>
- [74] ——, „FirstPersonControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/index.html#examples/en/controls/FirstPersonControls>
- [75] ——, „OrbitControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/#examples/en/controls/OrbitControls>
- [76] ——, „ThreejsTransformControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/?q=controls#examples/en/controls/TransformControls>
- [77] ——, „PointerLockControls,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/?q=controls#examples/en/controls/PointerLockControls>
- [78] ——, „ThreeJsClock,” letzter Zugriff am 14.03.2023. Online verfügbar: <https://threejs.org/docs/#api/en/core/Clock%0A>
- [79] M. Angelini, „AABB and OBB Picture,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://devdept.zendesk.com/hc/en-us/articles/360011559320-How-Collision-Detection-works-v2020->
- [80] T. Team, „PostProcessing.” Online verfügbar: <https://threejs.org/docs/#manual/en/introduction/How-to-use-post-processing>
- [81] Angular Team, „Angular: View encapsulation,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular.io/guide/view-encapsulation>
- [82] ——, „Angular: Component styles - (deprecated) /deep/,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular.io/guide/component-styles#deprecated-deep--and-ng-deep>
- [83] Joshua Colvin, „Understanding Angular ::ng-deep,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://www.joshuacolvin.net/understanding-ng-deep/>
- [84] 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/>
- [85] ——, „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/>

Abbildungsverzeichnis

1	Die Model-View-Controller Pattern [5]	6
2	Die Model-View-ViewModel Pattern [6]	6
3	Angular Material: Komponente Überblick [17]	10
4	ThreeJs Logo	12
5	Die Struktur von ThreeJs [20]	13
6	WebGL Logo	13
7	Cinema4D vs Maya [31]	19
8	Modellierung des Raumes in Cinema4D	20
9	Landingpage Design Mockups Gegenüberstellung	27
10	OberflächenDesign: Prototypen in Figma	29
11	Design Konzept 1 Page 1	30
12	Design Konzept 1 Page 2	31
13	Design Konzept 2	32
14	Logo der 3D-Portfolio-Gallery	33
15	Angular: Automatisch generierte Start-Webseite	35
16	Component-Hierarchy [11]	38
17	Der Moire-Effekt visualisiert [54]	45
18	Rendering mit und ohne Anti-Aliasing [55]	45
19	Vorgang des Ray-Tracing [56]	47
20	Landing Page	50
21	Projekt: Login Page	51
22	Navbar: authentifiziert vs noch nicht authentifiziert	55
23	Profilepage	56
24	Inhaltserstellungstool	57
25	Inhaltserstellung Datenklassen	58
26	Profileseite: 3D-Gallery-Portfolio Hinzufügen Knopf	59
27	Inhaltserstellungstool	59
28	Angular Material: horizontaler Stepper [72]	61
29	Inhaltserstellung Datenklassen	62
30	Metadaten Unterseite	63
31	Der Unterschied zwischen AABB und OBB [79]	69

Tabellenverzeichnis

Quellcodeverzeichnis

1	SCSS - Code Beispiel	12
2	CSS - Code Beispiel	12
3	Angular Three - Komponentenbasiertes 3D Scenen in HTML	14
4	Angular Three - App Cube	15
5	Terminalm - Angular aufsetzen, Installation der CLI, Configuration eines neuen Projektes, Starten des Projektes	34
6	Terminal - Angular Material Installation	35
7	Terminal - Bootstrap Installation	35
8	angular.json - Bootstrap Angular Verknüpfung	35
9	Terminal - Component erstellen	36
10	Beispiel für Interpolation in der 3D-Gallery	38
11	Beispiel für Property-Bindings in der 3D-Gallery	39
12	Beispiel für Event-Bindings in der 3D-Gallery	39
13	Beispiel für Two-Way-Bindings [48]	39
14	Routing in der 3D-Gallery	40
15	Routing über einen routerLink	40
16	Routingparamter in der 3D-Gallery	40
17	Snapshot der URL abfragen	41
18	Die URL subscriben	41
19	Eine Klasse Injectable machen	41
20	Constructor Injection	42
21	HttpClient Abfragen	42
22	Das Datenmodell eines Ausstellungsstückes	43
23	Canvas-Element in HTML	47
24	Canvas als View-Child initialisieren	47
25	WebGLRenderer anlegen	47
26	Animations-Schleife	48
27	Lichtsetzung in der 3D-Ausstellung	49
28	auth.service.ts - JWT und Localstorage	53
29	auth.interceptor.ts - add JWT to Request Header	54
30	OrbitControls initialisieren	65
31	FirstPersonControls initialisieren	65
32	PointerLockControls initialisieren	67
33	Logik der PointerLockControls	67
34	Controls entsperren	67
35	Controls updaten	68
36	Aktuelle Koordinaten des Mauszeigers einem 2D-Vektor zuweisen	71
37	Neuen Raycaster anlegen	71
38	Intersected Objects auslesen	71
39	Intersects als Ausstellungsstück identifizieren	71
40	Hover-effekt wieder entfernen	72
41	identifizieren des geklickten Ausstellungsstückes	73
42	initialisieren des Dialogfensters	73

43	Öffnen und Schließen des Dialogfensters	73
44	Constructor-Injection in der Dialog Komponente	74
45	Parent.component.scss - Changing Styling in Child Componentes by using :ngdeep	75

Anhang