

**Entwicklung einer immersiven Mixed-Reality-Anwendung zur virtuellen Tierbeobachtung und -fotografie**

Forschungsprojektdokumentation

Masterstudiengang

Advanced Information Technology (MIT)

Fischer, Silja-Marie, 333339

Krauß, Julia, 333145

Professor

Prof. Dr.-Ing. Sascha Seifert

Pforzheim, [DATUM]

**Eigenständigkeitserklärung**

Hiermit erklären wir, dass wir die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

Alle sinngemäß und wörtlich übernommenen Textstellen und Grafiken aus fremden Quellen wurden kenntlich gemacht.

Pforzheim, [Datum] (Unterschriften aller Bearbeiter)

**Inhaltsverzeichnis**

[1 Einleitung 1](#_Toc206060816)

[1.1 Motivation 1](#_Toc206060817)

[1.2 Projektziel 2](#_Toc206060818)

[1.3 Aufbau der Dokumentation 3](#_Toc206060819)

[2 Stand der Technik 4](#_Toc206060820)

[2.1 Extended Reality 4](#_Toc206060821)

[2.2 Head-Mounted Displays 7](#_Toc206060822)

[2.2.1 Begriff und Bedeutung 7](#_Toc206060823)

[2.2.2 Technische Funktionsweise 7](#_Toc206060824)

[2.2.3 Verfügbare Geräte 9](#_Toc206060825)

[2.3 Virtuelle Tierbeobachtung 9](#_Toc206060826)

[2.4 XR in Spielen 11](#_Toc206060827)

[2.4.1 Verfügbare Spiele 11](#_Toc206060828)

[2.4.2 Multiplayer 11](#_Toc206060829)

[2.5 Interaktion mit der Umgebung 11](#_Toc206060830)

[3 Anforderungen 12](#_Toc206060831)

[4 Technologieauswahl 13](#_Toc206060832)

[4.1 Hardware 13](#_Toc206060833)

[4.2 Entwicklungsumgebung 13](#_Toc206060834)

[4.3 Programmiersprache 14](#_Toc206060835)

[4.4 Multiplayer Plugin 15](#_Toc206060836)

[4.5 Architekturmodell 16](#_Toc206060837)

[5 Konzept 21](#_Toc206060838)

[5.1 Aufbau der Anwendung 21](#_Toc206060839)

[5.2 Grundlegende Architektur 22](#_Toc206060840)

[5.2.1 Datenbank 24](#_Toc206060841)

[5.2.2 Backend 25](#_Toc206060842)

[5.2.3 Frontend 26](#_Toc206060843)

[5.3 Interaktion 30](#_Toc206060844)

[6 Prototypenentwicklung 31](#_Toc206060845)

[6.1 Datenbank 31](#_Toc206060846)

[6.2 Backend 32](#_Toc206060847)

[6.3 Frontend 34](#_Toc206060848)

[6.3.1 Setup der Unity-Umgebung für die Meta Quest 3 34](#_Toc206060849)

[6.3.2 Implementierung 35](#_Toc206060850)

[6.3.3 Weiteres 38](#_Toc206060851)

[7 Zusammenfassung und Ausblick 40](#_Toc206060852)

[Glossar 41](#_Toc206060853)

[Abbildungsverzeichnis 42](#_Toc206060854)

[Tabellenverzeichnis 43](#_Toc206060855)

[Normen 44](#_Toc206060856)

[Software Tools 45](#_Toc206060857)

[Anhang A 46](#_Toc206060858)

[Literaturverzeichnis 49](#_Toc206060859)

**Abstract**

**Abkürzungsverzeichnis**

|  |  |
| --- | --- |
|  |  |
| API | Application Programming Interface |
| AR | Augmented Reality |
| CSV | Comma-Separated Values |
| ECS | Entity Component System |
| HTTP | Hypertext Transfer Protocol |
| LINQ | Language Integrated Query |
| MIT | Master Advanced Information Technology |
| MR | Mixed Reality |
| MVC | Model-View-Controller |
| MVVM | Model-View-ViewModel |
| SQL | Structured Query Language |
| UI | User Interface |
| VR | Virtual Reality |
| WWF | World Wide Fund of Nature |
| XR | Extended Reality |
|  |  |

# Einleitung

Thema VR in der Lehre:

* VR-Lehrumgebungen in zum Beispiel der Forstarbeit (Universität Freiburg, 2023)
  + Erstellung virtueller Abbildungen realer Waldbestände, Waldwachstum, forstliche Prozesse, Auswirkungen von Umweltveränderungen
  + Nutzung für Ausbildung von Forstexperten, Durchführung von Forschungsprojekten
  + Bereitstellung auch für externe Partner
* Hohes Unfallrisiko in Forstwirtschaft, also auch in Wald, VR-Training zur Unfallprävention, Integration in der Ausbildung, innovative Möglichkeit, Sicherheit zu erhöhen und Ausbildung praxisnah zu gestalten (Goebel & ORF-Wissenschaft, 2023)
* Aus/Weiterbildung in Forstwirtschaft muss umgestaltet werden, stetig neue Anforderungen an Waldbesitzer durch Klimaveränderungen, Corona hat virtuelle Lehre nochmal nötiger gemacht, Verbindung von Didaktik und Technologie stellt einen wesentlichen Mehrwert in der Lernerfahrung darstellt (Fell, 2021)
* oder Solararbeiten (PV-Trainer, o. J.)

## Motivation

Die fortschreitende Digitalisierung und die intensive Nutzung mobiler Geräte haben das Lernen und die Wahrnehmung von Informationen erheblich beeinflusst. Aktuelle Studien weisen darauf hin, dass die ständige Verfügbarkeit digitaler Medien die Aufmerksamkeitsspanne verringern kann. Eine Untersuchung der Technischen Universität Berlin und des Max-Planck-Instituts für Bildungsforschung zeigt, dass die Zeitspanne, in der die Gesellschaft ihre Aufmerksamkeit einem Thema widmet, immer kürzer wird, was auf eine "soziale Beschleunigung" hinweist [Mit der Informationsflut sinkt die Aufmerksamkeitsspanne der Gesellschaft](https://www.mpib-berlin.mpg.de/pressemeldungen/informationsflut-senkt-aufmerksamkeitsspanne?utm_source=chatgpt.com).

Aktuelle Forschungen zeigen allerdings, dass interaktive Lernmethoden, insbesondere durch Gamifikation und MR, die Motivation und das Engagement der Nutzer steigern können. Gamifizierte Elemente wie Herausforderungen, Belohnungen und interaktive Inhalte fördern nicht nur das Lernen, sondern erhöhen auch die kognitive Flexibilität und das langfristige Behalten von Informationen. Diese Technologien bieten eine effektive Lösung für die Verkürzung der Aufmerksamkeitsspanne und steigern gleichzeitig das Lernen, indem sie eine ansprechende, interaktive und erlebnisorientierte Lernumgebung schaffen (Role of gamified learning strategies in student’s motivation)

Mixed Reality (MR) bietet eine wertvolle Möglichkeit, das psychische Wohlbefinden zu fördern, indem sie virtuelle Naturerfahrungen schafft, die ähnliche Vorteile wie echte Naturaufenthalte bieten. Studien haben gezeigt, dass der Aufenthalt in natürlichen Umgebungen das Stressniveau senkt, die kognitive Leistung steigert und das emotionale Wohlbefinden verbessert. Diese positiven Effekte können auch in virtuellen Welten erzielt werden, was MR zu einer effektiven Methode macht, um psychische Gesundheit zu fördern, insbesondere wenn der direkte Zugang zur Natur nicht möglich ist (Berman et al., 2012). MR-basierte Anwendungen bieten so eine beruhigende Wirkung und können dabei helfen, Angstzustände zu reduzieren. (Into the wild … or not Virtual nature experiences benefit well-being regardless of human-made structures in nature, Nature in virtual reality improves mood and reduces stress, The Role of Immersive Experience in Anxiety Reduction)

Im Bereich der Tierbeobachtung und -fotografie bietet MR ebenfalls vielversprechendes Potenzial. Durch virtuelle Erlebnisse können Nutzer mit Tieren und ihrer natürlichen Umgebung interagieren, ohne die Umwelt zu belasten. Diese Form der Naturerfahrung ermöglicht es den Nutzern, das Verhalten von Tieren zu beobachten, ohne sie zu stören, und kann dazu beitragen, das Bewusstsein für den Naturschutz zu stärken und das Interesse an der Tierwelt zu wecken (Investigating Effects of Interactive Virtual Reality Games and Gender on Immersion, Empathy and Behavior Into Environmental Education).

Besonders in einer zunehmend urbanisierten und technologisierten Gesellschaft kann die virtuelle Naturerfahrung eine wertvolle Brücke zwischen den Nutzern und der Umwelt schlagen. Die Anwendung von MR zur Förderung des Naturschutzes und des psychischen Wohlbefindens stellt daher eine vielversprechende Lösung dar, um sowohl das Lernen zu optimieren als auch das Bewusstsein für die Bedeutung von Natur zu schärfen.

## Projektziel

Das Hauptziel dieses Projekts ist die Entwicklung einer Mixed-Reality-Anwendung, die den Nutzern eine realistische, interaktive Plattform zur Beobachtung und Fotografie von Tieren bietet. Die Anwendung soll es den Nutzern ermöglichen, in verschiedenen Umgebungen Tiere zu entdecken, mit ihnen zu interagieren und sie zu fotografieren.

Ein zentrales Element dieser Anwendung soll ein digitales Entdeckertagebuch sein, in dem die Nutzer ihre Fotos sammeln und detaillierte Informationen zu den Tieren einsehen können. Durch die Kombination von Elementen der virtuellen Realität und der realen Welt soll die Anwendung sowohl der Unterhaltung dienen, als auch für edukative Zwecke genutzt werden können.

Die Anwendung ermöglicht es, Tierarten in einer virtuellen Umgebung zu beobachten, in der sie sich frei bewegen können. Dabei lassen sich auch Tiere erleben, die in der Realität nur schwer zugänglich sind – etwa weil sie in entlegenen Regionen leben oder lediglich in zoologischen Einrichtungen gehalten werden. Die Darstellung orientiert sich dabei an ihren natürlichen Lebensräumen. So wird ein neuer Zugang zur Vielfalt der Tierwelt eröffnet, ohne in bestehende Ökosysteme eingreifen zu müssen.

Darüber hinaus soll die Anwendung die Nutzer dazu anregen, sich spielerisch mit der Natur auseinanderzusetzen und dabei ihr Wissen über Tiere zu vertiefen. Ziel ist es, das Interesse an biologischer Vielfalt zu fördern und ein stärkeres Bewusstsein für den Schutz von Tieren sowie ihrer natürlichen Lebensräume zu schaffen.

## Aufbau der Dokumentation

Die Dokumentation ist in folgende acht Kapitel gegliedert, die den gesamten Entwicklungsprozess des Projekts umfassen.

Zu Beginn wurde im ersten Kapitel eine Einführung in das Thema gegeben, die die Motivation für das Projekt erläutert und das Ziel des Projekts definiert.

Im zweiten Kapitel wird der Stand der Technik behandelt. Hierbei werden die aktuellen Grundlagen und Technologien im Bereich der Extended Reality (XR) vorgestellt werden. Dabei werden die verschiedenen Varianten wie Virtual Reality (VR), Augmented Reality (AR) und Mixed Reality (MR) erläutert. Darüber hinaus wird gezeigt, wie diese Technologien in Spielen und anderen Anwendungsbereichen eingesetzt werden.

Das dritte Kapitel widmet sich den Anforderungen des Projekts. Dabei werden sowohl funktionale als auch nicht-funktionale Anforderungen an das System formuliert. So wird eine strukturierte Grundlage für die Entwicklung und Implementierung der Anwendung geschaffen.

Im vierten Kapitel wird die Technologieauswahl beschrieben. Es wird erläutert, welche Hardware, Entwicklungsumgebung, Programmiersprache und Architekturmodelle für die Umsetzung des Projekts gewählt werden. Zusätzlich wird auf die Überlegungen eingegangen, die diese Entscheidungen beeinflussen.

Das fünfte Kapitel beschreibt das Konzept des Projekts. Hier wird der Aufbau des Prototyps erläutert, einschließlich der Struktur der Datenbank, der Erstellung von UML-Diagrammen sowie der Planung des Backend- und Frontend-Bereichs der Anwendung.

Im sechsten Kapitel geht es um die Implementierung und Prototypenentwicklung. Es wird beschrieben, welche wesentlichen Schritte zur Entwicklung des Prototyps geführt haben.

Das siebte Kapitel beschäftigt sich mit dem Testen und der Validierung der Anwendung, wobei die angewandten Testmethoden zur Überprüfung der Funktionsfähigkeit und zur Sicherstellung der Erfüllung der Anforderungen erläutert werden.

Abschließend gibt das achte Kapitel eine Zusammenfassung der Ergebnisse des Projekts und einen Ausblick auf mögliche Weiterentwicklungen und zukünftige Anwendungsmöglichkeiten.

# Stand der Technik

## Extended Reality

Extended Reality (XR), oder auch erweiterte Realität, beschreibt die Erweiterung der realen Welt um digitale Elemente (Zobel et al., 2018). Dies umfasst Augmented Reality (AR), Mixed Reality (MR) und Virtual Reality (VR) (Tremosa, 2025). Dabei ist das Ziel, eine immersive Umgebung zu schaffen, wobei Immersion nach Agrawal et al., 2019 als ein Zustand geistiger Versunkenheit und Aufmerksamkeitsverschiebung betrachtet werden kann, die das Ziel hat, das Bewusstsein von der physischen Welt zu entkoppeln. Diese Eigenschaften sollen auch für die Beobachtung von Tieren in einem zu entwickelnden Prototypen gelten.

Dieser Zustand tritt bei VR meist am stärksten auf, da hier die Welt vollständig digital dargestellt und die Realität darüber ausgeblendet wird. Merkmale für „vollständig“ sind die Wahrnehmung der Wirklichkeit und die zugehörigen physikalischen Eigenschaften. „Digital“ kann durch eine „in Echtzeit computergenerierte, interaktive, virtuelle Umgebung“ (Klein, 2009) verstanden werden. Grundlage für VR ist zwingend die Erstellung einer virtuellen Welt durch einen Computer. Entsprechend technische Voraussetzungen sind leistungsfähige Rechner, Hochleistungsgrafikkarten und eine hochauflösende Datenbrille. Sogenannte Cardboards[[1]](#footnote-2) werden zwar ebenfalls als VR-Geräte verkauft, stellen aber kein VR im engeren beziehungsweise korrekten Sinne dar (Mehler-Bicher & Steiger, 2021).

AR hingegen bezeichnet die Erweiterung der realen Umgebung durch virtuell eingeblendete Elemente, allerdings ohne Möglichkeit der Interaktion zwischen ihnen (Tremosa, 2025). Die reale Welt wird hierbei lediglich um computergenerierte Zusatzobjekte beziehungsweise Zusatzinformationen angereichert (Mehler-Bicher & Steiger, 2021).

MR beschreibt ebenfalls die Erweiterung der realen Welt um virtuelle Elemente, allerdings können hier Interaktionen zwischen realen und virtuellen Elementen stattfinden (Tremosa, 2025). Ein eingeblendetes Tier schwebt beispielsweise dabei nicht über dem Boden oder läuft durch Wände, sondern passt sich den physikalischen Eigenschaften an. Häufig werden trotz einer eindeutigen Unterscheidung die Begriffe AR und MR synonym verwendet (Mehler-Bicher & Steiger, 2021). Entsprechend werden Klassifikation, Vorteile und Nachteile in großen Teilen der Literatur ebenfalls synonym behandelt.

Auch wenn ein wachsendes Interesse im XR-Bereich beobachtet werden kann, kann von keiner belegbaren Marktreife entsprechender Produkte gesprochen werden. Jedoch bieten Eigenschaften wie Interaktivität, Medienreichtum oder auch Immersion einige Potenziale, wie sich insbesondere in der Werbeindustrie zeigt. Die genannten Eigenschaften fördern die Zugänglichkeit zu bestimmten Produkten oder Themen und können so die Kaufeigenschaft und Haltung gegenüber Werbung beeinflussen (Barta et al., 2025; Mehler-Bicher & Steiger, 2021). So werden heute in Webshops 3D-Objekte Brillen ins Gesicht projiziert (Living Mirror, als Anwendung über das Smartphone) oder geplante Küchen virtuell begutachtet (Living Architecture) (*Brille online anprobieren: Teste den virtuellen 3D-Simulator*, o. J.; *Virtuelle 3D-Küchenplanung | Der Wohnfuchs*, o. J.; Mehler-Bicher & Steiger, 2014). Für immersive Welten ist die Kategorie Living Environment relevant, denn diese beinhaltet Anwendungen, die reale Umgebungen abbilden und dabei eine Kombination mehrerer sensorischer Reize ergänzen. Das können 3D-Objekte oder auch Audiosequenzen sein (Mehler-Bicher & Steiger, 2014).

XR kann dabei nicht nur die Kaufbereitschaft erhöhen, sondern auch als Informations- und Kommunikationsmedium dienen. So kann XR nach Barta et al. (2025) unter anderem auch die soziale Interaktionen und damit Kommunikation beeinflussen. Eine schnellere Vermittlung von Inhalten, eine stärkere Aktivierung von Kommunikationsteilnehmern und die Reduktion kognitiver Dissonanz kann mittels XR-Technologien erreicht werden. Grund dafür ist eine Steigerung der Emotionalität und eine multisensorische Ansprache, was den Kommunikationsprozess nicht nur unterstützt, sondern verbessert (Mehler-Bicher & Steiger, 2014). Dieser Grund trägt auch dazu bei, weshalb XR verstärkt im Bildungsbereich eingesetzt wird. Neben einer anderen Art der Kommunikation sind auch die Reduktion von Suchzeiten und die zeitgleiche Ansprache unterschiedlicher Sinne Aspekte, die das Lernen vereinfachen. Geringe Kosten und die gefahrenfreie Erprobung handwerklicher Tätigkeiten sind weitere Vorteile, welche diese Technologie im Bildungsbereich beliebt machen (Mehler-Bicher & Steiger, 2014; Zobel et al., 2018).

Hürden, die eine Marktreife von XR-Technologien noch verhindern sind fehlende Nutzerakzeptanz, fehlende Alltagstauglichkeit und ethische Aspekte (von Eitzen, 2023). Auf der anderen Seite stehen noch technische Herausforderungen, ohne deren Überwindung eine Marktreife ebenso schwer werden könnte.

