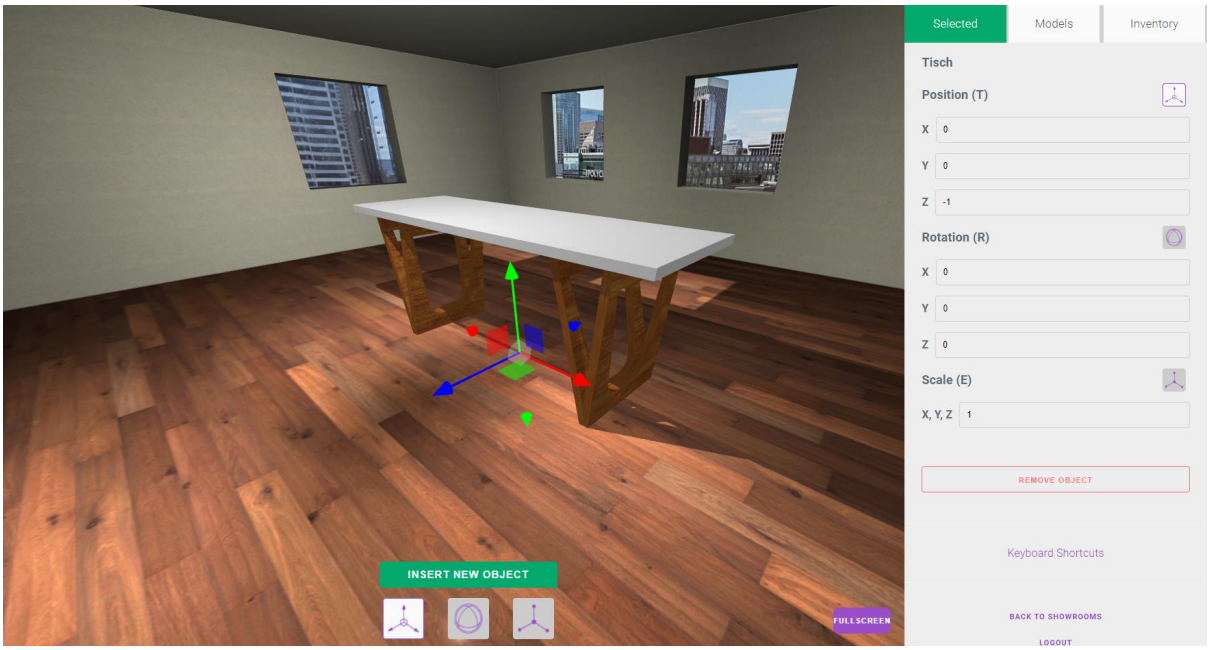
Fachhochschule Nordwestschweiz
Hochschule für Technik

INNOLOGY
[creatives behind the technologies]

Virtual Showroom Creator

Bachelor-Thesis



Studenten	Felix Kerkmann Fiorenzo Gramm
Experte	Stefan Arisona
Fachbetreuer/in	Madlaina Kalunder Alexandre Thomas
Auftraggeber	Francesco Bellanza
Projektnummer	22FS_IIT22
Studiengang	Informatik
Semester	Frühlingssemester 2022

Fachhochschule Nordwestschweiz, Hochschule für Technik

Windisch, August 19.08.2022

Abstract

Es gibt viele Strategien und Werkzeuge, um Kunden Produkte zu verkaufen. So können Produkte in einem Showroom präsentiert und von Kunden betrachtet werden. Dafür werden physische Produkte, Platz und je nach Produkt die passende Infrastruktur benötigt. Das kann nicht immer gewährleistet werden, so fehlt es zum Beispiel bei Messen an Platz. Produkte in einem virtuellen Raum zu präsentieren, löst dieses Problem. Daher soll ein Virtual Showroom Creator Kunden erlauben, eigene Showräume virtuell zu erstellen und so zu gestalten, dass ihre Produkte auch an Orten mit beschränktem Platz präsentiert werden können.

Es wurden zuerst Nutzerszenarien erfasst, die alle Anforderungen an einen optimalen Virtual Showroom Creator aufzeigen. Anhand dieser Anforderungen konnten einige bestehende Software-Lösungen untersucht werden, um zu erkennen ob einerseits bereits ein solches Produkt besteht und falls nicht, wie diese einzelnen Anforderungen umgesetzt wurden oder an was sie scheiterten. Anschliessend wurden einige Technologien und Frameworks untersucht, welche für eine Entwicklung einer solchen Applikation genutzt werden können.

Da im festgelegten Zeitrahmen keine ausgereifte Software-Lösung entwickelt werden kann, wurden ein Prototyp mit einer minimalen Menge an Anforderungen entwickelt. Dieser Prototyp erfüllt nur Kernanforderungen. Es zeigt sich, dass aktuelle Technologien und Frameworks effizient genutzt werden können.

Keywords:

Extended Reality, Virutal Reality, WebGL, Three.js, A-Frame, Nodejs

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ausgangslage

Die Innology GmbH ist ein Informatikdienstleister mit Schwerpunkt auf Extended Reality (XR). Kunden gibt es aus allen Branchen. Gemäss der Erfahrungen der Innology GmbH suchen ihre Kunden nach einer geeigneten Plattform um ihre Produkte nicht nur intern, sondern auch bei Kunden, Messen und Events vorstellen zu können. In vielen spezifischen Bereichen ist das sehr aufwendig und mitunter sehr kostenintensiv. Ein Whirlpool Verkäufer zum Beispiel, kann sein Sortiment nicht einfach von einem Ort an den anderen transportieren und präsentieren. Zudem wird für ein solches Vorhaben viel Platz, beziehungsweise eine geeignete Räumlichkeit benötigt. Doch diese Räumlichkeiten sind nicht immer wie gewünscht verfügbar. Ein Verkäufer muss entweder bei sich in der Firma einen oder mehrere Räume zur Präsentation zur Verfügung stellen oder an bestimmten Standorten einen kaufen, bzw. mieten und diesen an seine Bedürfnisse anpassen. Daher ist es eine naheliegende Möglichkeit, die Präsentation in die virtuelle Welt zu verlegen. Der Verkäufer ist dadurch weder an einen Ort, noch an eine Räumlichkeit gebunden. Großunternehmen haben ein genügend grosses Marketingbudget für die Entwicklung von zugeschnittenen Lösungen. Für kleine Unternehmen ist eine solche Investition nicht rentabel, da die Entwicklung eines solchen Produktes sehr kostspielig ist und oft keine Inhouse-Expertise vorhanden ist. Früher war diese Technik für viele Menschen noch neu und unverständlich. In den letzten Jahren hat sich laut (Petrov, 2022) die VR (Virtual Reality), AR (Augmented Reality) und MR (Mixed Reality) Branche schnell weiterentwickelt und dringt in immer mehr Bereiche des Lebens ein. Die Menschen nehmen sich immer mehr dieser Technologien an.

1.2 Problemstellung

Die Präsentation eines Produkts ist ein wichtiges Marketinginstrument, um den Bekanntheitsgrad einer Firma zu verbessern und die Verkaufsabschlüsse zu erhöhen. In Messen und klassischen Showrooms wird die ganze Raumgestaltung von Marketing-Experten umgesetzt, sowie zusätzliche Materialien wie Broschüren, Plakate oder Videos bereitgestellt. Damit das auch im digitalen Raum gelingt, müssen für das Einrichten des Raumes entsprechende Konfigurations- und Individualisierungsmöglichkeiten geboten werden. Dabei sollen sich Verkäufer und Kunde auch im virtuellen Showroom treffen und miteinander kommunizieren können. Es gibt aktuell keine Applikation, die einen virtuellen Showroom Konfigurator bereitstellt, der der Vision der Innology entspricht.

1.3 Vision der Innology

Um der Problematik der hohen Kosten für die Entwicklung von Einzellösungen entgegenzuwirken, soll ein VR Showroom Konfigurator für Produktpräsentationen als Software as a Service (SaaS) Lösung entwickelt werden. Somit können VR-Technologien auch für Klein- und Mittel- und Einzelunternehmen zugänglich gemacht werden. Die Plattform ermöglicht es Produktverkäufern und Geschäftsinhabern, sowie Marketing-Spezialisten einen VR-Showroom zu erstellen und zu bearbeiten, ohne Programmier- oder 3D-Kenntnisse zu haben. Da es sich um eine Web-Applikation handelt, kann diese einfach im Browser ausgeführt werden und benötigt keine sonstigen lokalen Installationen.

Der Nutzer kann zum Bearbeiten den Raum in seinem Browser betreten und dabei Objekte im Raum betrachten, bewegen, ausrichten und löschen, aber auch neue Objekte einfügen. So kann er zum Beispiel einen Tisch, den er verkaufen möchte, neben eine Wand platzieren und Modelle von Stühlen in den Raum laden und passend neben dem Tisch platzieren. Zudem soll er die Möglichkeit haben, all seine bereits bestehenden Räume zu verwalten. Dazu zählt das Bearbeiten, Löschen und das Hinzufügen eines neuen Raumes.

Der konfigurierte Showroom soll dann mit einer VR-Brille betreten werden können. Die Plattform soll dabei unabhängig sein, sich also nicht auf eine einzige Brille eines Herstellers beschränken. In diesem Projekt wurde ausschließlich die Oculus Quest 2 verwendet.

Ein Showroom ersetzt nicht den Verkäufer. Deshalb ist eine Lösung mit der Funktion für mehrere Personen gleichzeitig wichtig. So kann der Verkäufer mit dem Käufer agieren, wie er es in einem physischen Showroom machen kann. Der Verkäufer kann verschiedene Produkte ortsunabhängig bei Messen, Kunden oder in der eigenen Filiale in einem virtuellen Showroom präsentieren. Extern benötigt der Verkäufer weder seine Produkte noch entsprechende Räumlichkeiten.

1.4 Abgrenzung

Da die Entwicklung einer marktreifen Lösung den Umfang dieser Arbeit sprengen würde, geht es hier darum, einen Prototyp zu entwickeln. Welche Anforderungen erfüllt werden, wird im Kapitel *Anforderungen* genauer erläutert. Es ist also keine Lösung die direkt ausgeliefert und vermarktet werden kann, sondern zeigt auf, dass mit den gegebenen Technologien eine Lösung entwickelt werden kann.

Die Erstellung von 3D-Modellen ist kein Teil des Virtual Showroom Creators, sondern ist die Aufgabe des Verkäufers oder seines 3D Designers. Welche Merkmale eines 3D Modells dafür sorgen, dass ein Produkt gut und realitätsnah dargestellt werden kann, ist nicht Teil dieser Thesis. Das ist für die Entwicklung des Prototyps nicht von essenzieller Bedeutung. *Abbildung 1* verdeutlicht die Struktur und Systemgrenze des Virtual Showroom Creators. Durch das Verwenden von bereits bestehenden Standards können aber entsprechende Schnittstelle definiert werden.

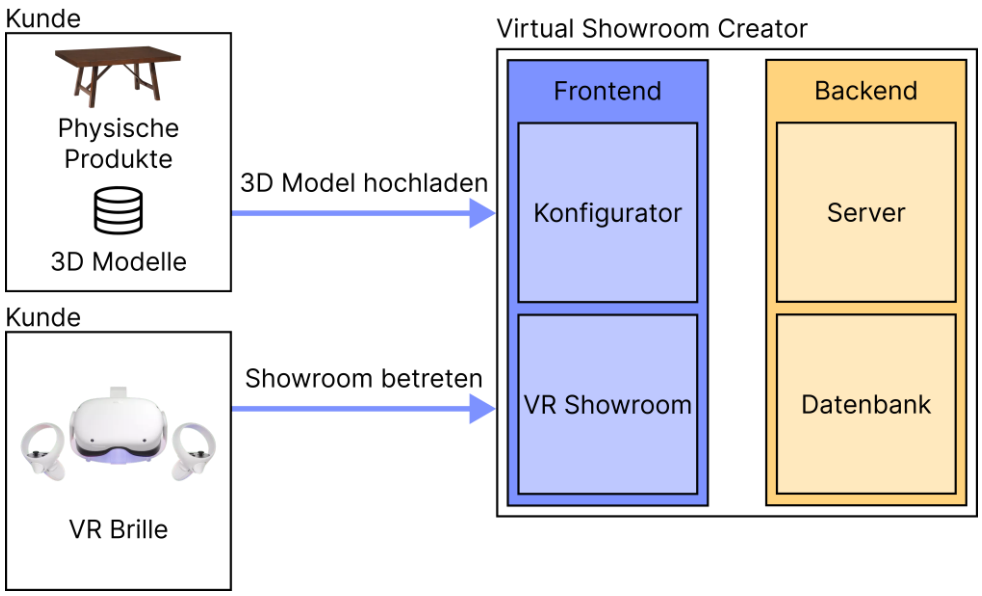


Abbildung 1: Darstellung der Systemgrenze des Virtual Showroom Creators und Aufteilung in Frontend und Backend sowie die zwei Kernkomponenten (Konfigurator und VR Showroom)

1.5 Methodik, Vorgehen, Aufbau des Dokuments

Die Arbeit ist so aufgebaut, dass zuerst erklärt wird, wie der Funktionsumfang des Virtual Showroom Creator aussieht. Darauffolgend beschreibt die Wissenschaftliche Recherche konzeptionelle und technische Faktoren. Danach kommt die Recherche unterschiedlicher bereits bestehender Produkte und deren Analysen. Im Anschluss daran werden einige Technologien aufgezeigt, mit denen es möglich ist, diese Applikation zu entwickeln. Nach der Technologie werden die Konzepte erläutert, mit den die Komponenten des Virtual Showroom Creators entwickelt wurden. Anschliessend wird detaillierter auf die Umsetzung mit den einzelnen Werkzeugen eingegangen. Am Ende werden die Ergebnisse zusammengetragen und weitere Empfehlungen vorgeschlagen.

2 Theoretischer Teil

2.1 Stakeholder-Analyse

Die Innology möchte eine wirtschaftliche Software, also eine Software die mehr Kapital einbringt als die Wartung und Entwicklung kostet. Neben der Innology sind auch die Verkäufer, also die Unternehmen, die Produkte verkaufen möchten, und auch die Käufer die wichtigsten Akteure. Das Ziel einer Stakeholder-Analyse ist es, möglichst früh in der Projektphase unterschiedliche Interessen aufzuzeigen, die während der Entwicklung berücksichtigt werden sollten, (Brugha & Varvasovszky, 2000). Es gibt verschiedene Parteien, die mit unterschiedlichem Interesse und Einfluss mit dem Projekt verbunden sind.

Die Innology bietet Produkte und Dienstleistungen an, mit denen Kunden ihre Produkte Käufern vorstellen können. Daher möchte sie ein Produkt anbieten, das seinen Kunden ermöglicht, eigene virtuelle Räume zu erstellen, die sie ihren Käufern präsentieren können. Da die Innology der Auftraggeber des Projektes ist, ist ihr Einfluss sehr gross.

Die Kunden der Innology grosses Interesse daran, ihre Produkte den Kunden zu präsentieren, um die Kaufwahrscheinlichkeit zu steigern, konkurrenzfähig zu bleiben und sich innovativ zu präsentieren. Falls ein entwickeltes Produkt nicht genutzt wird, bietet es auch der Innology keinen Wert. Die Erfahrung und das Wissen der Innology Kunden ist lässt sich massgebend in die Entwicklung einbinden und ist damit für den Erfolg von grosser Bedeutung.

Potenzielle Käufer, die Interesse an einem Produkt äussern und gerne mehr Informationen hätten, um eine Kaufentscheidung zu treffen, möchten das Produkt in verschiedenen Szenarien betrachten können. Wenn der Showroom nicht attraktiv genug ist, werden die Käufer ihn auf Dauer nicht nutzen. Potenzielle Käufer haben allerdings keinen direkten Einfluss auf das Projekt.

Der Produktdesigner erstellt ein 3D Modell zur Darstellung im Showroom. Er muss nicht zwingend dieselbe Person sein, die das Produkt entwickelt hat. Er muss ebenfalls nicht zwingend bei dem Verkäufer des Produktes angestellt sein. Der Designer möchte möglichst wenig Aufwand haben, die Modelle für den Virtual Showroom Configurator vorzubereiten. Zudem ist es nicht immer gegeben, dass der Verkäufer im Kontakt mit dem Designer steht. Im Idealfall sollen Modelle ohne weiteren Aufwand verwendet werden können. Auch die Designer der Modelle stehen nicht im direkten Kontakt mit dem Projekt und daher ist ihr Einfluss sehr gering.

Die IT-Abteilung der Innology ist für die Bereitstellung der Infrastruktur zur Ausführung der Software aber auch die Lagerung und Verfügbarkeit der Daten verantwortlich. Sie bietet ebenfalls die Schnittstelle zu den Kunden (Verkäufer) bei Problemen und bietet Unterstützung an. Um den Aufwand minimal zu

halten, sollte die Applikation auf einem möglichst kosteneffizienten Technologie Stack aufbauen. Zudem sollte sie fehlertolerant sein. Die Software wird zwar auf der Infrastruktur der Innology laufen, Einfluss auf das Projekt hat die IT der Innology allerdings nicht.

Konkurrierende Anbieter eines virtuellen Showroom Configurators möchten ihre eigenen Produkte vermarkten und sind daher nicht am Erfolg dieses Projekts interessiert. Allerdings haben sie auch keinen Einfluss auf das Projekt.

Die VR-Community profitiert davon, dass die VR-Technologie auch für kommerzielle Zwecke genutzt wird, das hat positiven Einfluss auf die Entwicklung und Verbreitung der VR-Technologie. Bessere und günstigere Hardware sowie ausgereiftere Anwender- und Entwicklungssoftware können die Folge sein. Die VR Community hat keinen Einfluss auf das Projekt selbst.

Ein Zwiebel-Diagramm zeigt vereinfacht, wie stark die einzelnen Akteure in dem Geschäftsprozess beteiligt sind (Alexander, 2004).Das in *Abbildung 2* dargestellte Zwiebel-Diagramm visualisiert hier die Stakeholder im Virtual Showroom Creator.

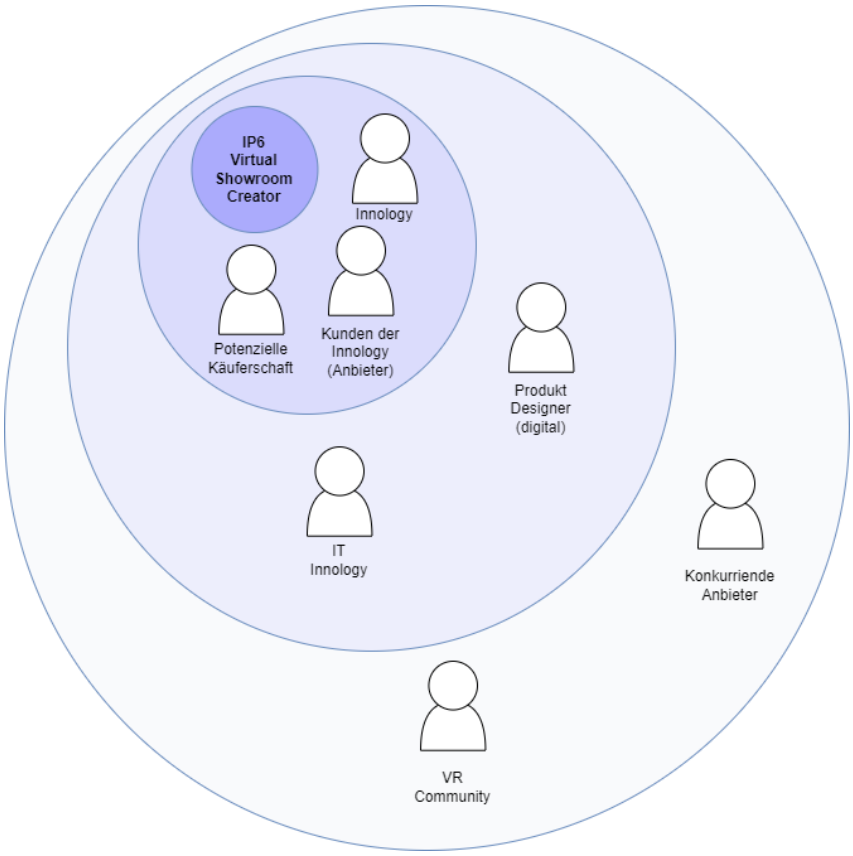


Abbildung 2: Die Akteure im äusseren Bereich nutzen den VR Showroom nicht selbst. Die in der Mitte stehen sporadisch mit den Virtual Showroom Creator in Kontakt. Die Akteure im inneren Bereich nutzen den VSC regelmässig und aktiv.

2.2 Customer Journey

Um präzisere Anforderungen erfassen zu können, die ein Virtual Showroom Creator haben muss, wurde ein Customer Journey erstellt. Dabei wird der Verwendungsprozess eines Nutzers vom ersten bis zum letzten Kontaktpunkt beschrieben. Daraus können im nächsten Schritt Kernfunktionen aber auch zusätzliche Quality of Life Funktionen extrahiert werden. Es geht also nicht um die Erstellung eines exakten Prototyps, sondern darum, eine genaue Vorstellung der Nutzererfahrung zu erhalten. Der Customer Journey wird aus Sicht eines Nutzers, also in diesem Falle einer Verkäuferin, beschrieben.

