

**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](#_Toc205560919)

[1.1 Motivation 1](#_Toc205560920)

[1.2 Projektziel 3](#_Toc205560921)

[1.3 Aufbau der Dokumentation 3](#_Toc205560922)

[2 Stand der Technik 5](#_Toc205560923)

[2.1 Extended Reality 5](#_Toc205560924)

[2.2 Head-Mounted Displays 8](#_Toc205560925)

[2.2.1 Begriff und Bedeutung 8](#_Toc205560926)

[2.2.2 Technische Funktionsweise 9](#_Toc205560927)

[2.2.3 Verfügbare Geräte 10](#_Toc205560928)

[2.3 Virtuelle Tierbeobachtung 11](#_Toc205560929)

[2.4 XR in Spielen 13](#_Toc205560930)

[2.4.1 Verfügbare Spiele 13](#_Toc205560931)

[2.4.2 Multiplayer 13](#_Toc205560932)

[2.4.3 Interaktion mit Umgebung 13](#_Toc205560933)

[3 Anforderungen 14](#_Toc205560934)

[4 Technologieauswahl 15](#_Toc205560935)

[4.1 Hardware 15](#_Toc205560936)

[4.2 Entwicklungsumgebung 15](#_Toc205560937)

[4.3 Programmiersprache 17](#_Toc205560938)

[4.4 Multiplayer Plugin 18](#_Toc205560939)

[4.5 Architekturmodell 18](#_Toc205560940)

[5 Konzept 24](#_Toc205560941)

[5.1 Aufbau des Prototyps 24](#_Toc205560942)

[5.1.1 Datenbank 25](#_Toc205560943)

[5.1.2 Backend 26](#_Toc205560944)

[5.1.3 Frontend 27](#_Toc205560945)

[5.2 Interaktion 31](#_Toc205560946)

[6 Implementierung und Prototypenentwicklung 32](#_Toc205560947)

[6.1 Datenbank 32](#_Toc205560948)

[6.2 Backend 33](#_Toc205560949)

[6.3 Frontend 35](#_Toc205560950)

[6.3.1 Setup der Unity-Umgebung für die Meta Quest 3 35](#_Toc205560951)

[6.3.2 Implementierung 36](#_Toc205560952)

[6.3.3 Weiteres 39](#_Toc205560953)

[7 Zusammenfassung und Ausblick 41](#_Toc205560954)

[Glossar 42](#_Toc205560955)

[Abbildungsverzeichnis 43](#_Toc205560956)

[Tabellenverzeichnis 44](#_Toc205560957)

[Normen 45](#_Toc205560958)

[Software Tools 46](#_Toc205560959)

[Literaturverzeichnis 47](#_Toc205560960)

**Abstract**

**Abkürzungsverzeichnis**

|  |  |
| --- | --- |
|  |  |
| API | Application Programming Interface |
| AR | Augmented Reality |
| ECS | Entity Component System |
| 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 |
| 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.



Figure 1: Smart Grid according to [BMWI 2013]

## 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 als Oberbegriff die Begriffe 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.

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 und „digital“ kann durch eine „in Echtzeit computergenerierte, interaktive, virtuelle Umgebung“" (BELEG) verstanden werden. Grundlage für VR ist zwingend die Erstellung einer virtuellen Welt durch einen Computer. Sogenannte Cardboards[[1]](#footnote-1) werden zwar ebenfalls als VR-Geräte verkauft, stellen aber kein VR im engeren, beziehungsweise korrekten, Sinne dar (Mehler-Bicher & Steiger, 2021). Entsprechend technische Voraussetzungen für VR sind leistungsfähige Rechner, Hochleistungsgrafikkarten und eine hochauflösende Datenbrille.

MR hingegen, bezieht die reale Welt weiterhin mit ein und erweitert die diese dabei um virtuelle Elemente. Ein Kernaspekt von MR, der insbesondere für die Unterscheidung zu AR relevant ist, besteht in den Interaktionsmöglichkeiten zwischen realen und virtuellen Elementen (Tremosa, 2025).

So wird die reale Umgebung auch bei AR um virtuelle Elemente erweitert, jedoch besteht hier keine Möglichkeit der Interaktion zwischen realen und virtuellen Elementen (Tremosa, 2025). Die reale Welt wird hierbei lediglich um computergenerierte Zusatzobjekte beziehungsweise Zusatzinformationen angereichert (Mehler-Bicher & Steiger, 2021). Häufig werden trotz einer eindeutigen Unterscheidung die Begriffe AR und MR, sowie die Bezeichnung Enhanced Reality 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 verzeichnet werden kann (Zobel et al., 2018), lässt sich weder für MR, noch in großen Teilen für AR, eine Marktreife belegen. Daher werden viele Szenarien mit Smartphones oder Tablets umgesetzt, wobei diese Anwendungen als Informations- und Kommunikationsmedium dienen und vorwiegend im Bereich Marketing platziert sind. Dort sind Eigenschaften wie Interaktivität, Medienreichtum oder auch Immersion entscheidende Hinweise für Kaufbereitschaft oder verbessern die Einstellung gegenüber Werbung (Barta et al., 2025; Mehler-Bicher & Steiger, 2021). Man könnte also von einer besseren Zugänglichkeit zu Produkten oder Themen mittels XR sprechen. Anwendungsgebiete wie Living Mirror (3D-Objekte werden zum Beispiel auf dem Kopf einer Person platziert) oder auch Living Architecture (eine Person kann durch Bewegung oder Kopfdrehungen einen Raum erfahren) bieten der Werbeindustrie weitreichende Potenziale, wobei für die Spieleindustrie insbesondere die Bereiche Living Game Mobile (augmentierte Spiele auf dem Smartphone) oder Living Environment von hoher Relevanz sind (Mehler-Bicher & Steiger, 2014). Living Environment sind alle Anwendungen, die reale Umgebungen abbilden und um zum Beispiel 3D-Objekte oder Audiosequenzen, also eine Kombination mehrerer sensorischer Empfindungen, ergänzen. Allerdings sprechen Mehler-Bicher & Steiger, 2014 von lediglich einem geringen Teil, der eine AR-Anwendung im engeren Sinne darstellt.