Technologisch herausfordernde Aspekte umfassen sowohl Software- als auch Hardware-Limitierungen, aber auch Themen wie Tracking und Interaktion. Der Bereich Interaktion umfasst nicht nur den Übergang unterschiedlicher Welten, sondern auch das Thema „Kollaboration“. So sind die Themen nahtlose Interaktion und Latenzen, sowie der Datenschutz und Echtzeitsynchronisation Themen, die die Entwicklung stabiler Multiplayeranwendungen beeinflusst. Allerdings zeigen gerade kollaborative virtuelle Umgebungen in einer immer digitalisierteren (Geschäfts-)Welt die Notwendigkeit für weitere Forschung und Standardisierung, wie Anwendungen wie virtuelle Meetingräume belegen (Lapschies, o. J.; Wolfenstein, 2025). Damit für solche Anwendungen die Akzeptanz steigt, werden immersive Anwendungen benötigt. Um aber eine sehr gute Immersion mit konsistenter Performance zu erschaffen, wird leistungsfähige Hardware benötigt. So ist beispielsweise die Verarbeitung hochauflösender Grafiken ohne zu überhitzen oder den Akku zu schnell zu entleeren, ein noch nicht vollständig optimiertes Feld (Molchanov, 2025; rankmagic, o. J.). Im Softwarebereich bestehen Herausforderungen darin, Anwendungen auf verschiedenen Systemen oder Browsern zur Verfügung zu stellen. Auch unterschiedliche oder proprietäre Datenformate oder Geräteschnittstellen erschweren die Interoperabilität (Lawton, 2024). Um Latenzen zu vermeiden und die Belastung von Geräten zu reduzieren, können Berechnungen in einer Cloud ausgeführt werden. Allerdings ist dies bei schlechter oder fehlender Internetverbindung nur eingeschränkt oder nicht möglich, was zu nicht ausführbaren Anwendungen führen kann (Molchanov, 2025; rankmagic, o. J.). Um in MR oder VR einen fließenden Übergang zwischen den Welten zu ermöglichen, sind fortschrittliche Tracking- und Renderingtechnologien erforderlich, welche derzeit noch ausbaufähig sind (Marcus, 2020; rankmagic, o. J.).

Der Bereich Ethik umfasst den Energie- und Ressourcenverbrauch und gesellschaftliche Auswirkungen (Lapschies, o. J.). Werden Augenbewegungen von einer Brille erfasst und diese Daten unrechtmäßig verwendet, sind Manipulation und Betrug Auswirkungen, welche die Relevanz des Themas Datenschutz belegen (Molchanov, 2025). Zudem können unwissentlich aufgenommene Bilder mit der Kamera von (VR)-Geräten als Möglichkeit der unzulässigen Datensammlung gesehen werden (Mehler-Bicher & Steiger, 2021). Daher werden Richtlinien für einen verantwortungsvollen Umgang mit immersiven Technologien immer bedeutender, da sie den betreffenden Personen ein Stück ihrer Kontrolle zurückgeben sollen. Darüber hinaus leiden viele Personen bereits nach kurzer Tragedauer einer Brille unter Anzeichen von Gesundheitsproblemen. Weit verbreitet ist in diesem Zusammenhang die sogenannte Simulationskrankheit. Diese ruft Symptome wie Übelkeit und Balanceverlust hervor (Mehler-Bicher & Steiger, 2021; Molchanov, 2025; Zobel et al., 2018). Ursache dafür ist die fehlerhafte Abstimmung von visueller Information und dem Gleichgewichtsorgan, wofür unzulängliche Immersion eine Ursache sein kann (Mehler-Bicher & Steiger, 2021; Zobel et al., 2018). Auch die Einschränkung der Bewegungsfreiheit oder körperliche Belastungserscheinungen wie Nacken- oder Schulterschmerzen können durch das Tagen einer XR-Brille hervorgerufen werden. Meist ist das Gewicht der am Körper verbauten Rechner- und Grafikleistung oder schlecht ausbalancierte Headsets Ursache solcher Symptome (Mehler-Bicher & Steiger, 2021; Molchanov, 2025). Auch Kopfschmerzen und Ermüdung, Trockenheit und Reizungen der Augen sind gängige Probleme (Molchanov, 2025). Einfache Gegenmaßnahmen sind die Begrenzung der Sitzungsdauer mit regelmäßigen Pausen und eine gute Körperhaltung (Molchanov, 2025). Wohingegen besseres Tracking oder höhere Bildwiederholraten als Gegenmaßnahmen im Verantwortungsbereich von Firmen liegt (Molchanov, 2025). Neben physischen Problemen kann es bei der Verwendung von XR-Technologien auch zu psychischen Problemen kommen. Dazu zählen Überreizung, Schlafstörungen oder Isolationsgefühle. Auch wenn diese Symptome bei Nicht-XR-Spielen beobachtet werden können, sind sie bei XR-Spielen der höheren Immersion wegen ausgeprägter zu beobachten. Insbesondere die Themen Gewaltverherrlichung, Suchtgefahr und Realitätsverlust erfordern eine höhere Aufmerksamkeit und gezielte Gegenmaßnahmen (Mehler-Bicher & Steiger, 2021).[[2]](#footnote-3)

Um diese Herausforderungen zu adressieren, stehen einige Themen im Fokus der Forschung. Ein Aspekt ist dabei die Frage, wie plattformübergreifende Inhalte entwickelt werden können, um XR-Anwendungen zugänglicher und hardwareunabhängig zu gestalten, was wiederum die Reichweite und Akzeptanz erhöhen kann (Molchanov, 2025). Auch ist die Integration von künstlicher Intelligenz (KI) ein Forschungsaspekt sowie Werkzeug, das unter anderem für natürlicher erscheinende Umgebungen oder Interaktionen eingesetzt wird. Ziel ist es beispielsweise die Gestenerkennung und Interpretation von Benutzerinteraktionen zu verbessern oder automatisch 3D-Inhalte zu generieren (adaptive Umgebungen) (Lapschies, o. J.; Molchanov, 2025; Wolfenstein, 2025). Auch KI-generierte personalisierte Inhalte, die Entwicklung realistischer Avatare und multisensorisches Feedback erfahren derzeit große Aufmerksamkeit (Lapschies, o. J.; Molchanov, 2025). Insbesondere im Spiele- und Bildungsbereich bietet XR laut einer Vielzahl von Quellen (Bitkom e.V, 2025; FREELANCE-PRESS-2, 2025; Molchanov, 2025; rankmagic, o. J.; Wolfenstein, 2025) große Potenziale. So beschäftigt sich ein Teil der Forschung mit der Frage, wie immersive und gefahrenfreie Lernumgebungen aufgebaut werden können. Das unfallfreie Testen von Prototypen, handwerklichen Aktivitäten oder Operationen ist neben der Aufrechterhaltung von Motivation und Aufmerksamkeit dabei das Hauptziel, das verfolgt wird.

Barta et al. (2025) stellen in ihrer Arbeit „Augmented reality experiences: Consumer‐centered augmented reality framework and research agenda” darüber hinaus eine Reihe von zukünftigen Forschungsfragen zur Verfügung, die Themen wie Nützlichkeit, gesellschaftliche Auswirkungen, Nutzung von generativer KI, Forschungsmethoden oder auch Marktpotenziale abdecken. Ein Auszug dieser Liste befindet sich in Anhang A.

## Head-Mounted Displays

### Begriff und Bedeutung

Head-Mounted Displays (HMDs) sind tragbare Anzeigegeräte, die direkt am Kopf des Nutzers befestigt werden und visuelle Inhalte unmittelbar vor den Augen darstellen. Sie ermöglichen ein immersives Erlebnis, bei dem der Nutzer in eine virtuelle (Virtual Reality, VR) oder erweiterte (Augmented Reality, AR) Umgebung eintaucht. Dies wird durch stereoskopische Darstellung erreicht, bei der für jedes Auge ein leicht versetztes Bild angezeigt wird, wodurch der Eindruck von Tiefe und Raum erzeugt wird. Dieser Effekt verstärkt das Gefühl der räumlichen Präsenz ("Presence"), also das Gefühl, tatsächlich in einer anderen Umgebung zu sein (Klinker et al., 2021, S. 56).

Ein entscheidendes Merkmal moderner HMDs ist das Head-Tracking. Dies beschreibt die Fähigkeit des Systems, die Kopfbewegungen des Nutzers zu erkennen und die virtuelle Umgebung entsprechend anzupassen. Dies geschieht mit Hilfe von verschiedenen Sensoren, die in das Headset integriert sind (Klinker et al., 2021, S. 62).

### Technische Funktionsweise

Die Funktionsweise von Head-Mounted Displays (HMDs) basiert auf mehreren Schlüsselkomponenten und Technologien, die zusammenarbeiten, um ein immersives Erlebnis zu schaffen:

**Displays und Linsen:**

HMDs verwenden kleine, hochauflösende Displays, die direkt vor den Augen des Nutzers positioniert sind. Um die visuelle Darstellung realistisch zu gestalten, sind diese Displays in der Regel mit speziellen Linsen ausgestattet. Diese Linsen, wie etwa Fresnel- oder Pancake-Linsen, verzerren das Bild, um eine klare und scharfe Darstellung trotz der geringen Distanz zwischen Display und Auge zu ermöglichen (Klinker et al., 2021, S. 60).

**Tracking und Sensorik:**

Eine der zentralen Technologien in HMDs ist das Tracking, das die Bewegungen des Nutzers in Echtzeit erfasst. Hierfür kommen verschiedene Sensoren wie Gyroskope (die Drehbewegungen messen) und Beschleunigungsmesser (die lineare Bewegungen erkennen) zum Einsatz. Diese Sensoren ermöglichen es dem HMD, die Bewegungen des Nutzers zu verfolgen und die virtuelle Umgebung so anzupassen, dass sie diesen Bewegungen entspricht. Häufig werden auch Magnetometer verwendet, um die Orientierung des Nutzers im Raum genauer zu bestimmen (Klinker et al., 2021, S. 62).

**Inside-Out-Tracking vs. Outside-In-Tracking:**

Beim Inside-Out-Tracking befinden sich die Kameras und Sensoren direkt im Headset. Diese Kameras erkennen die Umgebung des Nutzers und berechnen die Position und Orientierung des Geräts im Raum. Diese Technologie kommt oft bei modernen, autarken HMDs zum Einsatz (Klinker et al., 2021, S. 61).

Im Gegensatz dazu benötigt das Outside-In-Tracking externe Sensoren oder Kameras, die die Bewegungen des Nutzers im Raum verfolgen. Diese Technologie bietet eine höhere Präzision, da sie sich nicht nur auf das Headset, sondern auch auf externe Referenzen stützt (Klinker et al., 2021, S. 62).

**Passthrough-Technologie:**

Ein weiteres Merkmal moderner Mixed-Reality-Geräte ist die Passthrough-Technologie, bei der Kameras die reale Umgebung des Nutzers aufnehmen und auf den Displays des HMDs anzeigen. Diese Technologie ermöglicht es, dass der Nutzer seine Umgebung sehen kann, während virtuelle Inhalte darüber projiziert werden. So wird eine nahtlose Integration der realen und virtuellen Welt erreicht (Klinker et al., 2021, S. 217).

**Eye-Tracking:**

Einige fortschrittliche HMDs integrieren Eye-Tracking, eine Technologie, die die Blickrichtung des Nutzers erfasst. Dadurch können die dargestellten Inhalte an den Blickpunkt des Nutzers angepasst werden, was zu einer natürlicheren und immersiveren Interaktion führt. Eye-Tracking wird auch verwendet, um die Systemleistung zu optimieren, indem die Grafikressourcen gezielt auf den Bereich konzentriert werden, auf den der Nutzer schaut (Klinker et al., 2021, S. 122).

**Audiotechnologie und Haptisches Feedback:**

Moderne HMDs bieten häufig integrierte Audiotechnologien, wie z.B. räumliches Audio, das es dem Nutzer ermöglicht, die Herkunft von Geräuschen in der virtuellen Umgebung zu erkennen. Zusätzlich bieten viele Headsets haptisches Feedback, um die Immersion zu verstärken, indem sie Vibrationen oder andere taktile Reaktionen auf die Interaktion mit der virtuellen Welt zurückgeben (Klinker et al., 2021, S. 140).

### Verfügbare Geräte

Die Entwicklung der VR-Brillen entstand 1920 mit ersten Vorläufern in Form von Polarisationsbrillen. Das erste HMD, das sogenannte „Sword of Damocles“, wurde 1968 entwickelt und war so schwer, dass es an der Decke befestigt werden musste. Erst 1990 entstanden die ersten industriell genutzten Brillen, die Informationen in das Sichtfeld einblenden konnten. Allerdings waren diese immer noch teuer, schwer und technisch limitiert waren, sodass sie noch nicht für einen Massenmarkt zur Verfügung standen (Barff, o. J.; Manager, 2021). Erst 2012 wurden mit der Oculust Rift VR-Brillen für den Massenmarkt relevant. 2013 zog Google mit der Google Glass nach, scheiterten jedoch an der fehlenden gesellschaftlichen Akzeptanz und Cardboards boten preiswerte Alternativen (Zobel et al., 2018). Bis ins Jahr 2025 wurde die Hardware wurde deutlich leichtgewichtiger, benutzerfreundlicher und Akkulaufzeiten und Tragegefühl verbesserten sich zunehmend (Metamandrill, 2025; Molchanov, 2025). Brillen, die neu auf den Markt kommen bieten hochauflösende Displays, leistungsfähigere Chips und neben der Bedienung mit Controllern auch Augen- und Handtracking (Molchanov, 2025). Technische Fortschritte bestehen auch in höheren Bildwiederholraten und hochauflösender Grafik. Weitere Neuerungen sind die Einbindung von KI und die Entwicklung von MR-Brillen (Metamandrill, 2025). Diese gewinnen an Bedeutung, da sie einen nahtlosen Übergang zwischen realer und virtueller Welt ermöglichen (Metamandrill, 2025; Molchanov, 2025). Trotz sinkender Preise sind aber gerade hochwertige Produkte immer noch zu kostenintensiv für einen Massenmarkt (Molchanov, 2025).

Im Laufe der letzten Jahre haben sich verschiedene HMDs auf dem Markt etabliert. Diese Geräte variieren hinsichtlich ihrer Technologien, Funktionsweisen und Anwendungsmöglichkeiten. Hier ein Überblick über einige der aktuellen Modelle, die sich in den verschiedenen Bereichen wie Bildung, Gaming, Industrie und Mixed Reality etabliert haben:

## Virtuelle Tierbeobachtung

Die virtuelle Tierbeobachtung findet beispielsweise über Tiergärten und Zoos sowie weitere öffentliche Träger statt. Anwendungen umfassen von Videos auf Social-Media-Kanälen bis hin zu rein virtuellen Zoos unterschiedlichster Formen. Gründe hierfür sind vielfältig: Gefährlichen Tiere kann man auch in Zoos nicht nahe kommen, einige Tiere verstecken sich in ihren Gehegen und die Informationsbereitstellung mittels Schautafeln ist insbesondere bei jüngerem Publikum ineffektiv (Sukmawati et al., 2023).

Videos in den sozialen Medien können die Form von vorgedrehten Videos einzelner Tiere oder Livestreams rund um die Uhr und zu besonderen Zeiten annehmen. Diese Alternative hat sich insbesondere im Zuge der Einschränkungen der Covid 19-Pandemie, als Zoos geschlossen bleiben mussten und diese eine Alternative zum Zoobesuch bieten etabliert (Drews, 2020). Solche Liveaufnahmen sind nicht nur auf Zoos oder Tiergärten beschränkt, sondern können auch Aufnahmen von frei lebenden Tieren umfassen (Explore, o. J.).

Ein Bild, das Fenster, Gebäude, Mobiliar, Boden enthält.

Automatisch generierte BeschreibungEin Bild, das Boden, Im Haus, Fenster, Gebäude enthält.

Automatisch generierte BeschreibungNeben der Beobachtung von realen Tieren gibt es aber auch zahlreiche Varianten, die virtuelle Tiere in den Mittelpunkt stellen. Ein Beispiel sind AR-Einblendungen über das Smartphone. So bietet beispielsweise der WWF (World Wide Fund for Nature) eine solche Anwendung auf ihrer Webseite an und stellt darüber hinaus auch Informationen zu diesen Tieren zur Verfügung (WWF, 2024). Bei dieser Anwendung fehlt allerdings die Möglichkeit von Interaktionen, wie auf Abbildung 1 und Abbildung 2 zu sehen ist.

Abbildung 1 - Fehlerhafte Interaktion (WWF, 2024; eigene Aufnahme)

Abbildung 2 - Augmentierter Wolf (WWF, 2024; eigene Aufnahme)

XR-Erlebnisse können aber nicht nur in der eigenen Umgebung, sondern auch in Zoos selbst stattfinden. So bietet der Zoo Wuppertal Cardboards an, mit denen ein virtueller Rundgang durch einen Zoo erlebt werden kann (Stadt Wuppertal, o. J.).

Einen Schritt weiter geht der Anbieter Immotion, der ein VR-Fahrgeschäft anbietet. Hier kann bei einer Fahrt mit einer VR-Brille die Welt einer speziellen Tierart erlebt werden. Neben beispielsweise Gorillas werden auch Welten ausgestorbener Tiere angeboten (Immotion, o. J.; Zoo Leipzig, o. J.).

Aber auch rein virtuelle Zoos existieren, wie Beispiele aus China oder Belgien belegen (Gaia, o. J.; Nathalie, 2018). In diesen Zoos sind Betrachtung und Interaktion von und mit Tieren ohne Gefahren möglich. Zudem können auch hier ausgestorbene Tierarten wieder ins Leben gerufen werden. Ein weiterer Vorteil besteht darin, an keinen Ort gebunden zu sein und mit dem virtuellen Zoo an unterschiedliche Orte fahren zu können. Damit kann auch ein wichtiger Beitrag zum Tier- und Umweltschutz geleistet werden, indem Tiere in ihrer natürlichen Umgebung gezeigt werden, Aufklärung betrieben wird und konsequenterweise kein einziges Tier für einen solchen Zoo gefangen und eingesperrt werden muss (Gaia, o. J.).

