

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 Ema Halilovic, Lorenz Litzlbauer and Fabian Maar as part of their 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 offers a simple configuration process by selecting from a variety of exhibition rooms. Then the exhibition pieces can be placed there manually or automatically and supplied with additional information. Those interested can move freely through the three-dimensional web exhibition and view the exhibits with different types of interaction.

The following applications were used for the implementation: Angular is used in the front end for the SPA (single-page application) logic and ThreeJs for the 3D presentation. The server uses technologies such as Quarkus, Maven, JPA, Panach and Hibernate ORM. To ensure the safety of users, we use JWT (JsonWebToken)



Zusammenfassung

3D Portfolio Gallery ist eine Webapplikation, die von Ema Halilovic, 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 bietet dafür einen einfachen Konfigurationsprozess, indem aus verschiedenen Ausstellungsräumen ausgewählt wird. Dann können die Ausstellungsstücke dort manuell oder automatisch platziert und mit Zusatzinformation versehen werden.

Interessierte können sich in der dreidimensionalen Webausstellung frei bewegen und die Ausstellungsstücke mit verschiedenen Interaktionsarten anschauen. Für die Umsetzung wurden folgende Applikationen verwendet: Im Frontend wird Angular für die SPA (Single-Page-Applikation)-Logik und für die 3D-Darstellung, ThreeJs benutzt. Für den Server werden Technologien wie Quarkus, Maven, JPA, Panach und Hibernate ORM verwendet. Um die Sicherheit der User zu gewährleisten, verwenden wir JWT (JsonWebToken)

Inhaltsverzeichnis

1 Einleitung	1
1.1 Ausgangslage	1
1.2 Untersuchung Anliegen der individuellen Themenstellungen	1
1.3 Zielsetzung	2
1.4 Aufgabendifferenzierung	2
2 Technologien	3
2.1 Grundkriterien für das Backend [E]	3
2.2 PostgreSQL [E]	3
2.3 Quarkus [E]	3
2.4 IntelliJ IDEA [E]	4
2.5 Oracle Server [E]	5
2.6 Angular [L]	5
2.7 Webstorm [M]	10
2.8 ThreeJs [L]	10
2.9 Cinema 4D [M]	13
2.10 Figma [L]	15
2.11 Notion [E]	16
2.12 PlantUML [E]	16
3 Umsetzung	18
3.1 Plannung	19
3.2 Design	25
3.3 Initialisierung des Server	33
3.4 Initialisierung der Datenbank	33
3.5 Initialisierung der Landingpage [L]	33
3.6 Components [M]	36
3.7 Routing [M]	40

3.8 Angular Services [M]	41
3.9 Web Gallery Prototype	43
3.10 Fertige Landingpage	50
3.11 Userverwaltung [L]	50
3.12 Interaktions-Seite [M]	56
4 Zusammenfassung	59
4.1 Probleme und Lösungsstrategie	59
4.2 Zielerreichung	60
4.3 Erweiterungsvorschläge	60
5 Glossar	61
Literaturverzeichnis	VI
Abbildungsverzeichnis	XI
Tabellenverzeichnis	XII
Quellcodeverzeichnis	XIII
Anhang	XIV
	IV

1 Einleitung

1.1 Ausgangslage

Derzeit gibt es keine Möglichkeit für Designer, eine ansprechende Ausstellung im Internet zu erstellen, in welcher verschiedene Dateiformate wie Audio-, Bild-, Video-, 3D-Dateien dargestellt werden können.

1.2 Untersuchung Anliegen der individuellen Themenstellungen

Untersuchen, welche Webseiten-Technologien sich für die Darstellung von 3D - Modellen eignen. Mit dieser Technologie auseinandersetzen und eine userinteractive Webseite gestalten, welche benutzerfreundlich umgesetzt werden soll.

Dieses Anliegen lässt sich in folgende Teilbereiche unterteilen:

Untersuchen...

- eines geeigneten Suchalgorithmus um andere Ausstellungen zu finden
- einer geeigneten Art, um mit der 3D-Web-Ausstellung zu interagieren bzw. sich zu orientieren.
- einer geeigneten Art der Content Creation einer 3D-Web-Ausstellung.
- der geeigneten Software für die Continuous Integration und Continuous Delivery.
- wie die Daten gespeichert und geladen werden können.
- wie ein benutzerfreundliches UI umgesetzt werden kann, damit User weitere Informationen zu Ausstellungsstücken bekommen.

1.3 Zielsetzung

Eine Plattform bieten, auf der Designer eine 3D-Web-Ausstellung erstellen und Nutzer diese anschauen können. Diese ist interaktiv und verfügt über mehrere Konfigurationsmöglichkeiten, wie zum Beispiel das Aussehen des Ausstellungsraums.

1.4 Aufgabendifferenzierung

1.4.1 Ema

1.4.2 Lorenz

1.4.3 Fabian

2 Technologien

2.1 Grundkriterien für das Backend [E]

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

2.2 PostgreSQL [E]

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 [2]

2.3 Quarkus [E]

Unter Quarkus versteht man ein open source Framework, womit cloud-native Projekte in Java entwickeln werden können. Dieses 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 es eine Vielzahl von Extensions, welche durch Command-line-Befehle oder händisch zu Projekten hinzugefügt werden können.

[3, 4]

2.3.1 Maven [E]

Maven ist ein Tool, um den Kompilierungsprozess eines Projekts zu vereinfachen. Wenn man Projekte auf unterschiedlichen Geräten ausführen möchte kommt es dabei

oftmals zu Problemen. Meist sind lokale Konfigurationen der Auslöser dafür. Maven ermöglicht ebenso ein einheitliches System für Projektkonfigurationen, sodass diese nicht mehr manuell bei Gerätewechsel umgestellt werden müssen. Dabei wird die *pom.xml* Datei verwendet, welche eine der Hauptkomponenten ist für Maven-Projekte. [5]

In Quarkus-Projekten werden darin relevante Informationen gespeichert, wie zum Beispiel die verwendete Java Version oder alle verwendeten Extensions.

2.3.2 JDBC Driver - PostgreSQL [E]

JDBC Driver - PostgreSQL ist eine Extension für Quarkus Projekte, die Datenbankverbindungen zu PostgreSQL Datenbanken ermöglicht. Java JDBC gibt es als API für Java Anwendungen, jedoch unterscheidet diese sich von dem in Quarkus benutzten JDBC Driver.

// TODO alles umschreiben Das JDBC steht für *Java Database Connectivity* und .

Um eine Datenbankverbindung aufzubauen muss man in die Datei *application.properties* einige zusätzlichen Konfigurationen einfügen, wie den Pfad der Datenbank, die Art der Datenbank, den Nutzernamen und das Passwort 1:

Listing 1: Beispielkonfigurationen

```
1 quarkus.datasource.db-kind=postgresql
2 quarkus.datasource.username=meinUser
3 quarkus.datasource.password=meinPassword
4 quarkus.datasource.jdbc.url=jdbc:postgresql://localhost:5432/meineDatabase
```

2.3.3 Hibernate ORM [E]

// TODO

2.3.4 REST-Easy [E]

REST-Easy ist eine Erweiterung für Quarkus, die es ermöglicht, im Quarkus Projekt mit Jakarta RESTful Web Services zu arbeiten.

2.4 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 Entwick-

lungsumgebung 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. [6]

2.5 Oracle Server [E]

// TODO

2.6 Angular [L]

Um eine nachvollziehbare Auswahl zu treffen, muss verstanden werden, wie Angular funktioniert.

2.6.1 Allgemeines [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 [7] und wird von Milliardenkonzernen wie zum Beispiel Google oder PayPal verwendet. [8]

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. [9]

2.6.2 Dependencies [L]

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

RxJS

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 4.3 auf Seite 61). 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.

[10]

Webpack

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.

2.6.3 Vorteile/Auswahlkriterien [M]

Folgende Aspekte sprechen für die Nutzung Angular in diesem Projekt:

- komponentenbasiertes Programmieren; im Abschnitt 5 //TODO wird nochmals deutlich, wie dieser Aspekt genutzt wird.
- einfache und übersichtliche Einbindung von Libraries durch ngModules; wird behandelt in Kapitel NgModules 2.6.5
- 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 2.6.4
- Dependency Injection für die Verbindung von Front-End zum Back-End //TODO siehe Kapitel HTTP Services

2.6.4 Model-View-Controller und Model-View-ViewModel

Architecture [M]

Allgemein

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. [11]

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

[12] [13] [14]

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 entsprechend geändert. In Angular sind diese Modelle mit den Interfaces //TODO REFERENZIEREN 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.