Allerdings bieten gerade AR und MR-Anwendungen große Potenziale, insbesondere der Informationsmehrwert, in vielen Bereichen. So kann eine schnellere Vermittlung von Inhalten und eine stärkere Aktivierung von Kommunikationsteilnehmern stattfinden (Mehler-Bicher & Steiger, 2014). Nach Barta et al. (2025) verändert XR unter anderem soziale Interaktionen und sogar ethische Aspekte werden beeinflusst. Einige Untersuchungen deuten zudem darauf hin, dass die kognitive Dissonanz durch XR reduziert wird (Mehler-Bicher & Steiger, 2014). Dies, die Reduktion von Suchzeiten und die Ansprache mehrerer Sinne zeitgleich können das Lernen von Inhalten vereinfachen. Grund dafür ist eine Steigerung der Emotionalität und die multisensorische Ansprache, was den Kommunikationsprozess nicht nur unterstützt, sondern optimiert (Mehler-Bicher & Steiger, 2014). Auch die direkte Anwendung ohne ernsthafte Auswirkungen (“Learning by doing”) ist dabei einer der Hauptvorteile (Zobel et al., 2018), der neben den geringen Kosten vor allem im Bildungsbereich genutzt wird. Zudem bietet XR in handwerklichen Bereichen eine geschützte Umgebung, was zur Verringerung von Kollisionen und Unfällen führt (Mehler-Bicher & Steiger, 2014).

Herausfordernd ist allerdings die Nutzerakzeptanz, welche durch Faktoren wie die aktuell fehlende Alltagstauglichkeit einiger Brillen beziehungsweise Anwendungen (von Eitzen, 2023). Zudem ergeben sich auch auf den Feldern Technologie und Ethik Herausforderungen, die über die rein anwendungsbezogenen Herausforderungen hinaus gehen.

Technologisch herausfordernde Aspekte umfassen beispielsweise Hardware-Limitierungen, sowie die Bereiche Tracking und Interaktion. Um eine sehr gute Immersion zu erschaffen, wird leistungsfähige Hardware benötigt, um eine konsistente Performance zu gewährleisten. So ist neben Latenzen auch die Verarbeitung hochauflösender Grafik und weiterer Berechnungen, ohne zu überhitzen oder den Akku zu schnell zu entleeren, ein relevanter Punkt (Molchanov, 2025; rankmagic, o. J.). Im Softwarebereich ergeben sich 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 die Belastung von Geräten zu reduzieren werden so auch einige Berechnungen in einer Cloud ausgeführt, was allerdings bei schlechter oder fehlender Internetverbindung nur eingeschränkt oder nicht möglich ist (Molchanov, 2025; rankmagic, o. J.). Um in MR oder VR Welten einen fließenden Übergang zwischen der realen und der virtuellen Welt, beziehungsweise virtuellen Elementen, zu schaffen sind fortschrittliche Tracking- und Renderingtechnologien erforderlich, welche derzeit noch ausbaufähig sind. Dabei ist auch auf die Balance von immersiver Erfahrung und Benutzerfreundlichkeit zu achten (Marcus, 2020; rankmagic, o. J.). Der Bereich Interaktion umfasst allerdings 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, aber auch Standardisierung, wie Anwendungen wie virtuelle Meetingräume belegen (Lapschies, o. J.; Wolfenstein, 2025).

Im Bereich Ethik siedeln sich Herausforderungen wie der Energie- und Ressourcenverbrauch, aber auch gesellschaftliche Auswirkungen an (Lapschies, o. J.). So ist die Frage nach dem Datenschutz im Bereich XR sensibel, ein unbefugter Zugriff auf vertrauliche Informationen, wie zum Beispiel Augenbewegungen, zu Manipulation und Betrug führen kann (Molchanov, 2025). Zudem können Kameras an (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. Darüber hinaus leiden viele Personen nach kurzem Tragen der Brille schon unter Gesundheitsproblemen, wie der sogenannten Simulationskrankheit (Mehler-Bicher & Steiger, 2021; Molchanov, 2025; Zobel et al., 2018). Diese ruft Symptome wie Übelkeit und Balanceverlust hervor. Ursache dafür ist die fehlerhafte Abstimmung von visuellen Informationen und dem Gleichgewichtsorgan, wofür mitunter unzulängliche Immersion eine Ursache ist (Mehler-Bicher & Steiger, 2021; Zobel et al., 2018). Zudem muss, in Form einer Brille, die Rechner- und Grafikleistung am Körper verbaut werden, was die freie Bewegung im Raum der tragenden Person stark einschränken kann (Mehler-Bicher & Steiger, 2021) und bei schlecht ausbalancierten Headsets zu körperlicher Belastung wie Nacken- oder Schulterschmerzen führen kann (Molchanov, 2025). Auch Kopfschmerzen und Ermüdung, Trockenheit und Reizungen der Augen sind gängige Probleme (Molchanov, 2025). Einfache Gegenmaßnahmen seitens der nutzenden Personen sind die Begrenzung der Sitzungsdauer, eine gute Körperhaltung oder auch regelmäßige Pausen (Molchanov, 2025). Wohingegen Firmen mit besserem Tracking oder einer höheren Bildwiederholrate Verbesserungen schaffen könnten (Molchanov, 2025).

Neben physischen Problemen sind allerdings auch psychische Probleme zu adressieren. Dazu zählen Überreizung, Schlafstörungen oder Isolationsgefühle, welche durch XR hervorgerufen werden können. Bei XR-Spielen tritt in einer höheren Intensität als bei Nicht-XR-Spielen das Problem von Gewalt beziehungsweise Gewaltverherrlichung, Suchtgefahr sowie die Gefahr von Realitätsverlust auf den Plan (Mehler-Bicher & Steiger, 2021).[[2]](#footnote-2)

Gerade um diese Herausforderungen zu adressieren, aber auch um eine breitere gesellschaftliche Akzeptanz und größere wirtschaftliche Nutzung zu erreichen, stehen einige Themen im Fokus der Forschung. So soll mit der Entwicklung von plattformübergreifenden Inhalten XR zugänglicher und plattformunabhängiger, sowie über den Browser erlebbar gemacht werden, was die Reichweite und Akzeptanz von XR-Anwendungen erhöhen soll (Molchanov, 2025). Auch ist die Integration von künstlicher Intelligenz (KI) ein Forschungsaspekt sowie Werkzeug, das unter anderem für eine natürlicher erscheinende Umgebung eingesetzt wird. Ziel ist es beispielsweise die Gestenerkennung zu verbessern, Benutzerinteraktionen besser zu interpretieren oder automatisch 3D-Inhalte zu generieren (adaptive Umgebungen) (Lapschies, o. J.; Molchanov, 2025; Wolfenstein, 2025). Auch personalisierte Inhalte und immersive Erlebnisse durch realistische 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 können immersive und gefahrenfreie Lernumgebungen aufgebaut werden, die nicht nur Prototypen und Operationen testbar machen, sondern auch Motivation und Aufmerksamkeit aufrechterhalten. Barta et al. (2025) bieten in ihrer Arbeit „Augmented reality experiences: Consumer‐centered augmented reality framework and research agenda” eine Reihe von zukünftigen Forschungsfragen, die Themen wie Nützlichkeit, gesellschaftliche Auswirkungen, Nutzung von generativer KI, Forschungsmethoden oder auch Marktpotenziale abdecken. Ein Auszug dieser Liste befindet sich in Anhang (EINFÜGEN).