Diese Person wird im Customer Journey fortlaufend als Frau V. bezeichnet. Frau V. ist dabei zuständig für Marketing und Produktvorstellung in einem Unternehmen, das Möbel verkauft. Frau V. bereitet sich auf eine Messe vor, auf der viele Besucher erwartet werden. Leider steht dabei kaum Fläche für die

Vorstellung von Möbeln zur Verfügung. Als sie im Internet nach einer Lösung sucht, stösst sie auf den Virtual Showroom Creator der Innology GmbH. Nachdem sie auf der Seite genügend Information darüber gesammelt hat, entscheidet sie sich dafür. Auf der Login-Seite kann sie direkt ein neues Benutzerkonto erstellen und sich anmelden. Auf der Startseite findet sie Links zu Beispielszenen und Tutorials, die ihr erklären, wie der Virtual Showroom Creator aufgebaut ist und wie man ihn bedient. Danach legt sie direkt los und erstellt einen eigenen Showroom. Innerhalb des Konfigurators hat sie die Auswahl von einigen vorgegebenen Räumen, aus denen sie einen auswählt, der für sie passend erscheint. Dabei hat sie auch die Möglichkeit zwischen vordefinierten Flächen auszuwählen. Da der Raum zu Beginn leer ist, beginnt sie, eigene Modelle von ihrem Rechner hochzuladen. Sobald sie ein Modell hochgeladen hat, kann sie es im Raum platzieren. Sie kann neben einem Namen auch weitere Informationen zum Produkt wie eine Produktnummer angeben. Nun positioniert sie ihr Modell. Dabei kann sie nicht nur die Position ändern, sondern auch die Ausrichtung und Grösse anpassen. Nachdem sie den Raum konfiguriert hat und zufrieden ist, möchte sie selbst überprüfen, wie der Raum mit ihrer VR-Brille aussieht. Dabei teilt sie den soeben konfigurierten Showroom, der dann über einen ihr angezeigten Link erreichbar ist. Nun zieht sie ihre Oculus Quest 2 an und öffnet den Link im Browser. Sie sieht den Showroom direkt und kann sich darin bewegen und umsehen. Der Showroom ist nun für die Kundenpräsentation fertig.

2.3 Anforderungen

Aus dem im vorherigen Kapitel beschriebener Customer Journey wurden konkrete Anforderungen extrahiert. Diese wurden mit dem Auftraggeber besprochen. Die aus dem Customer Journey aufgelisteten Anforderungen wurden noch durch den Auftraggeber ergänzt. So stiessen gemäss der Erfahrung der Innology simple Animationen und austauschbare Materialien auf grosses Interesse. Aufgrund des Umfangs wurden einige Anforderungen, die nicht essenzieller Natur sind, nicht zum minimal brauchbaren Produkt hinzugefügt. Es wurden die Funktionen umgesetzt, die als essenziell gekennzeichnet sind. Die Anforderungen erlauben es, Showräume zu verwalten (Erstellen, Bearbeiten und Löschen) und diese mit verschiedenen Objekten auszustatten. Dabei können eigene Modelle hochgeladen und eingefügt sowie frei im Raum platziert werden. Animationen sind nicht Inhalt der Arbeit, trotzdem sind es Anforderungen, die nach dem Abschluss dieses Projekt umgesetzt werden können. Die *Tabelle 1* zeigt die Anforderungen in tabellarischer Form.

Anforderungen an den Konfigurator	**Priorität**
Als Kunde kann ich die Applikation im Browser bedienen.	Essenziell
Als Kunde kann ich ein Benutzerkonto mit meiner E-Mail-Adresse und einem Passwort einrichten.	Essenziell
Als Kunde kann ich mich auf der Plattform mit meiner E-Mail-Adresse und einem Passwort anmelden.	Essenziell
Als Kunde kann ich eine Liste all meiner bisher erstellten Showrooms einsehen.	Essenziell
Als Kunde kann ich einen neuen Showroom erzeugen.	Essenziell
Als Kunde habe ich die Möglichkeit, einen Showroom zu öffnen und zu bearbeiten.	Essenziell
Als Kunde kann ich ein neues Objekt in den Showroom einfügen und dabei ein 3D Modell einfügen.	Essenziell
Als Kunde kann ich verschiedene Variationen eines Objekts definieren. So kann ich zum Beispiel einem Objekt mehrere Materialen zuweisen.	Nicht essenziell
Als Kunde kann ich ein Objekt entlang der X-, Y- und Z-Achse verschieben.	Essenziell
Als Kunde kann ich ein Objekt um die X-, Y- und Z-Achse rotieren.	Essenziell
Als Kunde kann ich ein Objekt einheitlich vergrössern und verkleinern.	Essenziell
Als Kunde kann ich ein Objekt so animieren, so dass es sich entlang der X-, Y- und Z-Achse bewegt.	Nicht essenziell
Als Kunde kann ich ein Objekt so animieren, so dass es sich um die X-, Y- und Z-Achse dreht.	Nicht essenziell
Als Kunde kann ich ein Objekt so animieren, so dass sich die Grösse ändert.	Nicht essenziell
Als Kunde habe ich die Möglichkeit den Showroom als potenzieller Käufer zu betreten.	Essenziell

Als Kunde kann ich einen meiner Showrooms löschen.	Essenziell
Als Kunde kann ich zusammen mit dem potenziellen Käufer einen Showroom betreten. Dabei kann ich ihn führen und habe dieselben Möglichkeiten wie der potenzielle Käufer. So kann ich zum Beispiel das Material eines Produkts ändern und er sieht die Änderung ebenfalls.	Nicht essenziell
Als Kunde kann ich die Applikation im Browser bedienen.	Essenziell
Als Kunde kann ich Lichtquellen einfügen und wie Objekte bewegen.	Nicht essenziell

Tabelle 1: Anforderungen mit Priorisierung für den Konfigurator

Bemerkung zum Thema Animation: In dieser Anwendung ist nur eine sehr simple Animation gefordert. Es geht darum, dass ein Betrachter ein Objekt auswählen kann und dabei soll das Objekt so animiert sein, dass es optimal dargestellt werden kann. So soll ein Objekt, dass zum Beispiel aus mehreren Komponenten besteht, sich in dieses aufteilen und einzeln sichtbar werden.

Bemerkung zum Betreten mehrerer Personen gleichzeitig: Dabei handelt es sich um die Möglichkeit der Führung. Der Verkäufer, der den Showroom eingerichtet hat, soll mit dem Kunden den Showroom betreten können. Dabei soll er auch in der Lage sein für den Kunden die Produktvarianten auszuwählen. Das ist für Kunden hilfreich, die noch keine Erfahrung mit der VR-Technologie haben. So soll der Käufer sich mehr auf das Produkt als auf die Navigation und Steuerung konzentrieren können.

Anforderungen an die Betrachtung in VR	Priorität
Als Käufer kann ich mich frei im Showroom bewegen (5 DOF).	Essenziell
Als Käufer kann ich mich entlang der Achsen im Raum bewegen.	Essenziell
Als Käufer kann ich mich umschauen und so selbst bestimmen was ich wann sehe.	Essenziell
Als Käufer kann ich im Showroom auswählen welche Variante des Produkts ich anschauen möchte, zum Beispiel das Material.	Nicht essenziell
Als Käufer habe ich die Möglichkeit auf der Website den Showroom mit meiner eigenen VR-Brille anzuschauen (Showroom-On-Demand).	Nicht essenziell

Tabelle 2: Anforderungen und Priorisierung an die Betrachtung des Showrooms in VR

Bemerkung zum On-Demand Showroom: Für erfahrene Nutzer, die bereits über eigene VR-Ausrüstung verfügen, soll es möglich sein, den Showroom allein zu betreten. Potenzielle Käufer sollen damit in der Lage sein in einem Shop Produkte, die sie interessieren, selbständig betrachten zu können.

2.4 Wissenschaftliche Grundlage

2.4.1 Psychologische Faktoren beim Kaufentscheid

Es gibt viele Faktoren die den Kaufentscheid einer Person positiv als auch negativ beeinflussen können. Da es viele Faktoren gibt, die für jede Person individuell sind, ist es schwierig, allgemeine Faktoren, die für eine grosse Gruppe von Kunden zutrifft zu bestimmen. Gemäss (Vainikka, 2015) beeinflussen die Wahrnehmung, Interpretation, Achtsamkeit, Motivation, Persönlichkeit und Emotionen die Entscheidung einer Person massgeblich. Das Ziel des Marketings ist es, diese Faktoren so zu beeinflussen, dass sie die Person zu einer Kaufentscheidung führen. Dabei können nicht alle Faktoren gleichmässig beeinflusst werden. So ist es für eine Werbung, die man überspringen kann, um den Film den man gerade schaut weiter zu schauen, deutlich schwieriger eine Person zu involvieren wie zum Beispiel eine Testfahrt mit einem Auto. Ein physischer Showroom zeigt dem Kunden das Produkt nicht nur visuell, sondern kann auch eine Interaktion simulieren. Das kann weitere Emotionen auslösen, die sich auf die Kaufentscheidung auswirken. Es gibt ebenfalls Produkte, die zu gross für einen physischen Showroom sind. Ein virtueller Showroom kann hier die Lösung sein. Allerdings verliert dieser durch die nicht materiellen Produkte einen Teil der vorher beschriebenen Wirkung. Andererseits öffnen sich neue Möglichkeiten, da man nicht mehr auf den physischen Platz beschränkt ist. In einem virtuellen Showroom ist es ebenfalls möglich komplexe Anlagen in ihre Einzelteile zu zerlegen und alle Details einzeln anzeigen zu lassen.

Falls es gelingt den potenziellen Kunden so zu begeistern, kann dies die Wahrscheinlichkeit eines Kaufes fördern. Bei einer schlechten Performance besteht die Gefahr, dass der Kunde vergrault wird. Ist eine Person nicht geübt im Umgang mit der Technologie und ist überfordert, so wäre es eine negative Erfahrung. Um dieses Risiko zu minimieren, ist die Darstellung und Gestaltung eines solchen VR-Applikation ein wesentlicher Bestandteil.

2.4.2 3D Rendering in einem Browser

Ein wesentlicher Kernaspekt einer solchen Lösung ist das Darstellen von 3D-Welten in einem Browser. Den Prozess der Projektion einer 3D Szene mit einer Kamera auf ein 2D Bild, das auf dem Bildschirm angezeigt werden kann, wird als Rendering bezeichnet (Badler & Glassner, 2013). Bereits seit 1960 wurden Forschungen in der Computergrafik gemacht. So wurde von Professor Ivan Sutherland die Software «Sketchpad» entwickelt, die erlaubt einfache Formen darzustellen. Zu dieser Zeit wurden solche Berechnungen auf grossen, für diesen Zweck umgebauten Rechnern ausgeführt. Die Idee, dass solche und noch bessere Darstellungen auf handelsüblichen Geräten ohne explizite Anforderung an Software oder Hardware laufen, war kaum vorzustellen. Durch die Entwicklung der Informationstechnologie sowie die wachsenden Anforderungen der Anwender, verbesserte sich auch die Computergrafik. Sowohl durch die Interaktion mit grafischen Benutzeroberflächen der Betriebssysteme oder dem Internet wie auch grafikorientierter Software wie Spiele, Bildbearbeitung oder Filme, kommt fast jeder mit dem Thema in Kontakt (Orlamünder & Mascolus, 2008). So wurde unter anderem CAD (Computer Aided Design) entwickelt, mit dem mit Hilfe von Computern die Erstellung, Bearbeitung, Analyse oder Optimierung eines Designs ermöglicht wurde (Lalit Narayan, Mallikarjuna Rao, & Sarcar, 2008). Zu Beginn der 1980er Jahre wurde IRIS GL (Integrated Raster Imaging Graphics Library) als proprietäre Grafik-Schnittstelle für IRIX basierte grafische Computer veröffentlicht (Seddon, 2005). Erst 1992, also vor rund 30 Jahren wurde OpenGL veröffentlicht. OpenGL wurde als Alternative zu IRIS GL entwickelt. Die damaligen Entwickler versuchten dabei eine Grafik-Schnittstelle zu definieren, die plattformunabhängig ist. Heute wird OpenGL und auch dessen Nachfolger Vulkan von der Khronos Group entwickelt (Khronos Group, 2022). OpenGL ist jedoch nicht die einzige Grafik-Schnittstelle. Neben OpenGL gibt es weitere Schnittstellen wie Vulkan als Nachfolger von OpenGL, Microsofts proprietäres DirectX, Apples Metal und noch einige mehr.

Ursprünglich wurden Webbrowser nicht für 3D-Applikationen entworfen. Doch mit dem Fortschritt der dynamischen Inhalte und Client-Side Programmiersprachen, wuchs der Bedarf an einer 3D-Darstellung im Browser immer mehr. Nach und nach wurden immer mehr Technologien wie Silverlight, Flash und Shockwave entwickelt. Doch alle hatten den Nachteil, dass Erweiterungen im Browser installiert werden mussten (Nazarov & Galletly, 2013). Dann wurde es mit WebGL möglich, 3D Inhalte nur mit Verwendung von JavaScript in den meistverwendeten Browser darzustellen und dabei das volle Potenzial der Hardware zu nutzen. So können Elemente mit JavaScript API calls, also die Schnittstelle die WebGL bietet, gezeichnet werden. Dabei werden bereits vorhandene Canvas Elemente genutzt (Parisi, 2012).

2.4.3 Definition und Abgrenzung von Virtual Reality

Um zu verstehen, was Virtual Realitity bedeutet, muss zuerst virtual, oder zu Deutsch virtuell, genauer definiert werden. Laut Duden (Cornelsen Verlag GmbH, 2022) bedeutet virtuell im Kontext der Informationstechnologie:

> «nicht echt, nicht in Wirklichkeit vorhanden, aber echt erscheinend».

Dabei handelt es sich um nicht reale Objekte oder Umgebungen.

Dieses Konzept ist nicht neu, sondern es existiert bereits seit langem. So zum Beispiel in der Filmindustrie. Im Film «Matrix» wird eine perfekte virtuelle Welt erschaffen, in der die Menschen, die darin leben, sie nicht mehr von der Realität unterscheiden können. In der Fertigungstechnik werden virtuelle Prototypen erstellet, die auf einem Bildschirm dargestellt werden können, ohne dabei ein physischer Prototyp fertigen zu müssen. Wie in *Abbildung 3* können so zum Beispiel mithilfe von CAD Gebäude dargestellt werden.

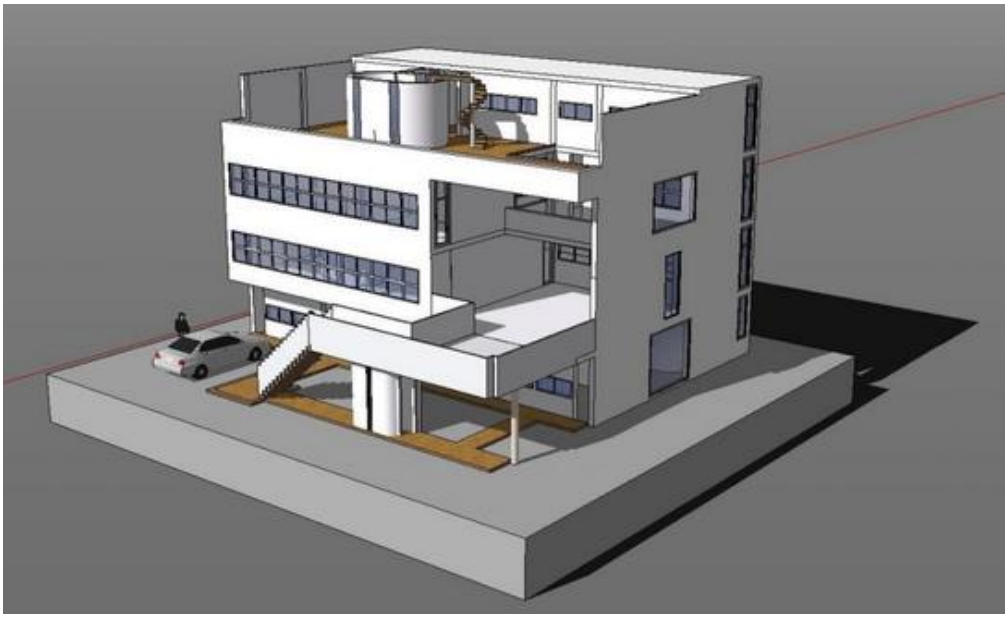


Abbildung 3: Eine Darstellung eines Gebäudes in drei Dimensionen.

Doch eine rein zweidimensionale Darstellung von dreidimensionalen Objekten ist nicht genug, um VR vollumfänglich zu beschreiben. Eine Definition von VR laut (Bryson, 1993):

> «*Virtual Reality bezieht sich auf die Verwendung von dreidimensionalen Displays und Interaktionsgeräten, um computergenerierte Umgebungen in Echtzeit zu erkunden*».

Laut dieser Definition entfernt sich der Begriff Virtual Reality von simplen Darstellungen auf einem Monitor hin zu einer dreidimensionalen Darstellung. Um 3D Inhalte auch korrekt darzustellen, werden entsprechende Darstellungsmöglichkeiten benötigt. Um dies für den Sehsinn des Menschen zu bewerkstelligen, werden stereoskopische Displays verwendet (Doerner, Broll, Jung, & Grimm, 2022). Als Stereoskopie bezeichnet man die visuelle Fertigkeit, die Umgebung in 3D wahrzunehmen. So kann eine Person die Entfernung zwischen sich und anderen Objekten einschätzen. Diese Fähigkeit des räumlichen Sehens hat der Mensch durch sein binokulares Sehen. Das bedeutet, dass er zwei Augen hat, die in dieselbe Richtung sehen (Dr. Lazarus, 2021). So kann mit zwei Displays, für jedes Auge eines, eine 3D Darstellung erreicht werden. Zu VR zählen aber nicht nur visuelle Eindrücke, sondern auch weitere wie das Hören, Berühren, oder haptisches Feedback. So gibt es zum Beispiel Handschuhe oder Kontroller, die mit Motoren ausgerüstet sind, um dem Nutzer ein haptisches Feedback zu übermitteln. Bereits sehr früh wurden verschiedene Arten von speziellen Eingabe- und Ausgabegeräten von VR-Systemen entworfen. Einige davon wie zum Beispiel Head-Mounted-Display (HMDs), Stereobrillen oder Tracking-Handschuhe können von Nutzern getragen werden. Neben der Darstellung ist auch die 3D Interaktion ein Bestandteil der virtuellen Realität. Es gibt diverse Geräte, welche die Position oder Orientierung verfolgen können, um so die Intentionen des Benutzers erfassen zu können. So werden zum Beispiel die Position und Orientierung des Kopfes gemessen. Dieses head-tracking bildet die Basis für viele VR-Systeme und ermöglichen es, abhängig vom Benutzer andere Bilder darzustellen. Bewegt sich ein Nutzer nun in der realen Welt, so bewegt er sich automatisch auch in der virtuellen Welt und es wird ihm eine neue Perspektive angezeigt. Das ermöglich dem Nutzer des VR-Systems ein immersives Erlebnis in der virtuellen Welt. Die Immersion wird oft als wesentliches Merkmal von VR System bezeichnet, doch leider ist der Begriff in der Literatur nicht eindeutig (Doerner, Broll, Jung, & Grimm, 2022).

Im Vergleich zu VR, die die Wahrnehmung des Nutzers der realen Umgebung durch eine virtuelle ersetzt, befasst sich die Augmented Reality (erweiterte Realität), oder kurz AR, damit, die reale Umgebung zu nutzen und weitere (virtuelle) Informationen hinzuzufügen. Dabei ist der Inhalt mehrheitlich real. Überwiegt jedoch der virtuelle Anteil den realen Anteil, ohne dabei rein virtuell zu sein, spricht man von Augmented Virtuality (AV) (Steve, Furness, Yuan, Iorio, & Wang, 2018). Definiert man ein Spektrum, das sich zwischen Realität und Virtualität (VR) erstreckt, wobei der Anteil der Realität kontinuierlich abnimmt und der der Virtualität zunimmt, vereinfacht es die verschiedenen Begriffe besser einzuordnen *Abbildung 4*.

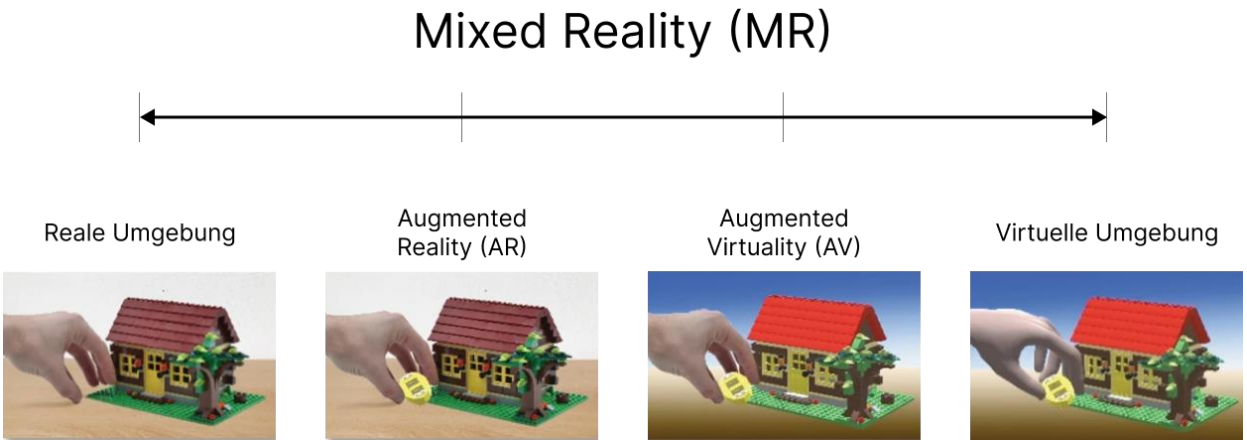


Abbildung 4: Das Spektrum zwischen der realen und rein virtuellen Umgebung (Milgram, Takemura, Utsumi, & Kishino, 1995).