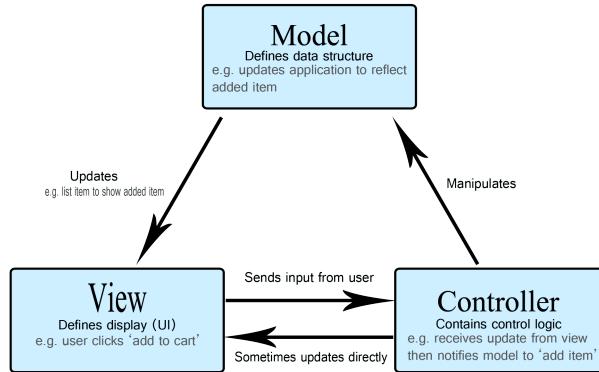


Abbildung 1: Die Model-View-Controller Pattern [12]

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 anzugeben. Die TypeScript-Files in Angular sind vergleichbar mit diesen Controllern. [14]

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.

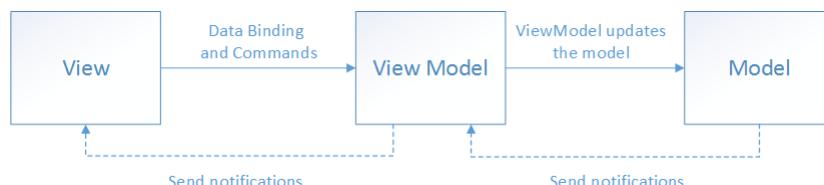


Abbildung 2: Die Model-View-ViewModel Pattern [13]

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 liefert es mit Daten. Dieser Vorgang beschreibt das Konzept des Data-Bindings //TODO REFERENZ 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. [13]

2.6.5 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 Inection //TODO REFERENZ abgerufen werden [15]
- 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. [16] [17] [18]

2.7 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.

2.8 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, 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.

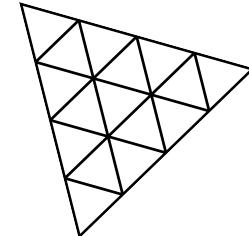


Abbildung 3: ThreeJs Logo

In ThreeJs werden Geometrie, Objekte und Materialien verbunden, um ein 3D-Objekt zu erstellen. Dabei kann die Struktur einer Szene der Abbildung 4 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, Materialien der 3D-Meshes werden außerhalb der Szene gespeichert und die Meshes referenzieren darauf.

[19, ThreeJs fundamentals]

2.8.1 Dependency

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

WebGL

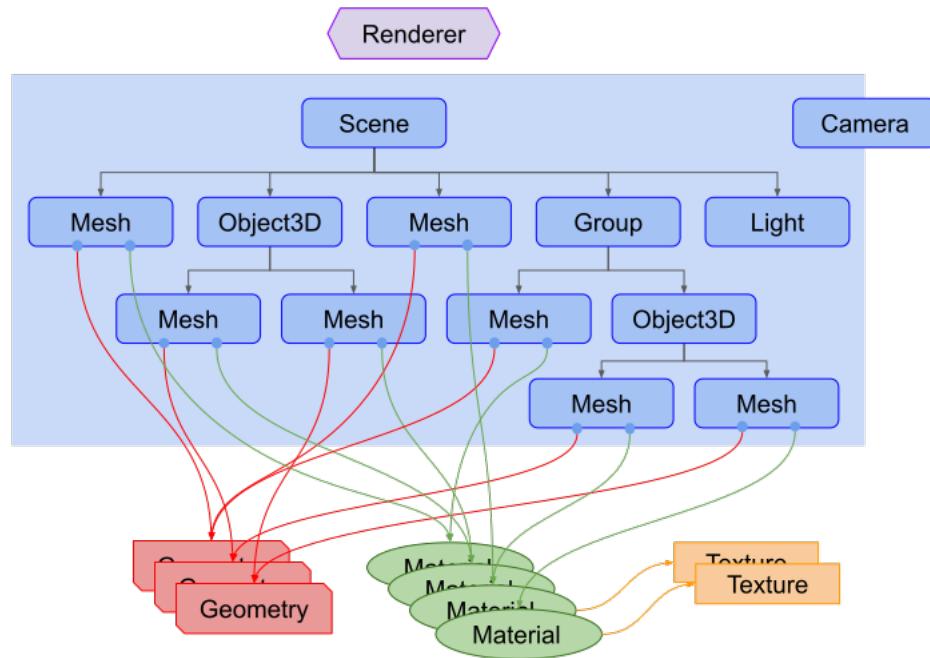


Abbildung 4: Die Struktur von ThreeJs [19]

WebGL ist eine lizenzenfreie Low-Level-API, die dafür benutzt wird, 3D-Grafiken im Web darzustellen. Sie basiert auf OpenGL ES 2.0 und benutzt auch dieselbe shading language GLSL wie OpenGL. Eine Low-Level-API 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. [20, WebGL Getting Started] [21]



Abbildung 5: ThreeJs Lo-
go

2.8.2 Auswahlkriterien

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)
 - Die Features des Projektes müssen damit umsetzbar sein
 1. Das Laden von 3D-Modellen aus 3D-Dateien und aus dem Internet
 2. Video-Texturen support

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. [22, 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.

WebGL Im Vorherigen Kapitel wurde sich bereits mit WebGL im Kontext mit ThreeJs befasst. In ThreeJs wird WebGL für die 3D-Darstellung gewählt. Doch gibt es auch WebGL als eigenständige Library. 2.8.1 Um nur eine einfache 3D-Szene in WebGL darzustellen, muss sehr viel Code geschrieben werden. Denn WebGL übernimmt keine Low-Level-Funktion. Es müssen Matrixrechnungen für die Transformationen von 3D-Objekt Manuel und Vertex buffers, die die Daten der Vertex Positionen, Normaldaten, Farben und Texturen, selbst programmiert werden und das in einer eigenen Programmiersprache. Deshalb war WebGL keine Option für das Projekt.

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 2: 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 3: 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 2 3 [23]

Ein Vorteil von Angular Three ist, dass durch nur wenige Zeilen Code 2 eine 3D-Szene mit Lichtern und Orientierungsfunktionen erstellt werden kann und die Business-Logik, App-Cube 3 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 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.

2.9 Cinema 4D [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 bie-

tet 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. [24]

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

Alle Funktionen	11
✓ 2D-Zeichnung	✓ 2D-Zeichnung
✓ 3D-Modellierung	✓ 3D-Modellierung
✗ Aktivitäts-Dashboard	
✗ Animation	
✓ Animationen und Übergänge	
✗ Bild-Nachverfolgung	
✗ Bildbearbeitung	
✗ Daten-Import / -Export	
✗ Digital Asset Management	
✓ Drag-and-Drop	
✓ Inhalt-Bibliothek	
✓ Medienimport	
✓ Rendering	
✓ Suche	
✓ Texteinblendung	
✓ Video-Inhalte	
✓ Videobearbeitung	
✗ Vorlagen	

Abbildung 6: Cinema4D vs Maya [25]

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.

2.9.1 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.

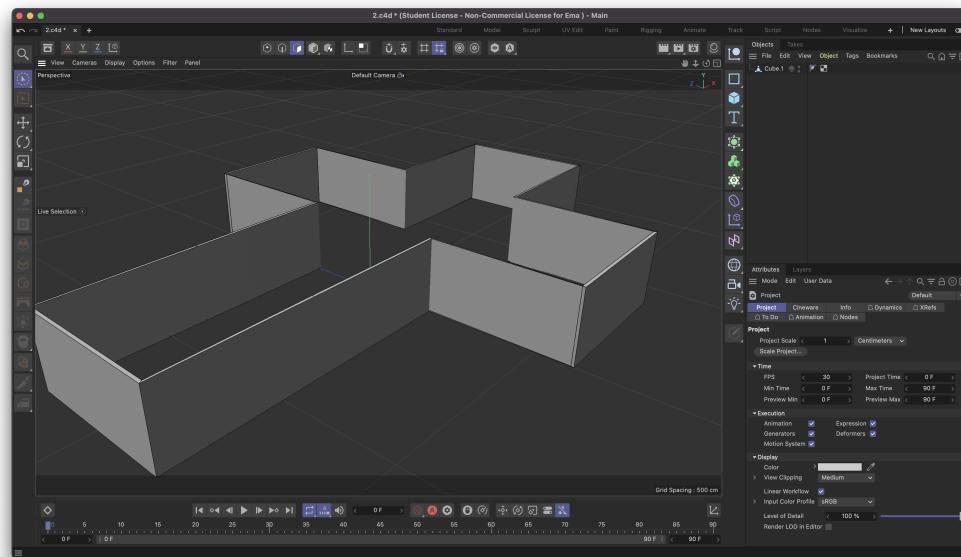


Abbildung 7: Modellierung des Raumes in Cinema4D

2.10 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.