## 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 Datenbrillen, die 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, da diese an der fehlenden gesellschaftlichen Akzeptanz scheiterten und Cardboards preiswerte Alternativen boten (Zobel et al., 2018). Bis ins Jahr 2025 wurde Oculus von Meta aufgekauft (Zobel et al., 2018) und die Hardware wurde deutlich leichtgewichtiger, benutzerfreundlicher und Akkulaufzeiten und Tragegefühl verbesserte sich (Metamandrill, 2025; Molchanov, 2025). Zwar sanken auch die Preise, aber gerade hochwertige Produkte sind für den Massenmarkt immer noch zu kostenintensiv (Molchanov, 2025), weshalb VR-Brillen häufig in gewerblichen oder Forschungseinrichtungen verwendet werden. Insbesondere MR-Brillen gewinnen an Bedeutung, da diese einen nahtlosen Übergang zwischen realer und virtueller Welt ermöglichen (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). Ein Blick in die Zukunft verrät Brillen, die höhere Bildwiederholraten und hochauflösendere Grafik bieten sollen. Weitere Neuerungen sind die Einbindung von KI und die Entwicklung von MR-Brillen (Metamandrill, 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 kann vor allem in Tiergärten und Zoos, unterschiedlichste Formen annehmen. Die virtuelle Beobachtung reich von Videos auf Social-Media-Kanälen bis hin zu rein virtuellen Zoos. 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 vorgedrehte Videos einzelner Tiere sein, allerdings können Videos auch Livestreams rund um die Uhr oder zu besonderen Zeiten sein. Diese Alternative etablierte sich insbesondere in den Anfängen der neuen Zwanzigern, als aufgrund von Pandemie, Zoos geschlossen hatten und dennoch die Möglichkeit von Tierbeobachtungen geboten werden sollte (Drews, 2020). Neben Liveaufnahmen in Zoos, existieren auch Liveaufnahmen von Tieren in ihrer natürlichen Umgebung und diese können so ebenfalls zur virtuellen Tierbeobachtung gezählt werden (Explore, o. J.).

Neben realen Tieren gibt es aber auch zahlreiche Varianten, bei denen sich die Beobachtung von Tieren auf virtuelle Tiere bezieht. So zum Beispiel AR-Einblendungen über das Smartphone, in dem auf dem Bildschirm Tiere im eigenen Wohnzimmer erscheinen können. So bietet auch der WWF (World Wide Fund for Nature, WWF) eine solche Möglichkeit auf ihrer Webseite an und stellt darüber hinaus auch Informationen zu diesen Tieren und Tier- beziehungsweise Naturschutz zur Verfügung (WWF, 2024). Diese Anwendungen sind allerdings aufgrund der fehlenden Interaktion mit der Umgebung dem AR-Bereich zuzuordnen, wie auf Ein Bild, das Fenster, Gebäude, Mobiliar, Boden enthält.

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

Automatisch generierte BeschreibungAbbildung 1 und Abbildung 2 zu sehen ist.

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

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

Neben AR-Beobachtung auf dem Smartphone in der eigenen Umgebung, können aber auch VR-Erlebnisse in Zoos stattfinden. So bieten die Stadt beziehungsweise der Zoo Wuppertal beispielsweise Cardboards an, mit denen ein virtueller Rundgang durch einen Zoo erlebt werden kann, bei dem zusätzlich Hintergrundinformationen vermittelt werden (Stadt Wuppertal, o. J.).

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

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

## XR in Spielen

### Verfügbare Spiele

### Multiplayer

### Interaktion mit 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

Die für die Entwicklung eines Prototyps zur Verfügung gestellte Brille ist die MetaQuest3. Diese wird (Stand Juli 2025) für 549,99€ von Meta angeboten und beinhaltet einen Snapdragon XR2 Gen 2-Prozessor, der nahtloses und hochauflösendes Passthrough bieten soll.[[3]](#footnote-3) Die Passthroughfunktion, die MR-Erlebnisse ermöglicht, wird über zwei Farbkameras und einen Tiefenprojektor erreicht, die eine originalgetreue Ansicht der Umgebung bieten können. Auch die Erfassung von Gesten ist mit der MetaQuest3 möglich, sodass eine Steuerung ohne Controller möglich ist, was eine intuitivere Steuerung ermöglichen kann. Zudem hat die MetaQuest3 Stereolautsprecher integriert, sodass eine verbesserte Immersion über die Ansprache mehrerer Sinne zeitgleich erreicht werden kann. Um diese Immersion visuell zu erreichen, besitzt die Brille ein 4K-Infinite-Display mit einer Auflösung von 2.064 x 2.208 Pixeln pro Auge und Aktualisierungsraten von 72 Hz, 90 Hz beziehungsweise 120 Hz. Höhere Bildwiederholraten führen zu geringerem Risiko, Anzeichen der Simulationskrankheit zu bekommen und 70 Hz sind an dieser Stelle sehr niedrig, bei kurzer Tragedauer oder einfachen Anwendungen aber noch ausreichend. Zudem helfen sogenannte „Pancake-Linsen“ gegen Stör- oder Streulicht, jedoch gilt zu beachten, dass das Tragen der Brille nur in Innenräumen vorgesehen ist. Um dennoch die Privatsphäre umstehender Personen zu ermöglichen, zeigt ein Signal den Einsatz der eingebauten Kamera an (*Meta Quest 3*, o. J.).

Diese Brille eignet sich also neben der Entwicklung von VR-Anwendungen auch zur Entwicklung von MR-Anwendungen und ermöglicht die Implementierung des vorgesehenen Prototyps.

## Entwicklungsumgebung

Um einen Prototypen entwickeln zu können, ist eine Entwicklungsumgebung nötig, die die Erstellung von VR-Anwendungen ermöglicht. Sehr bekannt und häufig verwendet sind Unity und Unreal Engine (Frank, 2022). Im Folgenden wird ein kurzer Überblick über beide Umgebungen gegeben und dann eine Entscheidung getroffen.