Da das Ziel dieser Arbeit ist, sich vom physischen Raum zu trennen, befasst sich die Arbeit nur mit der rein Virtuellen Umgebung. Es gibt zwar auch Anwendungen, in denen eine Lösung an einem anderen Punkt im Spektrum Sinn machen kann, für dieses Projekt ist das jedoch nicht der Fall.

2.4.4 Entwicklung der Virtual Reality

Die Entwicklung der VR-Technologie begann bereits in den 60er Jahren. 1965 schrieb Ivan Sutherland in seinen Forschungen über immersive Technologien über „The Ultimate Display“. Dabei beschrieb er seine Vision von einem Raum, mit dem ein Computer verbunden ist, der in der Lage ist, die Existenz der Materie im Raum zu kontrollieren (Sutherland, 1965). Die Vision virtueller Welten ist erstaunlich, da zu dieser Zeit der persönliche Computer noch nicht erfunden war. Bereits 1968 entwickelte Ivan Sutherland den ersten Prototypen eines Head-Mounted Systems mit einem Daten-Helm und einem mechanischen oder Ultraschall basierten Tracking System (*Abbildung 5*). Dieses System ermöglichte es, eine simple dreidimensionale Umgebung in der korrekten Perspektive darzustellen (Doerner, Broll, Jung, & Grimm, 2022).

Abbildung 5: Ivan Sutherlands frühe Umsetzung eines Head-Mounted-Displays. Quelle: (Sutherland, 1965)

Dieses System kann aufgrund seiner Durchsichtigkeit auch als das erste AR-System angesehen werden. Der Begriff „Augmented Reality“ wurde allerdings erst in den 90er Jahren geprägt. Bei einem Projekt für Boeing sollten Informationen im Sichtfeld angezeigt werden, damit die Arbeiter einfacher die Flugzeugkabel verlegen können (Caudell & Mizell, 1992). Mit „MARS“ wurde das erste mobile AR System präsentiert wie in *Abbildung 6* dargestellt (Feiner, Maclyntire, Höllerer, & Webster, 1979). Mit der Veröffentlichung von ARToolkit im Jahr 1998 (Kato & Billinghurst, 199) wurde das Computer Vision-basierte Tracking für AR verfügbar und löste weltweit eine große Forschungswelle aus.

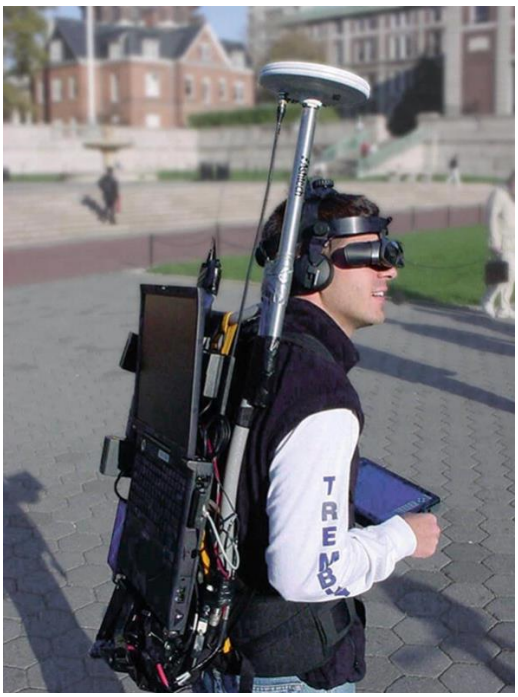


Abbildung 6: Das System MARS 1997. Quelle: (Steve, Furness, Yuan, Iorio, & Wang, 2018)

1999 veröffentlichte das 1993 gegründete Unternehmen Nvidia, GeForce Grafikchips. Mit diesen Grafikchips wurde 3D Hardware für den Konsumentenmarkt eingeführt. Doch es vergingen Jahre, in denen sich die Forschung der VR- und AR-Technologie auf Forschungsinstitute, grosse Industrieunternehmen und staatlichen Agenturen beschränkte. Doch als 2013 die Oculus Rift als erste „günstige" high-end VR-Brille auf den Markt kam, änderte sich dies schlagartig. Spätestens ab 2016, als Microsoft, Sony und HTC eigene VR-Systeme veröffentlichten, erlebte der VR-Markt einen Boom. Die AR-Technologie hingegen entwickelte sich langsamer, denn Microsofts HoloLens, eine technische Meisterleistung, konnte nicht gleichermassen für Begeisterung sorgen. Erst als 2017 einige grosse Softwareplattformen wie Apples ARKit oder Googles ARCore auf den Markt kamen, nahm die Entwicklung von AR-Applikationen zu (Doerner, Broll, Jung, & Grimm, 2022).

2.4.5 Navigation im virtuellen Raum

Die Bewegung im Raum erlernt der Mensch bereits sehr früh und automatisiert diesen Prozess vollständig. So ist er in der Lage, einen Zielpunkt anzuvisieren und ohne darüber nachzudenken steuert das Gehirn die Muskeln so, dass man sich automatisch auf das Ziel zu bewegt. Verfügt das VR-System über genügend Sensoren, kann diese Art der Bewegung auch im virtuellen Raum verwendet werden. Dabei ist die Bewegungsfreiheit allerdings stark von der physischen Umgebung beeinträchtigt. Laut LaViola

(Bowman, Kruijff, LaViola Jr., & Poupyrev, 2005) wird die Bewegung durch den virtuellen Raum in „Laufen“, „Steuern“, „Auswahlbasierte Bewegung“ und „Manipulationsbasierte Bewegung“ unterteilt.

Es gibt verschiedene Variationen des Laufens. Die natürlichste ist das Laufen im echten Raum, das dann in den virtuellen Raum übertragen wird. Dafür werden lediglich Sensoren des VR-Systems benötigt. Jedoch ist man dabei durch die physische Umgebung limitiert. Zudem gibt es die Möglichkeit, in der die Person die Bewegung des Laufens ausführt, sich aber nicht im Raum bewegt. Dabei wird spezielle Hardware, wie zum Beispiel eine reibungsarme Oberfläche mit Sensoren wie in *Abbildung 7* dargestellt, benötigt.

Abbildung 7: Eine reibungsarme Oberfläche zur virtuellen Fortbewegung Quelle: (Bowman, Kruijff, LaViola Jr., & Poupyrev, 2005).

Die am häufigsten verwendete Methode ist die des Steuerns. Dabei kann der Nutzer die Bewegungsrichtung selbst bestimmen. Der Nutzer gibt dabei entweder eine absolute Richtung, zum Beispiel entlang der Z-Achse, oder eine relative, zum Beispiel nach links, an. So kann mit einem Joystick die Bewegung relativ zur Ausrichtung der Kamera gestartet und gestoppt werden. Dabei bleibt die Person im physischen Raum stehen.

Es gibt auch die Möglichkeit einer auswahlbasierten Bewegung. Dabei gibt man dem Nutzer die Möglichkeit ein Ziel oder einen Weg auszuwählen, an den dann im virtuellen Raum navigiert werden kann. Das vereinfacht die Bewegung, da der Nutzer nicht darüber nachdenken muss, wie er dorthin kommt. Diese Bewegungen sind zwar nicht immer die natürlichsten, doch sie sind in der Regel einfach zu verstehen.

Ein Beispiel dafür wäre, dass ein Nutzer auf einer Karte eine Position auswählen kann und dann automatisch dorthin transportiert wird.

Neben der Translation, also der Änderung der Position, ist auch die Änderung der Ausrichtung ein wesentlicher Bestandteil der Navigation im virtuellen Raum. Die meisten VR-Brillen verfügen über genügend Sensoren, um die Rotation des Kopfes vollständig zu erfassen und in der virtuellen Umgebung zu replizieren. Das ist der natürlichste Weg und führt laut (Bowman, Kruijff, LaViola Jr., & Poupyrev, 2005) auch zur besseren räumlichen Orientierung.

2.5 Analyse bestehender virtueller Showroom Konfiguratoren

2.5.1 Der Konfigurator

Da die Thematik keinesfalls neu ist, gibt es bereits verschiedene Lösungen auf dem Markt. Ziel ist es, durch eine Recherche dieser Produkte, Ansätze und Ideen zu identifizieren, die auch auf den Virtual Showroom Creator angewendet werden können. Dabei wurde für jedes dieser Produkte versucht, die im *Customer Journey* beschriebenen Aktivitäten durchzuführen. Diese Abläufe wurden aufgenommen und ausgewertet. Der Fokus lag darauf, ob und wie diese Produkte die Anforderungen des Kapitels *Anforderungen* erfüllen. Somit ist es möglich, zu evaluieren, ob nicht bereits ein Produkt mit dem gewünschten Funktionsumfang existiert und andererseits einige Ansätze zu erfassen, mit denen eine Lösung entworfen und entwickelt werden kann. Es wurden 5 Produkte getestet. Dabei wurden Virtuloc, eine VR-Lösung für Geschäfts- und Weiterbildungszwecke (Virtuloc, s.r.o., 2022), Shapespark, eine VR-Lösung für Immobilien, Architektur und Büroräume (Shapespark sp. z o.o., 2022) sowie HEGIAS ebenfalls für Architekten (HEGIAS AG, 2022) untersucht. Zudem wurden FrameVR (Virbela, 2022) und VRdirect für virtuelle Touren durch Inneneinrichtungen (VRdirect GmbH, 2022) untersucht.

Nicht alle untersuchten Produkte konnten im Browser ausgeführt werden. Shapespark benötigte eine 3D-Modellierungs-Software, wie zum Beispiel Blender mit einem Plugin. Wobei die meisten die Konfiguration und das Betreten mit der VR-Brille durch den Browser lösten. HEGIAS löste das Betreten in der VR-Brille durch eine eigene App, die im Appstore des entsprechenden Anbieters heruntergeladen werden musste. Dabei können zwar eigene Features eingebaut werden und die Abhängigkeit zu externen Produkten ist geringer, dafür muss diese für verschiedene Systeme selbst entwickelt werden.

Bei der Ansicht aller Showrooms wurden zwei Ansätze angewendet. Einerseits wurden die Showrooms in einer Liste dargestellt, andererseits mit Kacheln. Der Vorteil der Kacheln war es, dass auf der Kachel eine Vorschau der Szene wie in *Abbildung 8* abgebildet ist. Dies ermöglicht einfacheres Erkennen und Finden eines Showrooms.

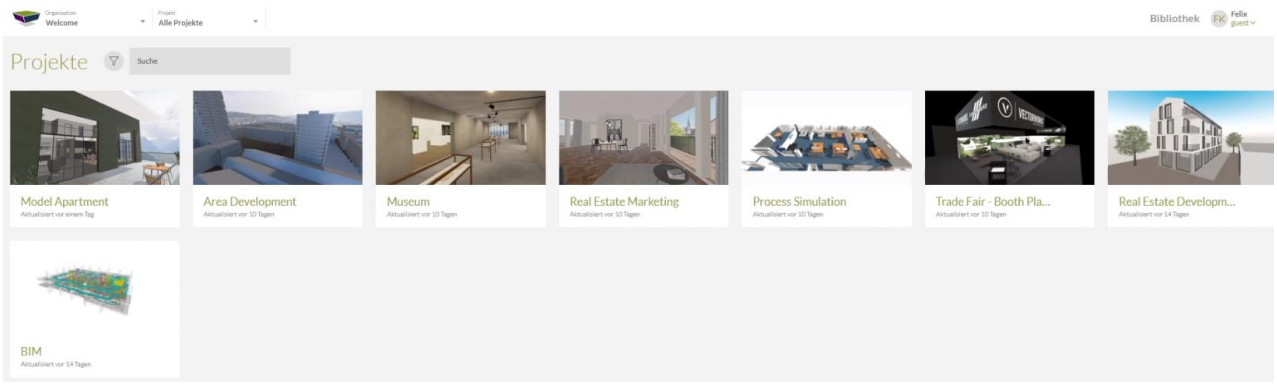


Abbildung 8: Ansicht der Showräume in HEGIAS.

Beim Erstellen eines neuen Showrooms erscheint ein neues Fenster, in dem der Name, eine Beschreibung und ein vorgefertigter Raum, wie in *Abbildung 9* dargestellt, ausgewählt werden kann. Der Raum ist dabei einer von mehreren vorgefertigten Räumen. Diese konnte man nicht bearbeiten, sondern nur die Elemente, die man in den Raum einfügt.

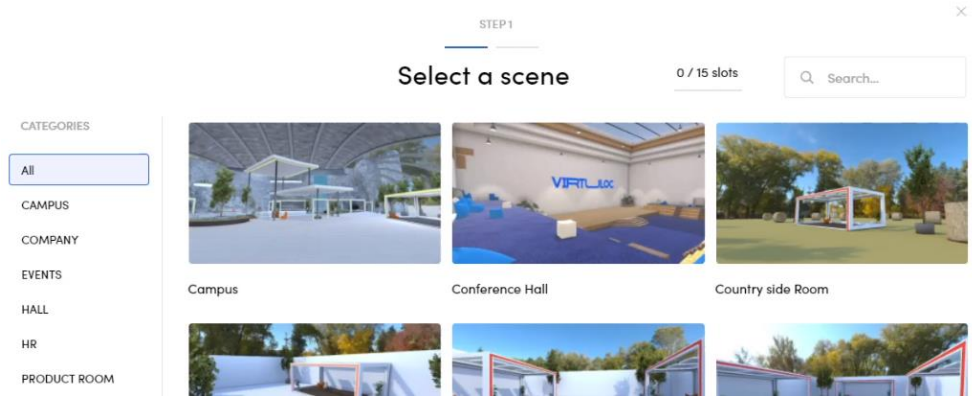


Abbildung 9: Auswahl aus vorgegebenen Räumen zum Erstellen eines neuen Showrooms in Virtuloc.

Eine Möglichkeit zu Darstellung verschiedener Aktionen ist ein Overlay, wie in *Abbildung 10* dargestellt, über der Kachel eines ausgewählten Showrooms. Dadurch können dem Nutzer direkt weitere Optionen angezeigt werden, ohne dabei den Bildschirm mit Optionen zu überhäufen, die der Nutzer aktuell nicht benötigt.

Abbildung 10: Eine Kachel mit den Interaktionen mit einer Showroom-Kachel in Virtuloc.

Um ein neues Objekt in den Showroom einzufügen, gab es mehrere Ansätze. Einer dieser Ansätze war es, dass Objekte über ein Inventar gehandhabt werden. Im Konfigurator wurde auf der rechten Seite das Inventar, wie in *Abbildung 11* dargestellt, mit allen Elementen angezeigt. Diese konnten dann mit der Maus in die Szene hineingezogen werden.

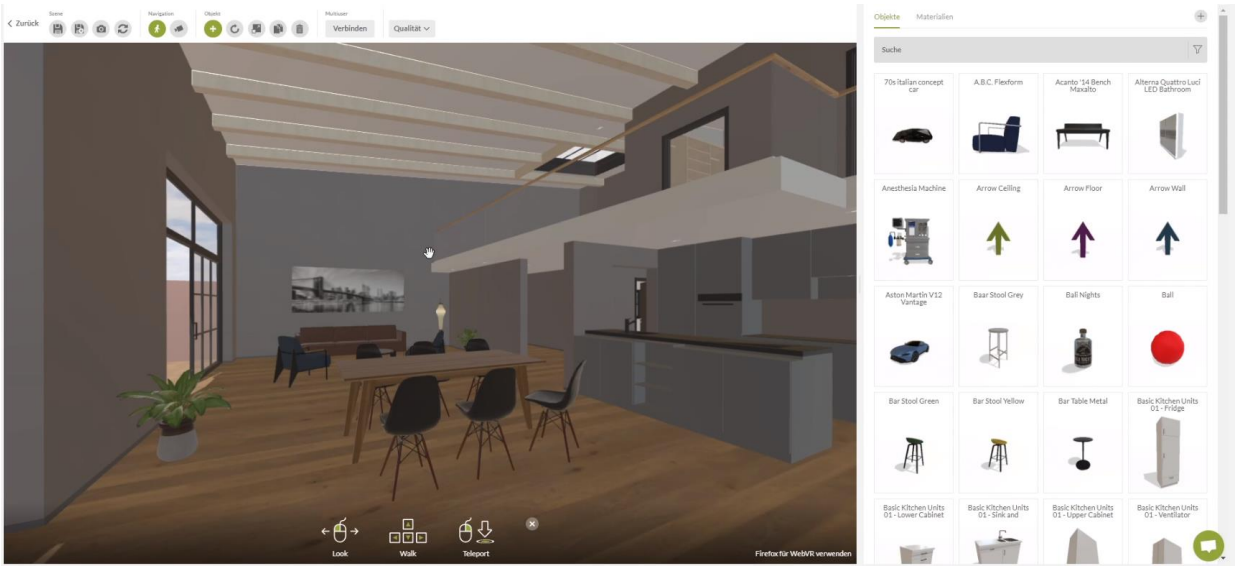


Abbildung 11: Der Showroom Konfigurator mit einem Inventar auf der rechten Seite in HEGIAS.

Eine andere Möglichkeit war, dass man die Datei direkt von ausserhalb des Browsers per Drag & Drop mit der Maus in die Szene ziehen konnte. Dieses Objekt wurde dann in die Szene geladen und konnte frei bewegt werden. Wie diese Bewegung der Objekte bewerkstelligt wird, wird später im Text genauer erläutert.

Das Wechseln von Produktvariationen, wie zum Beispiel dem Material, unterstützen nicht alle untersuchten Konfiguratoren. Ein Ansatz ist, dass Material ähnlich wie Modelle hochgeladen und dem Inventar hinzugefügt werden kann. Im Konfigurator unter dem Inventar im Reiter «Materialien» werden diese als Assets dargestellt. Das Wechseln geschieht dabei ebenfalls per Drag & Drop. Man kann dabei nicht bestimmen, welche Materialen vom Betrachter auf welche Objekte platziert werden können. Der Betrachter kann jedes Material auf jedes Objekt anwenden.

Shapespark nutzt dabei Komponenten. Ein Objekt kann angewählt werden und im Reiter rechts ein «Material-Picker», wie in *Abbildung 12* dargestellt, hinzugefügt werden. Darunter können dann mehrere in der Szene vorhandenen Materialen ausgewählt werden.

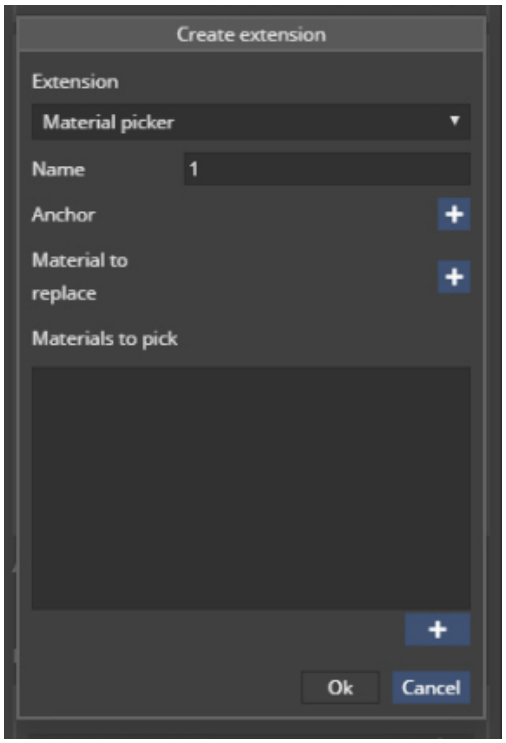


Abbildung 12: Der Material Picker in Shapespark, der ermöglicht, dass einem Objekt mehrere Materialen zugewiesen werden können, zwischen denen im VR-Showroom gewechselt werden kann.

Die Platzierung der Objekte im Raum wurde durchgehend gleich gelöst. Dabei gab es, wie in *Abbildung 13* dargestellt, zwei Möglichkeiten zur Manipulation eines Objektes, sobald man es mit der Maus ausgewählt hat. Die Eingabe der Werte erfolgt durch ein Feld.

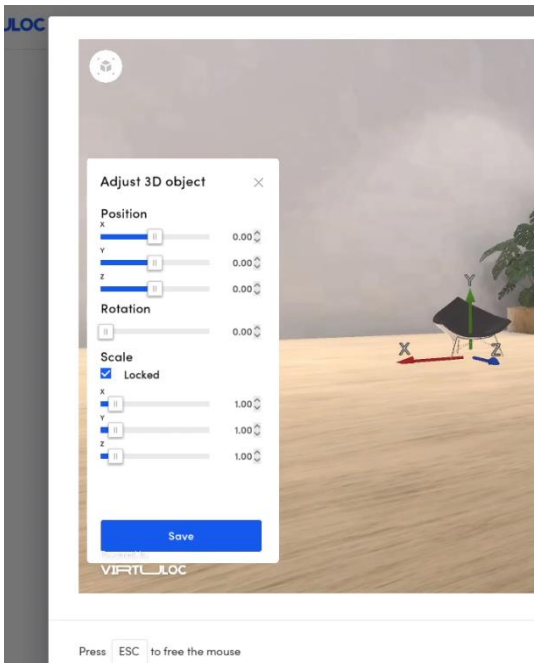


Abbildung 13: Die Anpassung der Position, Ausrichtung und Grösse in Virtuloc. Dabei gibt es neben den Eingaben unten rechts im Bild auch noch die Bewegung durch 3D-Gizmos.