2.10.1 Auswahl

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

2.11 Notion [E]

Notion ist eine Software, mit der so genannte *Workspaces* erstellt werden können. Diese ist verfügbar im Browser, als App in Windows-, MacOS- oder auf iOS- und Android-Geräten. Es besteht die Möglichkeit eine Seite zu erstellen und mit einem einzigen Befehl Unterseiten anzulegen, welche direkt verlinkt werden. Diese Seiten lassen sich mit beliebigem Inhalt füllen und durch einfache /-Befehle ist es möglich z. B. ein Kanban Board erstellen. Es ist sehr einsteigerfreundlich, da alle Funktionalitäten gut beschrieben sind und es benutzerfreundlich gestaltet ist. Workspaces lassen sich mit anderen Nutzern teilen, wodurch jeder Nutzer auf eine *single source of truth* Version des Workspaces zugreifen kann. Für die Bearbeitung in echt-Zeit wird Internet benötigt, falls dieses jedoch ausfällt, werden die Änderungen gespeichert und synchronisiert, sobald wieder eine Netzwerkverbindung aufgebaut wurde. [26]

Das Feature von geteilten Workspaces war für diese Arbeit nützlich, um Notizen nach Meetings und gemeinsame Termine festzuhalten, genauso um wichtige Links zu speichern und diese schnell wieder abrufen zu können.

2.12 PlantUML [E]

The screenshot shows a Notion workspace titled "3D Portfolio Gallery". The interface includes a header with a globe icon, a title, and a "Edit" button. Below the header, there's a section for "Helpful links and snippets for final thesis". A sidebar on the left lists "GOALS" and "Verbindung zwischen FE und BE". The main area is divided into sections:

- Specific:**
 - Backend
 - Three.js
 - Frontend
 - </> Code Examples
- General:**
 - Users
- GitHub Integration:**
 - A card for "GitHub - L0R3N7/3D-Portfolio..." showing a message about signing in.
 - A card for "Toggl Track" with a link to <https://track.toggl.com/timer>.
 - A card for "Login" with a link to <https://diplomarbeiten.berufsb...>.
 - A card for "3DProject" created with Figma, with a link to <https://www.figma.com/file/5oHgO...>.
 - A card for "angular-three/libs/documenta..." with a link to <https://github.com/nar...>.

Abbildung 8: Notion Workspace der Diplomarbeit

3 Umsetzung

Siehe tolle Daten in Tab. 1.

Siehe und staune in Abb. 9. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetur at, consectetur sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

	Regular Customers	Random Customers
Age	20-40	>60
Education	university	high school

Tabelle 1: Ein paar tabellarische Daten

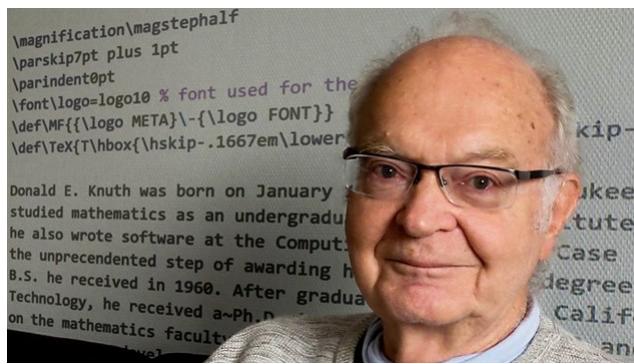


Abbildung 9: Don Knuth – CS Allfather

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetur a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetur. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus. Dann betrachte den Code in Listing 4.

Listing 4: Some code

```

1 # Program to find the sum of all numbers stored in a list (the not-Pythonic-way)
2
3 # List of numbers
4 numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
5
6 # variable to store the sum
7 sum = 0
8
9 # iterate over the list
10 for val in numbers:
11     sum = sum+val
12
13 print("The sum is", sum)

```

3.1 Planning

3.1.1 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

[27]

3.1.2 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. [28]

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

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:
 - Der*Die User*In 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 einen 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
 - Dabei können, aber nur einzelne Aspekte der Templates manuell verändert werden (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.

3.1.3 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. [29]

3.1.4 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 12 Angular Three) wurde überprüft, ob die Technologien auf die das Projekt aufbaut auch die Anforderungen (siehe Seite 21 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.

3.2 Design

3.2.1 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 [30].

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.

3.2.2 Corporate Design [M]

Das Corporate Design unterstützt die Corporate Identity und deren Ziele. Die Corporate Identity bezeichnet das äußere Erscheinungsbild eines Unternehmens . Hierbei sollen

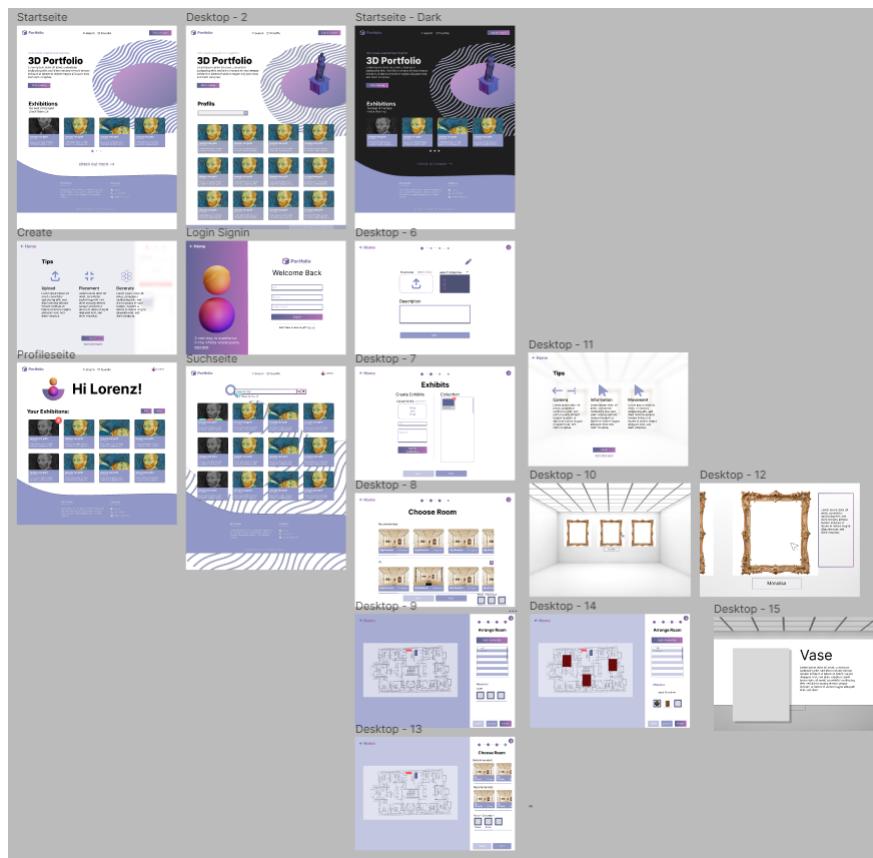


Abbildung 10: OberflächenDesign: Prototypen in Figma

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

[31]

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. [32]

//TODO Text über Bild + Bilder richtig anordnen Der erste Schritt des Designs war es, zwei unterschiedliche Konzepte der ersten Webseiten-Elemente zu erstellen. Anschließend