## XR in Spielen

### Verfügbare Spiele

### Multiplayer

### Interaktion mit der Umgebung

# Anforderungen

Tabelle 1: Anforderungen

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Kategorie** | **Anforderung** | **Muss / Sollte / Kann** |
| 1.1 | Umgebung | Der Nutzer bewegt sich frei auf einer begehbaren Strecke | Muss |
| 1.2 | Umgebung | Die Anwendung enthält verschiedene begehbare Umgebungen (z. B. Savanne, Wald) | Kann |
| 1.3 | Umgebung | Jede Umgebung enthält einzigartige Tiere | Muss |
| 2.1 | Tiere | Tiere verhalten sich realistisch (z. B. Fluchtverhalten) | Sollte |
| 2.2 | Tiere | Verschiedene Tiere sind nur zu bestimmten Tageszeiten aktiv | Sollte |
| 2.3 | Tiere | Tiere erscheinen zufällig | Sollte |
| 2.4 | Tiere | Die Häufigkeit des Auftretens hängt von der Tierart ab | Kann |
| 3.1 | Kamera | Der Nutzer fotografiert Tiere mit einer virtuellen Kamera | Muss |
| 3.2 | Kamera | Die Kamera bietet Zoom, Fokus und Belichtungsfunktionen | Kann |
| 3.3 | Kamera | Blitzlicht für Nachtaufnahmen ist verfügbar | Kann |
| 3.4 | Kamera | Fotos können optional bewertet werden | Kann |
| 4.1 | Tagebuch | Alle Fotos werden in einem Entdeckertagebuch gespeichert | Muss |
| 4.2 | Tagebuch | Das Tagebuch zeigt zu jedem Foto einen Tiersteckbrief mit wissenschaftlichen Infos | Muss |
| 4.3 | Tagebuch | Das Tagebuch kombiniert echte Fotos mit animierten Bildern | Sollte |
| 4.4 | Tagebuch | Nutzer können besondere Aufnahmen als „seltene Entdeckungen“ markieren | Kann |
| 4.5 | Tagebuch | Tagebuch ist personalisiert und trägt den Namen des Nutzers | Muss |
| 5. | Tageszeiten | Die Tageszeit verändert sich dynamisch (z. B. Morgen, Mittag, Nacht) | Sollte |
| 6.1 | Plattform | Die App ist für die Meta Quest 3 konzipiert | Muss |
| 6.2 | Benutzeroberfläche | Steuerung der Kamera und Navigation durch das Tagebuch sind intuitiv | Sollte |
| 6.3 | Benutzeroberfläche | Nutzer können durch Wischgesten im Tagebuch blättern | Kann |
| 7.1 | Account | Nutzer muss Account erstellen können | Muss |
| 7.2 | Account | Nutzer muss sich einloggen können | Muss |
| 7.3 | Account | Jeder Nutzer hat ein persönliches Tagebuch mit eigenen Fotos | Muss |
| 8.1 | Mixed Reality | Virtuelle Tiere werden in der echten Umgebung dargestellt | Muss |
| 8.2 | Mixed Reality | Tiere verhalten sich realistisch | Muss |
| 8.3 | Mixed Reality | Umweltelemente (z. B. Bäume, Felsen) erscheinen in der realen Umgebung | Sollte |
| 8.4 | Mixed Reality | Nutzer können mit der Umgebung interagieren | Sollte |
| 9.1 | Belohnungssystem | Fotos werden bewertet und mit Punkten belohnt | Kann |
| 9.2 | Belohnungssystem | Seltene Tiere und Entdeckungen bringen mehr Punkte | Kann |
| 9.3 | Belohnungssystem | Neue Kamerafunktionen oder Umgebungen können freigeschaltet werden | Kann |
| 9.4 | Missionssystem | Spieler erhalten Fotografie-Herausforderungen | Kann |
| 9.5 | Missionssystem | Fortschritt und Erfolge werden im Tagebuch dokumentiert | Kann |
| 10.1 | Erweiterungen | Regelmäßige Updates mit neuen Umgebungen und Tieren | Kann |
| 10.2 | Multiplayer | Kooperativer Multiplayer-Modus | Sollte |

# Technologieauswahl

Die Wahl der passenden Technologien ist entscheidend für die erfolgreiche Umsetzung des Projekts „Entwicklung einer immersiven Mixed-Reality-Anwendung zur virtuellen Tierbeobachtung und -fotografie“. Da die Hardware bereits durch die Verfügbarkeit an der Hochschule festgelegt war, lag der Fokus auf der Auswahl der geeigneten Entwicklungsumgebung, Programmiersprache und des Architekturmodells. Im Folgenden werden die in Betracht gezogenen Technologien beschrieben und die Entscheidung für die jeweils gewählte Lösung begründet.

## Hardware

Für die Entwicklung einer Anwendung wurde die MetaQuest3 zur Verfügung gestelltDiese enthält einen Snapdragon XR2 Gen 2-Prozessor, der nahtloses und hochauflösendes Passthrough bieten soll.[[3]](#footnote-4) Passthroughfunktion beschriebt die Möglichkeit die reale Welt durch die Brille zu sehen und darüber virtuelle Elemente einzublenden. Dies wird bei der MetaQuest3 über zwei Farbkameras und einen Tiefenprojektor erreicht, die eine originalgetreue Ansicht der Umgebung bieten können.Die Brille besitzt ein 4K-Infinite-Display mit einer Auflösung von 2.064 x 2.208 Pixeln pro Auge und Aktualisierungsraten von 72Hz, 90 Hz beziehungsweise 120 Hz. Höhere Bildwiederholraten führen zu geringerem Risiko von Anzeichen der Simulationskrankheit — 70 Hz sind für einfache Anwendungen allerdings ausreichend. Die sogenannten „Pancake-Linsen“ helfen gegen Stör- oder Streulicht, jedoch ist das Tragen der Brille nur in Innenräumen vorgesehen. Auch die Erfassung von Gesten und damit eine intuitive Steuerung ohne Controller ist mit der MetaQuest3 möglich. Zudem sorgen die integrierten Stereolautsprecher für eine parallele Ansprache mehrerer Sinne und erreichen damit eine verbesserte Immersion. (*Meta Quest 3*, o. J.).

Diese Brille eignet sich also neben der Entwicklung von VR-Anwendungen auch zur Entwicklung von MR-Anwendungen und erfüllt damit die Voraussetzung für die weitere Entwicklung.

## Entwicklungsumgebung

Um einen Prototypen entwickeln zu können, ist eine Entwicklungsumgebung nötig, die die Erstellung von MR-Anwendungen ermöglicht. Im Folgenden wird ein Überblick über die Umgebungen Unity und Unreal Engine gegeben, sowie eine Bewertung hinsichtlich der Eignung zur Entwicklung eines Prototyps durchgeführt. Beide Umgebungen sind bekannt und werden häufig verwendet (Frank, 2022).

**Unity** Unity bietet eine Entwicklung mit C# und wird als einsteiger- und generell als benutzerfreundlich beschrieben (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Schnelles Prototyping soll mit Unity möglich sein und die Einarbeitung wird durch eine große Community und eine gute Dokumentation unterstützt (Dadson, 2023). Zudem bietet Unity einen kostenlosen Unity Student Zugang an und ist aufgrund geringerer Lohnkosten auch im kommerziellen Bereich in der Regel günstiger als die Unreal Engine (Dadson, 2023). Eine Entwicklung ist für mehrere Zielplattformen möglich und dank eines großen Asset Store und vielen Plugins, ist die Entwicklung von MR-Anwendungen mit Unity besonders geeignet (Frank, 2022; Rocketbrush, 2024). Auch für die vorgegebene MetaQuest3 bietet diese Entwicklungsumgebung Vorteile, bestehend aus einer guten Metaintegration inklusive Passthrough, umfassendem Support und einem einfachen Deployment (Satya, 2023). Weiter ist die Verwendung von Unity ein Industriestandard und vor allem in der Gamingindustrie weit verbreitet. Einer der Gründe ist die gute Skalierbarkeit von Unity (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Nachteilig sind allerdings weniger realistische Grafikqualitäten, was die Entwicklung von AAA-Spielen[[4]](#footnote-5) einschränkt (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Unity eignet sich also besonders für eine einfache und schnelle Entwicklung von Prototypen, bringt bei der Entwicklung grafisch hochwertiger Spiele allerdings Einschränkungen mit sich.

**Unreal** Die Unreal Engine bietet hingegen eine Entwicklung mit C++ und wird als komplexer im Setup und als überdimensioniert für einfache Anwendungen und Prototypen beschrieben. Zudem soll Unreal von einem schwierigeren Einstieg begleitet werden (Dadson, 2023; Rocketbrush, 2024). Auch sind die Entwicklungskosten mit Unreal höher, was hauptsächlich auf wenige erfahrene Entwickler und Entwicklerinnen zurückzuführen ist. Allerdings müssen bei erhöter Softwarekomplexität auch höhere Anforderungen an die Hardware und das Performance-Budget, und damit an das Gesamtbudget, gestellt werden (Dadson, 2023; Frank, 2022). Allerdings ermöglicht dies auch die Entwicklung von AAA-Spielen, da größere Dateien verarbeitet werden können und dadurch eine hohe Leistung und Performance bei großen Projekte zu beobachten ist (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Zwar ist die Entwicklungsumgebung komplex, allerdings wird die Einarbeitung durch eine sehr große Community, viel Support und sogenannte Blueprints erleichtert (Frank, 2022; Rocketbrush, 2024). Zudem ist der Quellcode für Unreal frei verfügbar und es entsteht somit die Möglichkeit die Engine an die eigenen Bedürfnisse anzupassen. Unreal eignet sich also besonders für die Entwicklung von grafisch hochwertigen Spielen, ist allerdings komplexer, was diese Engine ungeeigneter für die schnelle und einfache Entwicklung eines Prototyps macht.

Da Unity geeigneter für die Entwicklung von Prototypen ist und einen guten Support für die zur Verfügung gestellte Brille bietet, wird diese Entwicklungsumgebung abschließend gewählt. Zwar bietet Unreal eine höhere Grafikqualität und Leistung, jedoch ist die erhöhte Komplexität bei entsprechenden Alternativen nicht zu rechtfertigen.

## Programmiersprache

Die Wahl der Programmiersprache hängt direkt mit der gewählten Engine sowie den technischen Anforderungen der Zielplattform zusammen. In diesem Projekt wurde Unity als Engine in Kombination mit der Meta Quest 3 als Zielgerät gewählt. Dennoch ist es sinnvoll, auch die Programmiersprachen anderer gängiger Engines zu betrachten, um die Entscheidung besser einordnen zu können. Es existieren zudem auch Frameworks für andere Geräte wie die Apple Vision Pro, bei denen bevorzugt mit Swift und RealityKit gearbeitet wird. Diese Optionen wurden jedoch aufgrund der klaren Ausrichtung auf die Meta Quest 3 nicht weiter verfolgt (Dörner et al., 2023).

C# ist die zentrale Programmiersprache von Unity. Die Engine bietet eine enge Integration dieser Sprache in ihre Entwicklungsumgebung, was es ermöglicht, auf eine umfangreiche API zuzugreifen, die speziell für interaktive 3D-Anwendungen – wie sie auch im Bereich der Mixed Reality zum Einsatz kommen – konzipiert ist (Freeman & Wagner, 2023). C# zeichnet sich durch eine gut strukturierte Syntax, objektorientierte Konzepte und eine hohe Lesbarkeit aus. Diese Eigenschaften erleichtern sowohl die initiale Entwicklung als auch die spätere Wartung und Erweiterung der Anwendung (Dörner et al., 2023).

Die Unreal Engine hingegen nutzt primär C++, eine Sprache, die eine sehr direkte Kontrolle über Speicher und Hardware bietet und damit insbesondere bei performancekritischen Anwendungen Vorteile aufweist. Gleichzeitig bringt C++ jedoch eine höhere Komplexität mit sich – insbesondere im Hinblick auf manuelles Speichermanagement und die allgemeine Fehleranfälligkeit (Stroustrup, 2022).

Zusätzlich zu C++ bietet Unreal mit Blueprints eine visuelle Programmiersprache an. Blueprints ermöglicht es, Logiken und Interaktionen auf visuelle Weise zu implementieren, ohne direkt in den Quellcode eingreifen zu müssen. Dieses Konzept eignet sich insbesondere für Prototyping oder für Entwickler:innen ohne tiefergehende Programmierkenntnisse (Johnson, 2023).

JavaScript wird primär im Webumfeld eingesetzt. Zwar existieren Frameworks, die eine Nutzung von JavaScript für 3D-Visualisierungen ermöglichen (z. B. Babylon.js oder A-Frame), jedoch ist die Sprache in Bezug auf Performance und direkte Engine-Integration in Unity ungeeignet. Die fehlende native Unterstützung in Unity würde zu zusätzlichen Hürden in der Entwicklung führen (Freeman & Wagner, 2023).

Nach Abwägung der verschiedenen Optionen stellt C# in Verbindung mit Unity die geeignetste Wahl für unser Projekt dar. Die geplante Anwendung soll eine Mixed-Reality-Erfahrung mit Tieren ermöglichen, bei der Interaktionen, Animationen und ein immersives Benutzererlebnis im Vordergrund stehen. Unity stellt hierfür eine stabile und ausgereifte Plattform bereit, während C# durch seine Flexibilität, Wartbarkeit und direkte Engine-Anbindung die technischen Anforderungen unseres Projekts optimal erfüllt.

## Multiplayer Plugin

<https://docs.normcore.io/essentials/why-normcore>

https://www.photonengine.com/fusion

## Architekturmodell

Zur Wahrung klarer Zuständigkeiten und zur Reduktion von Komplexität wird auf etablierte Entwicklungs- und Architekturmuster gesetzt. Wie in Kapitel 5 Konzept noch genauer erläutert werden wird, erfolgt hierfür eine Aufteilung in Frontend und Backend. Im Frontend sollen nachher all die Funktionen sein „die auf der Brille laufen“, also Szenen, animierte Tiere und Interaktionen. Die Datenverarbeitung und -bereitstellung hingegen, liegen im Verantwortungsbereich des Backends. Diese Aufteilung erleichtert die Trennung von Verantwortlichkeiten, reduziert die Komplexität und verbessert dadurch Lesbarkeit, Wartbarkeit und Testbarkeit des Codes.

Damit auch im Frontend und Backend selbst der Programmcode getrennt und damit test- und wartbar bleibt, ist es sinnvoll auf bewährte Architekturmodelle zurückzugreifen. Neben klassischen Aspekten sind auch XR-spezifische Anforderungen zu berücksichtigen, wie sich zum Beispiel bei Greifen eines Objekts zeigt: Es müssen für diesen Vorgang die Hände und die Position der Finger erkannt werden. Weiter muss eine physikalische Simulation erfolgen, die auch eine Kollisionserkennung erfordert. So dürfen Finger nicht durch ein Objekt hindurchgreifen und virtuelle Objekte nicht in realen Objekten verschwinden. Dies erfordert visuelles Feedback sowie eine Aktualisierung des Objektzustands, etwa der Positionskoordinaten. Im Folgenden wird ein Überblick über die Architekturmodelle Model-View-Controller (MVC), Model-View-ViewModel (MVVM) und Entity Component System (ECS) gegeben und eine Entscheidung getroffen, welches Modell bei den gegebenen Anforderungen als geeignet eingestuft wird.

**MVC** MVC ist ein Architekturmodell, welches aus den drei Schichten Model, View und Controller besteht, die jeweils unterschiedliche Aufgaben kapseln. Das Model enthält die Datenstruktur und ist für die Verbindung zu den Daten zuständig. Die View enthält visuelle Elemente und das User Interface (UI). Der Controller stellt die Koordinationsschnittstelle zwischen diesen beiden Schichten dar, indem er die Kommunikation zwischen Model und View anstößt. Zu beachten ist, dass Model und View dabei miteinander und nicht über den Controller kommunizieren (Capilla, 2004; Majdak, 2023; Tuch, 2024). Vorteilhaft ist die klare Trennung von UI und Logik, was einen schnellen Entwicklungsprozess fördert und dank des strukturierten Gerüsts auch die Planung, Testbarkeit und Wartbarkeit vereinfacht (Tuch, 2024). An dieser Stelle spielen wiederverwendbare Komponenten und eine effizientere Zusammenarbeit im Team durch die Möglichkeit paralleler Entwicklung eine große Rolle. So erleichtert dieses Architekturmodell auch die Integration neuer Technologien oder Module, was die Organisation größerer Projekte unterstützt. Beispielsweise hat ein Entwicklerteam von Unity selbst schon mit diesem Architekturmodell gearbeitet, um eine VR-Anwendung zu erstellen (Majdak, 2023; Tuch, 2024; Unity Developers, o. J.).

Allerdings ist Unity ist eine stark komponentenbasierte Umgebung, was im Kontext von XR-Anwendungen zu einer schwierigeren Trennung der einzelnen Funktionalitäten und einer Verschmelzung von Controller und View führt (Majdak, 2023; Vasconcelos, o. J.). So wird der Programmcode bei komplexeren XR-Anwendungen schnell unübersichtlich und die Views können schnell an Umfang zunehmen. Zudem steigt die Komplexität mit der Menge an Daten, was den Schluss zulässt, dass dieses Architekturmodell für kleine Anwendungen geeignet ist, bei komplexeren Projekten hingegen schnell eine Alternative benötigt wird (Majdak, 2023; Vasconcelos, o. J.).

Jedoch existieren auf dem MVC-Prinzip weiterentwickelte Architekturmodelle, die an XR-spezifische Bedarfe angepasst sind. So entwickelten (Benbelkacem et al., 2019) mit “MVC-3D” eine Erweiterung, um XR-Anforderungen zu adressieren. Kernaspekte waren das Hinzufügen einer “Library”-Komponente und das Weiterführen der View zur “interactive View”. Die Library kapselt komplexe Algorithmen, wie sie für Tracking, Gestenerkennung oder weitere Simulationen benötigt werden und die interaktive View enthält neue Komponenten, sodass sie nicht nur Nutzereingaben, sondern auch das Gerät und die Umgebung berücksichtigen kann. Dank der Auslagerung komplexerer Prozesse bleibt der restliche Programmcode übersichtlich und einfach wartbar. Änderungen an Algorithmen oder am Gerät selbst erforderten lediglich Anpassungen an der Library-Komponente. Zudem können mit dieser Architektur 3D-Informationen und komplexe Interaktionen besser gehandhabt werden (Benbelkacem et al., 2019). 2020 wurde dieses Modell um die Möglichkeit zur Entwicklung kollaborativer weiterentwickelt. Das „MVC-3DC“-Modell integriert Aspekte wie Kollaboration (Multiplayermodus), Verteilung, Interoperabilität und die Zusammenarbeit verschiedener Entwicklungsstellen und Systeme (Benbelkacem et al., 2020). Um ein kollaboratives System zu entwickeln, führen sie drei Methoden an: Centralized Mode, Duplicated Mode und Hybrid Mode (Benbelkacem et al., 2020; Dewan, o. J.; Fleury, o. J.). Alle diese drei Methoden beschreiben die Art und Weise eines möglichen Zusammenspiels zwischen einem Client und einem Server und stellen somit nicht zwangsläufig eine Erweiterung des Architekturmodells an sich dar.