Zudem wurde die Manipulation mit einem 3D-Gizmo ermöglicht. Ein 3D-Gizmo ist ein Werkzeug, das 3D Manipulationen eines Objektes erlaubt. Die Maus kann sich nur auf einer Fläche (2 Richtungen) bewegen. Allerdings können Objekte in einer dreidimensionalen Welt in drei Richtungen bewegt werden. Dabei muss die Eingabe des Nutzers so umgerechnet werden, dass eine Bewegung auf einer Fläche eine Bewegung im Raum wird. So kann man den Modus, also Bewegen, Drehen, Vergrössern wählen und mit der Maus mit dem auf dem Objekt dargestellten Gizmo interagieren. Ein, wie in *Abbildung 14* dargestelltes Gizmo, wird auch in vielen 3D-Applikationen wie Blender, Unity oder Autodesk eingesetzt.

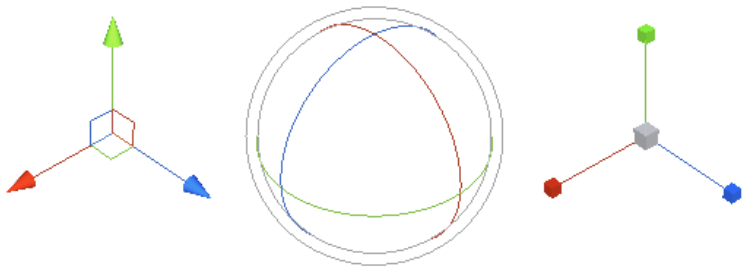


Abbildung 14: Die drei Formen eines 3D-Gizmos zur Manipulation der Position (links), Rotation (Mitte) und Skalierung (rechts).

Keines der untersuchten Produkte bot irgendeine Art von Animationsmöglichkeiten an. Höchstens importierte Modelle, die bereits eine Animation hatten, konnten in einigen Fällen abgespielt werden.

Das Betreten von mehreren Personen gleichzeitig war mehrheitlich identisch. Dabei konnte der Raum mit einem Link betreten werden. Alle Personen im Raum waren dabei durch einen Avatar zu sehen. Bei den Produkten, die über einen Materialwechsel verfügten, war dieser global, sprich jede Person sah dasselbe.

2.5.2 Das Betrachten des Showrooms in VR

Als Käufer möchte man sich frei im Showroom bewegen können. Dabei möchte man sich seine Position anpassen und herumschauen können. Für die Bewegung wurden zwei Ansätze angewandt. Einerseits konnte man sich mit den Sticks auf den Kontrollern bewegen. Die andere Methode war durch Teleport. Dabei muss die Person mit dem Kontroller auf die gewünschte Position auf den Boden zielen und eine entsprechende Taste betätigen. Beide Ansätze benötigen Kontroller und die Kenntnis der Person über diese. Da viele Personen bei der Bewegung im VR-Raum oft schnell Übelkeit (Thompson, 27) empfinden, gilt es hier, eine möglichst angenehme Variante zu finden. Das Herumschauen ist mit einer VR-Brille allerdings wesentlich simpler. Dabei kann einfach der Kopf so gedreht werden, wie es auch in der Realität nötig wäre, um ein Objekt zu sehen.

Um dem Betrachter zu erlauben, verschiedene Materialien auszuwählen, gab es zwei Ansätze. Der eine ist das Auswählen eines Icons oberhalb des gewünschten Objekts. Das Auswählen geschieht durch Zielen mit dem Kontroller. Dieses Zielen erlaubt es mit Hilfe der Raycasting-Methode auch Objekte auszuwählen, die ausserhalb der Reichweite sind (Argelaguet & Andujar, 2013). Diese Methode wird nicht nur häufig angewendet, sondern führt oft auch zu einer effizienteren Auswahl von Objekten (Bowman, Kruijff, LaViola Jr., & Poupyrev, 2005). Der Kontroller wird dabei in der Brille dargestellt. Zudem wird ein Strahl angezeigt, der zeigt, worauf der Kontroller gerichtet ist. Sobald der Strahl auf das Symbol trifft,

kann durch einen Tastendruck das Material gewechselt werden. Dabei kann die Person, wie in *Abbildung 15* dargestellt, zwischen den dargestellten Materialen entscheiden.

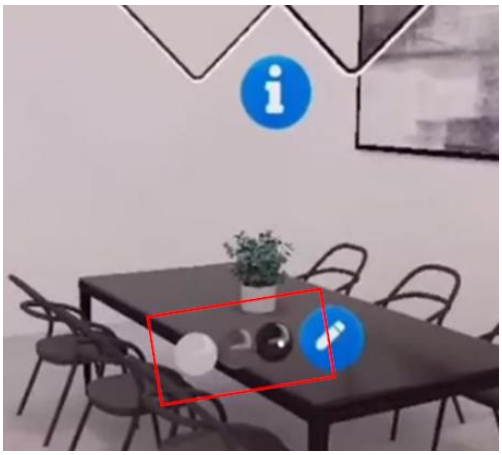


Abbildung 15: Die Darstellung der verschiedenen Materialen eines Objekts im VR-Showroom in Shapespark.

Beim zweiten Ansatz, wie in *Abbildung 16* sichtbar, hatte die Person zu jederzeit ein Menü auf ihrem linken Controller angezeigt. Die Person kann dann mit dem rechten Controller durch dieses Menü navigieren und ein Material auswählen. Nun ist es möglich dieses Material in die Szene und auf ein Objekt zu ziehen, auf das das Material angewendet werden soll. Dabei gibt es keine Einschränkungen, es ist möglich ein Steinmaterial auf ein Sofa zu platzieren oder umgekehrt.

Abbildung 16: Das Menü auf dem linken Controller im VR-Showroom in HEGIAS. Mit diesem Menü lassen sich Materialien auf Objekte und Oberflächen setzten, aber auch weitere Funktionen wie Chatoptionen werden dafür genutzt.

2.6 Technische Recherche

Um eine Softwarelösung zu entwickeln, gibt es viele Möglichkeiten. Neben einer von Grund auf eigenen Entwicklung „from Scratch“, über Bibliotheken, bis hin zu Frameworks gilt es, eine geeignete Lösung zu finden. Dieses Kapitel beinhaltet die Recherche unterschiedlicher Komponenten, die genutzt werden können, um einen Virtual Showroom Creator zu entwickeln. Dabei wurden eine Spiel-Engine wie Unity aber auch Frameworks wie A-Frame oder Babylon.js untersucht.

2.6.1 Spiel-Engine

Als Spiel Engine wurde in diesem Projekt Unity betrachtet, um herauszufinden, ob die Verwendung einer solchen Engine für die Darstellung des VR-Showrooms besser ist als das Umsetzten einer Web-Applikation. Um eine dreidimensionale Welt darzustellen, gibt es einige Möglichkeiten. Auch in Spielen geht es darum, einen oder mehrere Charaktere in einer virtuellen Welt zu manövrieren. Daher liegt die Verwendung einer Spiel-Engine, mit der auch solche Spiele produziert werden, nicht fern. Dabei gibt es verschiedene Spiele-Engines, einige davon auch ganz oder bis zu einem Punkt kostenlos. Bekannte Beispiele sind Unity, die Unreal Engine oder CryEnginge. Die Darstellung solcher dynamischen Welten, in denen sich der Nutzer bewegen kann, ist alles andere als simpel. Viele Komponenten greifen ineinander: die Darstellung einer 3D-Welt in Echtzeit, der Transfer der Daten zur Grafikkarte oder zum Hauptspeichers, die Erkennung der Eingabe diverser Geräte, die Darstellung auf unterschiedlicher Hardware, etc. Umso besser ist es, bereits bekannte und geeignete Lösungen zu verwenden. Viele dieser Engines bauen darauf auf, ohne sich um die vielen einzelnen Komponenten zu kümmern, eigene Szenarien umzusetzen. Dabei baut man eine Umgebung, die man haben möchte in einem Editor auf und kann zum Beispiel mit Verwendung von C# eigene Komponenten im Spiel programmieren. Die Engine erstellt daraus eine ausführbare Applikation. Allerdings ist die Applikation dann nicht plattformunabhängig. Das hat den Nachteil, dass für mehrere System auch mehrere Applikationen gebaut werden müssen. Dabei muss aber nicht alles erneut im Editor zusammengestellt werden. Ein Vorteil ist aber die Leistung. Dadurch, dass die Applikation für spezifische System erstellt werden kann, können Ressourcen auf einem System besser genutzt werden als bei herkömmlichen plattformunabhängigen Lösungen wie zum Beispiel eine auf Java basierende Lösung.

Die Schwierigkeit für dieses Projekt besteht darin, die Schnittstelle zwischen Web-Konfigurator und Hardware VR-Brille zu implementieren. Neben der Architektur müssen ebenfalls die Funktionalitäten implementiert werden, zum Beispiel Materialwechsel eines Modells. Wie dann von einer Datei, dem Showroom, verschiedene Materialien übermittelt werden, muss im Konzept genau definiert werden. Das

umfasst auch nur das Betrachten des Showrooms. Für das Bearbeiten muss eine andere Lösung entwickelt werden. Neben den technischen Herausforderungen muss die Applikation gleichzeitig performant und stabil laufen, im Web als auch in der Unity-Applikation. Die 3D-Modelle müssen in beiden Applikationen so identisch wie möglich dargestellt werden. Da Unity Applikationen auch für das Web entwickelt werden können (Unity Technologies, 2022), könnte eine zweite Applikation für den Showroom Konfigurator erstellt werden.

2.6.2 WebGL

WebGL (kurz für Web Graphics Library) ist eine JavaScript-API für die Darstellung von 2D- und 3D-Grafiken in jedem kompatiblen Webbrowser ohne die Verwendung von Plugins. WebGL ist vollständig in andere Webstandards integriert und ermöglicht die GPU-beschleunigte Nutzung von Physik und Bildverarbeitung. WebGL-Elemente können mit anderen HTML-Elementen gemischt und mit anderen Teilen der Seite oder dem Seitenhintergrund zusammengesetzt werden.

WebGL nutzt wie OpenGL GLSL als Shader Language. In der Computergrafik ist ein Shader ein Computerprogramm, das beim Rendern einer 3D-Szene die entsprechenden Helligkeits-, Dunkelheits- und Farbwerte berechnet, dieser Prozess wird als „Shading“ bezeichnet und wird auf der Hardware, in diesem Fall der GPU ausgeführt. Zu beachten sind ebenfalls die Positionen der Pixel, die Sättigung, der Kontrast sowie Texturen, die für das endgültige Bild von Bedeutung sind. Jeder Pixel auf dem Bildschirm hat eindeutige Koordinaten, denen dann die entsprechenden Werte zugewiesen werden. Je nach Rechenleistung und Änderungen passiert das bei 60 FPS (Frames per Second), über 60-mal pro Sekunde. Bei einer Monitorauflösung in Full HD (1920x1080) sind das bis 124’416'000 Änderungen pro Sekunde. WebGL ermöglicht die Kontrolle der gesamten Grafik im Webbrowser.

2.6.3 Three.js

Three.js ist eine browserübergreifende JavaScript-Bibliothek und API zur Erstellung und Anzeige animierter 3D-Computergrafiken in einem Webbrowser mit WebGL. Three.js nutzt WebGL zur Darstellung, reduziert dessen Komplexität allerdings. WebGL arbeitet auf einem niedrigen Abstraktionslevel. Das bedeutet konkret, dass WebGL nur Punkte, Linien und Dreiecke zeichnet und dies mit einem grossen programmiertechnischen Aufwand. Three.js hingegen übernimmt als Wrapper Funktionen wie Szenen, Lichter, Schatten, Materialien, Texturen, die ansonsten selbst vom Entwickler umgesetzt werden müssten (Three.js Fundamentals, 2022).

2.6.4 Babylon.js

Babylon.js ist eine 3D-Engine, die eine JavaScript-Library zur Darstellung von 3D-Grafiken in einem Webbrowser über HTML5 verwendet. Der Quellcode wird in TypeScript geschrieben und dann in eine JavaScript-Version kompiliert. Die JavaScript-Version steht dem Benutzer über NPM oder CDN (Content Delivery Network) zur Verfügung, der dann seine Projekte in JavaScript programmieren und auf die API der Engine zugreifen kann. Die Babylon.js 3D-Engine und der Anwendercode werden von allen Webbrowsern, die den HTML5-Standard und WebGL unterstützen, nativ interpretiert, um das 3D-Rendering durchzuführen. Zudem implementiert Babylon.js WebXR und kann dadurch auch VR-Inhalte mit gegebener VR-Ausrüstung darstellen.

Babylon.js bietet Tutorials in einem dafür programmierten Playground an. Man kann seine Playgrounds speichern, hochladen und teilen und so aus tausenden Projekten Beispiele ansehen und Inspiration sammeln. Neben dem Playground gibt es einen Editor, der von der Community geschrieben wurde und auf Github veröffentlicht wurde.

2.6.5 A-Frame

A-Frame ist ein Open-Source-Web-Framework zur Erstellung von Virtual-Reality Erlebnissen, das 2015 veröffentlicht wurde und noch heute verbessert wird. Ursprünglich wurde es vom MozVR Team entwickelt (Carpenter, 2015). Es basiert dabei auf HTML und ermöglicht somit auch für Neulinge einen einfachen Einstieg. Der Kern von A-Frame ist das Entity-Component Framework, das eine erweiterbare und komposite Struktur bietet.

Dabei unterstützt A-Frame viele VR-Headsets wie Vive, Rift, Windows Mixed Reality, Daydream, GearVR, Cardboard, Oculus Go und Quest und kann sogar für Augmented Reality Projekte verwendet werden, die eine einfache plattformunabhängige Entwicklung ermöglichen (A-Frame Introduction, 2022). Das Entwickeln mit A-Frame funktioniert mit einem einfachen Import von A-Frame durch CDN.

Szenen können in HTML definiert werden. Dabei können mit dem „a-entity“ Tag Objekte in die Szene geladen werden. Ihre Eigenschaften wie Position, Rotation, etc. werden als Attribute gespeichert. Es können verschiedene Elemente wie Modelle oder Lichtquellen definiert werden. Durch das Entity-Component System von A-Frame können verschiedene Funktionalitäten in Komponenten definiert und den Entitäten hinzugefügt werden. So kann zum Beispiel einer Entität eine „wasd-controls“ Komponente hinzugefügt werden die ab dann mit den Tasten „W“, „A“, „S“ und „D“ bewegt werden kann. Es können aber auch neue eigene Komponenten entwickelt oder aus anderen Quellen eingebunden werden. A-

Frame implementiert die Schnittstelle zwischen Browser und VR-Headset, sodass die gleiche Szene am Desktop-Browser, Mobile-Browser und auf einer VR-Brille angezeigt werden kann.

2.6.6 Speicherung der Daten

Relationales Modell

Eine relationale Datenbank ist eine Datenbank, die auf dem relationalen Datenmodell basiert. Ein System zur Verwaltung relationaler Datenbanken ist ein relational database management system (RDBMS). Viele relationale Datenbanksysteme verwenden SQL (Structured Query Language) zur Abfrage und Verwaltung der Datenbank.

Im relationalen Modell werden Daten in einer oder mehreren Tabellen mit Spalten und Zeilen organisiert, wobei jede Zeile durch einen eindeutigen Schlüssel identifiziert wird. Zeilen werden auch als Datensätze bezeichnet, Spalten werden auch als Attribute bezeichnet (siehe *Abbildung 17*).

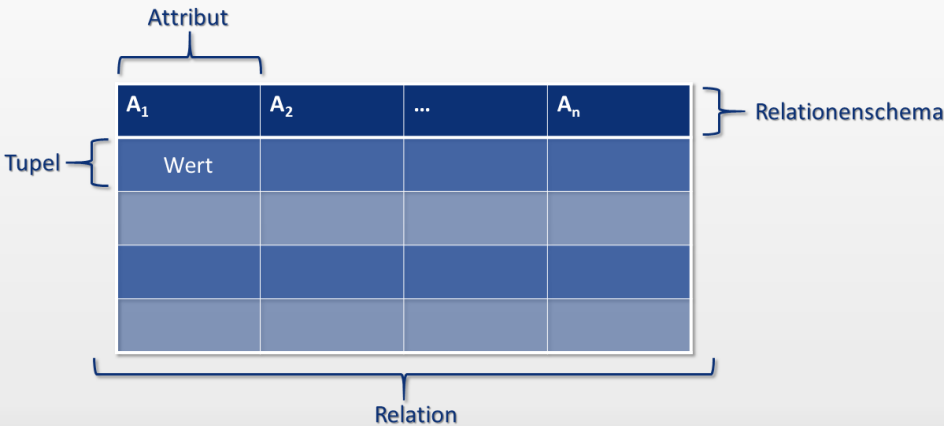


Abbildung 17: Schematische Darstellung des Relationsschema

Im Allgemeinen steht jede Tabelle für einen "Entitätstyp" (z. B. User oder Showroom). Die Zeilen stehen für Instanzen dieses Entitätstyps (z. B. "Kerkmann" oder "Showroom XYZ") und die Spalten für die dieser Instanz zugeordneten Werte (z. B. E-Mail-Adresse oder Grösse).

Jede Zeile in einer Tabelle hat ihren eigenen eindeutigen Schlüssel. Zeilen in einer Tabelle können mit Zeilen in anderen Tabellen verknüpft werden, indem eine Spalte für den eindeutigen Schlüssel der verknüpften Zeile hinzugefügt wird. Relationen zwischen unterschiedlichen Entitäten können one-to-many, one-to-one, many-to-one oder many-to-many sein. Beispielsweise kann ein User mehrere Showrooms haben (one-to-many).

Die Identifizierung erfolgt mittels Primary and Foreign Keys, sodass jedes Objekt in der Datenbank einen Unique Identifier hat und in allen Tabellen gefunden werden kann.

NoSQL

Eine NoSQL-Datenbank (Not only SQL) bietet einen Mechanismus für die Speicherung und den Abruf von Daten, die auf andere Weise modelliert sind als die in relationalen Datenbanken verwendeten tabellarischen Beziehungen. SQL-ähnliche Abfragesprachen werden unterstützt und neben SQL-Datenbanken können NoSQL Datenbanken in mehrsprachigen-persistenten Architekturen eingesetzt werden.

Zu den Beweggründen für diesen Ansatz gehören die Einfachheit des Designs, die einfachere horizontale Skalierung auf Cluster von Rechnern (was bei relationalen Datenbanken ein Problem darstellt) und eine feinere Kontrolle über die Verfügbarkeit. Die von NoSQL-Datenbanken verwendeten Datenstrukturen (z. B. Schlüssel-Wert-Paare, Graphen oder Dokumente) unterscheiden sich von denen, die standardmäßig in relationalen Datenbanken verwendet werden, wodurch einige Operationen in NoSQL schneller sind. Die besondere Eignung einer bestimmten NoSQL-Datenbank hängt von dem Problem ab, das zu lösen ist. Viele NoSQL-Speicher gehen Kompromisse bei der Konsistenz (im Sinne des CAP-Theorems) zu Gunsten von Verfügbarkeit, Partitionstoleranz und Geschwindigkeit ein (MongoDB Inc, 2022).

MongoDB

MongoDB ist ein quelloffenes, plattformübergreifendes, dokumentenorientiertes Datenbankprogramm. Als NoSQL-Datenbankprogramm eingestuft, verwendet MongoDB JSON-ähnliche Dokumente mit optionalen Schemas. MongoDB implementiert das ACID (Atomizität, Konsistenz, Isolation und Dauerhaftigkeit) Konzept.

MongoDB ist eine dokumentbasierte Datenbank, eine Unterkategorie von einem Key-Value Store. Jede Datenbank hat Collections, die Daten als Dokumente speichern. Jedes Dokument hat dabei eine eindeutige ID. Alle Daten zu einem Objekt werden in einer einzigen Instanz in der Datenbank gespeichert. Jedes Objekt kann sich von den anderen unterscheiden.

MongoDB unterstützt Feld-, Bereichsabfragen und Suchen mit Regular-Expressions. Abfragen können bestimmte Felder von Dokumenten zurückgeben und auch benutzerdefinierte JavaScript-Funktionen enthalten. MongoDB kann auf mehreren Servern betrieben werden, um die Last auszugleichen oder Daten zu duplizieren, damit das System auch bei einem Hardware-Ausfall weiterläuft. Über GridFS kann MongoDB auch als Dateisystem verwendet werden zum Speichern von Daten über mehrere Rechner hinweg (maximal 16 MB pro Datei) (MongoDB Inc, 2022).

MongoDB bietet mehrere unterschiedliche Module mit APIs für Node.js und andere Architekturen an, die das Benutzen und Verwalten der Datenbank vereinfacht.

2.6.7 Asset Speicher

Das Dateimanagement der 3D-Modelle ist wie die Benutzerverwaltung nach der E-Mail-Adresse aufgebaut. Für jeden Benutzer, der sich mit seiner E-Mail-Adresse und einem Passwort neu registriert, wird ein neuer Ordner mit der angegebenen E-Mail-Adresse erstellt. Da die E-Mail-Adresse einzigartig ist, kann es nicht zwei identische E-Mail-Adressen geben und es kommt zu keinen Konflikten untereinander. Wie in *Abbildung 18* dargestellt, hat jeder Benutzer in diesem Ordner sein Inventar an 3D-Modellen, in das er neue Modelle hochladen kann.

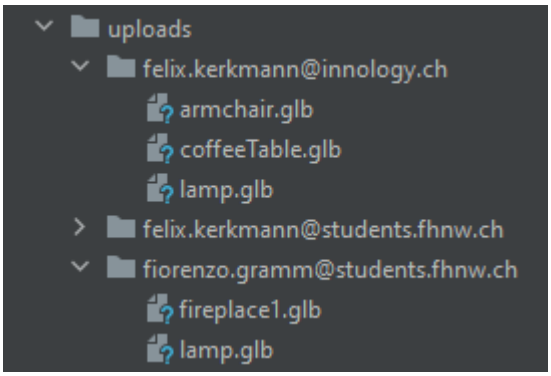


Abbildung 18: Screenshot des Verzeichnisses für die Datenablage der Modelle.