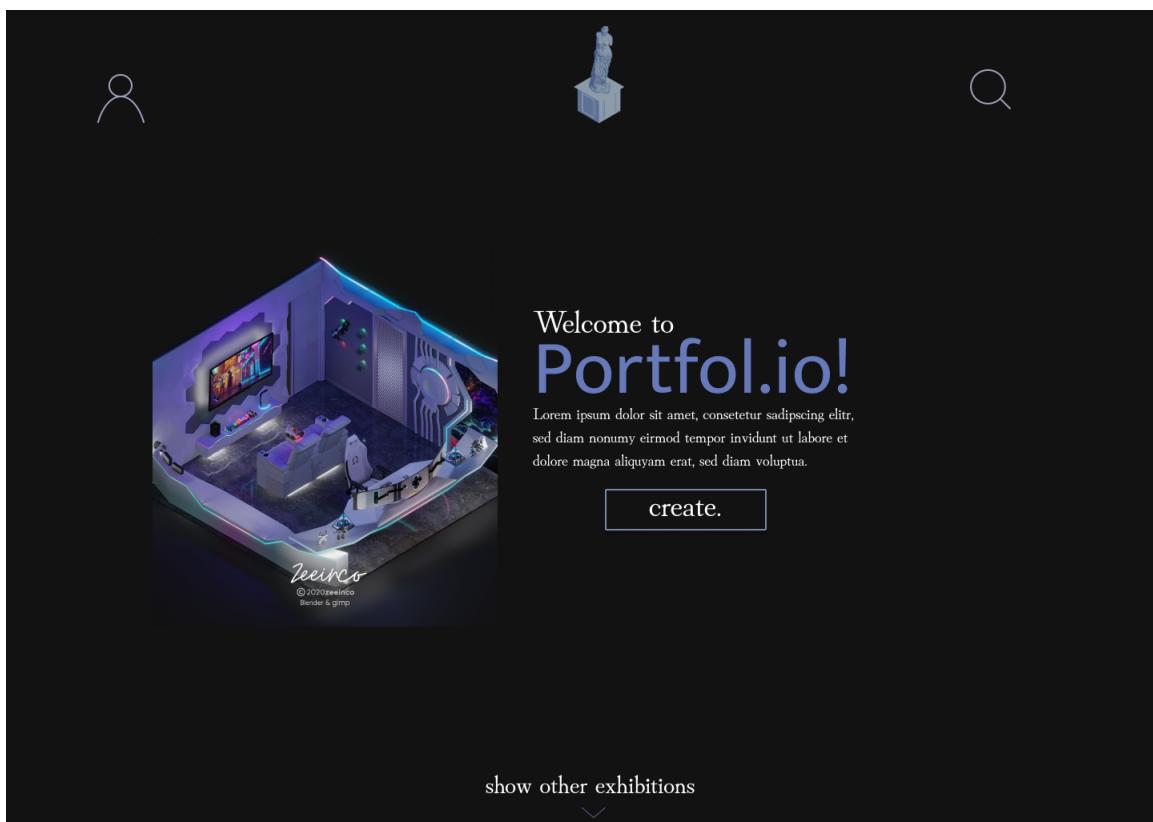


Abbildung 11: Design Konzept 1 Page 1

wurden diese verglichen und sich für eines entschieden. Dieses erste Grundkonzept wurde das Grundgerüst für den späteren Verlauf der Designarbeit.

3.2.3 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 (siehe Seite 21 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.

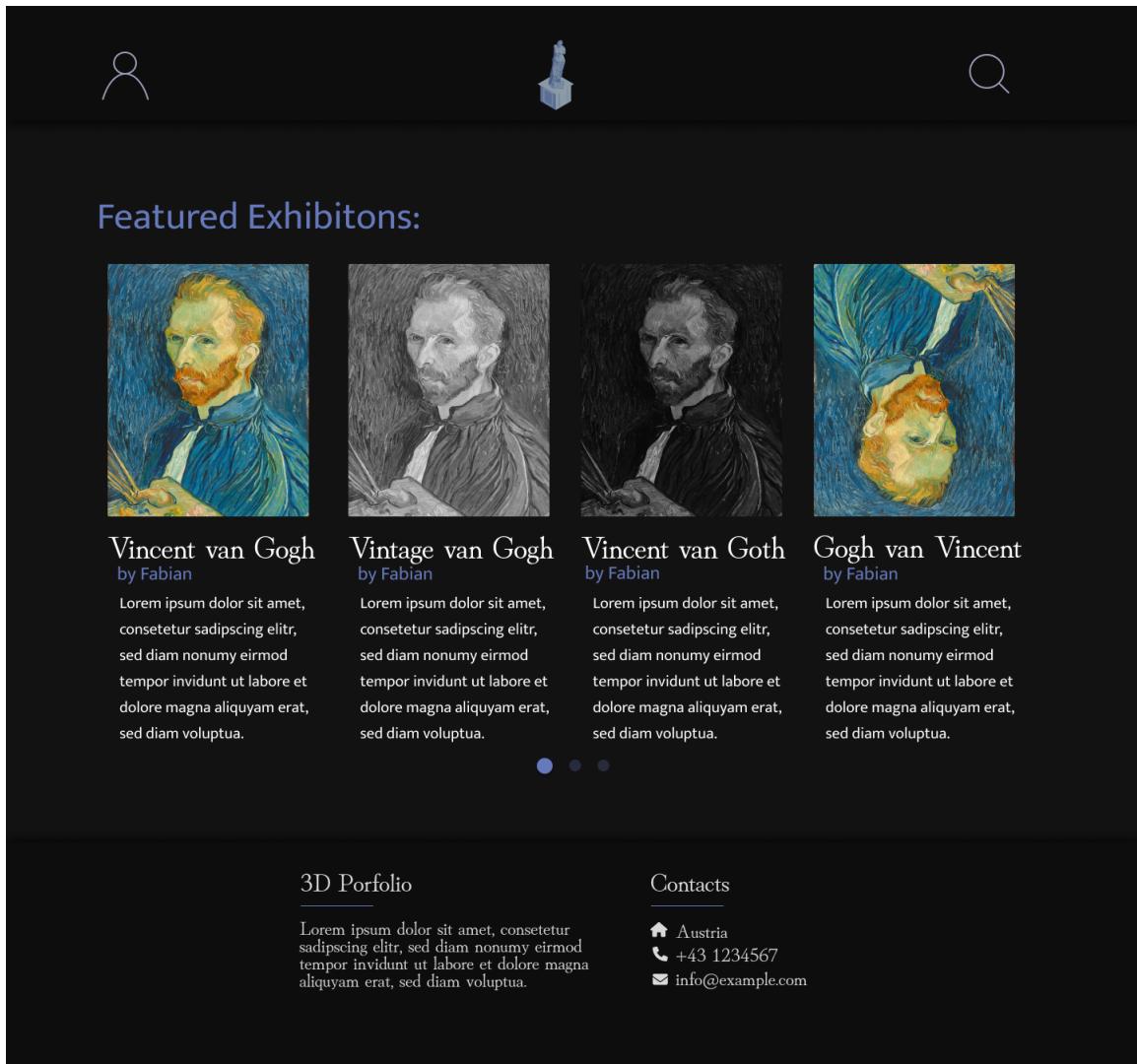


Abbildung 12: Design Konzept 1 Page 2



Abbildung 13: Design Konzept 2

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.

[33] [34]

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.