**MVVM** MVVM ist ebenfalls ein aus drei Schichten bestehendes Architekturmodell. Das Model (M) ist ebenfalls für die Datenhaltung beziehungsweise Bereitstellung zuständig, die View (V) enthält ebenfalls visuelle Informationen und das ViewModel (VM) stellt die Brücke zwischen diesen beiden Schichten dar (Dhawan, 2024; Majdak, 2023; Stonis, 2024). Bei der MVC-Architektur steuert der Controller aktiv UI-Eingaben wohingegen der Datenfluss beim ViewModel in der MVVM-Architektur automatisch durch sogenanntes „Data Binding“ gesteuert wird. Data Binding ermöglicht eine automatische Änderung von UI-Elementen bei Änderung der Daten und umgekehrt eine automatische Manipulation der Daten bei einer Nutzereingabe. Es ist also keine manuelle Definition und Steuerung eines Datenflusses nötig (adegeo, o. J.). Dies ermöglicht eine noch sauberere Trennung von UI und Logik, was aus Abbildung 3 deutlich hervorgeht: Wohingegen bei MVC eine Kommunikation zwischen View und Model stattfindet, ist bei MVVM das ViewModel der Vermittlungspunkt zwischen diesen beiden Schichten.

Ein Bild, das Text, Screenshot, Diagramm, Design enthält.

Automatisch generierte Beschreibung

Abbildung 3 - Unterscheidung MVC und MVVM (MVVM Pattern on Android | WOXAPP, o. J.)

Eines der Hauptmerkmale MVVMs besteht in der strikten Trennung und damit fehlenden Kopplung zwischen Anzeige und Daten. Dies macht dieses Architekturmodell insbesondere in UI-lastigen Anwendungen beliebt, da eine strikte Trennung von Verantwortlichkeiten die Zusammenarbeit im Team erleichtert (Dhawan, 2024; Majdak, 2023; Stonis, 2024). Das Modell kann wiederum aber auch für datengetriebene Anwendungen eingesetzt werden. Dank der strengen Entkopplung kann sich einer Vielzahl von Quellen bedient werden, ohne die View ändern zu müssen (Majdak, 2023). Folgende Vorteile gelten nicht nur für MVC, sondern auch für MVVM: erhöhte Wartbarkeit, Förderung der Wiederverwendbarkeit, effiziente Zusammenarbeit im Team und eine noch bessere Skalierbarkeit.

Nachteile des MVVM-Musters bestehen hauptsächlich in der Komplexität, da dieses Muster für kleine Anwendungen überdimensioniert sein und sich in der Einarbeitung etwas schwieriger gestalten kann. Häufig wird die Einarbeitung in das Data Binding als Ursache genannt (Majdak, 2023; Vasconcelos, o. J.). Zudem wurde das MVVM-Muster lange Zeit sehr selten in Verbindung mit Unity genutzt, da bis Oktober 2024 auf externe Lösungen zurückgegriffen werden musste (Majdak, 2023; Oriz, 2024; Vasconcelos, o. J.). Aktuell gibt es zudem keine (etablierten) Weiterentwicklungen von MVVM, die spezielle Anforderungen von XR-Anwendungen adressieren.

**ECS** Neben den bekannten, plattformunabhängigen Architekturmodellen bietet Unity selbst ein natives Modell an. Mit ECS können über 1.000 Objekte gleichzeitig verarbeitet werden, was das Modell sehr geeignet für große Anwendungen macht (Unity, o. J.). Jedoch ist dieses Modell laut Unity selbst hinsichtlich der XR Interaction Toolkit-Kompatibilität noch nicht ausgereift, was die Interaktionsmöglichkeiten einschränken kann (Unity, o. J.). Somit ist davon auszugehen, dass die XR-spezifischen Anforderung des Prototyps nicht ausreichend erfüllt werden könnten. Zudem ist dieses Architekturmodell vor allem für erfahrene Unityentwickler und –entwicklerinnen geeignet, die anspruchsvolle Spiele entwickeln wollen. Dies bedeutet auch ein anspruchsvolleres Debugging und eine eingeschränkte Teamarbeit, bietet aber erhebliche Performancevorteile (Unity, o. J.). Da Interaktionen nicht in dem Umfang möglich sind, wie sie für die Entwicklung des Prototyps benötigt werden, wird dieses Architekturmodell für die folgende Entscheidung und Entwicklung des Prototyps nicht weiter berücksichtigt.

Um die Aufgaben von Frontend und Backend bestmöglich zu unterstützen, wird jeweils ein eigenes Architekturmodell gewählt.

Da das Backend keine UI-lastigen Funktionen erfüllen muss und auch keine strenge Entkopplung von View und Model benötigt wird, ist die Verwendung des MVC-Musters geeignet. Eine Trennung von Funktionen und eine Entkopplung finden mit einer geringeren Komplexität statt. So beinhaltet das Backend wenige und klar definierte Schnittstellen sowie wenige bis keine gleichzeitigen Zustandsänderungen. Der Controller enthält also wenige und wenig komplexe Funktionen, daher ist die Verwendung des MVC-Musters ausreichend und MVVM wird aufgrund der erhöhten Komplexität für einen zeitgleich geringen Umfangs des Backends verworfen.

Im Gegensatz dazu ist insbesondere bei einer XR-Anwendung das Frontend in einem sehr viel höheren Maße UI- und damit interaktionslastig. So müssen beispielsweise 3D-Menüs im Raum schweben, Textfelder eingeblendet oder animierte Objekte angezeigt werden. Zudem können Interaktionen unterschiedliche Formen annehmen. Für die Menüauswahl können Controller, Handgesten oder die Stimme verwendet werden und Objekte können ebenfalls mit Controllern oder Handbewegungen manipuliert werden. Die Verwaltung von Interaktionen, multimodaler Eingabe und damit stetiger Zustandsänderungen erhöht somit auch die Komplexität des Frontends und benötigt eine Möglichkeit für einfach synchronisierbare Views. Eine Herausforderung in der Verwendung von MVVM im XR-Kontext besteht im Data Binding. Dieses musste bis zur Veröffentlichung von Unity 6 im Oktober 2024 extern entwickelt und in Unity eingebunden werden.[[5]](#footnote-6) Das native Data Binding in Unity ist also verhältnismäßig neu und so existieren derzeit noch wenige Erfahrungswerte oder Hilfestellungen(Oriz, 2024; Wong, 2025) Da die Verwendung des nativen Data Bindings auf der einen Seite komplex, auf der anderen Seite aber möglich ist und den Vorteil einfach synchronisierbarer Views verspricht, eignet sich das Architekturmodell somit für Anwendungen mit stetig wechselnden Zuständen.

Da für den zu entwickelnden Prototypen Interaktionen zwischen realen und virtuellen Elementen elementar sind, wird entsprechend ein Architekturmodell benötigt, was für UI- und interaktionslastige Anwendungen geeignet ist und datengetriebene Änderungen einfach ermöglicht. Damit eine gute Immersion möglich wird, muss das Architekturmodell die Reaktivität der Anwendung in hohem Maße unterstützen. Da der Entwicklungsaufwand nicht unverhältnismäßig hoch sein soll und auch die Anwendung selbst so einfach wie möglich bleiben soll, müssen Faktoren wie beispielsweise eine klare Trennung von Funktionalitäten oder die Wiederverwendbarkeit einzelner Komponenten berücksichtigt werden. So besteht zudem eine große Notwendigkeit von automatisierter Synchronisation vor manuell gehandhabten Zustandsänderungen, damit Fehlerquellen und Entwicklungsaufwand reduziert werden können. Dies kann mit Hilfe von Data Binding ermöglicht werden. Das Greifen eines Objekts würde mit einer MVC-Umsetzung die Erkennung der Hand, eine manuelle Berechnung der Physik und ein manuell eingebautes Feedback an die Person erfordern. Bei einer Umsetzung mit MVVM kann die Handposition gebunden werden, was bedeutet, dass View-Updates automatisch getriggert werden. So können View und Model ohne manuellen Aufwand konsistent gehalten werden, was entsprechend die Fehlerquellen bei der Entwicklung reduziert. Konsistenz mittels Data Binding, gute Interaktionsmöglichkeiten dank erhöhte Reaktivität und eine strukturierte Trennung von Verantwortlichkeiten rechtfertigen die erhöhte Komplexität des Architekturmodells. Bereits genannte Vorteile (Trennung von Daten und UI/Verantwortlichkeiten, Testing, Wartbarkeit) bieten zudem die Möglichkeit Models und ViewModels unabhängig von Unity zu testen, da ausschließlich die View plattformabhängig ist. Somit kann dieses Architekturmodell als für die Entwicklung geeigneter als MVC eingestuft werden und wird als Basisarchitekturmodell für das Frontend festgelegt. Es unterstützt zudem eine plattformunabhängigere Entwicklung und adressiert damit eine zentrale Herausforderung aktueller XR-Anwendungen (Molchanov, 2025).

Damit bei der Entwicklung auch spezielle Anforderungen für XR-Anwendungen adressiert werden können, wird dieses Architekturmodell auf Basis von (Benbelkacem et al., 2019) und (Benbelkacem et al., 2020) erweitert. Um Tracking und komplexere Berechnungen zu kapseln wird dem klassischen MVVM eine „Library“-Schicht hinzugefügt. Weiter wird die Trennung in Frontend und Backend als Voraussetzung für die Möglichkeit von kollaborativen Systemen verstanden. Damit Interaktionen zwischen realen und virtuellen Elementen möglich werden, soll die View Komponenten beinhalten, die auf unterschiedliche Nutzereingaben wie beispielsweise Gesten oder auch Controller reagieren können. Eine ausführliche Erläuterung des weiterentwickelten Architekturmodells findet sich in Kapitel 5.1.3 Frontend. Kernaspekte der Weiterentwicklung sind also eine zusätzliche Schicht, zusätzliche Komponenten in der View und die Aufteilung in Frontend und Backend als Teil des Architekturmodells.

# Konzept

## Aufbau der Anwendung

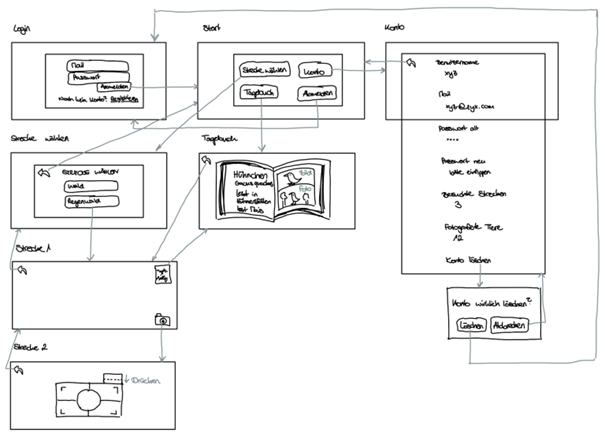
Einen Überblick über die graphische Benutzeroberfläche und damit die geplante Funktionsweise, sowie Navigationsmöglichkeiten der Anwendung bietet Abbildung 4.

Abbildung 4 - Mockup der Anwendung (eigene Zeichnung)

Die Benutzeroberfläche startet mit der Login-Szene, in der sich Personen mit E-Mail und Passwort anmelden oder neu registrieren können. Nach erfolgreichem Login erscheint das Hauptmenü (Szene „Start“) mit den Optionen Strecke wählen, Konto, Tagebuch und Abmelden.

Über die Schaltfläche Konto gelangt man in die Kontoszene, in der sich persönliche Daten bearbeiten und speichern lassen. Hier besteht zudem die Möglichkeit, das Konto zu löschen – sämtliche zugehörigen Daten werden entfernt, und es erfolgt automatisch eine Rückleitung zur Login-Szene. Wird eine Registrierung in der Login-Szene ausgewählt, führt dies direkt in die Kontoszene, in der alle erforderlichen Accountdaten eingegeben und gespeichert werden müssen. Von dort kann jederzeit über den Zurück-Pfeil in die Startszene navigiert werden.

Die Option Strecke wählen öffnet eine Szene, in der verschiedene Umgebungen – etwa Wald, Antarktis oder Regenwald – auswählbar sind. Jede Strecke bietet eigene Tiere und Umgebungselemente. Nach Auswahl einer Strecke beginnt das Spiel (Szene „Strecke 1“), in dem Tiere einzeln oder zeitgleich zufällig erscheinen können. Ihr Verhalten orientiert sich am realen Vorbild: Eine Schnecke kriecht unbeirrt über den Weg, während ein Reh bei Unterschreiten einer bestimmten Distanz flieht. Im MR-Kontext stellt die Interaktion zwischen virtuellen Elementen und realer Umwelt eine besondere Herausforderung dar: Beispielsweise darf ein Reh nicht durch einen realen Baum laufen oder in der Luft schweben, sondern muss korrekt verdeckt werden und auf dem realen Untergrund stehen. In Szene „Strecke 1“ können Nutzer außerdem ins Tagebuch oder zur Kamera wechseln.

Die Kamerafunktion (Szene „Strecke 2“) blendet eine virtuelle Kamera ins Sichtfeld ein. Über den Auslöser lassen sich Fotos von virtuellen Tieren aufnehmen, die automatisch im Tagebuch gespeichert werden. Jeder Tagebucheintrag umfasst eine Doppelseite mit Namen, Lebensraum, realistischem Bild, Aufnahmefoto und Datum der Aufnahme. Neue Aufnahmen erweitern das Tagebuch fortlaufend und bieten einen Überblick über alle entdeckten Tiere.

Ein Spiel kann jederzeit beendet werden, indem man im Hauptmenü die Option Abmelden wählt, was zur Login-Szene zurückführt.

Um die beschriebenen Funktionen und Interaktionen technisch umzusetzen, ist eine durchdachte Softwarearchitektur erforderlich. Die folgenden Abschnitte beschreiben daher den konzeptionellen Aufbau des Prototyps und zeigen, wie eine MR-Anwendung entwickelt werden kann, die animierte Tiere realitätsnah in den physischen Raum integriert und ihre Interaktion mit der Umgebung ermöglicht.

## Grundlegende Architektur

Im Folgenden wird die Architektur des Prototyps konzeptionell erläutert. Ziel ist es prototypisch eine MR-Anwendung zu entwickeln, die es ermöglicht, über die zur Verfügung gestellte MetaQuest3, animierte Tiere im realen Raum zu sehen und zu fotografieren. Darüber hinaus soll eine Interaktion zwischen dem animierten Tier und der realen Umgebung ermöglicht werden, zum Beispiel indem es bei einer zu geringen Distanz flieht und verschwindet. Bei der Entwicklung sollen mit der Architektur die Faktoren Skalierbarkeit und Multiplayermodus schon berücksichtigt werden, auch wenn diese nicht direkt implementiert werden. Der Faktor Skalierbarkeit soll über lose gekoppelte Schichten erreicht werden, sodass einfach Erweiterungen an Programm und Programmcode vorgenommen werden können. Zudem können mit diesem Faktor einzelne Schichten auf unterschiedlichen Systemen laufen und so kann bei Bedarf Rechenleistung hinzugezogen werden. Damit eine Einarbeitung schnell erfolgen kann und Erweiterungen einfach implementiert werden können, wird ein strukturierter und leicht verständlicher Aufbau vorausgesetzt. Um einen Multiplayermodus zu ermöglichen, wird eine zentrale Datenbasis (Server), eine Synchronisation zwischen darauf zugreifenden Parteien (Clients) und eine konfliktfreie Zustandsverwaltung benötigt. Aus diesen Gründen soll der Prototyp in drei Schichten unterteilt werden. Wie bereits angemerkt sollen die beiden Schichten Frontend und Backend unter Verwendung geeigneter Architekturmuster entwickelt werden.

Die erste Schicht ist die Datenbasis, sie enthält ausschließlich die Daten. Mit der Entkopplung der Datenbasis zum restlichen Programm kann die nachstehende Datenbank und Speicherumfang frei gewählt werden, was Skalierbarkeit ermöglicht und eine Voraussetzung für beispielsweise den Multiplayermodus ist. So können mehrere Personen zur selben Zeit auf die Daten zugreifen und mit entsprechenden Maßnahmen bleibt die Basis konsistent. Für eine Datenbank kann auf unterschiedliche Formen zurückgegriffen werden. Diese können Clouddiensten, On-Premise-Datenbanken oder lokale Simulationen umfassen. Eine weitere Unterscheidung kann zwischen Datenbanken, die strukturierte Daten in Form einer Tabelle enthalten, und jenen, die beispielsweise Dateien oder andere Medien enthalten, gezogen werden. Eine einfache Migration von einer Datenbank in eine andere ist nur dann möglich, wenn beide denselben Typ Daten speichern können.

Zugriff und Manipulation auf die Datenbank können geschützt über das Backend erfolgen. Das Backend, die zweite Schicht, stellt sicher, dass nur berechtigte Personen und Programme auf die für sie freigegebenen Daten zugreifen und diese manipulieren können. Die Implementierung einer rollenbasierten Autorisierung oder generellen Authentifizierung findet also im Backend statt. Zudem ermöglicht das Backend auch die Verarbeitung der Daten, was beispielsweise das umfängliche Löschen von Daten und Referenzen umfasst. Soll ein Account gelöscht werden, kann über das Backend auch die Löschung des zugehörigen Tagebuchs und der entsprechenden Fotografien und Tagebucheinträge erfolgen. Weiter kann das Backend komplexe Spiellogiken umfassen oder für die Erstellung einer Spieleinstanz zuständig sein. Nicht zuletzt stellt das Backend die Daten für die letzte Schicht bereit. Vorteilhaft bei dieser Trennung ist die Möglichkeit, die Logik von der Benutzeroberfläche getrennt zu halten und dabei unterschiedliche Technologien nutzen zu können. Beispielsweise kann für das Backend eine andere Programmiersprache verwendet werden als für das Programm, das die Daten später anzeigt. Für die Kommunikation der Schichten wird eine Schnittstelle benötigt, die unabhängig von Plattform oder Programmiersprache angesprochen werden kann. Eine solche Möglichkeit bieten standardisierte APIs (Application Programming Interface), welche die Kommunikation mehrerer Anwendungen, oder wie im vorliegenden Fall Schichten, ermöglichen. Eine API stellt eine strukturierte Programmschnittstelle dar, über die zum Beispiel das Frontend auf bestimmte Funktionen oder Daten zugreifen können. Die Kommunikation erfolgt dabei in der Regel über das Hypertext Transfer Protocol (HTTP), also das gleiche Protokoll, das auch dem Aufruf von Webseiten zugrunde liegt. Durch die Einhaltung etablierter Webstandards ist die API unabhängig vom verwendeten Frontend nutzbar, was langfristig zur Wartbarkeit und Skalierbarkeit beiträgt.