2.6.8 Nodejs

Node.js ist eine open-source, plattformübergreifende Back-End-JavaScript-Laufzeitumgebung, die auf der V8-Engine läuft und JavaScript-Code außerhalb eines Webbrowsers ausführt. Mit Node.js können Entwickler JavaScript zum Schreiben von Command-Line-Tools und für das serverseitige Scripting verwenden. Dabei werden Skripte serverseitig ausgeführt, um dynamische Webseiteninhalte zu erzeugen, bevor die Seite an den Webbrowser des Benutzers gesendet wird. Folglich stellt Node.js ein "JavaScript everywhere"-Paradigma dar, das die Entwicklung von Webanwendungen um eine einzige Programmiersprache herum vereinheitlicht, anstatt verschiedene Sprachen für serverseitige und clientseitige Skripte zu verwenden (OpenJS Foundation, 2022).

Node.js hat eine ereignisgesteuerte Architektur, die asynchrone I/O ermöglicht. Diese Design-Entscheidungen zielen darauf ab, den Durchsatz und die Skalierbarkeit in Webanwendungen mit vielen Eingabe- und Ausgabeoperationen sowie für Echtzeit-Webanwendungen zu optimieren (OpenJS Foundation, 2022).

Node.js ermöglicht die Erstellung von Webservern und Netzwerktools mithilfe von JavaScript und einer Sammlung von "Modulen", die verschiedene Kernfunktionen behandeln. Es werden Module für Dateisystem-I/O, Netzwerke (DNS, HTTP, TCP, TLS/SSL oder UDP) und andere Kernfunktionen bereitgestellt. Die Module von Node.js verwenden eine API, die die Komplexität beim Schreiben von Serveranwendungen verringern soll.

Nebenbei haben einige Webhosting Anbieter Node.js kompatible Plugins, die das Deployment der Applikation vereinfachen. Die Innology betreibt ihre Webseite über Novatrend, einem Webhosting Anbieter mit einem WHM Server und CPanel als Konfigurations-Tool.

3 Praktischer Teil

Im folgenden Abschnitt wird auf die praktische Umsetzung der im theoretischen Teil evaluierten Technologien und Konzepte eingegangen. Es wird die Architektur beschrieben und gezeigt, wie die Komponenten des Virtual Showroom Configurators zusammenhängen. In dem folgenden Kapitel wird dann auf die Umsetzung dieser Komponenten eingegangen.

3.1 Konzept

3.1.1 Konfigurator

Der Konfigurator ist die Komponente, mit der der Benutzer in seinem Browser einen Showroom erstellen und konfigurieren kann. Dabei soll der Konfigurator intuitiv bedienbar sein und sich an anderen Web-Applikationen orientieren, um die Nutzung für den Benutzer möglichst einfach zu gestalten. In *Abbildung 19* wird die Startansicht nach dem Login des Benutzers angezeigt. Der Benutzer hat die Möglichkeit, die Räume zu löschen, einen neuen Raum zu erstellen und den Raum zu bearbeiten. Auf der rechten Seite sieht er sein Inventar mit allen bereits hochgeladenen 3D-Modellen und kann dort weitere hinzufügen.

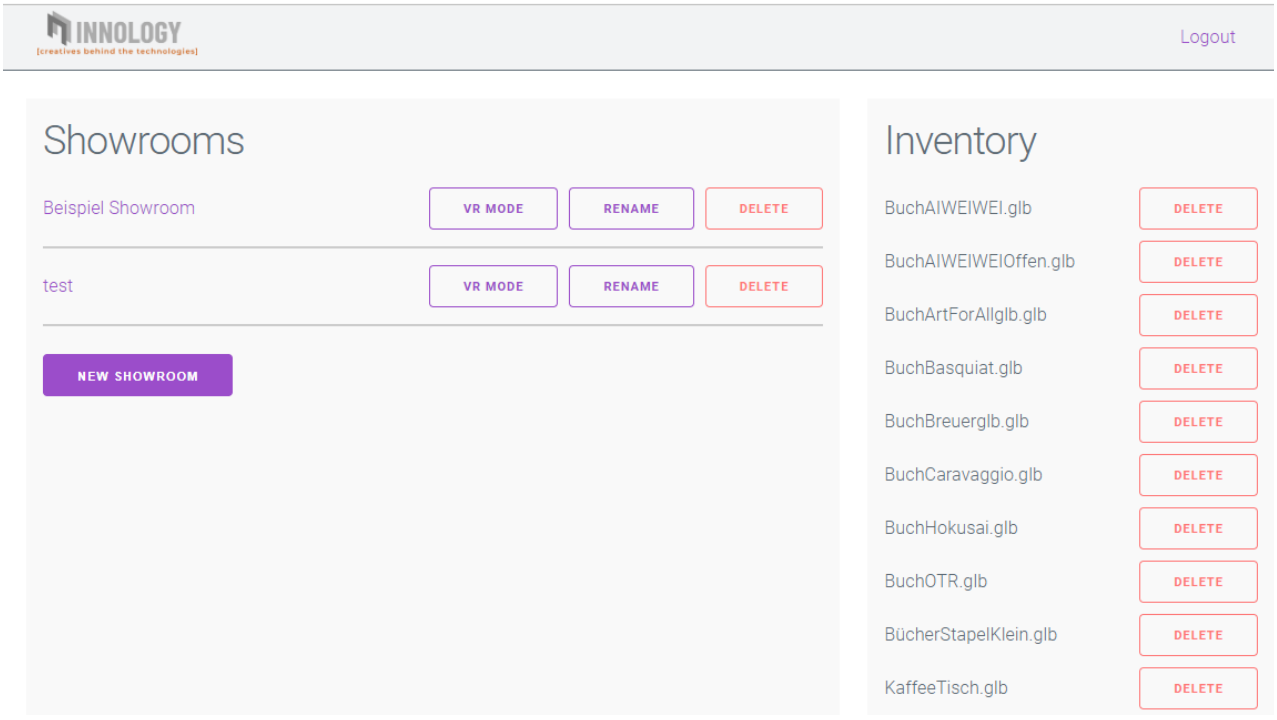


Abbildung 19: Die Startansicht zeigt die Liste aller Showräume und das Inventar mit allen Objekten, die dem Nutzer auf der Plattform zur Verfügung stehen.

Die Ansicht im Bearbeitungsmodus, die in *Abbildung 20* dargestellt ist, zeigt den Showroom mit allen Objekten. Dabei ist die Kamera so ausgerichtet, als ob die Person in diesem Raum steht.

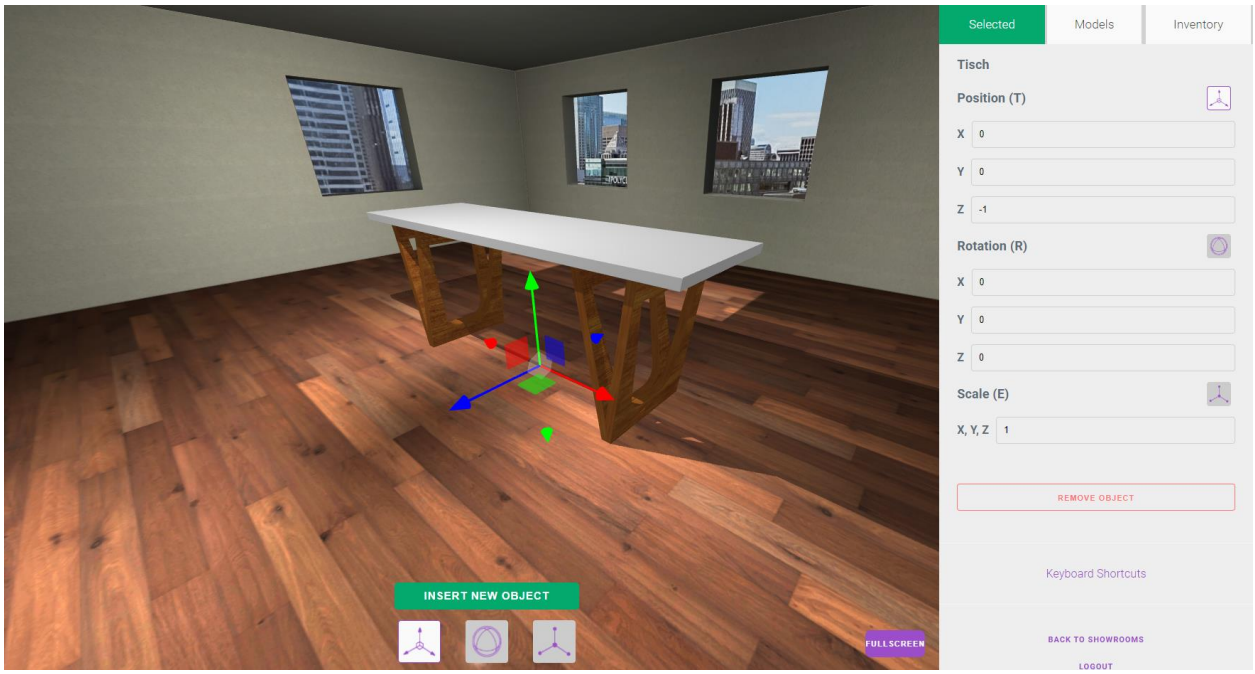


Abbildung 20: Die Bearbeitungsansicht in einem Showroom.

Der Benutzer kann sich mit seiner Maus und Tastatur in der Szene bewegen. Mit der Maus soll er zudem in der Lage sein, Objekte auszuwählen, die er dann bearbeiten kann. Auf der rechten Seite des Fensters ist eine Ansicht mit den Eigenschaften wie dem Namen, der Position oder der Grösse vom aktuell ausgewählten Objekt. So hat der Nutzer immer die Informationen über das Objekt und kann sich frei in der Szene bewegen. Sobald er ein Objekt ausgewählt hat, soll er über Eingabefelder in der rechten Ansicht oder über ein direkt am Objekt angezeigtes 3D-Gizmo das Objekt bewegen können.

3.1.2 Figma

Um bereits früh ein Feedback zu erhalten, wurden durchführbare Prototypen in Figma modelliert. Diese Prototypen beziehen sich auf die Prozesse im Konfigurator, also die Konfiguration im Browser. 3D-Prototypen, beziehungsweise VR-Prototypen können in Figma nicht erstellt werden. Für die Prototypen wurden einzelne Prozesse, wie zum Beispiel das Einfügen eines neuen Modells, in einzelne Ansichten aufgeteilt und durch interaktive Elemente so verknüpft, dass eine Person diesen Prozess ausführen kann. Dabei ist der ganze Ablauf rein visuell. Ziel war es, ein User Interface zu gestalten, das einfach zu bedienen sein sollte. Anhand solcher Prototypen ist es möglich, bereits früh Feedback von Personen mit unterschiedlichen Hintergründen, Erfahrungen und Kenntnissen zu erhalten. Der Vorteil eines Low Fidelity

Prototypen ist, dass man nicht alles vorher entwickelt haben muss, bis man solche Tests ausführen kann. Denn das Entwerfen eines wie in *Abbildung 21* gezeigten Figma-Prototyps geht wesentlich schneller als das Bauen eines komplett neuen User Interfaces im Front End.

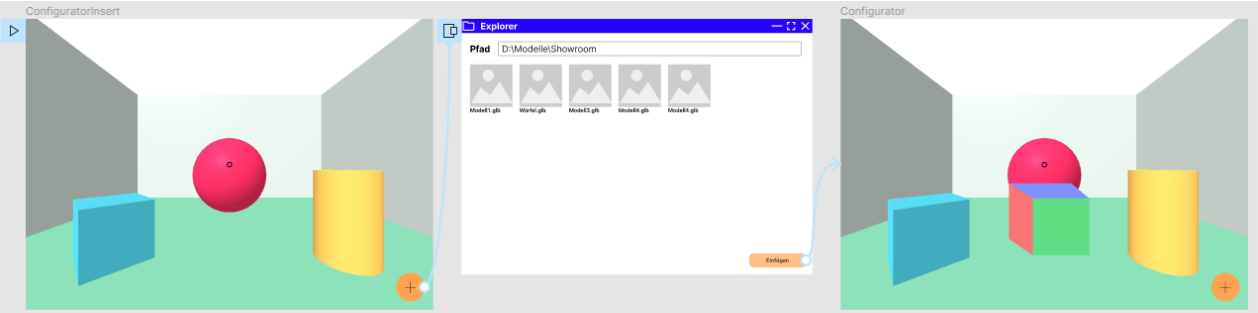


Abbildung 21: Figma Workflow von Einfügen eines neuen Objekts in die Szene.

3.1.3 Usability Testing

In der Mitte des Projektes wurde ein Usability Test mit drei unterschiedlichen Probanden unterschiedlicher Verkaufs- und VR-Erfahrung durchgeführt. Getestet wurde der Prototyp zum Web-Konfigurator und der Prototyp zum VR Showroom. Die Tester wurden über das Projekt aufgeklärt und in die Soft- und Hardware eingeführt. Der Prototyp des Konfigurators war zu diesem Zeitpunkt noch auf Figma. Der VR Showroom bestand aus einem einfachen Raum mit mehreren 3D-Modellen, in dem sich die Benutzer mit den Controllern teleportieren konnten. Der gesamte Usability Test wurde gefilmt.

Die Probanden sind einmal den Konfigurator durchgegangen und mussten Aufgaben lösen wie zum Beispiel „Erstelle einen neuen Showroom mit Namen Showroom 2“. Danach konnten sie in den VR Raum, sich darin frei bewegen und die vorhandenen 3D-Modelle betrachten. Anschliessend gab es jeweils ein Interview zum Konfigurator, sowie zum VR Showroom. Die Resultate des Usability Tests gehen aus der Auswertung der Fragen, sowie der Auswertung des Videomaterials des gesamten Usability Tests hervor.

Der Konfigurator hat die wichtigsten Funktionalitäten abgedeckt, die man als Verkäufer haben wollte: Erstellen eines neuen Showrooms, hochladen eigener 3D-Modelle sowie die Manipulation dieser. Es fehlte jedoch an Features, die das Produkt ansprechender machen. Raumgrösse, Tageszeit und Wetter wurden sich als editierbare Features gewünscht. Der Konfigurator hat seinen Zweck erfüllt und war benutzerfreundlich.

Im VR haben die Interaktionen mit den Controllern (Teleport) gut funktioniert und waren intuitiv. Der Massstab war richtig, aber die Materialien und Farben der 3D Modelle wurden unterschiedlich wahrge-

nommen. Für den einen war es realistischer als für den anderen. Jeder hat eine unterschiedliche Wahrnehmung daher kann ein Produkt nie komplett gleich für mehrere Leute sein. Hierbei ging es ebenfalls um Details, sodass dieser Punkt in der weiteren Entwicklung nicht priorisiert wurde. Niemand hatte währenddessen oder danach das Gefühl von Übelkeit oder Schwindel und es haben sich alle schnell an die virtuelle Umgebung gewöhnt. Die ausgewählten Technologien eignen sich somit für die weitere Umsetzung des Produktes.

3.1.4 Betrachten in VR

Das Betrachten des Showrooms mit einer VR-Brille ist die zweite fundamentale Komponente. Die Interaktionen in VR sind entweder mittels Handtracking durch die VR-Brille oder mittels Controller möglich. Controller sind im Vergleich zu Hand-Tracking bei mehreren VR-Ausrüstungen enthalten. Da Hand-Tracking vor allem für Menschen ohne Erfahrung mit VR-Kontrollern das Potenzial hat, einfacher zu sein, ist das Hand-Tracking durchaus eine valide Option, die weiter untersucht oder umgesetzt werden könnte. Controller sind auf Programmier-Ebene einfacher umzusetzen und nicht so fehleranfällig wie das Hand-Tracking, da ein Knopf entweder gedrückt wird oder nicht. Bei den Handbewegungen muss individuell ausgewertet werden, ob die Geste ein Ereignis auslöst oder nicht. Für die Bewegung in VR ist eine Interaktionsmöglichkeit in diesem Projekt unabdingbar, sodass die Applikation mit Controllern umgesetzt wurde.

Wie im Kapitel *Navigation im virtuellen Raum* beschrieben, ist Laufen die natürlichste Bewegungsart und dadurch eine nützliche Technik. Jedoch ist es das Ziel, möglichst unabhängig vom physischen Raum zu sein. Deshalb wurde die Bewegung, wie in *Abbildung 22* dargestellt, mit einem Teleport ergänzt. Somit kann sich der Nutzer über Teleport frei bewegen und hat die Möglichkeit sich auch physisch im Raum zu bewegen. Die einzige Gefahr dabei ist, dass die Person in ein Objekt oder in eine Wand läuft. Brillen, wie zum Beispiel die Oculus Quest 2, erkennen die Umgebung und warnen den Nutzer vor einer Kollision, indem sie die virtuelle Umgebung temporär ausblenden und die, durch die Kameras erfasste reale Umgebung anzeigen. Der Nutzer kann seinen Kopf bewegen und die Drehung wird exakt in der virtuellen Welt widergespiegelt. Neben dieser natürlichen Form der Rotation soll der Nutzer mit seinem rechten Joystick die Kamera jeweils um 90° nach links beziehungsweise rechts drehen können. Somit hat er die Möglichkeit, sich einfacher im Raum umzusehen.

Abbildung 22: Darstellung des Teleports aus Sicht des Benutzers.

Aufgrund der Usability Tests und der eigenen Erfahrung wurde der Teleport als Parabel umgesetzt. Sobald der Nutzer den VR Showroom betritt, wird über den Controllern der Text „Press X for Teleport“ bzw. „Press A for Teleport“ angezeigt. Durch das Drücken und wieder loslassen der entsprechenden in *Abbildung 23* dargestellten Tasten wird der Teleport ausgeführt.

Abbildung 23: Die Knöpfe auf den Oculus Quest 2 Controllern.

Der Benutzer soll vor dem Betreten in VR vom Verkäufer persönlich in die Hard- und Software eingewiesen werden und soll jederzeit Fragen zum Benutzen der Applikation an den Verkäufer stellen können.

3.2 Architektur

Das gesamte System ist eine Webapplikation und besteht wie in *Abbildung 24* dargestellt aus einem Backend einem Frontend und einer Datenbank. Ein Client schickt eine Anfrage (1) an den Server, dieser überprüft, ob der Client schon eingeloggt ist, oder nicht. Beim ersten Aufrufen der URL schickt der Server dann die Login-Seite an den Benutzer zurück. Beim Login wird dann erneut eine Anfrage an den Server geschickt, bei dem der User in der Datenbank verifiziert werden muss (2). Je nach Datenbankeintrag, den der Server erhält (3), wird der Benutzer eingeloggt und kommt auf die Landingpage oder zurück auf die Login Page (4).

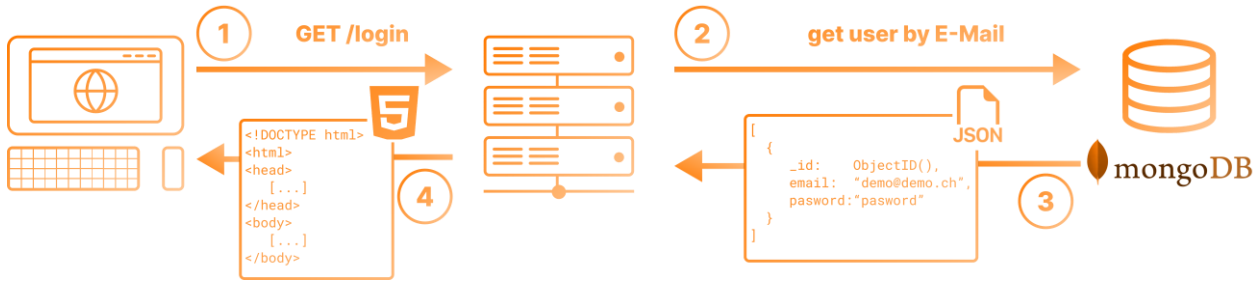


Abbildung 24: Darstellung des Logins mit Kommunikation vom Client dem Server und der Datenbank.

Anfragen die die Übersicht über die Showrooms und das Inventar, sowie Metadaten (z.B. Name des Showrooms) eines Benutzers betreffen und nicht spezifische Konfigurationen sind, werden über HTTP Anfragen bearbeitet. Das heisst, der Client sendet jedes Mal eine Anfrage an den Server, dieser bearbeitet diese und sendet eine entsprechende Antwort zurück. Die HTTP Request GET /showrooms beispielsweise sendet dem Benutzer die Seite zurück, auf der er alle seine erstellten Showrooms und sein Inventar sieht, mitsamt den möglichen Aktionen (bearbeiten, löschen, neu erstellen).

In *Abbildung 25* wird die Anfrage dargestellt, in der der Nutzer den Raum bearbeiten möchte. Der Editor Modus eines Showrooms wird ebenfalls mittels GET Request angefragt (1). Bei erfolgreicher Bearbeitung der Anfrage (2 + 3) wird zwischen Client und Server eine Socket Verbindung aufgebaut (4), ein Duplex Kanal, in dem beide Parteien kommunizieren können. Durch unterschiedliche Events werden dann Funktionen ausgelöst, auf Client sowie auf Serverseite. Jede Änderungsanfrage, beispielswiese von der X Koordinate der Position eines 3D-Modells wird nicht als separate Anfrage gesendet, sondern über den Websocket. Dadurch muss die Seite nicht neu gerendert werden und die Daten bleiben weiterhin persistent. Der Server bekommt den Änderungsauftrag und schreibt die Änderung in die Datenbank. Da der Raum und die Modelle mehrere 10 MB gross sein können, ist es performant diese nur einmal am Anfang zu laden. Grössere Änderungen, zum Beispiel das Hinzufügen eines neuen 3D-Modells erfolgen über ein

POST Request und die Seite wird neu erstellt, sodass das neue Objekt korrekt angezeigt wird und der Raum einheitlich gerendert wird.