Angular Material hat folgende Features:

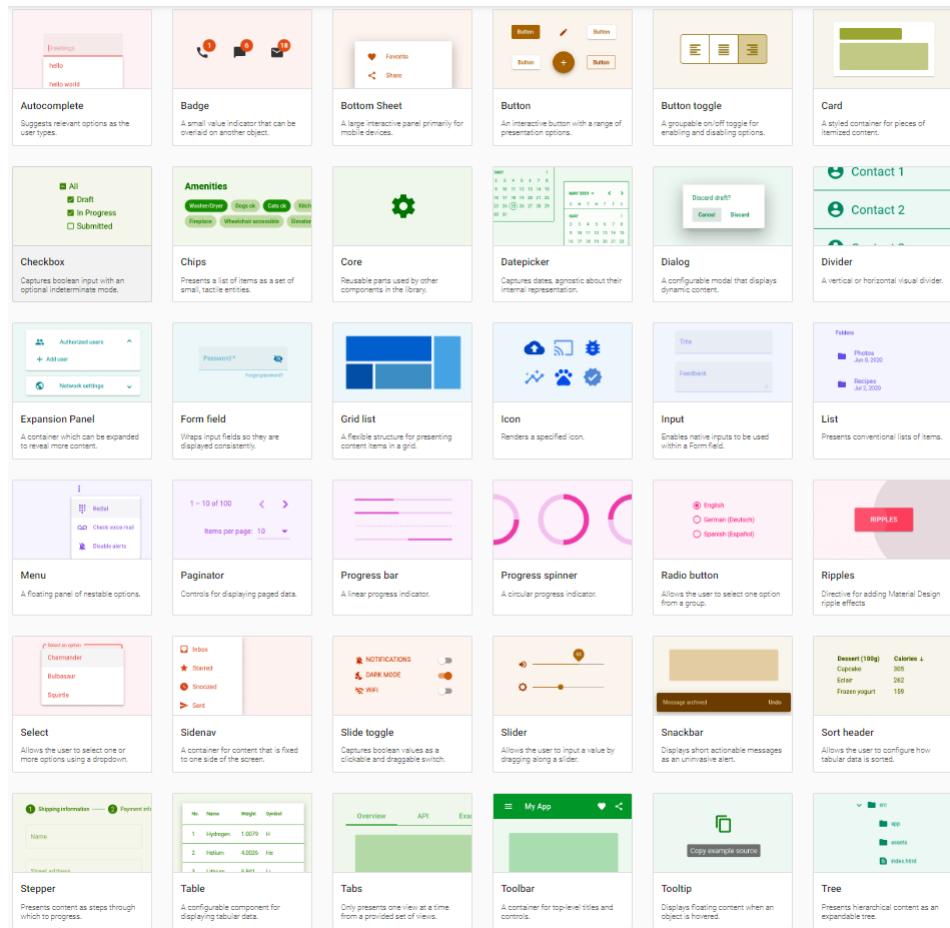


Abbildung 14: Angular Material: Komponente Überblick [37]

- 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 14 zeigt einen Teil der Komponenten, die Angular Material anbietet. Zu allen diesen Komponenten gibt es eine ausführliche Dokumentation mit Beispielen. [35] [36]

Material Design Spezifikation Material Design ist ein Designsystem, 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. Komponen-

ten sind interaktive Bauklötze für ein das User Interface. Komponenten haben alle eine Stati-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. Material Design befasst sich mit den Design-Fundamenten:

- Umgebung
- Layout
- Navigation
- Farbe
- Typography
- Ton
- Ikonografie
- Form
- Bewegung
- Interaktion
- Kommunikation
- maschinelles Lernen

[38]

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.

Bootstrap hat folgenden Features:

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

[34]

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.

Durch die Verschachtelung kann an redundanten Styles-Selektoren gespart werden, was die Style-Definition besser lesbarer macht. Im den Beispielen (siehe SCSS-Code-Beispiel 5 und natives CSS-Code-Beispiel 6) wird die gleiche Style-Definition von SCSS und CSS ausgedrückt, doch die Schreibweise unterscheidet sich.

[39, Sass Guide]

Listing 5: 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 6: 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 Beispiele5 6 [39, Sass Guide]

3.3 Initialisierung des Server

3.4 Initialisierung der Datenbank

3.5 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.

3.5.1 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.

npm - Node Package Manager Der Node Package Manager ist ein Softwareverwaltungstool zum Downloaden, Aktualisieren, Veröffentlichen und Verwalten von OpenSource-Programmen in der NodeJs-Umgebung. [40] [41]

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 7: 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  [
   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).

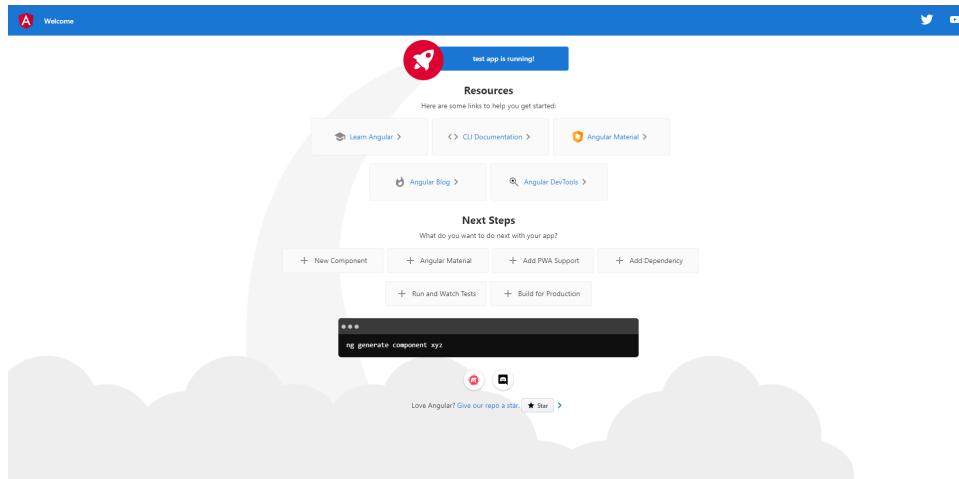


Abbildung 15: Angular: Automatisch generierte Start-Webseite

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. Bei NPM-Modulen kann zwischen lokalen und globalen Installationen unterschieden werden. 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 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. [42]

Installation der UI-Framework

Nach der Installation von Angular wurden die UI-Frameworks installiert. Angular Material konnte mithilfe des Befehls 8 mittels der Angular CLI in das Angularprojekt eingebunden werden.

Listing 8: Terminal - Angular Material Installation

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

Bootstrap konnte mithilfe des Befehls 9 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 10 veranschaulicht.

Listing 9: Terminal - Bootstrap Installation

```
1      npm i bootstrap
```

Listing 10: angular.json - Bootstrap Angular Verknüpfung

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

Installation von Three Js

Nach der Installation der UI-Libraries wurde ThreeJs installiert.

Mit dem ersten Befehl wird AngularJs eine JavaScript Liberay 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
```

3.6 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 11: Terminal - Component erstellen

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

3.6.1 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

3.6.2 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

[43]

3.6.3 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 3.5.1

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.

3.6.4 Hierarchie

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

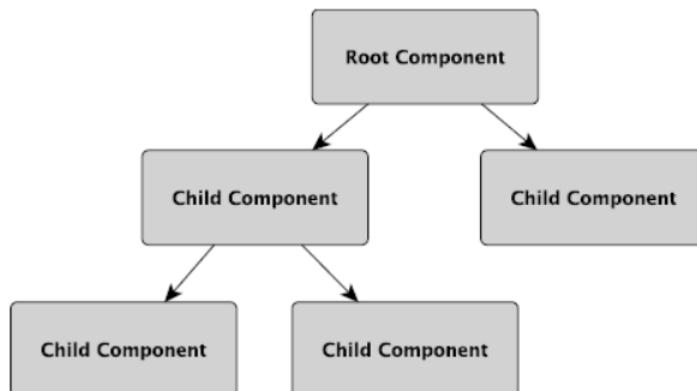


Abbildung 16: Component-Hierarchy [18]

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.

3.6.5 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 [18] [44]:

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 12: 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. [45]

Listing 13: 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. [46]

Listing 14: 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. [47]

Listing 15: Beispiel für Two-Way-Bindings [47]

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

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

3.7 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 16: 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.

3.7.1 Routingparameter

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

Listing 17: 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 18: 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 19: Die URL subscriben

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

[18]

3.8 Angular Services [M]

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

3.8.1 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. [18] [48]

3.8.2 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 [18]:

Listing 20: Eine Klasse Injectable machen

```

1  @Injectable({
2    providedIn: 'root',
3  })
4  export class GalleryService{
5    ..
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. [18]

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

Listing 21: Constructor Injection

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

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

3.8.3 HTTP Module

Um mit dem Backend über das HTTP-Protokoll kommunizieren zu können, wird eine HTTP-API //TODO REFERENZ API 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 [18] [49]:

Listing 22: 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.