Die letzte Schicht ist das Frontend. Es ist dafür zuständig, die bereitgestellten Daten (graphisch) darzustellen. Diese Schicht beinhaltet die Benutzeroberfläche[[6]](#footnote-7) und somit die Schnittstelle zwischen Anwendung und Mensch. Da auch die Darstellung von Inhalten einer gewissen Logik bedarf, enthält auch das Frontend Logiken. Diese beziehen sich aber auf die Präsentation von UI-Elementen und dürfen nicht mit der Backend-Logik verwechselt werden.

Vorteile einer solchen Architektur sind infolge der Trennung von Verantwortlichkeiten und einer damit einhergehenden Komplexitätsreduktion eine bessere Lesbarkeit, Wartbarkeit oder auch Testbarkeit des Programmcodes. So können einzelne Funktionen beispielweise im Frontend unabhängig vom jeweiligen System getestet oder gewartet werden, da nur ein kleiner Teil, die Anzeige, direkt an das System gekoppelt ist. Ein weiterer Vorteil ist das einfache Austauschen von Funktionen, wenn diese entkoppelt sind. So betreffen zum Beispiel Änderungen am UI nur die Anzeige und nicht das gesamte System. Auch die Wiederverwendbarkeit einzelner Funktionen steigt, da sich diese an mehreren Stellen wiederverwenden lassen. Zudem erleichtert ein solcher Aufbau die Zusammenarbeit im Team, da Funktionen mit geringen Abhängigkeiten entwickelt werden können. Für spätere Stadien sind auch die Skalierbarkeit und ein Multiplayermodus bereits benannte Faktoren, die mit dieser Architektur ermöglicht werden. Eine detailliertere Erläuterung der einzelnen Schichten erfolgt in den nachfolgenden Unterkapiteln.

### Datenbank

In einer produktiven Anwendung übernimmt die Datenbank eine zentrale und persistente Rolle als „Single Source of Truth“, was insbesondere für kollaborative Nutzung essenziell ist. Sie muss Konsistenz, Integrität und Verfügbarkeit gewährleisten, damit alle Beteiligten verlässlich mit denselben Daten arbeiten können. Eine klare Entkopplung der Datenhaltung zum restlichen System ermöglicht Skalierbarkeit sowie technologische Flexibilität. Darüber hinaus sind Zugriffskontrollen notwendig, um Sicherheit und modulare Zuständigkeiten zu gewährleisten.

Bei der Entwicklung des Prototyps liegt der Fokus auf der Weiterentwicklung eines Architekturmodells und der Implementierung von interaktiven Aspekten einer MR-Anwendung. Er liegt dabei explizit nicht auf der dauerhaften Verfügbarkeit von Daten oder der Performanz von Datenbanken und Abfragen. Somit sind die genannten Aspekte insbesondere für eine produktive Umgebung relevant, können allerdings für die Entwicklung eines Prototyps vernachlässigt werden. Daher wird eine Datenbank mit mehreren lokal gespeicherten Dateien simuliert. Dieses Vorgehen beziehungsweise diese Art der Datenhaltung ist in produktiven Systemen nicht ausreichend, ermöglicht aber eine schlanke und flexible Datenhaltung ohne zusätzlichen Konfigurationsaufwand. Dies wird im Kontext der Prototypentwicklung als ausreichend angesehen. CSV-Dateien (Comma-Separated Values) bieten hierfür eine einfache und gut lesbare Möglichkeit, strukturierte Daten wie Tabellen abzubilden. Sie sind ohne zusätzliche Software auswertbar, leicht versionierbar und plattformunabhängig nutzbar. Die gewählte Struktur der Daten erlaubt es darüber hinaus, die CSV-Dateien bei Bedarf in spätere relationale Datenbanksysteme zu überführen. Ein produktives System kann beispielsweise ein Cloudspeicher, wie beispielsweise von Google oder Amazon sein (*Amazon RDS für SQL Server-Datenbanken in der Cloud*, o. J.; *Cloud SQL for MySQL, PostgreSQL und SQL Server*, o. J.).

Bilder und Videos sind für die Anwendung ein relevanter Bestandteil und werden vorerst ebenfalls lokal gespeichert, aber auch hier ist das spätere Speichern in zum Beispiel Objekt-Datenbanken möglich (*Cloud-Objektspeicher – Amazon S3 – AWS*, o. J.).

### Backend

Das Backend wird mit der MVC-Struktur implementiert . Um den Zugriff auf die Datenbank zu kapseln, wird jedoch das „Repository“ als weitere Schicht eingefügt. Das Repository kapselt den Zugriff auf die Datenbank, beziehungsweise auf die lokale Datenbanksimulation. Ziel ist es, damit ein klar strukturiertes und modulares Backend zu implementieren, das so nicht nur erweiterbar und einfach wartbar ist, sondern zusätzlich noch Skalierbarkeit ermöglicht. Das MVC-Muster trennt dabei die Verantwortlichkeiten in Datenstruktur, Logik und Interaktionsschnittstelle. Im Folgenden werden die einzelnen Schichten genauer beleuchtet.

**Repository** Das Repository kommuniziert mit der Datenbank und übernimmt somit das Laden, Speichern, Aktualisierung und Löschen von Datensätzen. Dies erfolgt nahezu unabhängig davon, ob es sich bei der Datenbank um eine produktive Datenbank oder eine Simulation handelt. Da die Daten in strukturierter Form vorliegen, kann über eine entsprechende Abfragesprache auf sie zugegriffen werden. Eine solche Sprache wäre die Structured Query Language (SQL), oder in C# beispielsweise Language Integrated Query (LINQ), welches sich an SQL orientiert. Diese Schicht ist auch dafür zuständig, nach Extraktion der Daten aus der Datenbasis, diese in strukturierte und nutzbare Objekte zu schreiben. Damit sind Objekte gemeint, die vom aktuellen Programm weiterverarbeitet werden können. Mit dieser Umsetzung bleibt die Abfragelogik von der konkreten Speicherform entkoppelt, was spätere Technologie- oder Speicherwechsel vereinfacht.

**Model** Das **Model** beschreibt die Struktur der Datenobjekte des Prototyps, mit denen das Backend arbeitet. Es enthält keine Logik, sondern definiert lediglich die Eigenschaften und den Aufbau der Objekte (Klassen), die für die Programmlogik benötigt werden. Beispielsweise kann das die Klasse „User“ mit den Eigenschaften wie ID, Name oder E-Mail sein. Das Model wird also in Zusammenarbeit mit dem Repository verwendet, um aus externen Datenquellen gültige Objekte zu erzeugen, die dann im Controller weiterverarbeitet werden können.

**View** Die View wird im Backend nicht als graphische Benutzeroberfläche implementiert, sondern liegt in Form von API-Endpunkten vor. Diese Endpunkte stellen die sichtbare Oberfläche des Backends dar, indem sie den Zugriff auf ausgewählte Funktionen und Daten ermöglichen und als Kommunikationsschnittstelle genutzt werden. So kann beispielsweise das Frontend auf die Daten einzelner Tiere zugreifen oder über einen Endpunkt einen neuen Account erstellen. Die technische Umsetzung erfolgt über standardisierte REST-Schnittstellen, was die Interoperabilität mit anderen Systemen und Plattformen sicherstellt.

**Controller** Die zentrale Stelle im Backend des Prototyps ist der Controller. Dieser stellt nicht nur die Kommunikation mit der Außenwelt her, sondern kommuniziert auch mit dem Model und dem Repository und stößt somit die Kommunikation mit der Datenbasis an. So fungiert der Controller als Koordinationsinstanz, die Datenfluss, Verarbeitungslogik und Sicherheit kontrolliert. Zusätzlich enthält der Controller weitere Logik. Sollte eine Person einen Account anlegen, wird so beispielsweise durch den Controller auch das Erstellen eines zugehörigen Tagebuchs angestoßen. Im Rahmen einer möglichen Erweiterung zum Multiplayersystem wäre der Controller zudem die zentrale Stelle zur Synchronisierung von Zuständen, zur Validierung eingehender Änderungen durch verschiedene Clients und zur Konfliktvermeidung.

Die klare Trennung in Repository, Model, View und Controller ermöglicht es,Änderungen oder Erweiterungen gezielt auf einzelne Schichten zu beschränken. Bei einer Migration der Datenbasis muss nur eine Anpassung des Repositorys umgesetzt werden. Änderungen an Controller oder Model sind nicht notwendig. Ebenso kann eine Erweiterung um Authentifizierungslogik oder Validierung der Daten klar abgegrenzten Bereichen erfolgen, was die Wartbarkeit und Erweiterbarkeit des Systems langfristig sichert.

### Frontend

Dem Frontend liegt als Basisarchitekturmodell das MVVM-Muster zugrunde. Um XR-spezifischen Anforderungen wie Interaktionsmöglichkeiten gerecht zu werden, wird dieses nach (Benbelkacem et al., 2019) zu MVVM-3D erweitert. In Verbindung mit dem Backend erfüllt es zudem die Voraussetzung für eine Erweiterung zu MVVM-3DC nach (Benbelkacem et al., 2020). Im Folgenden wird ein detaillierterer Überblick über MVVM-3D gegeben und Implementierungsmöglichkeiten werden für MVVM-3DC dargelegt.

MVVM-3D teilt sich in die Schichten Model, View, ViewModel auf, wird allerdings um eine Library-Schicht ergänzt. Die folgenden Abschnitte bieten einen Überblick über diese vier Komponenten.

**Model** Das Model enthält die Struktur der Daten, beschreibt also, wie Objekte aufgebaut sind und welche Eigenschaften sie haben. Beispielsweise definiert das Model die Datenklasse Tier und legt die Eigenschaften Name, Art, Lebensraum fest. Models enthalten darüber hinaus allerdings keine Logik. Da das Frontend keine eigenen Objektdaten speichert, sondern vom Backend empfängt, benötigt das Frontend eine Möglichkeit auf die API des Backends zuzugreifen. Damit Verantwortlichkeiten deutlich getrennt bleiben, wird eine Schicht namens „Services“ als Schnittstelle zum Backend eingeführt und dem Model zugeordnet. Diese kapselt API-Abfragen und erstellt mit den empfangenen Daten Objekte. Wiederverwendbarkeit und Wartbarkeit spielen auch an dieser Stelle eine große Rolle. So kann ein Service zum Beispiel von mehreren Models genutzt werden oder aber mehrere Model-Objekte erzeugen. Somit hat eine Änderung am Model nicht zwangsläufig eine Änderung am Service zur Folge und umgekehrt betreffen Änderungen am Service nicht automatisch auch die Datenstruktur. Das Model beschreibt an der Stelle also wie ein Objekt aufgebaut ist (Klasse) und ein Service erstellt diese Objekte auf Basis realer Daten (Instanzen). Diese zusätzliche Komponente ist dient lediglich der Vereinfachung des Programmcodes und soll nicht explizit eine Erweiterung des Architekturmodells darstellen, da ein Einsatz des Architekturmodells in Szenarien denkbar ist, in denen keine API-Abfragen benötigt werden.

**View** Die View enthält alle UI-Elemente. Im MR-Kontext sind neben beispielsweise Buttons und Textfelder auch 3D-Darstellungen oder Animationen eingeschlossen. So zeigt die View beispielsweise ein Tier basierend auf seinen Model-Daten an oder stellt einer Person ein Menü mit Buttons zur Navigation zur Verfügung. Die View enthält eine Benutzeroberfläche, die einem Menschen die Möglichkeit zur Interaktion bietet. Diese Interaktion darf im Zuge der Anpassung an XR-Anforderungen nicht nur ein Klickevent sein, wie es in beispielsweise Web- oder mobilen 2D-Anwendungen der Fall ist, sondern muss darüber hinausgehen. So sind neben Controllereingaben (analog zur Maus) auch Eingaben mit den Händen, den Augen oder der Stimme relevant. Um für Tiere zum Beispiel einen Fluchtverhalten zu ermöglichen, muss auch die Position einer Person beziehungsweise die Position des Geräts selbst als Nutzereingabe verstanden werden. Dies kann über Tracking-Systeme der XR-Hardware erreicht werden. Die View muss daher zwingend für diese multimodale Interaktion Komponenten beinhalten, die diese Nutzereingaben verwalten können.

**Library** Die Library ist eine für das MVVM-3D-Muster neu eingeführte Schicht, die komplexe Berechnungen kapselt, die das ViewModel verkomplizieren würden. Beispiele für solche Anwendungsfälle sind das Tracking von Objekten, Verhalten bei Interaktion und damit auch die Bewegungslogik eines Tieres. So kann die konkrete Implementierung von Bewegungsabläufen für jedes Tier separat implementiert werden und das ViewModel ruft bei einem bestimmten Abstand zu einer Person oder in einer vordefinierten Situation einen bestimmten Bewegungsablauf auf.

**ViewModel** Das ViewModel ist das zentrale Steuerungselement in der Architektur. Zentral für diese Funktion sind sogenannte Events[[7]](#footnote-8), Observer-Patterns[[8]](#footnote-9) und auch Reactive-Patterns[[9]](#footnote-10). Das ViewModel empfängt Änderungen von View oder Model, verarbeitet diese und leitet sie weiter. Dabei müssen View und Model nicht direkt miteinander kommunizieren. Beispielsweise empfängt das ViewModel bei der Änderung einer E-Mail-Adresse die neue Adresse und aktualisiert automatisch das Model, wodurch wiederum eine Modeländerung dem ViewModel gemeldet wird und infolgedessen automatisch die View und damit die Anzeige geändert wird. Dafür müssen die Verbindungen zwischen View und ViewModel zur Laufzeit überwacht werden. Dies geschieht, indem sogenannte Properties im ViewModel mit UI-Elementen verknüpft werden. Properties können als Felder oder Variablen verstanden werden, die zusätzlich eine Logik enthalten. Diese Logik kann zum einen den Zugriff auf die Variable steuern, oder aber die Möglichkeit bieten ein UI-Element zu verknüpfen. Diese Verknüpfung ermöglicht eine automatische Synchronisation von Eingabe und Variablenwert bei Änderungen (Stonis, 2024). So kann im vorigen Beispiel das Eingabefeld für die E-Mail-Adresse mit einer E-Mail-Property im ViewModel verknüpft werden. In diesem Ablauf sorgt dann das Event bei der Änderung der E-Mail-Property dafür, dass im Rahmen des Observer-Patterns alle relevanten Elemente informiert werden, und das Reactive-Pattern übernimmt die direkte Synchronisation zwischen Property, Model und View. Ein weiterer Vorteil des Architekturmodells, neben diesem Verhalten und der strikten Trennung von Aufgabenbereichen, besteht in der Möglichkeit Berechnungen aus der Library hinzuzuziehen. Ein Beispiel hierfür ist die Flucht eines Tieres, bei der dem ViewModel eine Nutzeraktion in Form einer zu geringen Distanz zwischen Person und Tier, von der View gemeldet werden kann. Das ViewModel setzt dann die Logik „Flucht“ um und nutzt dabei die Fluchtimplementierung des jeweiligen Tieres aus der Library-Komponente. Ist der Bewegungsablauf abgeschlossen, kann das ViewModel dem Model die neue Position des Tieres weiterleiten, die daraufhin an das Backend weitergeleitet und dort in einer Datenbank gespeichert wird. Diese Änderung an der Datenbasis, und damit am Model, wird zurück an das ViewModel geleitet. Dort wird eine Aktualisierung der Anzeige, also View, angestoßen. Umsetzbar ist dieses Verhalten mittels Data Binding, welches UI-Elemente automatisch mit den entsprechenden Properties verknüpft. Wie bereits erläutert, kann sich bei Änderung an der Datenbasis die Anzeige automatisch aktualisieren und umgekehrt bei einer Nutzereingabe automatisch die Datenbasis ändern.[[10]](#footnote-11)

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte BeschreibungEine zusammenfassende Darstellung der MVVM-3D-Architektur und Zusammenhänge ist in Abbildung 5 dargestellt.

Abbildung 5 - MVVM-3D (eigene Darstellung)

Um neben XR-spezifischen Anforderungen auch die Entwicklung eines kollaborativen XR-Systems und damit die Grundlagen für einen Multiplayer-Modus zu ermöglichen, wird nachfolgend die Weiterentwicklung von MVVM-3D zu MVVM-3DC nach (Benbelkacem et al., 2020) ausgeführt.

Wie im Kapitel 4.5 Architekturmodell schon angemerkt, gibt es für die Entwicklung von kollaborativen Systemen drei Methoden. Mit der Grundstruktur von Datenbank, Backend und Frontend ist allerdings die Methode „Centralized Mode“ sehr geeignet. Diese Methode beschreibt einen zentralen Server als „Single Source of Truth“ und mehrere Clients, die jeweils auf denselben Datenstand zugreifen (Benbelkacem et al., 2020; Dewan, o. J.; Fleury, o. J.). Das Backend und die Datenbank stellen im Kontext des Prototyps den Server dar, der alle Anfragen und Datenoperationen verarbeitet. Mehrere Instanzen des Frontends übernehmen die Rolle von unterschiedlichen Clients und können unabhängig voneinander auf das Backend/den Server zugreifen. Dieses Modell ermöglicht es mehreren Benutzern, gleichzeitig mit derselben Anwendung zu interagieren, wobei Konsistenz und Synchronisation durch den zentralen Server gewährleistet werden. Für eine Umsetzung würde dies bedeuten, dass zum Beispiel Koordinaten von Tieren oder getrackte Gesten zwingend an das Backend gesendet werden und dort entweder in einem temporären oder persistenten Speicher gehalten werden müssen. Weiterhin wird ein Mechanismus benötigt, der Änderungen an die Clients propagiert, damit diese synchronisiert werden können. Die Kommunikation über zyklische GET-Anfragen ist hierfür nicht ausreichend, da sie nicht nur ineffizient, sondern auch anfällig für Verzögerungen und Synchronisationsprobleme ist. Dies setzt nicht nur eine Anfrage-basierte Kommunikation voraus (Pull-Prinzip), sondern auch eine Realisation von serverseitigen Benachrichtigungen der Clients (Push-Prinzip). Da das Backend, sowie die relevanten Teile des Frontends (Services) in C# implementiert werden, ist eine C# beziehungsweise .NET-Lösung ausreichend. Eine Möglichkeit in diesem Kontext wäre der Einsatz von SignalR. Diese Bibliothek ermöglicht es, serverseitige Ereignisse unmittelbar an die verbundenen Clients weiterzuleiten, wodurch eine bidirektionale Kommunikation zwischen Server und Clients hergestellt wird. Dabei können Nutzeraktionen auf mehreren Clients in Echtzeit synchronisiert werden (Fletcher, 2023).