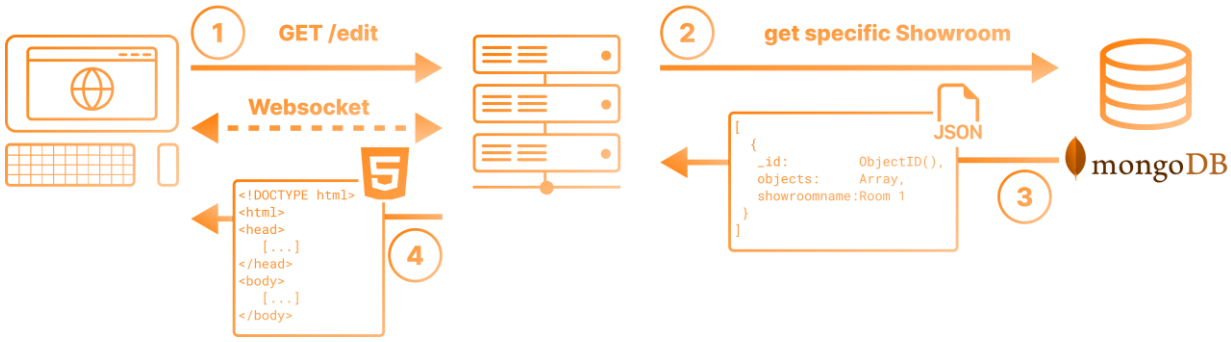


Abbildung 25: Darstellung der Anfrage einen Showroom im Konfigurator zu öffnen.

Das Betreten des Showrooms in VR, wie in *Abbildung 26* dargestellt, erfolgt ebenfalls mittel HTTP Request (1). Der Benutzer muss sich wie oben beschrieben im Browser der VR-Brille anmelden und kann dann in den VR-Modus mittels GET Request (1-4). In diesem Modus wird nur der Showroom angezeigt ohne weitere Funktionen.

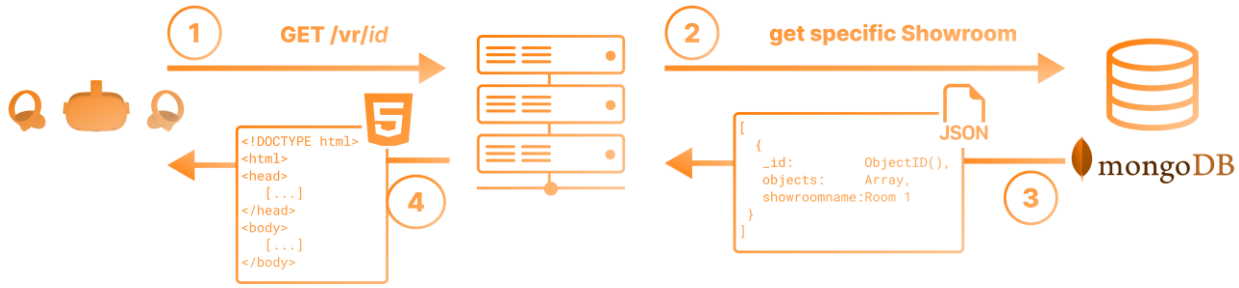


Abbildung 26: Darstellung der Anfrage in einem Showroom in VR zu betreten.

Die Architektur ist als SaaS (Software as a Service) Plattform aufgebaut. Nutzer müssen sich anmelden und können dann die Web-Applikation nutzen. Die Daten werden persistent vom Server, beziehungsweise der Datenbank gespeichert. Neben der VR-Brille und dem Webbrowser gibt es keine weiteren Abhängigkeiten.

3.3 Frontend

3.3.1 A-Frame

Statt einer Hardware spezifischen Lösungen wie Unity wurde mit A-Frame und Babylon.js eine VR kompatible Webapplikation gewählt, die das Produkt plattform- und hardwareunabhängig macht.

A-Frame oder Babylon.js vereinfacht es, die Versionen aktuell zu halten. Gäbe es in Unity ein Update, müsste man dieses Update installieren und die jeweiligen Applikationen neu erstellen und bereitstellen, vorausgesetzt das Update in Unity hätte einen wesentlichen Einfluss auf die Applikation (zum Beispiel stark verbesserte Performance). Das Aktualisieren mit A-Frame oder Babylon.js hingegen reduziert sich auf das Anpassen der Versionsnummer im Import. Für diesen Prozess wäre nicht einmal ein Neustart des Servers notwendig. Der Konfigurator sollte unabhängig von der VR Applikation im Browser laufen. Dadurch reduziert sich die Komplexität und das Endprodukt wird kompakter. Eine Umsetzung mit Unity wäre jedoch grundsätzlich möglich.

Sowohl A-Frame als auch Babylon.js sind potente Kandidaten für die Umsetzung. Beide werden laufend aktualisiert und beide wurden bereits für 3D und VR-Projekte verwendet. Beide nutzen, direkt oder indirekt, WebGL und können somit in allen herkömmlichen Browsern 3D-Inhalte darstellen. Ebenfalls bieten beide die Nutzung von WebXR an, um mit unterschiedlicher VR-Hardware zu funktionieren. A-Frames Kernfunktion ist jedoch die Darstellung von VR-Welten und bietet eine native VR Kompatibilität ohne weiteren Aufwand an. Zudem werden von A-Frame auch mehrere Dateitypen neben dem glTF unterstützt. Zusätzlich unterstützt A-Frame mit den Komponenten auch die Implementierung eigener Funktionalitäten. Da A-Frame alle Kernfunktionen beinhaltet und direkte VR-Unterstützung anbietet, wurde es Babylon.js vorgezogen.

3.3.2 Object Selector und Selectable

Da Objekte im Raum bewegt werden müssen, muss dem Nutzer eine Möglichkeit geboten werden, Objekte auszuwählen. Dafür wurde eine Komponente in A-Frame entwickelt. Für die Auswahl gibt es zwei Komponenten. Erstens der `ObjectSelector`. Das ist eine Komponente, die einerseits speichert, welches Objekt vom Nutzer ausgewählt wurde und andererseits weitere Events basierend auf der Auswahl auslöst. Die zweite Komponente ist das `Selectable`. Die Funktion des `Selectable`'s ist es, zu erkennen, wenn es angeklickt wurde und dies dem `ObjectSelector` mitzuteilen. Der `ObjectSelector` muss einem Element in der Szene als Attribut hinzugefügt werden. Es muss nicht zwingend ein eigenes Entity sein, es macht es aber einfacher es, wie in *Abbildung 27* dargestellt, in einer eigener Entität mit entsprechender ID zu platzieren.

```
<a-entity id="ObjectSelector" object-selector></a-entity>
```

Abbildung 27: Object-Selector in der A-Frame-Scene.

Sobald die Komponente das erste Mal einer Entität hinzugefügt wird, wird die `init` Funktion der Komponente ausgeführt. In dieser wird die Komponente dann initialisiert. Der `ObjectSelector` erstellt die **Transform Controls (3D-Gizmo)**, sobald ein Objekt ausgewählt wird. Diese werden wie in *Abbildung 28* dargestellt, zu Beginn initialisiert, jedoch noch keinem Objekt hinzugefügt, werden also (noch) nicht angezeigt. Die Transform Controls werden dabei von **Three.js importiert**.

```
import { TransformControls } from 'three/examples/jsm/controls/TransformControls'

AFRAME.registerComponent('objectselector', {

    init: function(){
        const camera        = this.el.sceneEl.camera;
        const renderer      = this.el.sceneEl.renderer;
        const scene         = this.el.sceneEl.object3D;
        transformControls   = new TransformControls(camera, renderer.domElement);
        scene.add(transformControls);

        this.el.addEventListener('onValueChanged',
                                 changeSelectionHandler);
       [...]
});
```

Abbildung 28: Initialisierung des Object-Selectors. Die Registrierung der Listener auf die Events sind identisch und wird daher nur durch einen dargestellt.

Nach der Initialisierung des Gizmos, werden die Listener für die Events gesetzt. Der Vorteil von Events ist, dass diese individuell, auch ausserhalb von A-Frame, zum Beispiel durch Tastatureingaben, ausgelöst werden können. So wird durch das Drücken der „R“-Taste das `changeTransformManipulation` Event ausgelöst, das im `ObjektSelector` dann den Modus des Gizmos auf Rotation wechselt. *Abbildung 29* zeigt die Funktion, die beim Auswählen eines Objekts ausgeführt wird. Die Funktion nimmt dabei immer einen Parameter **Event**, in der weitere Informationen zum Event gespeichert werden können. Bei der Auswahl eines Objektes wird das **angeklickte Objekt** mitgegeben. Beim Wechsel des Objektes wird zuerst das alte Objekt abgewählt und danach erst das neue ausgewählt.

```
function changeSelectionHandler(event) {
    const newSelection = event.detail.selectedObject;

    if(objectIsAlreadySelected(newSelection)){
        return;
    }

    deselectOldSelection();
    selectObject(newSelection);
}
```

Abbildung 29: Die Methode changeSelectionHandler welche ausgeführt wird, wenn das Event onSelectionChange ausgelöst wurde.

In `deselectOldSelection` wird die alte Auswahl entfernt und es werden, falls ein Objekt ausgewählt wurde, die TransformControls (Gizmo) entfernt. In `selectObject` wird eine Kopie des alten Objekts erstellt und danach wird die neue Auswahl in einer Variablen gespeichert. Die Kopie des alten Objekts wird benötigt, da bei einem Update eines Elements die neuen und alten Werte mitgegeben werden. Das sorgt dafür, dass eine Änderung in der Datenbank nur dann ausgeführt wird, wenn der Showroom auf dem Client auf demselben Stand ist wie der in der Datenbank. In *Tabelle 3* sind alle Events mit einer Beschreibung und dem Parameter beschrieben, auf die der `ObjectSelector` reagiert.

Name des Events	Beschreibung	Event Details

onChangeSelection	Ein neues Objekt wurde ausgewählt.	{
		newSelection: Objekt
		}
		• neues Element
		• null, auf kein Objekt ausgewählt wechseln

changeTransformanipulation	Wechselt den Modus des 3D-Gizmos.	{
		mode: Modus als String
		}
		• „translate“
		• „rotate“
		• „scale“

onValuesChange	Werte eines Objekts der aktuellen Auswahl wurden geändert.	{
		name: Name des Objekts
		[key]: Namen der Attribute
		[newValue]:Neue Werte
		}

onObjectSubmit	Das ausgewählte Objekt soll aktualisiert werden. Dabei werden nur die geänderten Werte an den Server gesendet.	{
		name: Name des Objekts
		}

onFailedUpdateValues	Ein Update eines Objekts vom Server konnte nicht ausgeführt werden. Falls das Objekt noch in der Szene ist, wird es aktualisiert	{
		name: Name des Objekts
		[keys]: Namen der Attribute
		[oldValues]: Alte Werte
		[newValues]: Neue Werte
		}

onRemoveSubmit	Ein Objekt aus der Szene entfernen.	{
		name: Name des Objekts
		}

onFailedRemove	Das Entfernen des gewünschten Objekts in der Datenbank schlug fehl und die Seite lädt neu.	{
		name: Name des Objekts
		}

Tabelle 3: Alle Events des Object-Selectors

Die Komponente `Selectable` ist hierbei wesentlich simpler aufgebaut. `Selectable` nutzt mit **schema** die Möglichkeit Daten in der Komponente direkt zu speichern. Im Vergleich zu JavaScript Variablen können diese von A-Frame direkt, zum Beispiel über `getAttribute` genutzt werden. Wie in *Abbildung 30* dargestellt, verfügt jedes Objekt, das man auswählen kann über einen Namen, der beim Laden der Szene durch ejs und json2html direkt durch den in der Datenbank gesetzt wird. Zudem benötigt jedes `Selectable` noch den `ObjectSelector`. Bei der Initialisierung wird der `ObjectSelector` über die id gesucht und gesetzt. Danach wird noch der Listener gesetzt, der ausgelöst wird, sobald das Objekt mit der Maus geklickt wird. Dabei löst die Komponente das vorher beschriebene **onChangeSelection** Event aus und übergibt sich selbst in den **Event Details** an.

```
AFRAME.registerComponent('selectable', {
    schema: {
        name:{type: 'string', default: 'No Name'},
        objectSelectorController: {default: ''},
    },
    init: function(){
        this.data.objectSelectorController =
                                        document.querySelector('#ObjectSelector');
        this.el.addEventListener('click', () => {
            console.log("Click on " + this.data.name);
            this.data.objectSelectorController.emit('onChangeSelection',
                                                    selectedObject: this.el});
        })
    }
});
```

Abbildung 30: Die A-Frame Komponente des selectable.

3.4 Backend

3.4.1 HTTP Requests

Die Verwaltung der Showrooms der Benutzer erfolgt über das HTTP Protokoll. Der Benutzer kann CRUD (Create, Read, Update, Delete) Anfragen an den Server schicken, diese werden bearbeitet und eine Antwort zurückgeschickt. Da die Daten hauptsächlich im JSON Format gespeichert und verwendet werden, sind die POST Anfragen als `application/json` codiert. Um Datenverlust zu verhindern sind die Dateiuploads von anderen Daten getrennt und als `multipart/form-data` codiert. Der Benutzer kann lediglich eine .glb Datei hochladen, diese wird im Inventar angezeigt und mit einem neuen Request kann das 3D-Modell mit einem selbst gewählten Namen einem Showroom hinzugefügt werden. Die unterschiedlichen Anfragen auf einen spezifischen Showroom oder ein spezifisches 3D-Modell erfolgen mittels ID, die in der Anfrage als Parameter mitgegeben wird.

3.4.2 Socket.io

Das WebSocket-Protokoll ermöglicht die Interaktion zwischen einem Webserver und einem Browser in einem Duplexkanal. Die Kanäle laufen über die HTTP-Ports 443 und 80 und sind somit HTTP kompatibel. Der Informationsaustausch erfolgt in Echtzeit. Die Nachrichten werden zwischen Client und Server hin und hergeleitet.

Für die Bearbeitung eines Showrooms und das Platzieren von 3D-Modellen ist das REST Prinzip ungeeignet, da bei jeder kleinsten Änderung jedes Mal eine neue Anfrage gesendet, bearbeitet und eine Antwort zurückgeschickt werden muss. Die Serverlast, sowie das ständige Neuladen, machen die Applikation langsam und benutzerunfreundlich. Daher wird beim Start der Applikation ein Websocket geöffnet. Sobald ein Benutzer in den Bearbeitungsmodus des Showrooms geht, verbindet sich der Client auf den Socket und die 3D-Modell Attribute werden über diese Verbindung an den Server geschickt und in die Datenbank übertragen. Bei einem Verbindungsabbruch sind die Daten persistent und werden einfach neu aus der Datenbank gelesen. Die Verbindung steht nur unter der Edit-URL.

3.4.3 Fehlersuche

Im Verlauf einer Web-Applikation können unterschiedliche Fehler auftreten. Diese zu finden ist nicht immer einfach. Da Client und Server miteinander kommunizieren und nicht direkt miteinander verbunden sind und auch nebenläufig funktionieren, können viele verschiedene Fehler in verschiedenen Reihenfolgen auftreten, die nicht immer einfach reproduzierbar sind. So wurden einzelne, wie in *Abbildung 31* dargestellte Debug-Statements an verschiedenen Stellen im Code eingefügt, die es erlauben, auf Knopfdruck spezifische Fehlermeldungen zu werfen.

```
// Aufruf des Debug-Statements im Code
askToThrowExceptionWhenDebuggingFlagIsSet('debugFlag', 'Error message');
```

Abbildung 31: Befehl für das Fragen zum Werfen einer Fehlermeldung, wenn ein entsprechendes Flag gesetzt ist.

Dabei blockiert die Applikation und wartet auf die Eingabe in der Konsole. Ein Beispiel wäre ein Update eines Objekts im Raum. Der Benutzer verschiebt es in seinem Browser. Dabei werden die aktuellen Werte an den Server übertragen. Ein möglicher Fehler könnte sein, dass es ungültige alte Werte enthält, also der Stand auf dem Client entspricht nicht dem der Datenbank. Dies könnte zum Beispiel auftreten, wenn zwei Updates nicht in derselben Reihenfolge beim Server ankommen, wie sie vom Nutzer ausgeführt werden. Dieser Effekt ist nicht einfach zu reproduzieren. Empfängt der Server nun eine solche Update Anfrage, erscheint in der Konsole (des Servers) die Frage (letzte Zeile in *Abbildung 32*), ob eine entsprechende Fehlermeldung geworfen werden sollte. Nun ist es möglich mit einer Eingabe von „y“ diese zu werfen, obwohl die Daten korrekt sind.

```
// Ausgabe der Konsole
[…]
positionX from 1.7646667041876034 to 1.5029822821279024
positionZ from -0.7362292835644297 to -0.994609704670731
For user "fiorenzo.gramm@students.fhnw.ch" on showroom with ID
"62d92441388927f5f549169e".
Throw Error message?[y|any]
```

Abbildung 32: Konsolenausgabe, wenn ein Debug-Statement erreicht wird.

Da diese Abfragen den Server blockieren und nicht in einer produktiven Umgebung genutzt werden sollten, können, wie in *Abbildung 33* dargestellt, beim Start der Applikation Flags gesetzt werden.

```
// Starten der Applikation
npm start -- debugFlag
```

Abbildung 33: Start des Servers mit einem Debug-Flag. das "--" trennt dabei die npm Parametern mit denen der Applikation.

Falls die Flags nicht gesetzt sind, wird dieser Prozess nicht ausgeführt und die Applikation läuft unterbruchfrei. Mit diesem Vorgang kann auch geprüft werden, ob sich die Applikation im Fehlerfall korrekt verhält.

3.5 Datenbank

3.5.1 DB Konzept Entscheidung

Im Konzept kann jeder Showroom unterschiedliche Attribute und 3D Modelle haben und unterschiedlich aussehen, sodass eine dokumentenbasierte Datenbank als durchaus sinnvoll evaluiert wurde. Dafür wurde MongoDB verwendet, da diese relativ einfach aufzusetzen ist und eine gute API für Node.js mittels eines Nodes Modul bereitgestellt wird. In der Entwicklung wurde MongoDB Atlas, eine MongoDB Instanz, die auf einem AWS Server von Amazon läuft, verwendet.

3.5.2 DB Architektur

Für die Benutzerverwaltung wurde eine Collection erzeugt die jeden User mit Object ID, E-Mail-Adresse und Passwort speichert. Die Abbildung zeigt einen Datensatz aus der `users` Collection. Sicherheitsbedenken werden im Kapitel *Sicherheit* behandelt.

Für jeden neuen User wird eine eigene Collection mit der E-Mail-Adresse erstellt. Um die Daten kompakt und an einem Ort zu halten, hat jeder User eine Collection in der, wie in *Abbildung 34* dargestellt, alle seine Showrooms als einzelne Einträge gespeichert werden.

Abbildung 34: Screenshot eines Showrooms in MongoDB.

Jeder Showroom soll alle Konfigurationen, Modelle und ihre Eigenschaften und seine eigenen Daten haben. Um die unterschiedlichen Showrooms unterscheiden zu können, benötigt es das Feld `_id`, sowie einen Showroom-Namen für den Benutzer. Es folgen die unterschiedlichen Konfigurationsmöglichkeiten und ein Array mit den 3D-Modell Eigenschaften. Um die Modelle einzigartig zu machen und um Komplikationen zu vermeiden, ist ebenfalls eine `_id`, sowie ein Modellname für den Benutzer notwendig. Da

die Modelle nicht in der Datenbank gespeichert werden, muss der Pfad, beziehungsweise der Dateiname, gespeichert werden. Dazu kommen alle A-Frame relevanten Eigenschaften wie zum Beispiel die Position.

Im Rahmen dieses Projektes wurde lediglich ein statischer Raum verwendet, dadurch ist das Feld `room` für weitere Implementierungen nutzbar, um dem Benutzer die Möglichkeit zu geben, aus unterschiedlichen Räumen wählen zu können. Das Attribut `light` soll ähnlich funktionieren wie `objects` und ist in diesem Produkt ebenfalls als Erweiterbarkeit gedacht, in dem der Benutzer eigene Lichtquellen hinzufügen kann, um seine Modelle zu bestrahlen.

Jedes 3D-Modell hat eine `entity`. Dabei handelt es sich um den A-Frame-Typen der als HTML-Tag verwendet wird. Als Erweiterung könnte man auf diese Weise andere A-Frame-Typen wie zum Beispiel `Light` verwenden. `Filename` gibt den im Filesystem gespeicherten Namen des 3D-Modelles an, sodass es im Frontend unter dem entsprechenden Pfad gefunden und importiert werden kann. Das Attribut `modelname` ist der Name des Modells der im Konfigurator angezeigt werden soll. Dieser kann vom Benutzer angegeben werden. Dieselbe 3D-Modell Datei kann dadurch mehrfach eingebunden werden, sodass der `modelname` ebenfalls eine einzigartige ID ist und nur ein Modell und nicht alle mit derselben Datei geändert werden. Position, Rotation und Scale spiegeln das Objekt in der Scene wider. Die Standardwerte sind so gewählt, dass ein neues Modell beim Einfügen direkt in der Mitte des Raumes platziert wird.