3.8.4 Interface [M]

Interfaces werden verwendet, um typischer 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 [18]:

Listing 23: 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  }
```

3.9 Web Gallery Prototype

3.9.1 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 für die Entwicklung verwendet. //TODO Referenzieren

3.9.2 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. [50] [51]

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 //TODO Referenz
- die Kameraposition und Renderbereich (Clipping)
- der Anti-aliasing Prozess
- die Art des Render-Algorithmus
- weitere atmosphärische Effekte

[51]

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. [51]

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.

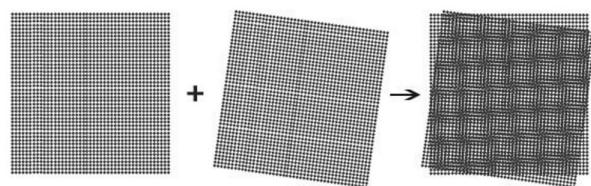


Abbildung 17: Der Moire-Effekt visualisiert [53]

- 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.

[52]

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. [51]

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 3D-Modell aus, in dem Zacken an Polygon-Kanten entstehen, ein Moiré-Effekt auftritt (siehe Abbildung 17) Verweis oder generell unerwünschte Muster auftreten.[51]

//TODO Bild wird oben angezeigt

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,

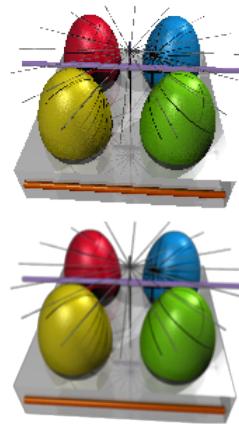


Abbildung 18: Rendering mit und ohne Anti-Aliasing [54]

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. [55]

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. [51]

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. [51]

Ray-Casting

Beim Ray-Casting 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 //TODO-Referenz näher erklärt und angewandt. [51]

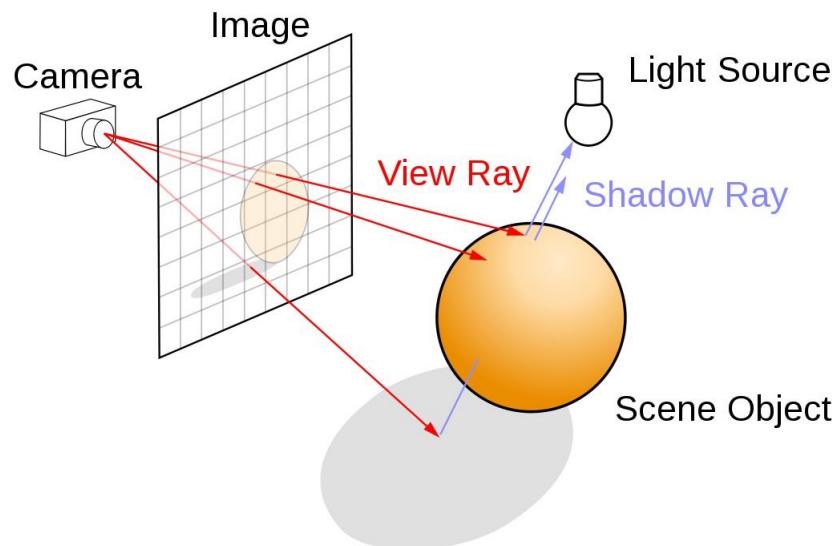


Abbildung 19: Vorgang des Ray-Tracing [55]

Ray-Tracing

Der Ray-Tracing Prozess funktioniert ähnlich wie der Prozess des Ray-Castings. 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

[55]

//TODO Bild wird oben angezeigt

Rendering in Three.js

Für das Rendern der 3D-Gallery wird der in Three.js inkludierte Renderer von WebGL //TODO Referenz verwendet. Um das gerenderte 3D-Objekt in das DOM //TODO Referenz zu integrieren, wird das HTML-Element <canvas> verwendet. [56]

Listing 24: 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. [57]

Listing 25: 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. [56]

Listing 26: 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. [58]

Listing 27: Animations-Schleife

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

3.9.3 Resizing [L]

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

3.9.4 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. [59]
- 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. [60]
- Das AmbientLight belichtet die gesamte Szene gleich, kann dabei jedoch keine Schatten werfen. [61]
- Das DirectionalLight wirft Licht parallel in eine bestimmte Richtung. Da es unendlich weit weg erscheint, wird es oft als Tages- oder Sonnenlicht verwendet. [62]
- Das HemisphereLight wird über der Szene positioniert und besitzt eine Himmel- und Bodenfarbe mit Verlauf. Diese Lichtquelle kann keine Schatten werfen [63]
- Das PointLight wirft Licht von einem Punkt in alle Richtungen. Es soll eine Glühbirne simulieren. Hierbei kann ebenfalls die Reichweite des Lichtes und ein Dimmeffekt bestimmt werden. [64]
- Das RectAreaLight strahlt Licht in Form eines Rechtecks aus. Damit können zum Beispiel Fenster simuliert werden [65]
- Das SpotLight wirft Licht in Form eines Kegel [66]

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

Listing 28: Lichtsetzung in der 3D-Ausstellung

```

1  const bulbGeometry = new THREE.SphereGeometry(.02, 16, 8);
2  const bulbLight = new THREE.PointLight( 0xffffffff, 3, 1000, 2 );
3  const bulbMat = new THREE.MeshStandardMaterial( {
4      emissive: 0xffffffff,
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.

3.10 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 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.



Abbildung 20: Landing Page

3.11 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 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 ihrem Registrationsstatus auch besuchen kann.

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

3.11.1 JWT - Json Web Token [H]

3.11.2 Implementation im Backend [H]

3.11.3 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.

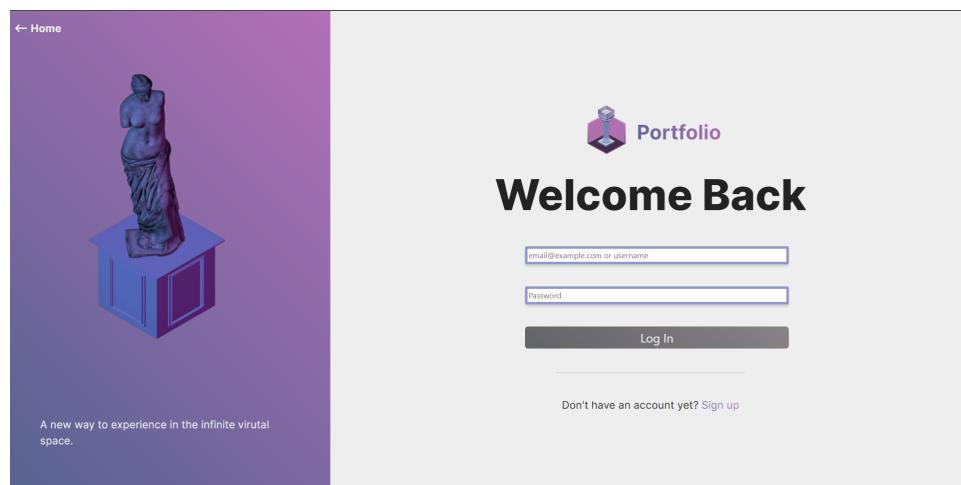


Abbildung 21: Projekt: Login Page

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 3.11.3) 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.

Bei den Ansätzen kann zwischen den reactive Forms und den Template-Driven-Forms unterschieden werden.

Template-Driven-Forms Bei den Template-Driven-Forms TODO template und reactive forms erklären [18, Bookmonkey - 12 Formularverarbeitung und Validierung: Iteration IV]

Token-Verwaltung

Um das Problem der Anmeldung zu lösen, wurde der JWT im LocalStorage des Browsers gespeichert. Im Codeausschnitt (siehe 29) 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 3.11.3) 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 29: 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`. [67] [68]

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. [68]

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. [68]

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. [68]

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. [18, 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 30).

Listing 30: 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()

```

```

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.

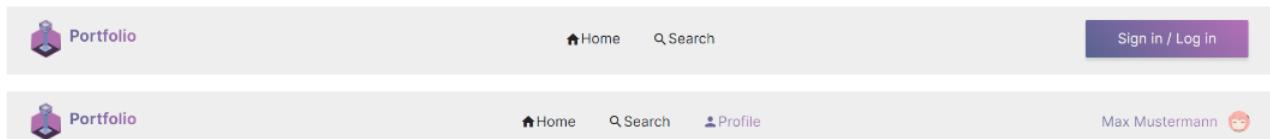


Abbildung 22: Navbar: authentifiziert vs noch nicht authentifiziert

3.11.4 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.