Wenn in einer späteren Multiplayeranwendung eine Person eine Weitere auf ein Tier aufmerksam machen möchte, ergibt sich folgender Ablauf: Eine Zeigegeste der einen Person wird von der View als Nutzereingabe registriert und diese wird an das ViewModel weitergeleitet und dort verarbeitet. Die daraus entstehende Änderung an den Daten wird an das Model weitergereicht, wodurch die Änderungen über die Services an das Backend weitergeleitet werden. Die Daten werden dort verarbeitet und verändern die Datenbasis. Diese Änderung führt zu Propagation dieser Änderungen an die Services aller Clients. Diese Änderungen erreichen über die Services die ViewModels, wodurch eine automatische Aktualisierung der View erfolgt. In diesem Falle wird das Tier, auf das gedeutet wurde, bei der zweiten Person auf der Brille hervorgehoben. Der Ablauf einer solchen Kollaboration ist in Abbildung 6 dargestellt.

Ein Bild, das Text, Diagramm, parallel, Plan enthält.

Automatisch generierte Beschreibung

Abbildung 6 - Ablauf Kollaboration (eigene Darstellung)

Diese Weiterentwicklung soll in dieser Arbeit jedoch ausschließlich konzeptionell betrachtet werden, eine prototypische Umsetzung findet mit dem MVVM-3D-Modell statt. Details der Umsetzung können dem nachfolgenden Kapitel entnommen werden.

## Interaktion

# Prototypenentwicklung

In den nachfolgenden Unterkapiteln werden einige Aspekte der Prototypimplementierung erläutert. Dabei wird insbesondere auf die Umsetzung von MVVM-3D und Interaktionen eingegangen.

## Datenbank

Für eine Implementierung der Datenbasis wurden insgesamt fünf CSV-Dateien angelegt, lokal gespeichert und über GitHub versioniert. So war ein Austausch der Daten möglich, ohne auf ein produktives System zurückgreifen zu müssen. Alle Daten liegen in textform vor und können später problemlos in eine reale, relationale Datenbank überführt werden. Abbildung 7 zeigt den Aufbau und die Zusammenhänge der Datenbasis. Aus der Abbildung gehen die Klassen Animal, User, Diary und Diaryentry, sowie die Verbindungsklasse DiaryDiaryentry hervor.

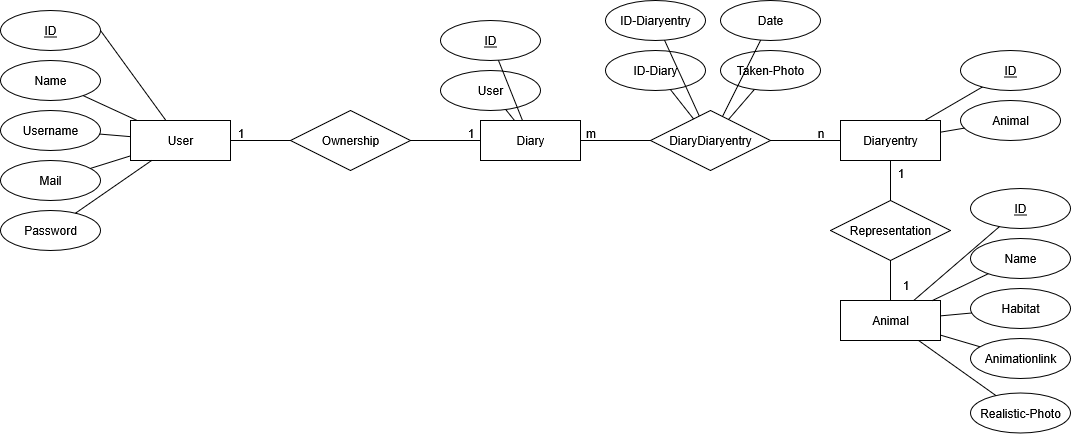


Abbildung 7 - ER-Diagramm Datenbank (eigene Darstellung)

Die Klasse Animal hält Informationen zu den Tieren, die virtuell auf der Brille eingeblendet werden sollen: ID, Name und Lebensraum des Tieres, sowie die Pfade zum animierten 3D-Objekt und zum realistischen Foto[[11]](#footnote-12) Damit die Tabelle im weiteren Verlauf in einer relationalen Datenbank gespeichert werden kann und da Bilder und Videos in einem speziellen Unity-Ordner abgelegt werden, werden für Medien lediglich die Pfade zu den Ablageorten gespeichert.

Die Klasse User enthält die Daten User-ID, Name, Username, Mail und Passwort. Das Speichern von Passwort und Mail als Klartext kann für eine produktive Anwendung eine Sicherheitslücke bedeuten. Daher ist es sinnvoll mit der Überführung der Daten in ein produktives System, einen externen Dienst, wie zum Beispiel Google Firebase, einzubinden. Die Authentifizierung kann dann darüber abgewickelt und sensible Daten können dort sicher gespeichert werden.

Die Klasse Diary enthält die Informationen ID und zugehöriger User. Diese Klasse kann zusätzlich einfach erweitert werden und Personalisierungseinstellungen wie Farbe oder Schriftart hinzufügen. Für jeden User ist nur ein Tagebuch vorgesehen.

Die Klasse Diaryentry enthält die Informationen ID und ID des zugehörigen Tieres. Diese Klasse stellt eine eigene Entität dar, da ein Tagebuch aus mehreren Einträgen besteht und nicht nur aus Verbindungen zu Tieren. Damit Tagebucheinträge nicht für jeden User angepasst werden müssen, sondern standardisiert für jedes Tier gelten, enthält ein Eintrag keine userspezifischen Daten wie Datum einer Aufnahme oder die Aufnahme selbst. Eine Optimierungsmöglichkeit besteht in der Übertragung des Attributs „realistisches Foto“ von Animal in Diaryentry, da dieses Attribut ausschließlich im Kontext eines Tagebucheintrags verwendet wird. Für jedes Tier ist nur ein Tagebucheintrag vorgesehen. Ein Eintrage kann hingegen mit mehreren Tagebüchern in Verbindung stehen und umgekehrt kann ein Tagebuch mehrere Einträge enthalten.

Die Klasse DiaryDiaryentry stellt eine Verbindungsklasse zwischen einem userspezifischen Tagebuch und einem tierspezifischen Tagebucheintrag dar und enthält verbindungsspezifische Informationen wie ID des Tagebuchs, ID des Eintrags, Datum und den Speicherpfad zu einem aufgenommenen Foto. Diese Verbindungsklasse ist notwendig, da ein Tagebuch mehrere unterschiedliche Einträge enthalten kann, ein Tagebucheintrag wiederum in unterschiedlichen Tagebüchern aufgenommen werden kann. Datum und aufgenommenes Foto personalisieren den Tagbucheintrag, ohne dass der Tagebucheintrag selbst angepasst werden muss. Dies ist zudem eine Möglichkeit, den benötigten Speicherplatz zu reduzieren, da keine redundanten Informationen gespeichert werden.

Die gewählte Struktur erlaubt eine klare Trennung zwischen generischen Inhalten und nutzerspezifischen Informationen. Gleichzeitig bleibt sie erweiterbar und leicht in eine relationale Datenbank überführbar, was die Skalierbarkeit und spätere Migration unterstützt.

## Backend

Das Backend wurde mit C# implementiert, weiter wurden LINQ-Ausdrücke, sowie das Framework ASP .NET verwendet.

Alle Repositories sind gleich aufgebaut. Sie implementieren eine Verbindung zu den jeweiligen CSV-Dateien und die Möglichkeit Daten aus diesen Dateien auszulesen. Dafür wird die Datei geladen und jede Zeile repräsentiert die Informationen eines individuellen Objekts. Die jeweiligen Spalten stellen die zugehörigen Attribute dar. Um ein verwertbares Objekt zu erstellen, werden die entsprechenden Informationen einer Zelle auf die Attribute eines C#-Objekts gemappt. Entsprechend werden Änderungen gespeichert, indem Attributwerte eines C#-Objekts in entsprechende Zellen der CSV-Datei geschrieben werden. Welche Objekte genau aus der Datenbasis geladen werden, wird über LINQ-Ausdrücke definiert. LINQ ermöglicht es im Programmcode, SQL-ähnliche Abfragen an eine Datenbasis zu senden. So ist es beispielsweise möglich alle Objekte einer Tabelle zu laden, oder nur solche, die eine bestimmte Bedingung erfüllen. Mit dem Ausdruck „FirstOrDefault“ wird beispielsweise das erste Tier aus der Datenbank gelesen, welches eine übereinstimmende ID mit der Abfrage hat. Da jede ID einzigartig ist entspricht das erste Tier mit einer bestimmten ID gleichzeitig dem einzigen Tier mit einer bestimmten ID.

Wie bereits ausgeführt enthalten die Models lediglich die Datenstruktur. Um die enthaltenen Klassen allerdings instanziieren oder in die Datenbank schreiben zu können, sind für jedes Attribut Getter- und Setter-Methoden implementiert.

Die Controller haben eine Verbindung zu den Repositories implementiert und erhalten darüber die Daten, die dem Frontend zur Verfügung gestellt werden sollen. Für diese Bereitstellung wurde das Framework ASP.NET Core verwendet. Dieses Framework ist eine moderne, plattformunabhängige Technologie von Microsoft zur Entwicklung von Webanwendungen und REST-APIs (*What Is ASP.NET Core?*, o. J.). Zudem fügt sich ASP.NET nahtlos in das C#-Ökosystem ein, was eine konsistente Entwicklung über Backend und Datenverarbeitung hinweg ermöglicht. Beispielsweise können Endpunkte wie „GetAll“ zur Verfügung gestellt werden, sodass alle verfügbaren Objekte einer Tabelle geladen werden. Aber auch Endpunkte, hinter denen eine komplexere Logik steht, sind implementiert. So steht für die Klasse Tier im Animal-Controller der Endpunkt „Create“ zur Verfügung, worüber das Frontend die Erstellung eines neuen Tieres anfragen kann. Der Controller erstellt hierfür eine neue ID, indem über das Repository alle bestehenden Tiere und damit alle vergebenen IDs ausgelesen werden und die nächste freie Nummer verwendet wird. Als zweites wird das neue Tier mit der neuen ID über das Repository in die Datenbank geschrieben. Parallel wird mit der neuen Tier-ID ein neuer Tagebucheintrag erstellt. Hierfür wird über das Diaryentry-Repository analog eine neue Tagbucheintrag-ID vergeben und diese beiden Informationen in der Datenbank gespeichert. Hingegen bei einer Löschanfrage für ein Tier, werden zuerst alle verknüpften Tagebucheinträge aus der Datenbank abgefragt und gelöscht und erst wenn dies abgeschlossen ist, wird das Tier aus der Datenbank gelöscht. So werden verwaiste Referenzen vermieden und der Speicherbedarf minimiert.

Ein Bild, das Text, Diagramm, Schwarzweiß, parallel enthält.

Automatisch generierte BeschreibungIn Abbildung 8 sind alle verwendeten Models, Repositories und Controller mit zugehörigen Attributen und Methoden aufgeführt.

Abbildung 8 - UML-Diagramm Backend (eigene Darstellung)

## Frontend

### Setup der Unity-Umgebung für die Meta Quest 3

Für die Entwicklung der Anwendung wird die Unity-Engine eingesetzt. Die Installation erfolgt über den Unity Hub. Dieser dient als zentrale Verwaltungsoberfläche für unterschiedliche Unity-Versionen und Projekte. Über den Unity Hub wird die ausgewählte Unity-Version inklusive der erforderlichen Module für die Zielplattform Android installiert. Dazu zählen das Android Build Support Modul sowie das Android SDK, NDK und OpenJDK. Diese sind für die Ausführung der Anwendung auf der Meta Quest 3 notwendig.

Die Meta Quest 3 basiert auf einem Android-Betriebssystem. Daher werden Anwendungen als APK-Dateien benötigt. Aus diesem Grund wird im Unity Editor das Build-Target auf Android umgestellt. Die Einstellungen umfassen unter anderem die Ausrichtung auf die ARM64-Architektur und die Aktivierung des XR-Plugins. Das XR-Plugin ermöglicht die Integration von Funktionen für Virtual- und Augmented Reality. Dazu gehören das Tracking von Headset und Controllern sowie die Darstellung der virtuellen Umgebung.

Die Übertragung der Anwendung auf die Meta Quest 3 setzt die Aktivierung des Entwicklermodus voraus. Zuerst wird das Headset mit einem Meta-Konto verknüpft. Anschließend erfolgt die Aktivierung des Entwicklermodus über die Meta Horizon App auf einem verbundenen Smartphone. Danach kann Unity die erstellte APK-Datei direkt auf das Gerät installieren und starten.

Die Übertragung der Anwendung auf die Meta Quest 3 erfolgt über die „Build and Run“-Funktion. Das Headset muss dazu per USB-C-Kabel mit dem Entwicklungsrechner verbunden sein. Unity erstellt während des Buildprozesses eine APK-Datei, installiert diese automatisch auf dem Gerät und startet die Anwendung. Dieser Vorgang ermöglicht eine schnelle und einfache Testmöglichkeit. So können Fehler frühzeitig erkannt und Anpassungen effizient umgesetzt werden.

### Implementierung

Das Frontend wurde in großen Teilen mit C# entwickelt, hauptsächlich in der View wurde der Programmcode um spezielle Befehle für Unity ergänzt. In Unity werden Benutzeroberflächen und 3D-Umgebungen in sogenannten „Scenes“ organisiert, die als einzelne Anwendungsabschnitte fungieren. Es ist möglich auf einer graphischen Oberfläche zum Beispiel einen Button an eine bestimmte Position zu verschieben und dort zu beschriften oder Form, Farbe und Platzhalter zu verändern. Diese Änderungen können im Inspector vorgenommen werden. Der Inspector zeigt Eigenschaften eines ausgewählten Elements an und bietet nicht nur die Möglichkeit für graphische Anpassungen, sondern auch für Verknüpfungen von UI-Elementen und beispielsweise Variablen.

Die Frontend-Models spiegeln die Backend-Datenstrukturen wider und enthalten neben der Beschreibung der Eigenschaften ebenfalls Getter- und Setter-Methoden. Um die Eigenschaften im Unity-Inspector sichtbar und serialisierbar zu machen, werden sie mit dem Attribut „SerializeField“ versehen. Für die automatische Aktualisierung der UI werden zusätzlich Events und Properties genutzt.

In den Services werden diese Models für die Erstellung von Objekten verwendet. Ein zentraler Service initialisiert einen HTTP-Client, der für die Kommunikation mit den verschiedenen API-Endpunkten des Backends verwendet wird. Welche Endpunkte aufgerufen werden, bestimmen die spezifischen Services. So ruft beispielsweise der User-Service den API-Endpunkt für „lade alle User“ auf und erhält als Antwort alle User zurück. Diese werden dann instanziiert, in eine serialisierbare Liste geschrieben und an die ViewModels weitergegeben. Ein Service ist dabei nicht zwangsläufig einem ViewModel zugeordnet, sondern kann von mehreren ViewModels angesprochen werden.

Die ViewModels implementieren die Logik, die als Schnittstelle zwischen View und Model notwendig ist. Dabei wurde reines C# verwendet, was die Wiederverwendbarkeit und Testbarkeit erhöht. Eine Aufgabe des ViewModels ist beispielsweise die API-Abfrage in den Services anzustoßen. Darüber erhält das ViewModel die entsprechenden Objekte, schreibt es sie in sogenannte „ObservableCollections“. Damit können Änderungen automatisch erkannt und UI-Elemente direkt aktualisiert werden. Für die Integration in Unity wird dies mit einem PropertyChanged-Mechanismus kombiniert. Neben dem Laden, Aktualisierung und Löschen von Daten, ist das ViewModel aber auch für speziellere Logiken zuständig. Das UserViewModel setzt zum Beispiel das Data Binding um, was im weiteren Verlauf dieses Kapitels noch ausführlicher betrachtet wird. Das DiaryViewModel hingegen ist für die Zusammenführung von Tagebuch, Einträgen, personalisierten Verbindungen und den zugehörigen Tieren zuständig. Dies umfasst neben einem initialen Laden auch das „Blättern“ im Tagebuch. Das AnimalViewModel hingegen setzt die Steuerung und den Bewegungsablauf eines Tieres um. Aufgrund der unterschiedlichen Implementierungen für jedes Tier und der damit einhergehenden Komplexität, befindet sich die konkrete Implementierung in der Library-Schicht. Auch das ViewModel ist nicht einem einzelnen Service oder einer einzelnen View zugeordnet, sondern kann von unterschiedlichen Views und Services angesprochen werden.

Die Library ist die Schicht, die komplexe Algorithmen und Berechnungen kapselt, um das ViewModel schlank zu halten. So implementiert die Library die genannten Bewegungsabläufe und Animationstrigger, sowie eine MonoBehaviourBridge. „MonoBehaviour“ ist eine unityspezifische Basisklasse, die unterschiedliche Funktionen anbietet, welche für die Animation eines Tieres unerlässlich sind. Da das ViewModel in C# geschrieben und dabei weitestgehend unabhängig von der Plattform Unity ist, benötigt es eine Möglichkeit dennoch MonoBehaviour-Funktionen aufzurufen. Die MonoBehaviourBrücke ist genau dafür zuständig. Sie nimmt das Objekt, dessen Funktion aufgerufen werden soll und erstellt daraus ein unitykompatibles Objekt sodass die entsprechende Funktion dann ausgeführt werden kann. Diese Brücke könnte auch im ViewModel selbst implementiert werden. Um die Unityintegration im C#-Code möglichst gering zu halten und so die Plattformunabhängigkeit zu fördern, befindet sie sich allerdings getrennt in der Library.