**Unity** Unity bietet eine Entwicklung mit C# an und wird unter anderem aus diesem Grund als Einsteiger- aber auch generell als Benutzerfreundlicher beschrieben (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Schnelles Prototyping soll mit Unity möglich sein und die Entwicklung wird durch eine große Community und eine gute Dokumentation unterstützt (Dadson, 2023). Zudem bietet Unity einen kostenlosen Unity Student Zugang an und ist auch im kommerziellen Bereich in der Regel günstiger als beispielsweise die Unreal Engine (Dadson, 2023). Eine Entwicklung ist für mehrere Zielplattformen möglich und dank eines großen Asset Stores, sowie vielen Plugins, ist die Entwicklung von VR-Anwendungen mit Unity besonders geeignet (Frank, 2022; Rocketbrush, 2024). Auch für die vorgegebene MetaQuest3 bietet diese Entwicklungsumgebung Vorteile, da eine gute Meta Integration inklusive Passthrough existiert, Meta viel Support für Unity bietet und das Deployment auf der Brille als „einfach“ beschrieben wird (Satya, 2023). Weiter ist die Verwendung von Unity führender Industriestandard und vor allem in der Gamingindustrie weit verbreitet, da Unity auch sehr gut skalierbar ist (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Nachteilig sind allerdings berichtete Performanceprobleme, insbesondere, wenn die verwendeten Geräte nicht besonders leistungsstark sind, und weniger realistische Grafikqualitäten, was die Entwicklung für AAA-Spiele[[4]](#footnote-4) einschränkt (Dadson, 2023; Frank, 2022; Rocketbrush, 2024). Unity eignet sich also besonders für eine einfache und schnelle Entwicklung von Prototypen, kann aber für die Entwicklung grafisch hochwertiger Spiele Einschränkungen bedeuten.

**Unreal** Die Unreal Engine bietet hingegen eine Entwicklung mit C++ an und wird von einem komplexeren und schwierigeren Einstieg begleitet (Rocketbrush, 2024). Zudem wird Unreal als komplexer im Setup und als überdimensioniert für einfache Anwendungen und Prototypen beschrieben (Dadson, 2023). Allerdings wird die Einarbeitung durch eine sehr große Community und viel Support erleichtert (Rocketbrush, 2024) und Unreal bietet darüber hinaus mittels BluePrints eine NoCode-Möglichkeit an (Frank, 2022). Auch hat diese erhöhte Komplexität höhere Kosten zufolge, da weniger Personen erfahren in der Entwicklung mit Unreal sind und bei der Entwicklung mit höherer Softwarekomplexität auch höhere Anforderungen an die Hardware und das Performance-Budget gestellt werden müssen (Dadson, 2023; Frank, 2022). Jedoch kann die erhöhte Komplexität mit qualitativen Vorteilen gerechtfertigt werden. Unreal wird häufiger als Unity in der Entwicklung von AAA-Spielen eingesetzt, da größere Dateien verarbeitet werden können und insbesondere bei komplexen 3D-Szenen und großen Projekten hohe Leistung und Performance beobachtbar sind (Dadson, 2023; 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, bringt aber eine gesteigerte Komplexität mit sich, was diese Engine ungeeigneter für die schnelle und einfache Entwicklung eines Prototypen macht.

Da Unity geeigneter für die benutzerfreundliche Entwicklung von Prototypen ist und einen guten Support für die zur Verfügung gestellten Brille bietet, wird diese Entwicklungsumgebung für die weitere Arbeit gewählt. Zwar bietet Unreal eine höhere Grafikqualität und Leistung, jedoch ist die erhöhte Komplexität aufgrund der Vorteile die Unity bietet 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

Um eine klare Trennung von Zuständigkeiten zu erreichen und die Komplexität zu reduzieren, soll für den Code auf ein Architekturmodell zurückgegriffen werden. Dafür wird im Voraus eine Aufteilung der Anwendung in Frontend und Backend festgelegt. Im Frontend sollen nachher all die Funktionen sein „die auf der Brille laufen“, also Szenen, wie sich Objekte bewegen, Interaktionen, etc. Das Backend hingegen soll für die Verarbeitung von Daten zuständig sein. Also die Bereitstellung von Daten über eine API (Application Programming Interface, BELEG) und weiter für Analysen, Authentifizierung und im weiteren Verlauf für die Erstellung einer Spieleinstanz. So dürfen Spielstände oder Tagebücher nicht gelöscht oder zurückgesetzt werden, nur weil die VR-Brille abgesetzt wird. Zusätzlich soll noch eine Datenbankschicht eingeführt werden, die lediglich die Daten speichert. Zugriff und Manipulation kann dann geschützt über das Backend erfolgen. Vorteile einer solchen Architektur sind infolge der Trennung von Verantwortlichkeiten und einer damit einhergehenden Komplexitätsreduktion eine bessere Lesbarkeit, Wartbarkeit oder auch Testbarkeit. 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 User Interface (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 ist auch die Skalierbarkeit ein wichtiger Faktor, der mit dieser Architektur ebenfalls steigt. So kann bei Bedarf beispielsweise mehr Rechenleistung für das Backend hinzugezogen werden.[[5]](#footnote-5) (BELEG FÜR VORTEILE FEHLT).

Damit auch in den einzelnen Schichten (Frontend, Backend) der Programmcode getrennt und damit test- und wartbar bleibt, ist es sinnvoll auf bewährte Architekturmodelle zurückzugreifen. 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 für welche Schicht mit den gegebenen Anforderungen als geeignet eingestuft wird. Zu berücksichtigen sind im Frontend neben klassischen Anforderungen auch XR-spezifische Anforderungen, 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 berechnet werden, die auch eine Kollisionserkennung erfordert. So dürfen Finger nicht durch ein Objekt hindurchgreifen und virtuelle Objekte nicht in realen Objekten verschwinden. Dies erfordert auf der visuellen Seite eine entsprechende Rückmeldung, sowie eine Aktualisierung des Objektzustands, wie zum Beispiel Positionskoordinaten.

**MVC** MVC ist ein Architekturmodell, welches aus drei Schichten besteht, welche unterschiedliche Funktionen beinhalten. Das Model (M) ist für die Verbindung zu den Daten und die Logik zuständig. Die View (V) enthält alles bezüglich visueller Elemente und UI und der Controller stellt die Brücke zwischen diesen beiden Schichten dar, indem er die Kommunikation zwischen Model und View anstößt (Capilla, 2004; Majdak, 2023; Tuch, 2024). Vorteile sind hierbei 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). Auch 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 Unity selbst schon mit diesem Architekturmodell gearbeitet, um eine VR-Anwendung zu erstellen (Majdak, 2023; Tuch, 2024; Unity Developers, o. J.). So ist auch die Anpassung des MVC-Modells möglich, da besondere Anforderungen für zum Beispiel XR-Anwendungen einfach zu integrieren sind.