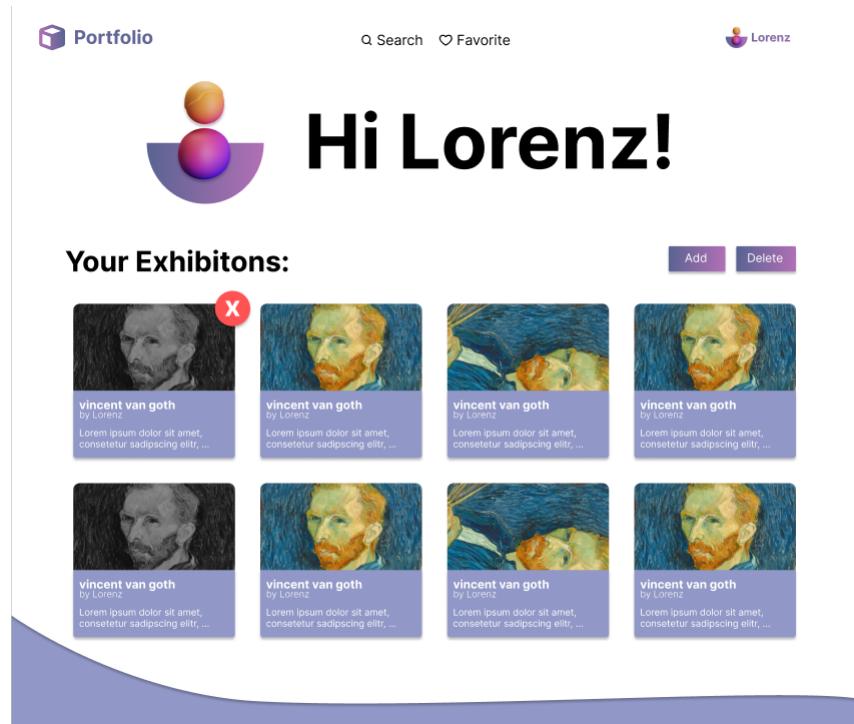


Abbildung 23: Profilepage

3.12 Interaktions-Seite [M]

3.12.1 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 [69]
- 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. [70]
- OrbitControls - Der*die Nutzer*in kann mittels linker Maustaste um ein bestimmtes Ziel kreisen. Durch die rechte Maustaste kann dieses Ziel geändert werden. [71]
- //TODO

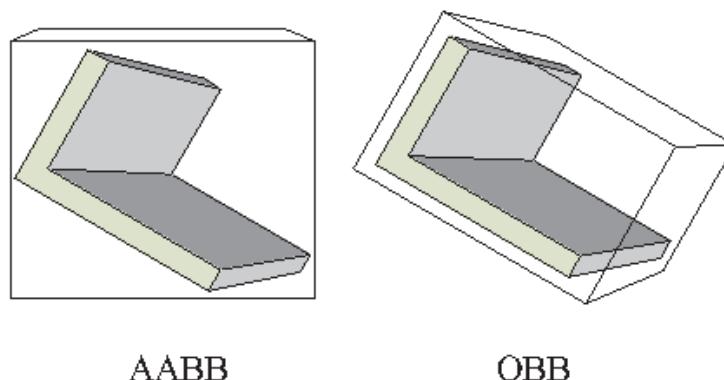


Abbildung 24: Der Unterschied zwischen AABB und OBB [72]

3.12.2 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.

3.12.3 Bounding Box

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

- Axis Aligned Bounding Box (AABB)

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

- Oriented Bounding Box (OBB)

Die Oriented Bounding Box 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 //TODO Referenz. 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 //TODO 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.

4 Zusammenfassung

In diesem Projekt machte das Diplomarbeitsteam große Erfahrungen und Fortschritte im Bereich Team-, 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.

4.1 Probleme und Lösungsstrategie

4.1.1 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. [73]

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 31)

Listing 31: 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 ein globaler Wert verändert werden muss, das 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. [74] [75]

4.2 Zielerreichung

4.3 Erweiterungsvorschläge

Aufzählungen:

- Itemize Level 1
 - Itemize Level 2
 - Itemize Level 3 (vermeiden)
1. Enumerate Level 1
 - a. Enumerate Level 2
 - i. Enumerate Level 3 (vermeiden)

Desc Level 1

Desc Level 2 (vermeiden)

Desc Level 3 (vermeiden)

5 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. [76]

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. [77]

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

Direktiven [M]

//TODO

Pipes [M]

//TODO

API [M]

//TODO

Optionals [M]

//TODO

Literaturverzeichnis

- [1] 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>
- [2] T. P. G. D. Group, „PostgreSQL: About,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://www.postgresql.org/about/>
- [3] Quarkus, „What is Quarkus?” letzter Zugriff am 08.03.2023. Online verfügbar: <https://quarkus.io/about/>
- [4] ——, „Creating Your First Application,” letzter Zugriff am 08.03.2023. Online verfügbar: <https://quarkus.io/guides/getting-started>
- [5] T. A. S. Foundation, „Maven – Introduction,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://maven.apache.org/what-is-maven.html>
- [6] JetBrains, „IntelliJ IDEA - the Leading Java and Kotlin IDE,” letzter Zugriff am 08.03.2023. Online verfügbar: <https://www.jetbrains.com/idea/>
- [7] A. Ivanos, „AngularEvidence.” Online verfügbar: <https://stackdiary.com/front-end-frameworks/>
- [8] A. P. Sofiya Merenchy, „AngularEvidence2.” Online verfügbar: <https://clockwise.software/blog/best-angular-applications/>
- [9] Angular Team, „What is Angular,” letzter Zugriff am 20.02.2023. Online verfügbar: <https://angular.io/guide/what-is-angular>
- [10] ReactiveX Team, „Documentation Introduction for ReactiveX,” letzter Zugriff am 24.02.2023. Online verfügbar: <https://reactivex.io/intro.html>
- [11] R. Gravelle, „AngularArchitecturePattern.” Online verfügbar: <https://www.htmlgoodies.com/javascript/the-model-view-viewmodel-pattern-and-angular-development/>
- [12] M. Team, „MVCmdn.” Online verfügbar: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
- [13] ——, „MVVM.” Online verfügbar: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm%0A>
- [14] ——, „MVC.” Online verfügbar: <https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview>
- [15] Angular Team, „AngularProviders.” Online verfügbar: [AngularProviders](#)
- [16] ——, „Angular ngModules,” letzter Zugriff am 20.02.2023. Online verfügbar: <https://angular.io/api/core/NgModule>

- [17] ——, „AngularNgModulesAPI.” Online verfügbar: <https://angular.io/guide/ngmodule-api>
- [18] D. K. Johannes Hoppe, Ferdinand Malcher, „AngularBuch, - Grundlagen, fortgeschrittene Themen und Best Practices,” 2020.
- [19] ThreeJs Team, „Fundamentals of ThreeJs,” letzter Zugriff am 26.02.2023. Online verfügbar: <https://threejs.org/manual/#en/fundamentals>
- [20] Khronos Foundation, „Webgl - Getting Started - Wiki,” letzter Zugriff am 26.02.2023. Online verfügbar: https://www.khronos.org/webgl/wiki/Getting_Started
- [21] 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>
- [22] „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))
- [23] Angular Three Team, „Angular Three: Our first scene,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular-three.netlify.app/docs/first-scene>
- [24] M. Team, „Cinema4D,” letzter Zugriff am 22.02.2023. Online verfügbar: <https://www.maxon.net/de/cinema-4d>
- [25] G. Team, „C4DvsMaya.” Online verfügbar: <https://www.getapp.de/compare/2036947/2051550/cinema-4d/vs/maya>
- [26] Notion, „Reference,” letzter Zugriff am 12.03.2023. Online verfügbar: <https://www.notion.so/help/reference>
- [27] I. S.-M. F. Schwab, „Ideenfindung,” in *Systemplanung und Projektentwicklung*, Wien, 2013, ch. 2, letzter Zugriff am 1.03.2023.
- [28] ——, „Agile Vorgehensmodelle,” in *Systemplanung und Projektentwicklung*, 2013, ch. 6.4.
- [29] ——, „Prototyping,” in *Systemplanung und Projektentwicklung*, Wien, 2013, ch. 6.1.3.
- [30] 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>
- [31] P. D. F.-R. Esch, „Corporate Identity,” letzter Zugriff am 3.03.2023. Online verfügbar: <https://wirtschaftslexikon.gabler.de/definition/corporate-identity-31786>
- [32] ——, „Corporate Design,” letzter Zugriff am 3.03.2023. Online verfügbar: <https://wirtschaftslexikon.gabler.de/definition/corporate-design-30453>
- [33] 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/>
- [34] 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