3.6 Technischer Ablauf

Sobald die Applikation gestartet wird, öffnet der Server seine Socket-Verbindung und wartet unter einer definierbaren IP-Adresse auf http-Anfragen. Sobald eine Anfrage eintrifft, wird die Route vom Router auf Zulässigkeit überprüft. Wenn der Benutzer noch nicht eingeloggt ist, hat er keinen Zugriff auf die anderen Seiten und wird direkt zum Login weitergeleitet. Der Router gibt die Anfrage dann an den, für den Fall zuständigen Controller weiter. In den Controllern ist die Logik implementiert, wie der Server mit den Anfragen umgehen soll. Bei erfolgreichem Login wird mittels des Express-Session Moduls von Node.js ein Cookie gesetzt, das angibt, ob ein Benutzer eingeloggt ist oder nicht. Das Cookie speichert die E-Mail-Adresse des Benutzers, sodass das Backend immer darauf Zugriff hat.

Beispiel der Bearbeitung der GET-Anfrage auf `/showrooms` nach dem Routing:

1. Es wird ein Datenbankmodell erzeugt mit der E-Mail-Adresse des eingeloggten Benutzers.
2. Mit diesem Modell wird ein Datenbank Aufruf abgesetzt, um alle Showrooms des Benutzers zu bekommen.
3. Es wird auf das Filesystem zugegriffen und alle Modellnamen für das Inventar geladen.
4. Die Showrooms werden über ein Template in einen HTML String gerendert.
5. Das Inventar wird über ein Template in einen HTML String gerendert.
6. Es wird auf vorherige Fehler geprüft, um eventuell eine Fehlermeldung mitzugeben.
7. Über Embedded JavaScript (EJS) wird die gesamte Website mit den individuellen Einträgen in ein EJS-Template gerendert.
8. Das EJS wird gleichzeitig mit allen anderen Includes (Header und Footer) zu einem HTML Dokument zusammengeführt und schlussendlich an den Benutzer gesendet.
9. Im Browser hat der Benutzer dann Zugriff auf die öffentlichen JavaScript-Dateien, die die Seite je nach Klick interaktiv machen.

Bei dieser Anfrage sind die Aufrufe auf die Datenbank und das Filesystem nur zum Lesen. Je nach Anfrage werden die entsprechenden Einträge und Dateien geändert oder gelöscht.

4 Fazit

4.1 Zielerreichung

Das Ziel dieser Arbeit war es, ein minimales Produkt zu definieren und zu entwickeln, mit dem man virtuelle Showrooms erstellen kann. Dabei wurden einige innovative, aber nicht fundamentale Anforderungen nicht umgesetzt. Die Fortschritte der VR-Technologie bieten Werkzeuge an, die dafür gut genutzt werden können. Obwohl sich für diese Arbeit auf ein Framework konzentriert wurde, gibt es unterschiedliche valide Möglichkeiten, eine Software-Lösung zu entwickeln. Der umgesetzte Prototyp erlaubt es, Showrooms anzulegen und zu verwalten. Zudem können Objekte hochgeladen und frei im Raum platziert werden. Der Konfigurierte Showroom kann dann in einer VR-Brille betreten werden. Dabei kann der Benutzer sich physisch und durch Verwendung der Controller bewegen und umschauen. Diese Arbeit zeigt einerseits, dass es möglich ist, mit den vorhandenen Technologien und Frameworks eine Lösung zu entwickeln, und andererseits, welche Technologien und Frameworks sich anbieten.

4.2 Herausforderungen und Probleme

Das WebXR API erlaubt es, die Hardware der Ausrüstung wie die VR-Brille oder die Controller zu nutzen. Aus dem Setup und Testversuchen wurde mit der Zeit klar, dass dafür eine sichere Verbindung (https) zum Host benötigt wird. Startet nun die Applikation (der Server) und man verbindet sich auf den localhost (via http ohne s) funktioniert zunächst alles. Die Szene wird dabei korrekt dargestellt. Wechselt man dann allerdings in den VR-Modus, wird eine generische zwei Bildern dargestellt (in der Darstellung innerhalb des Browsers), auch auf der Brille. Es wird also nicht für jedes Auge ein separates Bild angezeigt. Die Controller werden ebenfalls nicht erkannt. Erst wenn man eine sichere Verbindung via https zum host aufbaut, wird die VR-Ausrüstung erkannt und es funktioniert. Da es keinerlei Fehlermeldung gab und nur eine „falsche“ Darstellung, war zunächst nicht klar es lag. Durch verschiedene Tests mit glitch.com wurde dann der „Fehler“ erkannt.

Die Darstellung des Showrooms in VR variiert in unterschiedlichen Browsern. Auf der Oculus Quest 2 im FireFox Reality Browser ruckelt es gelegentlich und der Raum bewegt sich mit. Auf dem Oculus eigenen Browser treten diese negativen Effekte nicht auf. A-Frame nutzt die WebXR API erst seit Version 1.1.0. Vorherige Versionen nutzen nur die inzwischen veraltete WebVR API. Der Oculus eigene Browser hat keine WebVR Unterstützung. Erst beim Umstellen der Version auf 1.3.0 konnte der Showroom auch im Oculus eigenen Browser betreten werden.

Zudem gab es noch Probleme mit den Modellen. Im Verlauf des Projektes wurden unterschiedliche Modelle von unterschiedlichen Anbietern wie TurboSquid oder CGTrader verwendet, um Objekte einzufügen und zu testen. Neben gltf wurden ebenfalls die Dateiformate fbx und obj verwendet. Dabei wurden die Modelle oft nicht dargestellt. In den meisten Fällen auch ohne Fehlermeldungen. Was geholfen hat, war das Importieren in Blender und das anschliessende Exportieren. Es ist nicht klar, was genau der Fehler war, jedoch konnten die Modelle dann dargestellt werden. Aufgrund mangelnder Fehlermeldungen ist es schwierig, Schlüsse daraus zu ziehen, ob der Fehler an einem problematischen Export des Anbieters oder doch bei A-Frame oder Three.js liegt.

Als der Applikation Licht und Schatten hinzugefügt wurden, fiel auf, dass Modelle, die in die Szene geladen wurden Fehler in der Darstellung der Schatten zeigten. Auch hier gab es keine Fehlermeldungen. Nach einiger Recherche wurde ein Post von MaX (MaX, 2020) gefunden in dem beschrieben wurde, dass es am doppelseitigen Material liegt. Um dieses Problem zu beheben, muss für jedes Material in Three.js angegeben werden, dass die Schattenseite die Rückseite ist.

In der technischen Recherche wurde untersucht, wie man Controller einfach einbinden und verwenden kann. In einzelnen Beispielen und Tutorials hat das einigermaßen gut funktioniert. Es wurde die Komponente „input-listen“ von Michael (Michael, 2022) übernommen. Das Mapping ist nicht immer korrekt man muss gelegentlich die Seite neu laden, um die Controller am gleichen Ort in VR und im realen Raum zu haben.

Da A-Frame nur unter https läuft, konnten wir in der Entwicklung die VR nur eingeschränkt testen, da jede Änderung erst auf vsc.innology.ch bereitgestellt werden musste, um sie zu testen. Im Laufe des Projektes haben sich die Prioritäten stärker Richtung Konfigurator verschoben und die Features, die ursprünglich für den VR Showroom gedacht waren, wurden in das Kapital *Weiterführende Schritte und Empfehlungen* verschoben.

4.3 Diskussion

Der Vorteil der Innology liegt im Kundenkontakt. Die Kunden haben verschiedene Produkte, die sie anbieten und verkaufen möchten. Daher ist es von grossen Nutzen von dem Wissen und der Erfahrung, die diese mitbringen zu profitieren. Dadurch könnte einerseits mehr auf die Bedürfnisse der einzelnen Kunden eingegangen werden. Aber es wäre auch wertvoll zu erfahren, welche Werkzeuge, Strategien oder Medien sie bisher nutzen, um ihre Produkte zu vermarkten.

Für dieses Projekt wurde A-Frame als Framework verwendet. In der Technischen Recherche wurden jedoch einige Alternative vorgestellt. Es wäre auch eine Hybrid-Lösung möglich. Dabei ist der Web-Konfigurator zum Beispiel mit A-Frame umgesetzt, die Applikation, in der man den Showroom betritt, könnte zum Beispiel mit Unity umgesetzt werden. Damit könnte man von den Stärken beider Technologien gleichzeitig profitieren, wobei dies auch die Entwicklung und Wartung beeinflusst.

Im Verlauf des Projekts änderte sich die Performance im VR-Showroom. So fängt es je nach Anzahl der Elemente im Showroom und der Anzahl der Polygone im Modell an zu ruckeln. Dieses Ruckeln verhindert ein angenehmes Erlebnis im Showroom und sollte daher verhindert werden. Ob und wie sich unterschiedliche Technologien wie eine Lösung mit Unity, A-Frame oder Babylon.js im Kontext der Performance unterscheiden, wurde nicht überprüft. Um einen Vergleich machen zu können, müsste man die gleichen Szenen und gleiche, oder zumindest ähnliche, Beleuchtung umsetzten. Für eine finales Produkt ist das ein wesentlicher Faktor, für die Entwicklung eines Prototyps hatte dies aber geringe Priorität.

4.4 Weiterführende Schritte und Empfehlungen

Mit der entwickelten Applikation wurde ein Fundament gelegt, auf dem aufgebaut werden kann. Werden die im Kapitel Diskussion genannten Punkte berücksichtig, kann eine gute Software-Lösung umgesetzt werden.

4.4.1 Usability Tests

Es wurden zwar einige Tests mit unterschiedlichen Kandidaten durchgeführt, doch diese sind nicht für ein breites Klientel repräsentativ. Die Kundschaft, die den VR-Showroom betritt, kann sehr unterschiedlich sein. Da die Wirkung, wie im Kapitel *Psychologische Faktoren beim Kaufentscheid* erwähnt, nicht zwangsweise positiv sein muss, ist das Kundenfeedback wichtig.

Neben den Funktionen des Konfigurators ist auch die Bedienung wichtig. Wenn ein Verkäufer die Funktionen hat, die er benötigt, sie aber nicht finden oder nutzen kann, bringt ihm auch die beste Applikation nichts. Daher sollte eine ausführliche Einführung mit Tutorials und/oder einem Wiki bereitgestellt werden, in denen ein Verkäufer sich schnell einlesen kann.

4.4.2 Modelle

Eine weitere Empfehlung ist das Einführen eines klar definierten Importstandards für Modelle. Für einen Nutzer ist es frustrierend, wenn er ein Objekt einfügt, es aber fehlerhaft oder gar nicht dargestellt wird. Um dies zu verhindern, müssen klare Merkmale definiert sein, die für ein Modell gelten müssen. Ein Merkmal wäre zum Beispiel die Grösse. Die meisten Formate unterstützen keine Einheiten. Daher kann

ein Modell mal in Millimetern oder in Metern angegeben werden. Dabei ist es wichtig, ob ein Objekt 1 Meter oder einen Kilometer breit ist. Ist das Objekt viel zu gross oder klein, ist es unter Umständen nicht mal sichtbar. Das gilt es zu verhindern. Ob beim Einfügen eine Einheit angegeben werden kann oder ob alle Elemente auf eine Einheitsgrösse skaliert werden, muss allerdings in einem weiteren Schritt genauer untersucht werden, im besten Fall mit Experten, die Erfahrung mit 3D-Design haben.

4.4.3 Bibliothek von Modellen

Wie im vorherigen Kapitel beschrieben, ist das Sammeln von geeigneten Modellen aufwendig und kann kostspielig werden. Da Unternehmen eigene Modelle haben, diese jedoch nicht zwingend vergleichbar mit anderen Objekten sind oder der Raum an sich zu leer erscheint, empfiehlt sich eine Bibliothek an vordefinierten 3D-Modellen. Dadurch konzentriert dich der Verkäufer nur um seine eigenen Produkte und muss nicht noch andere Modelle suchen und eventuell sogar kaufen.

4.4.4 Darstellung der Showrooms und des Inventars

Da es bei vielen Modellen im Inventar oder auch Showrooms, die nur mit einem Namen dargestellt werden, schnell unübersichtlich wird, empfiehlt es sich, diese Ansicht zu erweitern. Eine wie in *Abbildung 8* dargestellte Sicht in Bildern vom Showroom oder den Modellen, würde die Übersichtlichkeit deutlich steigern. Dabei könnte dem Nutzer zum Beispiel die Möglichkeit gegeben werden, in jedem Showroom eine Aufnahme zu machen welche dann direkt als Vorschau für diesen Showroom verwendet werden kann. Für die Modelle wäre eine Automatische Lösung möglich, in der nach dem Upload das Modell gerendert wird und auf mit einem neutralen Hintergrund als Vorschau verwendet wird.

4.4.5 Cloud

Amazon AWS, Microsoft Azure und andere Anbieter offerieren einen Clouddatenspeicher, den man ortsunabhängig nutzen kann. Dateien unterschiedlicher Formate werden extern auf mehreren Servern abgespeichert. Der Service ist flexibel, gut skalierbar, zuverlässig und hoch verfügbar. Für unterschiedliche Anwendungsgebiete gibt es unterschiedliche Arten, Objektspeicher, Dateispeicher oder Blockspeicher. Der Zugriff erfolgt über eine mitgelieferte API.

In einer Applikation, bei der mehrere Benutzer gleichzeitig Daten anfragen, ist diese keine skalierbare Lösung. Daher empfiehlt es sich dafür die Cloud zu nutzen. Werden die Assets in einer Cloud gespeichert können diese unabhängig, effizienter und ohne den Server zu belasten von vielen verschiedenen Benutzern angefragt und geladen werden. Während der Entwicklung des Projektes waren wenige Benutzer und dementsprechend wenige Filesystem Calls aktiv.

4.4.6 Sicherheit

Der Benutzer muss sich für die Nutzung des Produktes registrieren, das erfolgt über die Eingabe einer korrekten E-Mail-Adresse, und einem Passwort. Das Format der E-Mail-Adresse wird überprüft, es muss einen individuellen Teil, das @-Zeichen, sowie eine Domäne haben. Es erfolgt keine Verifizierung, ob die E-Mail-Adresse wirklich existiert. Das Passwort darf nicht leer sein, sonst gibt es keine Restriktionen.

Der Fokus dieses Projektes lag auf dem Erstellen eines Konfigurators von Showrooms für Virtual Reality. Es wurde aus Gründen der Priorität eine simple Benutzerverwaltung erstellt, die den Anforderungen entspricht. Die Passwörter werden unverschlüsselt abgespeichert und Sicherheitsmechanismen wie die Verifizierung der E-Mail-Adresse wurden nicht eingeführt. Es wäre möglich durch ungültige Anfragen den Server zu überlasten oder ihn zum Absturz zu zwingen. Mit der rechtlichen Situation wurde sich ebenfalls nicht auseinandergesetzt. Es empfiehlt sich das ganze Sicherheitskonzept neu aufzubauen und umzusetzen.

Literaturverzeichnis

A-Frame Introduction. (2022). (maintained by Supermedium) Retrieved 8 8, 2022, from https://aframe.io/docs/1.3.0/introduction/

Alexander, I. (2004). *A Better Fit - Characterising the Stakeholders.* Scenario Plus.

Argelaguet, F., & Andujar, C. (2013). *A survey of 3D object selection techniques for virtual environments.*

Badler, N., & Glassner, A. (2013). *3D Object Modeling.* Chapel Hill, North Carolina, United States: University of North Carolina at Chapel Hill.

Bowman, D., Kruijff, E., LaViola Jr., J., & Poupyrev, I. (2005). *3D User Interfaces.* Boston: Addison-Wesley.

Brugha, R., & Varvasovszky, Z. (2000). *Stakeholder analysis: a review, Health Policy and Planning, Volume 15.* Retrieved 08 13, 2022, from https://www.bwl-lexikon.de/wiki/stakeholderanalyse/

Bryson, S. (1993). *IEEE 1993 Symposium on Research Frontiers in Virtual Reality.*

Carpenter, J. (2015, 12 15). *Introduction to A-Frame*. Retrieved 8 8, 2022, from https://aframe.io/blog/introducing-aframe/

Caudell, T., & Mizell, D. (1992). *Augmented Reality: An Application of Heads-Up Display Technology to Manual.* Seattle.

Cornelsen Verlag GmbH. (2022). *Duden | virtuell | Rechtschreibung, Bedeutung, Definition, Herkunft*. (Cornelsen Verlag GmbH) Retrieved 8 5, 2022, from https://www.duden.de/rechtschreibung/virtuell

Doerner, R., Broll, W., Jung, B., & Grimm, P. (2022). *Virtual and Augmented Reality (VR/AR).* Springer.

Dr. Lazarus, R. (2021, 07 11). *3D Vision is more important than you think*. Retrieved 8 5, 2022, from https://www.optometrists.org/vision-therapy/vision-therapy-for-lazy-eye/7-signs-your-child-might-have-a-lazy-eye/stereopsis-more-than-3d-vision/

Feiner, S., Maclyntire, B., Höllerer, T., & Webster, A. (1979). *A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment.*

HEGIAS AG. (2022). *Hegias*. Retrieved Juni 23, 2021, from https://hegias.com/

Kato, H., & Billinghurst, M. (199). *Marker tracking and HMD calibration for a video-based augmented reality conferencing system.*

Khronos Group. (2022, 2 13). (Khronos Group) Retrieved 8 4, 2022, from https://www.khronos.org/opengl/wiki/History_of_OpenGL

Khronos Group. (n.d.). *WebGl Overview*. Retrieved 8 4, 2022, from https://www.khronos.org/webgl/

Lalit Narayan, K., Mallikarjuna Rao, K., & Sarcar, M. (2008). *Computer Aided Design and Manufacturing.* New Dehli: Prentice-Hall.

MaX. (2020, 05 27). *Stackoverflow*. Retrieved from Stackoverflow: https://stackoverflow.com/questions/62048075/lights-in-a-frame-have-striped-artifacts-depending-on-distance-from-them

mdn web docs. (2022, 6 3). (Mozilla) Retrieved 7 27, 2022, from https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API#browser_compatibility

Michael. (2022, 08 19). *inspiredtoeducate.net*. Retrieved from inspiredtoeducate.net: http://inspiredtoeducate.net/inspiredtoeducate/build-a-aframe-io-scene-on-oculus-quest-with-teleportation/

Milgram, P., Takemura, H., Utsumi, A., & Kishino, F. (1995). *Augmented reality: a class of displays on the reality-virtuality continuum.*

MongoDB Inc. (2022). *MongoDB*. (MongoDB Inc) Retrieved 08 09, 2022, from https://www.mongodb.com/de-de/nosql-explained

moz://a Mixed Reality. (n.d.). (Mozilla) Retrieved 7 22, 28, from https://mixedreality.mozilla.org/firefox-reality/

Nazarov, R., & Galletly, J. (2013). *Native browser support for 3D rendering and physics* . Thessaloniki, Greece.

OpenJS Foundation. (2022, 08 09). *nodejs*. (OpenJS Foundation) Retrieved 08 09, 2022, from https://nodejs.org/en/about/

Orlamünder, D., & Mascolus, W. (2008). *Computergrafik und OpenGL.* München: Carl Hanser Verlag.

Parisi, T. (2012). *WebGL Up and Running.* Sebastopol: O'Reilly Media .

Petrov, C. (2022, 6 3). *techjury*. Retrieved 7 26, 2022, from https://techjury.net/blog/virtual-reality-statistics/

PixelPerfect Studios. (n.d.). Retrieved 8 4, 2022, from https://pixelperfect-studios.com/history-of-3d-rendering/

Seddon, C. (2005). *OpenGL Game Development.* Wordware Pub.

Shapespark sp. z o.o. (2022). *shapespark*. Retrieved Juni 23, 2021, from https://www.shapespark.com/

Steve, M., Furness, T., Yuan, Y., Iorio, J., & Wang, Z. (2018). *All Reality: Virtual, Augmented, Mixed (X), Mediated (X,Y), and Multimediated Reality.*

Sutherland, I. (1965). *The Ultimate Display.*

Thompson, S. (27, 4 2020). *VIRTUALSPEECH*. Retrieved 7 22, 28, from https://virtualspeech.com/blog/motion-sickness-vr

Three.js Fundamentals. (2022). Retrieved 8 8, 2022, from https://threejs.org/manual/#en/fundamentals

Unity Technologies. (2022, 07 30). *Unity Documentation - Building your WebGL application*. Retrieved 8 8, 22, from https://docs.unity3d.com/Manual/webgl-building.html

Vainikka, B. (2015). *Psychological Factors Influencing Consumer Behaviour.* Centria University of Applied Science.

Virbela. (2022). *FRAMEVR*. Retrieved Juni 23, 2021, from https://framevr.io/

Virtuloc, s.r.o. (2022). *Virtuloc*. (Virtuloc, s.r.o.) Retrieved Juni 23, 2021, from https://www.virtuloc.com/

VRdirect GmbH. (2022). *VRdirect*. Retrieved Juni 23, 2021, from https://www.vrdirect.com/

wikipedia. (n.d.). *nodejs wikipedia*. (wikipedia) Retrieved 08 09, 2022, from https://de.wikipedia.org/wiki/Node.js

Ehrlichkeitserklärung

«Hiermit erkläre ich, die vorliegende Bachelorarbeite selbständig und nur unter Benutzung der angegebenen Quellen verfasst zu haben. Die wörtlich oder inhaltlich aus den aufgeführten Quellen entnommenen Stellen sind in der Arbeit als Zitat bzw. Paraphrase kenntlich gemacht. Diese Bachelorthesis ist noch nicht veröffentlicht worden. Sie ist somit weder anderen Interessierten zugänglich gemacht noch einer anderen Prüfungsbehörde vorgelegt worden.»