Neben einigen Vorteilen hat das MVC-Modell allerdings im Kontext von XR-Anwendungen auch einige Schwachstellen. Unity ist eine stark komponentenbasierte Umgebung, was zu einer schwierigeren Trennung der einzelnen Funktionalitäten und einer Verschmelzung von Controller und View führt (Majdak, 2023; Vasconcelos, o. J.). Jedoch ist dies auch außerhalb von Unity beobachtbar. So wird der Programmcode bei komplexeren XR-Anwendungen generell schnell unübersichtlich und die Views sind schnell überladen. Zudem steigt die Komplexität mit der Menge an Daten, was vorerst den Schluss zulässt, dass dieses Architekturmodell geeignet für kleine Anwendungen ist, aber bei komplexeren nicht ideal (Majdak, 2023; Vasconcelos, o. J.).

Jedoch bietet das MVC-Modell gute Möglichkeiten diese Schwachstellen mit XR-spezifischen Erweiterungen zu adressieren. So entwickelten (Benbelkacem et al., 2019) eine Erweiterung des Modells um den spezifischen Anforderungen von XR gerecht zu werden, das „MVC-3D“-Modell. Kernaspekte war das Hinzufügen einer „Library“-Komponente, die komplexe Algorithmen, wie sie für Tracking, Gestenerkennung oder weitere Simulationen benötigt werden, kapselt. Weiterhin wurde die View zur „interaktiven View“, sodass sie nicht nur Nutzereingaben, sondern auch das Gerät und die Umgebung berücksichtigen kann. Dank der Auslagerung komplexerer Prozesse blieb 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 von derselben Gruppe auch dieses Modell um die Möglichkeit zur Entwicklung kollaborativer weiterentwickelt. Das „MVC-3DC“-Modell integriert Aspekte wie Kollaboration, Verteilung, Interoperabilität und die Zusammenarbeit verschiedener Nutzer und Systeme (Benbelkacem et al., 2020). Um ein kollaboratives System zu entwickeln (Dhawan, 2024; Majdak, 2023; Stonis, 2024), führen sie nach (BELEG: Fleury, Dewan) drei Methoden an: Centralized Mode, Duplicated Mode und Hybrid Mode. Alle diese drei Methoden beschreiben die Art und Weise eines möglichen Zusammenspiels zwischen einem Client und einem Server und somit keine Erweiterung des Architekturmodells an sich.

**MVVM** MVVM ist ebenfalls ein Architekturmodell mit drei Schichten, wobei das Model (M) ebenfalls für die Datenhaltung beziehungsweise Bereitstellung zuständig ist, die View (V) ebenfalls visuelle Informationen enthält und das ViewModel (VM) ebenfalls die Brücke zwischen den anderen beiden Schichten darstellt (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 es Elementen sich bei Änderung der Daten automatisch zu aktualisieren und wiederum können Daten bei Manipulation der UI automatisch geändert werden, es ist also keine manuelle Definition eines Datenflusses nötig (adegeo, o. J.). Dies ermöglicht eine noch sauberere Trennung von UI und Logik. Abbildung 3 zeigt diesen Unterschied deutlich. Es lässt sich erkennen, dass das Model und die View im MVVM-Modell zu keinem Zeitpunkt miteinander interagieren, bei MVC findet allerdings eine Kommunikation zwischen diesen beiden Schichten statt.

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

Automatisch generierte Beschreibung

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

Einer der Hauptvorteile von MVVM besteht also in der strikten Trennung und damit fehlenden Kopplung zwischen Anzeige und Daten, was dieses Architekturmodell insbesondere in UI-lastigen Anwendungen beliebt macht (Dhawan, 2024; Majdak, 2023; Stonis, 2024), allerdings auch gut für datengetriebene Anwendungen einsetzbar ist, vor allem, wenn diese Daten aus unterschiedlichen Quellen stammen (Majdak, 2023). Gleiche Vorteile wie für die MVC-Architektur gelten auch für MVVM, aufgrund der stärkeren Trennung allerdings in einem höheren Maße: gute Testbarkeit, 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 kann und sich darüber hinaus in der Einarbeitung etwas schwieriger gestalten kann. Ursache dafür ist die Einarbeitung in das Data Binding (Majdak, 2023; Vasconcelos, o. J.). Dieses war lange Zeit auch ein Grund, weshalb das MVVM-Muster nicht oder sehr selten in Verbindung mit Unity genutzt wurde. Dort gab es bis zur Veröffentlichung von Unity 6 kein natives Data Binding, weshalb auf externe Lösungen zurückgegriffen werden musste (Majdak, 2023; Oriz, 2024; Vasconcelos, o. J.). Aus diesen Gründen gibt es noch keine (etablierten) Weiterentwicklungen von MVVM, die spezielle Anforderungen von XR-Anwendungen adressieren.

**ECS** Unity selbst bietet neben den bekannten, plattformunabhängigen Architekturmodellen ein natives Modell an. Das sogenannte ECS kann über 1.000 Objekte gleichzeitig verarbeiten und ist daher sehr gut für große Anwendungen geeignet (Unity, o. J.). Jedoch ist dieses Modell laut Unity selbst hinsichtlich der XR Interaction Toolkit-Kompatibilität noch nicht ausgereift (Unity, o. J.), was die Interaktionsmöglichkeiten einschränken kann und somit die XR-spezifischen Anforderung des Prototyps nicht ausreichend erfüllen könnte. Zudem ist dieses Architekturmodell vor allem für Personen geeignet, die erfahren in der Entwicklung mit Unity sind und anspruchsvolle Spiele entwickeln wollen und für die Performancevorteile sowohl ein anspruchsvolleres Debugging, sowie eingeschränkte Teamarbeit in Kauf nehmen wollen (Unity, o. J.). Da der geplante Prototyp stark interaktionsbasiert sein soll und dieser Anwendungsfall für ECS noch nicht ausgereift ist, wird dieses Architekturmodell für die Entwicklung des Prototyps nicht weiter berücksichtigt.

Die Entscheidung, welches Architekturmodell für das Backend und das Frontend gewählt werden soll, ist eine Entscheidung zwischen MVC oder MVVM. Um die Aufgaben der beiden Schichten bestmöglich zu unterstützen, werden Backend und Frontend getrennt betrachtet.