Die View-Schicht ist für die Verknüpfung von UI-Elementen und den dahinterliegenden Funktionen im ViewModel zuständig. Damit ist diese Schicht die Einzige, die vollständig unityspezifisch ist. Wie bereits beschreiben, kann über den Inspector beispielsweise ein Button mit einem Event, einer Variablen oder einer Funktion der entsprechenden View verknüpft werden. Diese Funktionen können dann einen Szenenwechsel auslösen oder komplexe Logiken ausführen. Ein Wechsel der Szene wird direkt in der View implementiert, komplexere Logik wie Speichern von Userdaten, die Anmeldung oder das Blättern im Tagebuch werden im ViewModel implementiert, aber aus der View heraus aufgerufen. Damit Funktionen im ViewModel aufgerufen werden können, erben alle Views von der unityspezifischen Basisklasse MonoBehaviour, die Lebenszyklusmethoden wie Start() bereitstellt. In dieser Funktion wird nicht nur die Verbindung zum ViewModel hergestellt, sondern es wird auch auf die Daten des aktuellen Users zugegriffen, damit diese in der aktuellen Szene verfügbar sind. Unity selbst bietet keine Möglichkeit, bei einem Szenenwechseln Parameter wie den aktuellen User zu übergeben. Daher ist eine Hilfe mit einer Klasse „SessionData“ und einem Attribut „User“ umgesetzt worden. Für Weiterentwicklungen ist es möglich diese Klasse um zum Beispiel Spielstände zu erweitern, damit diese nicht bei einem Szenenwechsel gelöscht werden.

Ein Überblick über diesen Aufbau und die resultierenden Verknüpfungen zeigt Abbildung 9.

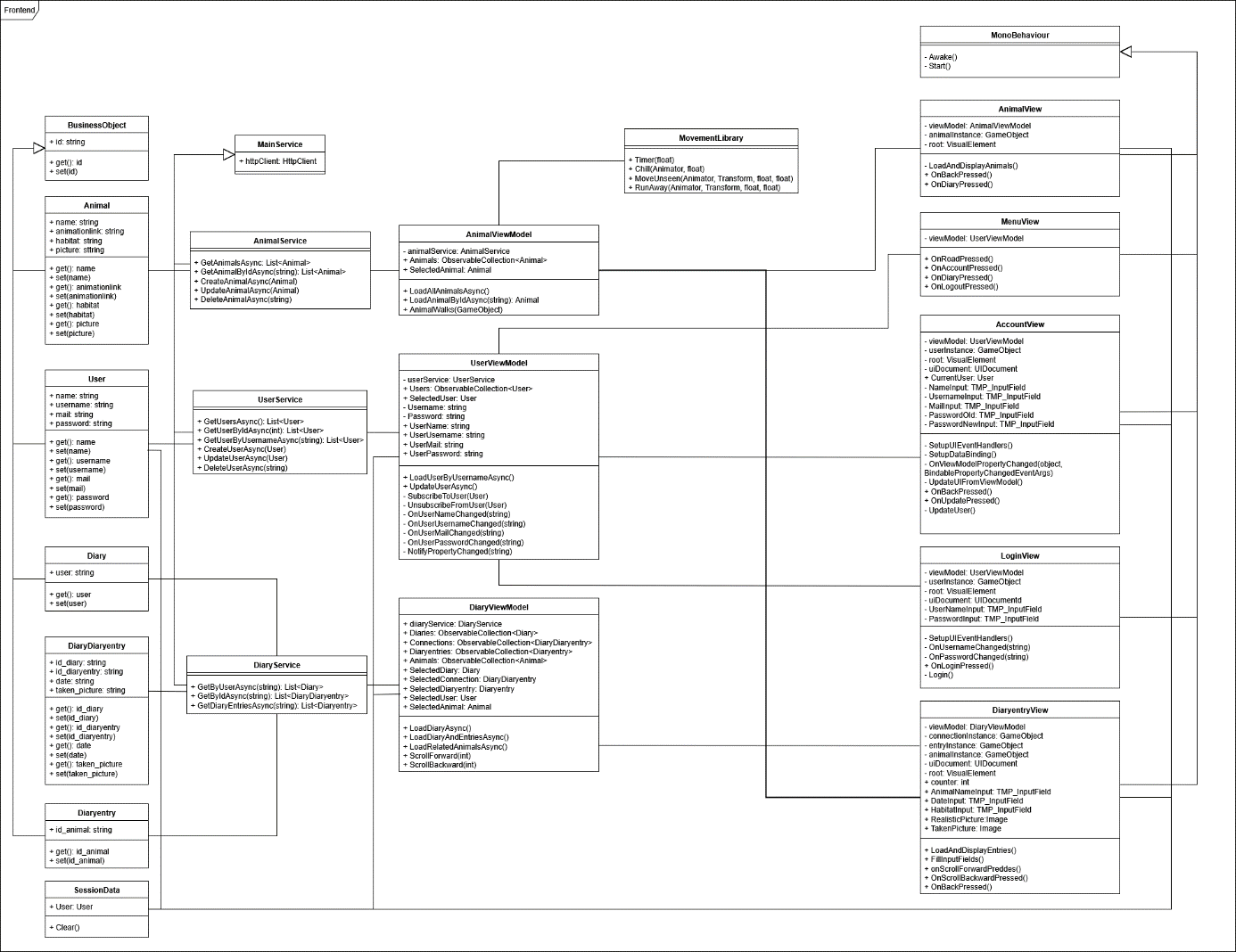


Abbildung 9 - UML-Diagramm Frontend (eigene Darstellung)

Mit diesem Aufbau sind Verantwortlichkeiten nach dem MVVM- beziehungsweise MVVM-3D-Muster getrennt. Zentral für das Zusammenspiel zwischen View und ViewModel, und damit der zentrale Vorteil dieser Architektur, ist das Data Binding. Wie im Kapitel 4.5 Architekturmodell erläutert, ermöglicht es automatische Änderungen an der UI bei Änderungen an Properties und umgekehrt die direkte Übernahme von Benutzereingaben in das ViewModel. Nachfolgend wird die Umsetzung von Data Binding in Unity am Beispiel des Anwendungsfalls „Passwort ändern“ erläutert, da jenes elementar für die Funktion der weiterentwickelten Architektur ist.

Für den Prototypen wurde das Data Binding mittels UI Toolkit und einer Implementierung mit Properties und Events realisiert. Die relevanten Felder im User-Model sind mit „SerializeField“ und „Serializable“ versehen, damit Unity sie serialisieren und in der graphischen Oberfläche anzeigen, weiter mit Funktionen oder Variablen verknüpfen kann. Für die Synchronisierung zwischen View und ViewModel verwenden die ViewModels ObservableCollections und implementieren das Interface INotifyBindablePropertyChanged. Über dieses Interface können Änderungen an Properties sofort als Event gemeldet und daraufhin die UI aktualisiert werden. Das Speichern eines Passworts verdeutlicht diesen Ablauf. Die UserView enthält zwei Eingabefelder – eines für das aktuelle Passwort und eines für die Eingabe eines neuen Passworts. Beim Start der Szene werden diese Felder über die Start()-Methode mit entsprechenden Properties im ViewModel verknüpft. Diese Verknüpfung erfolgt über onValueChanged-Listene. Damit wird bei einer Eingabe diese direkt an die passende Property im ViewModel übergeben wird. Dort löst die Property ein Änderungs-Event aus, wodurch wiederum alle gebundenen UI-Elemente automatisch aktualisiert werden. So erscheint eine Passwortänderung sofort auch im Feld für das aktuelle Passwort. Damit die Änderung am Account nicht nur in der aktuellen Szene, sondern im gesamten Spiel wirksam wird, muss sie explizit gespeichert werden. Dies geschieht über einen Button, der eine Übertragung der Daten an das Backend auslöst.

Diese Architektur ermöglicht eine interaktionslastige MR-Anwendung, die in weiteren Entwicklungsschritten um einen Multiplayermodus erweitert werden kann und einfach Skalierbar, Testbar und Wartbar ist.

### Weiteres

Einen Überblick über weitere relevante Implementierungen bieten die nachfolgenden Unterkapitel.

**Animationen** Da für die virtuelle Tierbeobachtung ein realistisches Verhalten von virtuellen Tieren vorausgesetzt wird, werden 3D-Modelle mit animierten Komponenten verwendet. Diese Animationsclips umfassen beispielsweise Laufen, Ablegen oder Rennen. Allerdings sind die einzelnen Animationen nur wenige Sekunden lang und längere Bewegungsabläufe können nur dann erstellt werden, wenn sie mit mehreren (unterschiedlichen) Animationen zusammengesetzt werden. Damit Medieninhalte, wie animierte 3D-Objekte oder Bilder, zur Laufzeit geladen werden können, müssen sie im Ordner Resources abgelegt werden, sofern kein alternatives Ladesystem verwendet wird.

Um ein animiertes Tier in Unity anzuzeigen, wird in der AnimalView über den Animationslink des aktuellen Tierobjekts das zugehörige 3D-Modell geladen und daraus ein GameObject erstellt. Dieses Objekt wird mit der MonoBehaviourBridge verknüpft und im ViewModel wird die Funktion für den Bewegungsablauf dieses Objekts aufgerufen. Zusätzlich wird diesem Objekt ein Animator hinzugefügt und der AnimatorController[[12]](#footnote-13) geladen. Der Animator stellt die Schnittstelle zur Steuerung von Animationen bereit und der AnimatorController ist für die konkrete Steuerung der Animationen zuständig. Das ViewModel ruft im Anschluss die MovementLibrary aus der Library-Schicht auf, diese hat den konkreten Bewegungsablauf des Objekts implementiert. So werden einzelne Animationen aneinandergefügt, indem Transitionen erstellt und bestimmte Parameter (zum Beispiel bools, floats, trigger) gesetzt werden. Dazu bietet Unity eine graphische Oberfläche, die die Verknüpfung von Zuständen (einzelnen Animationen) über Transitionen ermöglicht. Hier kann festgelegt werden, ob ein Übergang an einen Parameter geknüpft ist oder erst nach einer bestimmten Zeit erfolgt. In der MovementLibrary können dann die Parameter gesetzt werden und darüber die Übergänge zwischen Animationen gesteuert werden. Ein „Running“ kann zum Beispiel nur dann erfolgen, wenn der Parameter „isRunning“ auf „true“ gesetzt wird. Das ermöglicht zum Beispiel den Ablauf „RunAway“, bei dem auf ein vorwärts rennen zuerst ein nach rechts drehen und dann nach rechts rennen erfolgt. AnimatorController-Parameter lösen also die Transition aus, aber die MovementLibrary setzt diese Parameter.

Eine Herausforderung stellen dabei die vordefinierten Animationen selbst dar, da diese beispielweise festgelegte Ausgangspositionen- oder Ausrichtungen haben. Diese sind unter Umständen schreibgeschützt, sodass trotzt programmiertem Übergang unerwünschte unnatürliche Effekte, wie plötzliche Sprünge oder unerwartetes Drehen des 3D-Modells nicht vermeidbar sind. Auch die Interaktion mit der Umwelt stellt an dieser Stelle eine große Herausforderung dar, da einzelne Animationen nicht in kleinere Bestandteile aufgespalten und auf spezielle Bedürfnisse angepasst werden können.

**Interaktion**

# Zusammenfassung und Ausblick

Notizen

ViewModel durch Data Binding direkte (wir wollen ja schnelle Änderungen) und automatische Änderungen an View ermöglicht → hey, vielleicht können wir dadurch die Reaktionsfähigkeit des Systems bissl erhöhen, was a) für unsere Fotografiererei ganz sinnvoll sein kann und b) wenn annähernd Echtzeit kann man evtl was gegen die Simulationskrankheit machen und c) kann man ggf auch fürs Thema Spiele einsetzen und d) fällt uns bestimmt auch noch was zum Thema Reaktionsfähigkeit und Medizin ein

Glossar

|  |
| --- |
|  |

Abbildungsverzeichnis

[Abbildung 1 - Fehlerhafte Interaktion (WWF, 2024; eigene Aufnahme) 10](https://d.docs.live.net/2F16C520D8C484C7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc206059912)

[Abbildung 2 - Augmentierter Wolf (WWF, 2024; eigene Aufnahme) 10](https://d.docs.live.net/2F16C520D8C484C7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc206059913)

[Abbildung 3 - Unterscheidung MVC und MVVM (MVVM Pattern on Android | WOXAPP, o. J.) 18](https://d.docs.live.net/2F16C520D8C484C7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc206059914)

[Abbildung 4 - Mockup der Anwendung (eigene Zeichnung) 21](https://d.docs.live.net/2F16C520D8C484C7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc206059915)

[Abbildung 5 - MVVM-3D (eigene Darstellung) 28](https://d.docs.live.net/2F16C520D8C484C7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc206059916)

[Abbildung 6 - Ablauf Kollaboration (eigene Darstellung) 30](https://d.docs.live.net/2F16C520D8C484C7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc206059917)

[Abbildung 7 - ER-Diagramm Datenbank (eigene Darstellung) 31](#_Toc206059918)

[Abbildung 8 - UML-Diagramm Backend (eigene Darstellung) 34](https://d.docs.live.net/2F16C520D8C484C7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc206059919)

[Abbildung 9 - UML-Diagramm Frontend (eigene Darstellung) 37](#_Toc206059920)

Tabellenverzeichnis

Normen

Software Tools

Anhang A

Die folgende Liste stellt eine Reihe zukünftiger Forschungsfragen im XR-Kontext dar. Die Liste ist vollständig aus (Barta et al., 2025) übernommen und ins Deutsche übersetzt.

* + Inhalte
    - Nützlich (Utilitarian)
      * Welchen tatsächlichen Effekt hat nützlicher AR-Content auf die kognitive Verarbeitung von Konsument:innen im Rahmen des Zwei-Prozess-Modells der Informationsverarbeitung?
      * Welche Gestaltungsprinzipien können die kognitive Verarbeitung in AR optimieren, um die kognitive Belastung zu reduzieren?
      * Wie beeinflussen Unterschiede in visuellen Details, Farbe und Kontrast von AR-Inhalten kognitive Prozesse wie die Aufmerksamkeit während Entscheidungsfindungen?
    - Hedonistisch (Hedonic)
      * Wie beeinflussen unterhaltsame Elemente in AR-Inhalten die Emotionen von Konsument:innen? Wie entwickeln sich diese Emotionen über die Dauer einer längeren AR-Erfahrung?
      * Welche Arten von hedonistischem Content in AR (z. B. Gamification, interaktive Erzählformen) sind am effektivsten, um positive emotionale Reaktionen hervorzurufen?
      * Wie wirkt sich unterhaltsamer AR-Content auf das Wohlbefinden der Konsument:innen aus (z. B. Stressniveau, Stimmung, Glück)?
    - Sozial (Social)
      * Wie beeinflusst die Integration sozialer Elemente (z. B. Sharing-Funktionen) die wahrgenommene Glaubwürdigkeit und Vertrauenswürdigkeit der präsentierten Informationen?
      * Wie wirken sich unterschiedliche soziale Elemente (z. B. virtuelle Avatare, Echtzeit-Interaktionen) auf das Konsumentenverhalten aus?
      * Welche Rolle spielen Empfehlungen von Gleichgesinnten und sozialer Beweis (social proof) bei Konsumentscheidungen?
    - Generative KI (GenAI)
      * Welchen Einfluss hat individuell anpassbarer GenAI-Content auf das Engagement der Konsument:innen?
      * Wie unterscheiden sich die Reaktionen der Konsument:innen auf GenAI-Content im Vergleich zu manuell erstelltem Content in AR-Umgebungen?
      * Wie kann KI-gesteuerte, echtzeitbasierte Anpassung an die Umgebung in AR die Nutzererfahrung verbessern?
  + Kontext
    - Produkttyp
      * Wie beeinflusst AR das Vertrauen und die Entscheidungsfindung von Konsument:innen bei risikoreichen Produkten im Vergleich zu mittel- und niedrig-risikohaften Produkten?
      * Wie helfen AR-Funktionen dabei, wahrgenommenes Risiko zu verringern und Vertrauen bei risikoreichen Produkten zu stärken?
      * Wie beeinflusst die Vertrautheit der Konsument:innen mit einem Produkt deren Reaktion auf AR-unterstützte Produktpräsentationen?
    - Marken
      * Wie wirkt sich AR auf die Markenwahrnehmung und -bindung bei lokalen im Vergleich zu globalen Marken aus?
      * Wie beeinflusst der Einsatz von AR die Wahrnehmung und Einstellung der Konsument:innen gegenüber bekannten versus unbekannten Marken?
      * Welche Unterschiede zeigen sich in den emotionalen Reaktionen auf AR-Erlebnisse bei der Interaktion mit vertrauten vs. unbekannten Marken?
    - Umgebung
      * Wie beeinflussen multisensorische AR-Erfahrungen (z. B. Kombination aus visuellen, auditiven und olfaktorischen Reizen) die Bewertung und das Verhalten der Konsument:innen?
      * Wie wirkt sich die wahrgenommene Privatsphäre der Umgebung auf die Bereitschaft der Konsument:innen aus, persönliche Informationen zu teilen und mit AR-Features zu interagieren?
      * Wie beeinflusst die Kongruenz des AR-Raumes (z. B. virtuelle Produkte in passenden Umgebungen) die Wahrnehmung von Produktqualität und Relevanz?
    - Business-to-Business (B2B)
      * Wie beeinflusst der Einsatz von AR in Mitarbeiterschulungen Lernresultate, Wissenserhalt und Arbeitsleistung im Vergleich zu traditionellen Schulungsmethoden?
      * Wie wirkt sich AR-gestützte Remote-Zusammenarbeit auf Entscheidungsfindung, Projekteffizienz und Markteinführungszeiten in Produktentwicklungsteams aus?
      * Welche Auswirkungen haben virtuelle AR-Ausstellungen auf Kund:innenbindung und Conversion-Rates im Vergleich zu traditionellen Marketingansätzen?
  + Gerät (Device)
    - Wie unterscheiden sich Bedienbarkeit und Benutzerfreundlichkeit von AR-Apps auf mobilen Geräten, Computern und AR-Brillen, und wie wirkt sich das auf die Akzeptanz bei Konsument:innen aus?
    - Wie beeinflussen virtuelle Spiegel die Wahrnehmung von Passform, Qualität und Kaufabsicht im Vergleich zu traditionellen Einkaufsmethoden?
    - Welche kognitiven und emotionalen Unterschiede ergeben sich bei der Nutzung von AR über mobile Geräte versus AR-Brillen?
  + Konsument:innen
    - Soziodemografische Aspekte
      * Welche spezifischen AR-Funktionen sprechen jüngere im Vergleich zu älteren Konsument:innen besonders an?
      * Wie kann AR-Content kulturell vielfältig angepasst werden?
      * Welche Strategien können genutzt werden, um AR-Apps benutzerfreundlicher für Konsument:innen mit unterschiedlichen technischen Fähigkeiten zu gestalten?
    - Ethische Bedenken
      * Wie nehmen Konsument:innen die ethische Vertretbarkeit emotionaler Beeinflussung durch AR im Marketing und Einzelhandel wahr?
      * Wie wird die ethische Frage bewertet, dass AR zu Impulskäufen verleiten kann?
      * Welche Strategien können helfen, Datenschutzbedenken zu verringern und Vertrauen in AR-Technologien aufzubauen?
  + Methodik
    - Messmethoden
      * Wie können physiologische Messmethoden wie Eye-Tracking und Herzfrequenzmessung zur Bewertung von Aufmerksamkeit und emotionalen Reaktionen in AR-Erfahrungen eingesetzt werden?
      * Wie beeinflussen AR-Erfahrungen Konsumentenverhalten, z. B. Kaufentscheidungen oder Rückgabequoten?
      * Wie entwickeln sich Konsumentenwahrnehmung und -nutzung von AR-Technologie bei wiederholtem Kontakt?
    - Forschungsmethoden
      * Wie lassen sich Langzeitstudien gestalten, um Veränderungen im Konsumentenverhalten und in der Einstellung zu AR nachzuvollziehen?
      * Welche spezifischen Muster lassen sich durch Mixed-Methods-Forschung (Kombination von qualitativen und quantitativen Daten) identifizieren?
      * Wie unterscheiden sich AR-Erlebnisse in natürlichen Umgebungen von solchen in kontrollierten Laborsituationen hinsichtlich der Konsumentenreaktionen?