Windisch, 19.08.2022

Name: Felix Kerkmann

Unterschrift: 

Name: Fiorenzo Gramm

Unterschrift: 

Anhang

A Usability Tests vom 26.05.2022

Ablauf

1. Einführung in unser Projekt und folgenden Ablauf
2. Einführung Hardware
3. Einführung Prototyp
4. Konfigurator Test
5. Showroom Test
6. Interview
7. Aktueller Stand

Ablauf einzelner Test

Konfigurator durchklicken:

1. Login
2. Showroom öffnen
3. Neuen Showroom erstellen aus Vorlage
4. Objekt hinzufügen
5. Position, Rotation, Skalierung des Objektes ändern
6. Neues Material hochladen
7. Showroom öffnen
8. Material im Showroom ändern

Showroom:

1. Herumlaufen
2. Controller Teleportation
3. Objekte betrachten

Interview:

1. Fragen zum Konfigurator
2. Fragen zum Showroom

Proband 1

Vorkenntnisse:

- Keine VR Erfahrung
- Kann nicht gut mit Computern
- Kein Verkäufer
- Keine Konfigurationserfahrung
- Noch nie 3D Modelle benutzt

Analyse Konfigurator:

- Überfordert mit den Aufgaben
- Angefangen einfach rumzuklicken
- Langsam ein Verständnis entwickelt
- Zu schnell durch
- Abstraktionslevel nicht verstanden
- Nicht verstanden wie die Applikation funktionieren soll
    - Web Konfig -> Web Showroom auf der Brille
- Sinn verstanden

Analyse Showroom:

- Überfordert mit den Controllern
- Steif gestanden
- Langsame Bewegung
- Browser im VR nicht verstanden
- Angefangen einfach rumzuklicken
- Begeistert von der Technologie
- «Oh sieht gut aus»
- Teleport anfangs nicht verstanden
- Ganz langsame Bewegungen
- Bewegungen im RL um den VR Tisch herum
- Schnell klicken für Teleport, ohne die Position anzuschauen
- «Am Tisch ist der Rand»
- Konzept verstanden

Interview Erkenntnisse:

Konfigurator

- Vorstellung: Vordefinierte Objekte kombinieren
- Farbe, Grösse der Objekte am wichtigsten
- Objekt Typ wechseln und ganzes Objekt bewegen / entfernen / hinzufügen
- Raumgrösse soll selbst definiert werden können
- Hauptanwendungsfeld Innenarchitektur
- Animationen wichtig für realitätsnähe -> sieht schöner aus
- Licht, Wetter, Tageszeit total wichtig

Showroom

- Orientierung im Raum gut
- War nicht immer bewusst, wo man im realen raum stand -> Blaue Randlinien super
- Massstab gut
- Schnelle Angewöhnung an Umgebung
- Kein Schwindel oder ähnliches
- Mehr Objekte, ansprechendere Gestaltung
- Objekte waren realistisch

Ideen, Wünsche

- Konfiguration komplett im VR
- Objekte hinzufügen, bearbeiten in VR
- Unterschiedliche Modelle zum gleichen 3D Objekt

Proband 2

Vorkenntnisse:

- Ein bisschen VR Erfahrung
- Erfahrung mit Showrooms
    - U.a. in der eigenen Firma
- Erfahrung mit Computern
- Verkaufserfahrung
- Noch nie 3D Modelle benutzt

Analyse Konfigurator:

- Aufgaben zu einfach, will immer noch andere Sachen machen
- Aber auf eine Aktion beschränkt
- Schnelles Verständnis entwickelt
- Abstraktionslevel in VR nicht verstanden
- Sinn verstanden

Analyse Showroom:

- Überfordert mit den Controllern
- Bisschen angespannt gestanden
- Langsame Bewegung
- Teleport am Anfang nicht verstanden
- Schnell Interesse verloren
- Materialien waren nicht realitätsnah
- Konzept verstanden

Interview Erkenntnisse:

Konfigurator

- Vorstellung: Vordefinierte Objekte kombinieren
- Drinsitzen und Fahren
- Komplette Anlagen -> Stein der durch die Asphaltanlage fährt
- Treppen hoch und runter + mehrere Räume
- Ganze Gebäude unterschiedlicher Massstab
- Objekte selbst bearbeiten nur ein Teil des Objektes soll eine andere Farbe haben
- CAD
- Einzelne Bereiche eines Objektes bearbeiten
- Animationen, Wetter unwichtig
- Man sollte Produkte spüren, öl riechen, wärme spüren von Motoren
- Licht, Tageszeit nice to have
- Man arbeitet mit dem was man hat (CAD)
- Demo, direkte Beschreibung was man machen kann, Guideline, selbsterklärend

Showroom

- War nicht immer bewusst, wo im VR Room man stand
- War immer bewusst wo im realen Raum
- Gatter gut
- Massstab gut
- Farbe Material schlecht
- Schnell an Umgebung gewöhnt
- Kein Schwindel oder ähnliches
- Probleme mit der VR Brille als Brillenträger
- Interagieren war einfach
- Teleport war einfach
- Interessante Erfahrung
- Funktionalitäten langen für Verkäufer

Proband 3

Vorkenntnisse:

- keine VR Erfahrung
- keine Erfahrung mit Showrooms
- keine Erfahrung mit Computern
- keine Verkaufserfahrung
- Noch nie 3D Modelle benutzt

Analyse Konfigurator:

- Aufgaben zu kompliziert
- überfordert
- Abstraktionslevel nicht verstanden
- Viel Hilfe benötigt

Analyse Showroom:

- Controllerhandling einfach
- Locker gestanden
- normale Bewegung
- Schnell Interesse verloren, zu langweilig

Interview Erkenntnisse:

Konfigurator

- Zu kompliziert
- Funktionalitäten werden abgedeckt
- Licht, Wetter, Tages Zeit, Animation nice to have

Showroom

- War immer bewusst wo man im RL steht
- War immer bewusst wo man im VR steht
- Massstab okay
- Schnell an Umgebung gewöhnt
- Kein Schwindel oder ähnliches
- Interagieren war einfach
- Teleport war einfach

Fazit

Konfigurator

- Deckt die wichtigsten Funktionalitäten ab
- Zu Kompliziert
    - Guideline die einen durchführt
- Erweiterbarkeit Licht, Tageszeit, Wetter, Animationen
- Raumgrösse definierbar machen
- Einzelne Bereiche eines Objektes bearbeiten

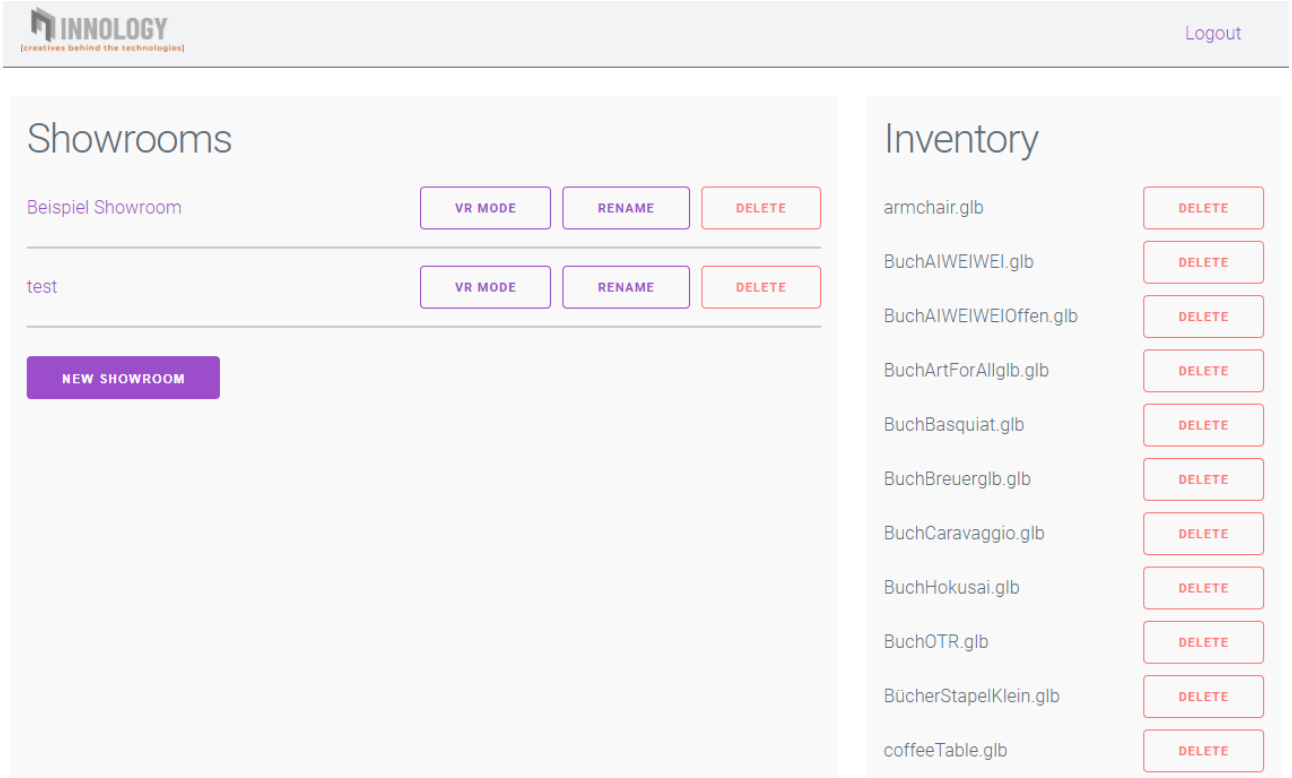
- Gute Einführung für Workflow – Nutzer Bedienfreundlich
- Welche Konfigurationen eigenen sich für die Plattform / VR
- Finetuning im VR (z.B. Licht)
- Wie sieht der raum aus – Standardräume + Freie Möglichkeit

Showroom

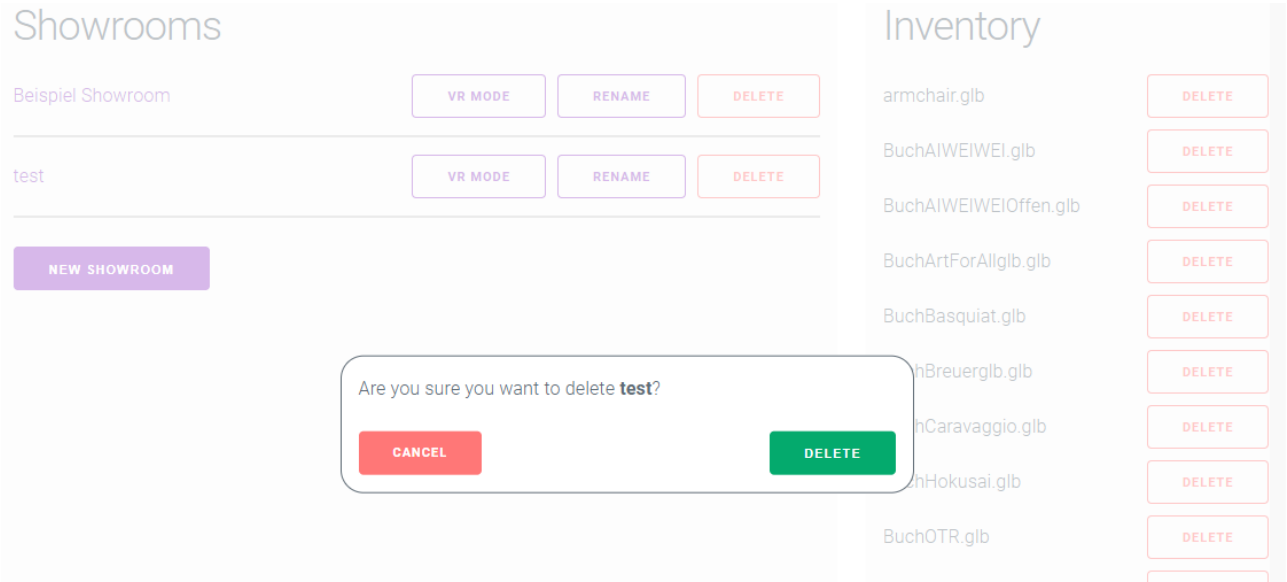
- Materialien, Farben sind Ansichtssache
- Raum sehr gut, Interaktionen mit Controllern gut
- Massstab gut
- Schnelle Gewöhnung
- Kein Übelgefühl, Schwindel
    - WebVR eignet sich gut

- Demo für Controller - Detailliert
- Handtracking?
- Draussen zeigen – drinnen zeigen
- An bestehenden Lösungen inspierieren

B Screenshots des Web-Konfigurators

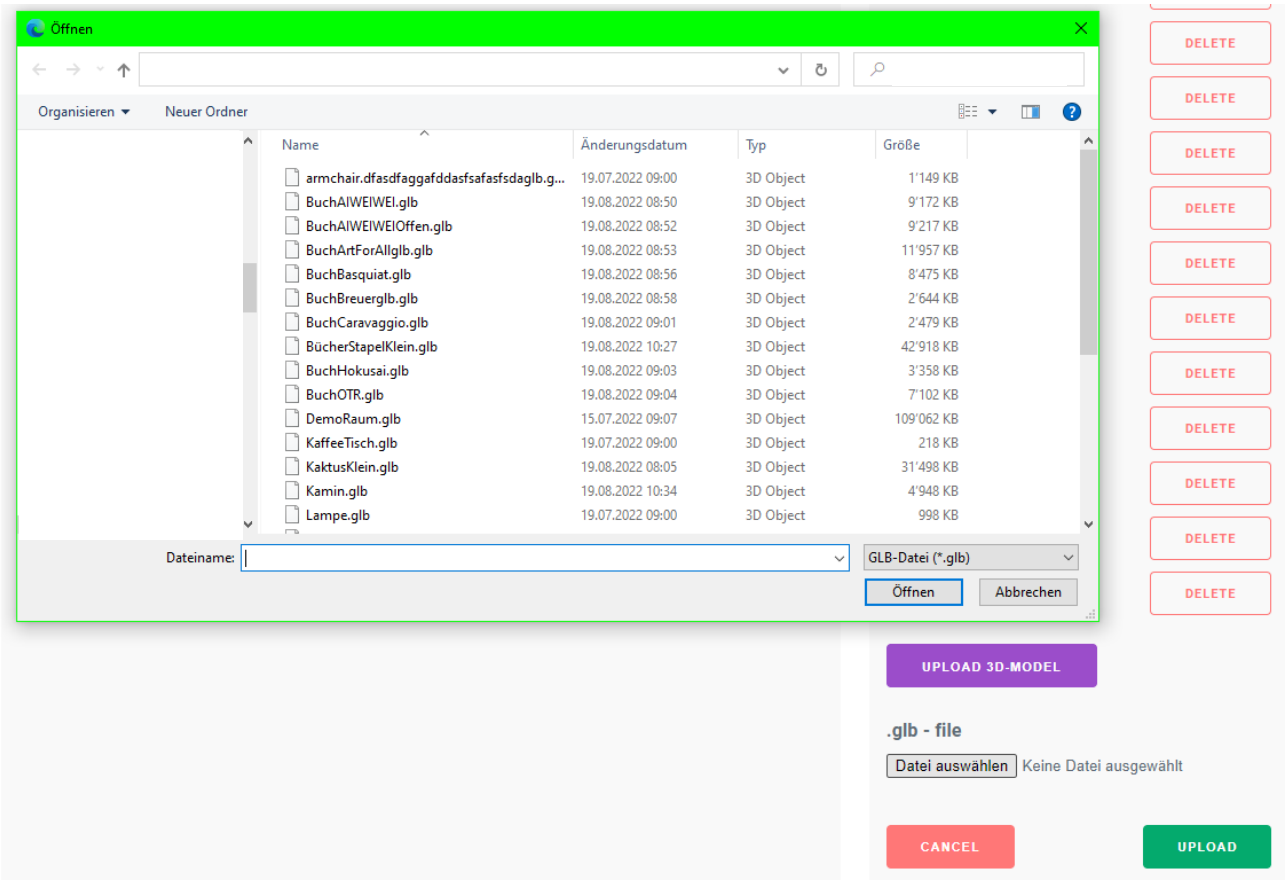


Screenshot der Startsicht mit den unterschiedlichen Showrooms und dem Inventar.

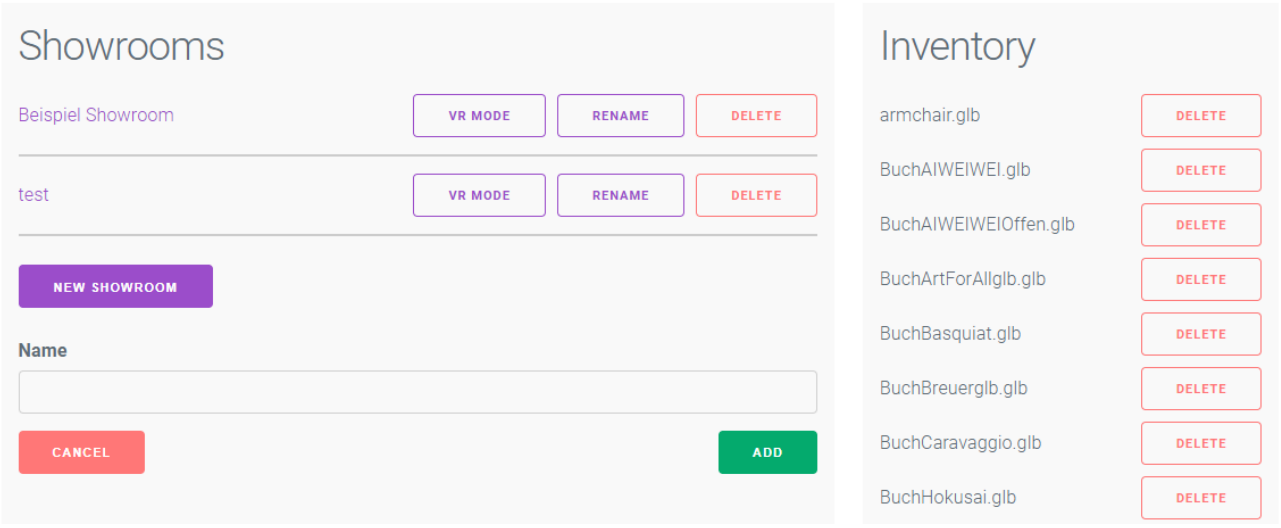


Pop-Up Dialog zum Löschen eines Elements.

B Screenshots des Web-Konfigurators

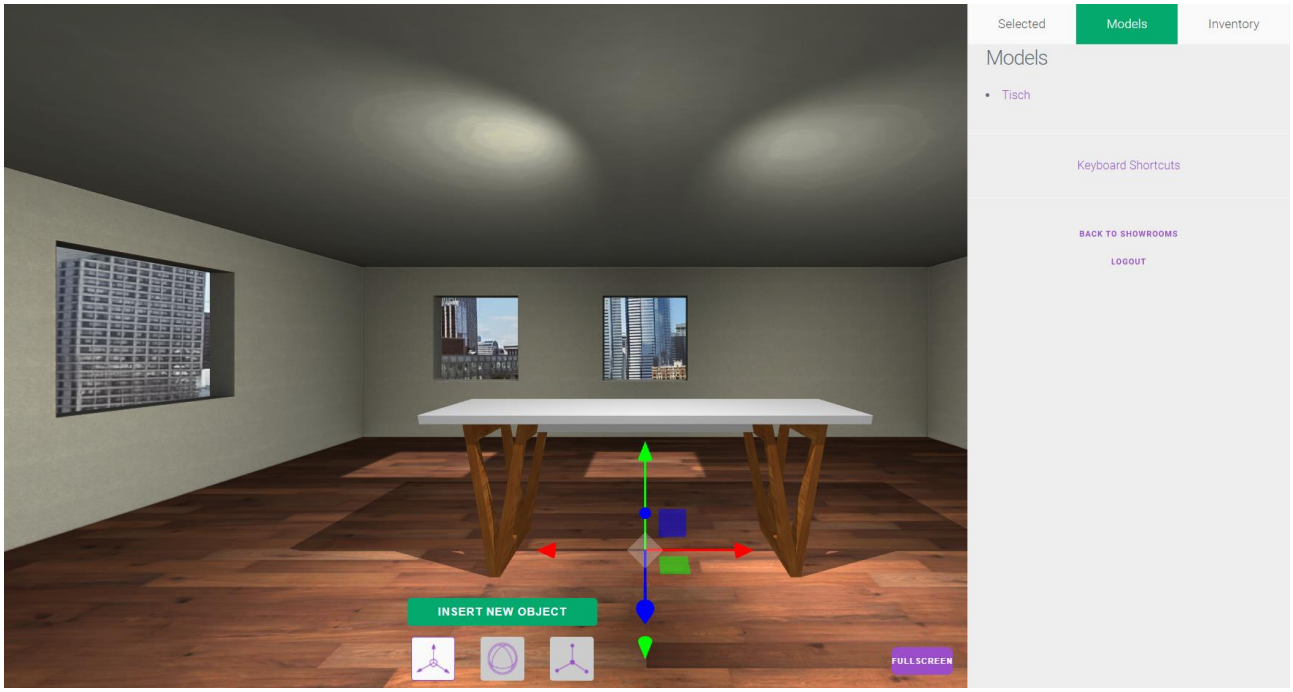


Modell im Datei-Explorer auswählen und ins Inventory hochladen.



Erstellen eines neuen Showrooms.

B Screenshots des Web-Konfigurators



Darstellung im Konfigurator im Menü „Models“ welches alle Modelle im Raum auflistet die via Mausklick ausgewählt werden können.