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 findet immer noch statt, zudem ist die Komplexität geringer. 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 des zu 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 werden oder animierte Objekte angezeigt werden. Zudem können Interaktionen unterschiedlichste Formen annehmen und für die Menüauswahl könnten Controller, Handgesten oder die Stimme verwendet werden, Textfelder könnten sich durch Augenbewegungen automatisch anpassen und Objekte könnten 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 die Anforderungen an einfach synchronisierbare Views. Eine Herausforderung in der Verwendung von MVVM im XR-Kontext ist die Anwendung des Data Bindings. Dieses musste bis Oktober 2024, also bis Unity 6, extern entwickelt und in Unity eingebunden werden.[[6]](#footnote-6) Das native Data Binding in Unity ist also verhältnismäßig neu und es kann auf entsprechend wenig Erfahrungswerte zurückgegriffen werden (Oriz, 2024; Wong, 2025). Zudem sind derzeitige fehlende etablierte Weiterentwicklungen im XR-Bereich eine Herausforderung, erschwert durch eine komplexere Einarbeitung, da das Verständnis von Data Binding zentral für die Implementierung ist. Da seit Ende 2024 allerdings die Verwendung des Data Bindings möglich ist, erscheint das MVVM-Muster als eine geeignete Wahl für die Entwicklung eines Prototypen im Unitykontext. Dieses Muster eignet sich somit sehr gut für Anwendungen mit stetig wechselnden Zuständen, da diese Änderungen mit Hilfe des Data Bindings automatisch umgesetzt werden können. 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 zudem die Fehlerquellen bei der Entwicklung reduziert. Aus diesen Gründen ist die Verwendung von MVVM sinnvoller und rechtfertigt die erhöhte Komplexität des Architekturmodells. Bereits genannte Vorteile (Trennung von Daten und UI/Verantwortlichkeiten, Testing, Wartbarkeit) bieten die Möglichkeit Models und ViewModels unabhängig von Unity zu testen, da ausschließlich die View plattformabhängig ist. Somit dient dieses Architekturmodell der Entwicklung einer plattformunabhängigeren Entwicklung, was nach (Molchanov, 2025) eines der aktuellen Herausforderungen von XR-Anwendungen adressiert. Auch die Wiederverwendbarkeit einzelner Komponenten spielt dabei eine wichtige Rolle.

Da für den zu entwickelnden Prototypen Interaktionen zwischen realen und virtuellen Elementen eine große Rolle spielen, wird entsprechend ein Architekturmodell benötigt, was zum einen für UI- und interaktionslastige Anwendungen geeignet ist aber auch datengetriebene Änderungen einfach ermöglicht. Das Modell muss in hohem Maße reaktiv sein, damit eine gute Immersion möglich ist. Da der Entwicklungsaufwand nicht übertrieben hoch sein soll, müssen Faktoren wie beispielsweise eine klare Trennung von Funktionalitäten berücksichtigt werden, auch damit die Anwendung nicht unnötig komplex und unübersichtlich wird. Somit besteht 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. Aufgrund der Möglichkeit unterschiedliche Eingaben geeignet zu verwalten und über eine reaktive Zustandsverwaltung eine gute Interaktion zwischen virtuellen und realen Elementen zu ermöglichen, stellt die MVVM-Architektur eine geeignetere Lösung als MVC dar und wird aus diesen Gründen als Basisarchitekturmodell gewählt.

Damit neben der klassischen Umsetzung auch spezielle Anforderungen für XR-Anwendungen adressiert werden können, wird dieses Architekturmodell auf Basis von (Benbelkacem et al., 2019) und (Benbelkacem et al., 2020) allerdings zusätzlich erweitert. Um Tracking und komplexere Berechnungen zu kapseln wird dem klassischen MVVM eine „Library“-Schicht hinzugefügt (Benbelkacem et al., 2019) und um die Möglichkeit eines kollaborativen Systems zu gewährleisten, dient die Trennung in Frontend und Backend (Benbelkacem et al., 2020). Damit Interaktionen zwischen realen und virtuellen Elementen möglich ist, soll die View mehrere Komponenten beinhalten, die auf unterschiedliche Nutzereingaben wie beispielsweise Gesten oder auch Controller reagieren kann. Der besseren Verständlichkeit wegen wird diese Schicht nicht nach (Benbelkacem et al., 2019) umbenannt, sondern soll weiterhin als „normale“ View verstanden werden. Eine ausführliche Erläuterung des weiterentwickelten Architekturmodells findet sich in Kapitel Konzept-Frontend. Kernaspekte der Weiterentwicklung sind also eine zusätzliche Schicht, zusätzliche Komponenten in der View und die Aufteilung in Frontend und Backend als Grundvorrausetzung für die Entwicklung des Frontends.

# Konzept

## Aufbau des Prototyps

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 über die Architektur der Anwendung die Faktoren Skalierbarkeit und Multiplayermodus allerdings schon berücksichtigt werden. Der Faktor Skalierbarkeit soll über lose gekoppelte Schichten erreicht werden, sodass einzelne Schichten auf unterschiedlichen Systemen laufen können und bei Bedarf Rechenleistung hinzugezogen werden kann. Vor allem ist damit aber auch die Möglichkeit gemeint, einfach Erweiterungen an Programm und Programmcode vornehmen zu können. Dies setzt einen strukturierten und leicht verständlichen Aufbau voraus, sodass eine Einarbeitung schnell erfolgen kann und Erweiterungen einfach implementiert werden können. Der Faktor Multiplayermodus setzt eine zentrale Datenbasis, Synchronisation zwischen Clients und konfliktfreie Zustandsverwaltung voraus. Aus diesen Gründen soll der Prototyp in drei Schichten unterteilt werden und zwei dieser drei Schichten mit bewährten Architekturkonzepten aufgebaut werden.

Die erste Schicht stellt die Datenbasis dar, sie enthält lediglich die Daten. Dabei kann auf unterschiedliche Programme zurückgegriffen werden, für die Simulation von relationalen Datenbanken reicht beispielsweise ein lokaler Dateiordner mit unterschiedlichen Tabellen. In einem späteren Schritt können diese Tabellen dann in einer richtigen Datenbank abgelegt und veröffentlicht werden. Mit der Entkopplung der Datenbasis zum restlichen Programm kann die nachstehende Datenbank und Speicherumfang frei gewählt werden, was Skalierbarkeit ermöglicht und eine Voraussetzung für den Multiplayermodus sein kann. So können mehrere Personen zur selben Zeit auf die Daten zugreifen und die Basis bleibt konsistent.

Die zweite Schicht, das Backend ermöglicht einen gesicherten Zugriff auf die relevanten Daten. So kann sichergestellt werden, dass nur berechtigte Personen und Programme auf die für sie freigegebenen Daten zugreifen und diese manipulieren können. Die Implementierung einer Authentifizierung oder rollenbasierten Autorisierung findet also im Backend statt. Zudem ermöglicht das Backend auch die Verarbeitung der Daten, was komplexe Spiellogiken umfassen kann oder das umfängliche Löschen von Daten sicherstellt. Soll ein Account gelöscht werden, kann über das Backend auch die Löschung des zugehörigen Tagebuchs und der entsprechenden Fotografien und Tagebucheinträge erfolgen. 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 dafür unterschiedliche Technologien nutzen zu können. Beispielsweise kann für das Backend eine andere Programmiersprache verwendet werden als für die letzte Schicht. Relevant hierbei ist eine Schnittstelle, die unabhängig von zum Beispiel Plattform oder Programmiersprache angesprochen werden kann. Ein solches Konzept sind standardisierte REST-APIs, welche die Kommunikation mehrerer Anwendungen, oder wie im vorliegenden Fall Schichten, ermöglichen.