Literaturverzeichnis

|  |
| --- |
| **Pop, A. (2008):** Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages. Dissertation an der Linköpings Universitet, Linköping, Schweden, Department of Computer and Information Science, 2008. |
| **Fritzson, P.; Aronsson, P.; Bunus, P.; Engelson, V.; Saldamli, L.; Johansson, H.; Karstöm, A. (2002):** The Open Source Modelica Project. In Tagungsband „2nd International Modelica Conference“, 18.-19. März 2002, München, S. 297-306. |
| **BMWI (2013):** E-Energy: Online: [http://www.digitale-technologien.de /DT/Navigation/DE/Service/Abgelaufene\_Programme/E-Energy/e-energy.html](http://www.digitale-technologien.de/DT/Navigation/DE/Service/Abgelaufene_Programme/E-Energy/e-energy.html) [last access: 20.02.2018]. |

adegeo. (o. J.). *Data binding overview—WPF*. Abgerufen 31. Juli 2025, von https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/

Agrawal, S., Simon, A., Bech, S., Bærentsen, K., & Forchhammer, S. (2019). Defining Immersion: Literature Review and Implications for Research on Immersive Audiovisual Experiences: 147th AES Pro Audio International Convention. *Journal of Audio Engineering Society*, *68*(6), 404–417. https://doi.org/10.17743/jaes.2020.0039

*Amazon RDS für SQL Server-Datenbanken in der Cloud*. (o. J.). Amazon Web Services, Inc. Abgerufen 6. August 2025, von https://aws.amazon.com/de/rds/sqlserver/

Barff. (o. J.). *Als sich die Virtualität mit der Realität vermischte*. Abgerufen 19. April 2025, von https://www.barff.de/die-geschichte-der-augmented-reality

Barta, S., Gurrea, R., & Flavián, C. (2025). Augmented reality experiences: Consumer-centered augmented reality framework and research agenda. *Psychology & Marketing*, *42*(2), 634–650. https://doi.org/10.1002/mar.22143

Benbelkacem, S., Aouam, D., Zenati-Henda, N., Bellarbi, A., Bouhena, A., & Otmane, S. (2019). *MVC-3D: Adaptive Design Pattern for Virtual and Augmented Reality Systems* (No. arXiv:1903.00185). arXiv. https://doi.org/10.48550/arXiv.1903.00185

Benbelkacem, S., Zenati-Henda, N., Aouam, D., Izountar, Y., & Otmane, S. (2020). MVC-3DC: Software architecture model for designing collaborative augmented reality and virtual reality systems. *Journal of King Saud University: Computer and Information Sciences*, *32*(4), 433–446.

Bitkom e.V. (2025, Januar 6). *Immer mehr Deutsche nutzen Augmented Reality | Presseinformation | Bitkom e. V.* https://www.bitkom.org/Presse/Presseinformation/Immer-mehr-nutzen-Augmented-Reality

*Brille online anprobieren: Teste den virtuellen 3D-Simulator*. (o. J.). Abgerufen 9. August 2025, von https://www.misterspex.de/l/pg/100508

Capilla, R. (2004). Software Architectures for Designing Virtual Reality Applications. *Lecture Notes in Computer Science*. https://www.academia.edu/14693693/Software\_Architectures\_for\_Designing\_Virtual\_Reality\_Applications

*Cloud SQL for MySQL, PostgreSQL und SQL Server*. (o. J.). Google Cloud. Abgerufen 6. August 2025, von https://cloud.google.com/sql

*Cloud-Objektspeicher – Amazon S3 – AWS*. (o. J.). Amazon Web Services, Inc. Abgerufen 6. August 2025, von https://aws.amazon.com/de/s3/

Dadson, C. (2023, Juni 13). *Unreal vs. Unity*. Design4Real. https://design4real.de/unreal-vs-unity/

Dewan, P. (o. J.). *Architectures for Collaborative Applications*.

Dhawan, H. (2024, Dezember 13). MVVM Architecture: Boost Efficiency in App Development. *Neuronimbus*. https://www.neuronimbus.com/blog/unlocking-efficiency-with-mvvm-architecture-a-modern-approach-to-model-view-architecture/

Drews, I. (2020, April 21). *Virtuelle Stunden im Zoo | KÄNGURU Magazin*. https://www.kaenguru-online.de/themen/medien/virtuelle-stunden-im-zoo

Explore. (o. J.). *Explore.org*. Abgerufen 12. April 2025, von https://explore.org/

Fell, T. (2021, Mai 26). *Mit der VR-Brille in den Wald – Immersive Learning News*. https://www.immersivelearning.news/2021/05/26/mit-der-vr-brille-in-den-wald/

Fletcher, P. (2023, Juli 13). *Einführung in SignalR*. Einführung in SignalR. https://learn.microsoft.com/de-de/aspnet/signalr/overview/getting-started/introduction-to-signalr

Fleury, C. (o. J.). *Modèles de conception pour la collaboration distante en environnements virtuels distribués: De l’architecture aux métaphores*.

Frank, M. (2022, Dezember 18). Unity vs. Unreal—Welche Engine ist besser? *nobreakpoints*. https://blog.nobreakpoints.com/unity-vs-unreal-engine/

FREELANCE-PRESS-2. (2025, Januar 6). *CES 2025: Augmented Reality: Vom Trend zum Mainstream? - Das FotoPortal*. https://www.dasfotoportal.de/augmented-reality-vom-trend-zum-mainstream

Gaia. (o. J.). *Zomertour 2023 | GAIA*. Abgerufen 18. März 2025, von https://www.gaia.be/nl/campagnes/zomertour-2023

Goebel, T. & ORF-Wissenschaft. (2023, Oktober 7). *Virtual-Reality-Training macht Forstarbeit sicherer*. science.ORF.at. https://science.orf.at/stories/3221561/

Heckl, J., Benedikt Peter, & CipEquinus. (o. J.). *JochenHeckl/DataBinding: Data binding for Unity gameobjects. If you like MVVM User Interfaces, this is where to start.* Abgerufen 30. April 2025, von https://github.com/JochenHeckl/DataBinding/tree/main

Hosseini, S. M. (2025, März 7). *MVVM in Unity: A Developer’s Real-World Take on UI and Logic Separation*. https://www.linkedin.com/pulse/mvvm-unity-developers-real-world-take-ui-logic-hosseini-gbl0e

Immotion. (o. J.). *IMMOTION | The Global Leader in Immersive Edutainment*. IMMOTION | The Global Leader in Immersive Edutainment. Abgerufen 18. März 2025, von https://edu.immotion.co

Klein, G. (2009). *Visual Tracking for Augmented Reality: Edge-based Tracking Techniques for AR Applications*. VDM Publishing.

Lapschies, S. (o. J.). *Die Zukunft der VR- und AR-Software im Jahr 2025: Was Geschäftskunden wissen müssen*. Abgerufen 18. April 2025, von https://unboundxr.de/blogs/die-zukunft-von-vr-und-ar-software-im-jahr-2025-was-geschaftskunden-wissen-mussen

Lawton, G. (2024, März 7). *Metaverse Interoperability Challenges and Impact | TechTarget*. Search CIO. https://www.techtarget.com/searchcio/tip/Metaverse-interoperability-challenges-and-impact

Majdak, M. (2023, Oktober 20). *MVVM vs MVC: Key Differences and Use Cases*. Startup House. https://startup-house.com/blog/mvvm-vs-mvc-comparison

Manager, A. (2021, September 28). AR/VR Blog—Blick in die Geschichte. *Augmented Virtual Reality*. https://www.ar-vr-manager.de/ar-vr-geschichte/

Marcus. (2020, November 4). *Virtual Reality & Augmented Reality im Handwerk*. 21 grad. https://www.vaillant.de/21-grad/technik-und-trends/das-spiel-mit-der-realitaet-virtual-reality-und-augmented-reality/

Mehler-Bicher, A., & Steiger, L. (2014). *Augmented Reality: Theorie und Praxis*. De Gruyter Oldenbourg. https://doi.org/10.1524/9783110353853

Mehler-Bicher, A., & Steiger, L. (2021). Augmentierte und Virtuelle Realität. In A. Hildebrandt & W. Landhäußer (Hrsg.), *CSR und Digitalisierung: Der digitale Wandel als Chance und Herausforderung für Wirtschaft und Gesellschaft* (S. 243–258). Springer. https://doi.org/10.1007/978-3-662-61836-3\_16

*Meta Quest 3: Mixed-Reality-Headset der nächsten Generation*. (o. J.). Abgerufen 2. August 2025, von https://www.meta.com/de/quest/quest-3/

Metamandrill. (2025, Februar 13). *Erwartete Virtual Reality- und Augmented Reality-Headsets im Jahr 2025*. Metamandrill.com. https://metamandrill.com/de/erwartete-vr-und-ar-headsets-im-jahr-2025/

Molchanov. (2025, Januar 28). AR/VR Trends in 2025. *Innowise*. https://innowise.com/de/blog/ar-vr-trends/

Nathalie. (2018, Januar 8). *China VR Zoo*. Schweizer Virtual Reality News. https://vr-room.ch/2018/01/08/erster-vr-zoo-in-china-eroeffnet/

Oriz, E. (2024, Oktober 29). *Mini-tutorial: Data binding with UI Builder and C# in 5 minutes - Technical Articles*. Unity Discussions. https://discussions.unity.com/t/mini-tutorial-data-binding-with-ui-builder-and-c-in-5-minutes/1544817

PV-Trainer. (o. J.). *PV Trainer | e-learning with Virtual Reality*. PV-Trainer. Abgerufen 25. März 2025, von https://www.pv-trainer.com

rankmagic. (o. J.). *Der Einfluss von Augmented Reality auf Nutzersignale & SEO 2025 | Rankmagic*. Abgerufen 18. April 2025, von https://rankmagic.net/blog/der-einfluss-von-augmented-reality-auf-nutzersignale-seo-2025/

Rocketbrush. (2024, August 17). *Unity vs Unreal: What to Choose in 2025?* https://rocketbrush.com/blog/unity-vs-unreal-engine-which-one-should-you-choose-in-2024

Satya. (2023, Dezember 25). *Creating a Mixed Reality App for Meta Quest 3 Using Unity: A Step-by-Step Guide*. https://www.linkedin.com/pulse/creating-mixed-reality-app-meta-quest-3-using-unity-step-by-step-dev-vumnc

Stadt Wuppertal. (o. J.). *smart.zoo—Ein virtuelles Zooerlebnis*. smart.wuppertal. Abgerufen 18. März 2025, von https://smart.wuppertal.de/projekte/smart.zoo.php

Stonis, M. (2024, Oktober 9). *Model-View-ViewModel—.NET*. https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

Sukmawati, F., Santosa, E. B., & Rejekiningsih, T. (2023, April 1). *Design of Virtual Reality Zoos Through Internet of Things (IoT) for Student Learning about Wild Animals. | EBSCOhost*. https://doi.org/10.18280/ria.370225

Tremosa, L. (2025, März 12). *Beyond AR vs. VR: What is the Difference between AR vs. MR vs. VR vs. XR?* The Interaction Design Foundation. https://www.interaction-design.org/literature/article/beyond-ar-vs-vr-what-is-the-difference-between-ar-vs-mr-vs-vr-vs-xr

Tuch, R. (2024, Februar 27). *Benefits of MVC for App Development: 9 Advantages for 2025*. https://all-win-solutions.com/benefits-of-mvc-architecture/

Unity. (o. J.). *ECS für Unity*. Unity. Abgerufen 29. April 2025, von https://unity.com/ecs

Unity Developers. (o. J.). Ethereum Towers—Virtual Reality and Web Metaverse. *Unity Developers*. Abgerufen 9. April 2025, von https://unitydevelopers.co.uk/case-study/vr-ethereum-towers/

Unity Technologies. (2024, April 25). *Unity - Manual: Data binding*. https://docs.unity3d.com/2023.2/Documentation/Manual/UIE-data-binding.html

Universität Freiburg. (2023). *Gläserne Wälder durch Augmented Reality: XR Future Forest Lab startet an Universität Freiburg—Archiv der Hochschul- und Wissenschaftskommunikation (bis 08/2024)*. https://kommunikation.uni-freiburg.de/pm/2023/waelder-augmented-reality

Vasconcelos. (o. J.). *Andre Vasconcelos—Portfolio & Blog*. Abgerufen 29. April 2025, von https://avasconcelos114.github.io/portfolio

*Virtuelle 3D-Küchenplanung | Der Wohnfuchs*. (o. J.). Der Wohnfuchs • Das Möbelhaus in Rheinberg. Abgerufen 9. August 2025, von https://www.wohnfuchs.com/kuechenwelten/virtuelle-3d-kuechenplanung/

von Eitzen, I. M. (2023). *Faktoren zur Akzeptanz von Virtual Reality Anwendungen*.

*What is ASP.NET Core? | .NET*. (o. J.). Microsoft. Abgerufen 7. August 2025, von https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core

Wolfenstein, K. (2025, März 5). *IEEE VR 2025: XR/AR/VR/MR Themen und Schwerpunkte der 32. IEEE Conference on Virtual Reality and 3D User Interfaces*. Xpert.Digital. https://xpert.digital/ieee-vr-2025/

WWF. (2024, Juni 19). *Heimische Wildtiere per Augmented Reality erleben*. https://www.wwf.de/aktiv-werden/augmented-reality

Yang, C. (2024). *Vovgou/loxodon-framework* [C#]. https://github.com/vovgou/loxodon-framework (Ursprünglich erschienen 2017)

Zobel, B., Werning, S., Metzger, D., & Thomas, O. (2018). Augmented und Virtual Reality: Stand der Technik, Nutzenpotenziale und Einsatzgebiete. In C. de Witt & C. Gloerfeld (Hrsg.), *Handbuch Mobile Learning* (S. 123–140). Springer Fachmedien. https://doi.org/10.1007/978-3-658-19123-8\_7

Zoo Leipzig. (o. J.). *Gorilla Trek*. Zoo Leipzig. Abgerufen 18. März 2025, von https://www.zoo-leipzig.de/tiere-erlebniswelten/erlebniswelten/pongoland/gorilla-trek/

1. Cardboards sind Pappvorrichtungen, in die ein Smartphone geschoben werden kann, welches dann ein Bild mit einem Eindruck räumlicher Tiefe erstellt. [↑](#footnote-ref-2)
2. So wurde nach einer Testphase eines VR-Spiel die Möglichkeit eines simulierten Suizids unterbunden, da dieser Testpersonen psychisch stark belastet hatte (Mehler-Bicher & Steiger, 2021). [↑](#footnote-ref-3)
3. Hochauflösend ist an dieser Stelle ein Werbeversprechen und muss nicht mit der subjektiven Wahrnehmung zusammenfallen. [↑](#footnote-ref-4)
4. Spiele mit hohem Budget und meist hochwertiger Grafik und aufwendigen Spielmechanismen. Meist von großen Entwicklerstudios produziert. [↑](#footnote-ref-5)
5. Beispiele für 2D und 3D-Anwendungen finden sich in (Heckl et al., o. J.; Hosseini, 2025; Unity Technologies, 2024; Yang, 2017/2024). [↑](#footnote-ref-6)
6. Benutzeroberfläche kann als graphische Schnittstelle zwischen Computer und Mensch verstanden werden, wohingegen die Nutzeroberfläche mehr Schnittstellen (zum Beispiel Kommandozeile) umfasst, die auch von Maschinen oder Programmcode genutzt werden können. [↑](#footnote-ref-7)
7. Event: Änderungssignal [↑](#footnote-ref-8)
8. Observer-Pattern: Objekt „lauscht“ auf Änderungen eines anderen Objekts, erfährt Änderungen sofort [↑](#footnote-ref-9)
9. Reactive-Pattern: Systematische Beobachtung von Datenflüssen und Reaktion darauf [↑](#footnote-ref-10)
10. Hierbei soll nur von den technischen Möglichkeiten gesprochen werden. Sicherheitskonzepte sollten in einer realen Anwendung die direkte und ungeprüfte Manipulation an der Datenbasis verhindern. [↑](#footnote-ref-11)
11. Es wird eine Unterscheidung zwischen „realistischem Foto“, also einem Foto eines realen Tieres und „aufgenommenem Foto“, also dem Foto, das ein User in der Anwendung von einem virtuellen Tier aufnehmen kann, gemacht. [↑](#footnote-ref-12)
12. Ein Controller in diesem Kontext bezieht sich auf die Steuerung von Animationen und nicht auf die Steuerung zwischen Model und View, wie er im Kontext der MVC-Architektur verstanden wird. Der Controller hat an dieser Stelle keine Verbindung zum Architekturmodell. [↑](#footnote-ref-13)