- [35] S. J. . J. Team, „Angular Material Tutorial,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://www.javatpoint.com/angular-material>
- [36] E. Ighosewe, „What Is Angular Material,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://upstackhq.com/blog/software-development/what-is-angular-material>
- [37] Angular Team, „Angular Material - components,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://material.angular.io/components/categories>
- [38] G. Team, „Material Design: Introduction,” letzter Zugriff am 5.03.2023. Online verfügbar: <https://m2.material.io/design/introduction>
- [39] Sass Team, „Sass Basics,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://sass-lang.com/guide>
- [40] w3schools Team, „What is npm?” letzter Zugriff am 6.03.2023. Online verfügbar: https://www.w3schools.com/whatis/whatis_npm.asp
- [41] npm Team, „About npm,” letzter Zugriff am 6.03.2023. Online verfügbar: <https://docs.npmjs.com/about-npm>
- [42] 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/>
- [43] Angular Team, „AngularComponentOverview.” Online verfügbar: <https://angular.io/guide/component-overview>
- [44] ——, „BindingSyntax.” Online verfügbar: <https://angular.io/guide/binding-syntax>
- [45] ——, „AngularPropertyBinding.” Online verfügbar: <https://angular.io/guide/property-binding>
- [46] ——, „AngularEventBinding.” Online verfügbar: <https://angular.io/guide/event-binding>
- [47] ——, „AngularTwoWayBinding.” Online verfügbar: <https://angular.io/guide/two-way-binding>
- [48] ——, „AngularArchitectureService.” Online verfügbar: <https://angular.io/guide/architecture-services>
- [49] ——, „AngularHTTPClient.” Online verfügbar: <https://angular.io/guide/http>
- [50] A. Team, „AdobeRendering.” Online verfügbar: <https://www.adobe.com/de/products/substance3d/discover/3d-rendering.html#:~:text=Beim 3D-Rendering wird aus, Modell sein 2D-Bild gerendert.>
- [51] Norman I. Badler, Andrew S. Glassner, „Rendering3DModels.” Online verfügbar: http://gamma.cs.unc.edu/courses/graphics-s09/LECTURES/3DModels_SurveyPaper.pdf
- [52] M. D. Team, „RenderArten.” Online verfügbar: <https://magic-3d.de/was-sind-3d-renderings/>
- [53] P. Team, „MoireEffekt.” Online verfügbar: <https://www.printer-care.de/de/drucker-ratgeber/moire-effekt>
- [54] J. Herzog, „AntiAliasing.” Online verfügbar: [https://de.wikipedia.org/wiki/Antialiasing_\(Computergrafik\)#/media/File:EasterEgg_antialiasing.png](https://de.wikipedia.org/wiki/Antialiasing_(Computergrafik)#/media/File:EasterEgg_antialiasing.png)

- [55] N. Team, „RayTracingRasterization.” Online verfügbar: <https://developer.nvidia.com/discover/ray-tracing>
- [56] T. Team, „ThreejsWebGLRenderer.” Online verfügbar: <https://threejs.org/docs/#api/en/renderers/WebGLRenderer>
- [57] Angular Team, „AngularViewChild.” Online verfügbar: <https://angular.io/api/core/ViewChild>
- [58] T. Team, „ThreejsCreateAScene.” Online verfügbar: <https://threejs.org/docs/?q=scene#manual/en/introduction/Creating-a-scene>
- [59] ——, „StandardLight.” Online verfügbar: <https://threejs.org/docs/index.html#api/en/renderers/WebGLRenderer.physicallyCorrectLights>
- [60] ——, „LightProbe.” Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/LightProbe>
- [61] ——, „AmbientLight.” Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/AmbientLight>
- [62] ——, „DirectionalLight.” Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/DirectionalLight>
- [63] ——, „HemisphereLight.” Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/HemisphereLight>
- [64] ——, „PointLight.” Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/PointLight>
- [65] ——, „ReactAreaLight.” Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/RectAreaLight>
- [66] ——, „SpotLight.” Online verfügbar: <https://threejs.org/docs/?q=light#api/en/lights/SpotLight>
- [67] Wiki Contributors, „Persistenz,” letzter Zugriff am 11.03.2023. Online verfügbar: <https://de.wiktionary.org/wiki/Persistenz>
- [68] 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
- [69] T. Team, „DragControls.” Online verfügbar: <https://threejs.org/docs/?q=Control#examples/en/controls/DragControls>
- [70] ——, „FirstPersonControls.” Online verfügbar: <https://threejs.org/docs/index.html#examples/en/controls/FirstPersonControls>
- [71] ——, „OrbitControls.” Online verfügbar: <https://threejs.org/docs/#examples/en/controls/OrbitControls>
- [72] M. Angelini, „AABBandOBBCPicture.” Online verfügbar: <https://devdept.zendesk.com/hc/en-us/articles/360011559320-How-Collision-Detection-works-v2020>
- [73] Angular Team, „Angular: View encapsulation,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular.io/guide/view-encapsulation>

- [74] ——, „Angular: Component styles - (deprecated) /deep/,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://angular.io/guide/component-styles#deprecated-deep--and-ng-deep>
- [75] Joshua Colvin, „Understanding Angular ::ng-deep,” letzter Zugriff am 09.03.2023. Online verfügbar: <https://www.joshuacolvin.net/understanding-ng-deep/>
- [76] 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/>
- [77] ——, „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 [12]	8
2	Die Model-View-ViewModel Pattern [13]	8
3	ThreeJs Logo	10
4	Die Struktur von ThreeJs [19]	11
5	ThreeJs Logo	11
6	Cinema4D vs Maya [25]	14
7	Modellierung des Raumes in Cinema4D	15
8	Notion Workspace der Diplomarbeit	17
9	Don Knuth – CS Allfather	19
10	OberflächenDesign: Prototypen in Figma	26
11	Design Konzept 1 Page 1	27
12	Design Konzept 1 Page 2	28
13	Design Konzept 2	29
14	Angular Material: Komponente Überblick [37]	31
15	Angular: Automatisch generierte Start-Webseite	35
16	Component-Hierarchy [18]	38
17	Der Moire-Effekt visualisiert [53]	45
18	Rendering mit und ohne Anti-Aliasing [54]	46
19	Vorgang des Ray-Tracing [55]	47
20	Landing Page	50
21	Projekt: Login Page	51
22	Navbar: authentifiziert vs noch nicht authentifiziert	55
23	Profilepage	56
24	Der Unterschied zwischen AABB und OBB [72]	57

Tabellenverzeichnis

1	Ein paar tabellarische Daten	18
---	--	----

Quellcodeverzeichnis

1	Beispielkonfigurationen	4
2	Angular Three - Komponentenbasiertes 3D Scenen in HTML	12
3	Angular Three - App Cube	13
4	Some code	19
5	SCSS - Code Beispiel	33
6	CSS - Code Beispiel	33
7	Terminalm - Angular aufsetzen, Installation der CLI, Configuration eines neuen Projektes, Starten des Projektes	34
8	Terminal - Angular Material Installation	36
9	Terminal - Bootstrap Installation	36
10	angular.json - Bootstrap Angular Verknüpfung	36
11	Terminal - Component erstellen	37
12	Beispiel für Interpolation in der 3D-Gallery	39
13	Beispiel für Property-Bindings in der 3D-Gallery	39
14	Beispiel für Event-Bindings in der 3D-Gallery	39
15	Beispiel für Two-Way-Bindings [47]	40
16	Routing in der 3D-Gallery	40
17	Routingparameter in der 3D-Gallery	41
18	Snapshot der URL abfragen	41
19	Die URL subscriben	41
20	Eine Klasse Injectable machen	42
21	Constructor Injection	42
22	HttpClient Abfragen	42
23	Das Datenmodell eines Ausstellungsstückes	43
24	Canvas-Element in HTML	47
25	Canvas als View-Child initialisieren	48
26	WebGLRenderer anlegen	48
27	Animations-Schleife	48
28	Lichtsetzung in der 3D-Ausstellung	49
29	auth.service.ts - JWT und Localstorage	53
30	auth.interceptor.ts - add JWT to Request Header	54
31	Parent.component.scss - Changing Styling in Child Components by using :ngdeep	59

Anhang