Die letzte Schicht ist das Frontend. Dieses ist dafür zuständig, die bereitgestellten Daten (graphisch) darzustellen. Diese Schicht beinhaltet die sogenannte Benutzeroberfläche[[7]](#footnote-7) und somit die Schnittstelle zwischen Mensch und Anwendung. Es ist zu beachten, dass auch diese Schicht bis zu einem gewissen Maße Logik enthält und nicht nur die reine Darstellung von Inhalten beinhaltet.

Der grundlegende Aufbau des Prototyps wird in <ABB> dargestellt. Eine detailliertere Darstellung der einzelnen Schichten erfolgt in den nachfolgenden Unterkapiteln.

<Hier noch Abbildung einfügen>

### 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 vom 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 zwischen virtuellen und realen Elementen einer MR-Anwendung und nicht auf der Sicherstellung und dauerhaften Verfügbarkeit von Daten oder der Performanz von Datenbanken und Abfragen. Daher sind die genannten Aspekte für eine produktive Umgebung weiter relevant, können allerdings für die Entwicklung eines Prototypen vernachlässigt werden. Daher wird eine Datenbank mittels mehrerer lokal gespeicherten Dateien simuliert. Dieses Vorgehen beziehungsweise diese Art der Datenhaltung ist in produktiven Systemen nicht ausreichend. Allerdings ermöglicht es eine schlanke und flexible Datenhaltung ohne zusätzlichen Konfigurationsaufwand, was im Kontext der Prototypentwicklung ausreichend ist. 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. Zudem lassen sich Änderungen direkt nachvollziehen und die Dateien problemlos über gängige Werkzeuge wie Git oder GitHub gemeinsam pflegen. Die gewählte Struktur erlaubt es darüber hinaus, die CSV-Dateien bei Bedarf in spätere relationale Datenbanksysteme zu überführen, da sie einem tabellarischen Format folgen. Ein solches System kann beispielsweise ein Cloudspeicher, wie beispielsweise von Google oder Amazon (*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 ebenfalls relevant und werden vorerst ebenfalls lokal gespeichert, aber auch hier ist das spätere Speichern in zum Beispiel NoSQL-Datenbanken möglich (*Cloud-Objektspeicher – Amazon S3 – AWS*, o. J.).

### Backend

Das Backend soll nach dem Muster MVC implementiert werden. Um den Zugriff auf die Datenbank zu kapseln, wird jedoch eine weitere Schicht, das „Repository“ eingefügt. Ziel ist es, damit ein klar strukturiertes und modulares Backend zu implementieren, das so nicht nur erweiterbar und einfach wartbar ist, sondern zusätzlich noch die Skalierbarkeit ermöglicht. Das MVC-Muster trennt dabei die Verantwortlichkeiten in Datenstruktur, Logik und Interaktionsschnittstelle, mit dem Repository auch den Zugriff auf die Datenbank, beziehungsweise lokalen Dateien, die die Datenbank simulieren. 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 zum Beispiel eine klassische Datenbank oder um eine Simulation mit zum Beispiel CSV-Dateien handelt. Da die Tabellen in strukturierter Form vorliegen, kann über eine entsprechende Abfragesprache auf die Daten zugegriffen werden. Eine solche Sprache wäre 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 verwertbare Objekte zu schreiben. Damit sind Objekte gemeint, die vom aktuellen Programm, welchen beispielsweise Sprachenabhängig ist, weiterverarbeitet werden können. Mit dieser Umsetzung bleibt die eigentliche Geschäftslogik von der konkreten Speicherform entkoppelt, was spätere Technologie- oder Speicherwechsel vereinfacht.

**Model** Das **Model** beschreibt im MVC-Muster die Struktur der Datenobjekte, mit denen das Backend arbeitet. Es enthält keine Logik, sondern definiert lediglich die Eigenschaften und den Aufbau der Objekte, den sogenannten Klassen, die für die Programmlogik benötigt werden. Im Kontext des Prototyps wäre das zum Beispiel eine Klasse „User“ mit Eigenschaften wie ID, Name oder E-Mail. Das Model wird also im Zusammenspiel 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 direkt implementiert und als graphische Benutzeroberfläche umgesetzt, 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 von außen, als Kommunikationsschnittstelle mit dem Backend genutzt werden können. 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 in einer MVC-Architektur und auch im zu implementierenden Backend hat der Controller inne. 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. 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. So fungiert der Controller als Koordinationsinstanz, die Datenfluss, Verarbeitungslogik und Sicherheit kontrolliert. Im Rahmen einer möglichen Erweiterung zum Multiplayersystem wäre der Controller zudem die zentrale Stelle zur Konfliktvermeidung, zur Synchronisierung von Zuständen und zur Validierung eingehender Änderungen durch verschiedene Clients.

Die klare Trennung in Repository, Model, View und Controller ermöglicht eine **modulare Architektur.** Änderungen oder Erweiterungen lassen sich dadurch gezielt auf einzelne Schichten beschränken. Beispielsweise kann die CSV-basierte Datenbasis bei einem späteren Übergang zu einer relationalen Cloud-Datenbank durch eine Anpassung des Repositorys umgesetzt werden, ohne dass Änderungen an Controller oder Model notwendig wären. Ebenso kann eine Erweiterung um Authentifizierungslogik, Benutzerrollen oder Datenvalidierung in 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 und in Verbindung mit dem Backend erfüllt es zudem die Voraussetzung für eine Erweiterung zu MVVM-3DC nach (Benbelkacem et al., 2020). Im Folgenden soll ein detaillierterer Überblick über MVVM-3D gegeben und Implementierungsmöglichkeiten für MVVM-3DC dargelegt werden.

MVVM-3D teilt sich in die bekannten Schichten Model, View, ViewModel auf, wird allerdings um eine Library-Schicht ergänzt.

**Model** Das Model enthält Datenklassen, beschreibt also, wie Objekte aufgebaut sind und welche Eigenschaften ein Objekt hat. Beispielsweise definiert das Model die Datenklasse Tier und legt die Eigenschaften Name, Art, Lebensraum fest. Models enthalten darüber hinaus allerdings keine Logik. Da die Daten über das Backend an das Frontend weitergegeben werden und Daten von zum Beispiel Tieren oder Nutzern nicht im Frontend abliegen und die Weitergabe über eine API erfolgt, sind auch API-Abfragen im Frontend nötig. 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 damit Objekte. Vorteil davon sind zudem Wiederverwendbarkeit, ein Service kann zum Beispiel von mehreren Models genutzt werden oder aber mehrere Model-Objekte erzeugen, und vereinfachte Wartbarkeit. So 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 baut diese Objekte dann tatsächlich auf (Instanzen). Diese zusätzliche Komponente ist lediglich der Vereinfachung des Programmcodes gedacht und soll nicht explizit eine Erweiterung des Architekturmodells darstellen, da ein Einsatz des Modells in Szenarien denkbar ist, in denen keine API-Abfragen nötig sind.

**View** Die View enthält alle UI-Elemente, damit eingeschlossen sind 3D-Darstellungen oder Animationen. 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 ist damit eine Nutzeroberfläche, 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 darüber hinausgehen. So sind neben Controllereingaben (analog zur Maus, oder Berührungen) auch Eingaben mit den Händen (Klicken als Ersatz für den Controller oder Kommunikationsgesten), den Augen oder der Stimme relevant. Um für Tiere zum Beispiel einen Fluchtradius zu ermöglichen[[8]](#footnote-8), muss auch die Position einer Person beziehungsweise die Position des Geräts selbst als Nutzereingabe verstanden 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 komplexere Berechnungen kapselt, die ein ViewModel verkomplizieren würden. Beispiele für solche Anwendungsfälle sind das Tracking von Objekten, genaueres Verhalten bei Interaktion und damit beispielsweise 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. Die Library enthält also die konkrete Implementierung von komplexeren oder individuellen Logiken.

**ViewModel** Das ViewModel ist das zentrale Steuerungselement und enthält Events[[9]](#footnote-9), Observer-Pattern[[10]](#footnote-10) und auch Reactive Patterns[[11]](#footnote-11). Das ViewModel empfängt Änderungen von View oder Model und leitet sie weiter, ohne dass View oder Model selbst miteinander kommunizieren müssen. Vorteilhaft ist neben der sehr strikten Trennung von Aufgabenbereichen auch die Möglichkeit Berechnungen aus der Library hinzuzuziehen. So empfängt das ViewModel bei der Änderung einer Email-Adresse also die neue Adresse und aktualisiert automatisch das Model, wodurch wiederum eine Modeländerung dem ViewModel gemeldet wird und dadurch automatisch die View und damit die Anzeige geändert wird. Ein anderes Beispiel ist die Flucht eines Tieres, bei der seitens View dem ViewModel eine Nutzeraktion, in Form einer zu geringen Distanz zwischen Person und Tier, gemeldet werden kann. Das ViewModel setzt dann die Logik „Flucht“ um und nutzt dabei die konkrete Flucht-Implementierung des jeweiligen Tieres aus der Library-Komponente. Ist der Bewegungsablauf abgeschlossen, kann das ViewModel dem Model dann die neue Position des Tieres weiterleiten, die dann an das Backend weitergeleitet werden und dort in einer Datenbank gespeichert werden. Diese Änderung an der Datenbasis und damit am Model wird zurück an das ViewModel geleitet, wo eine Aktualisierung der Anzeige, also View, möglich wäre. Umsetzbar ist dieses Verhalten mittels Data Binding, welches UI-Elemente automatisch mit den entsprechenden Datenquellen verknüpft. Wie bereits erläutert kann sich bei Änderung an der Datenbasis damit die Anzeige automatisch aktualisieren und umgekehrt bei einer Nutzereingabe automatisch die Datenbasis ändern.[[12]](#footnote-12) Dafür müssen diese Verbindungen zur Laufzeit überwacht werden und 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, wodurch Änderungen an einem Element automatisch registriert werden. Diese Verknüpfung ermöglicht eine automatische Synchronisation von Eingabe und Variablenwert bei Änderungen (BELEG).

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

Automatisch generierte BeschreibungEine zusammenfassende Darstellung der MVVM-3D-Architektur ist in Abbildung 4 dargestellt.

Abbildung 4 - MVVM-3D (eigene Darstellung)

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

Wie im Kapitel Architekturmodell schon angemerkt gibt es für die Entwicklung von kollaborativen Systemen drei Methoden. Mit der Grundstruktur von Datenbank, Backend und Frontend ist die Methode „Centralized Mode“ gut umsetzbar. Dies meint einen zentralen Server, der alle Änderungen speichert und mehrere Clients, die alle auf denselben Datenstand zugreifen (BELEG). Für einen möglichen Prototypen fungiert das Backend als Server, der alle Anfragen und Datenoperationen verarbeitet. Mehrere Instanzen des Frontends übernehmen die Rolle von unterschiedlichen Clients und können unabhängig voneinander auf das Backend, also 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. Dies setzt voraus, dass nicht nur eine Anfrage-basierte Kommunikation (Pull-Prinzip), sondern auch eine serverseitige Benachrichtigung der Clients (Push-Prinzip) realisiert wird. Die Kommunikation über zyklische GET-Anfragen ist hierfür nicht ausreichend, da sie nicht nur ineffizient, sondern auch anfällig für Verzögerungen und Synchronisationsprobleme ist. Stattdessen wird eine Lösung benötigt, bei der das Backend aktiv Änderungen an die Clients übermittelt. Da das Backend, sowie die relevanten Teile des Frontends (Model inklusive Services und ViewModels) in C# implementiert werden, ist eine C# beziehungsweise .NET-Lösung ausreichend. Eine Möglichkeit hierfür 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 auf synchronisiert werden (Fletcher, 2023).

Entsprechend ergibt sich folgender Ablauf, sollte beispielsweise eine Person auf ein Tier zeigen, welches dann für eine zweite Person zum Beispiel mit einem Effekt oder einer Umrandung hervorgehoben werden soll. Die 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 somit 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, hervorgehoben. Dieser Ablauf ist in Abbildung 5 dargestellt.

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

Automatisch generierte Beschreibung

Abbildung 5 - 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, in dem eine Übersicht die Architektur noch einmal verdeutlicht.

## Interaktion

# Implementierung und Prototypenentwicklung

In den nachfolgenden Unterkapiteln wird die Implementierung des Prototyps dargestellt. Dabei wird insbesondere auf die Umsetzung von MVVM-3D und Interaktion eingegangen.

## Datenbank

Für eine Implementierung der Datenbank, beziehungsweise Simulation, wurden insgesamt fünf CSV-Dateien angelegt. Die Dateien wurden lokal gespeichert und über GitHub versioniert. So war ein Austausch der Daten möglich, ohne auf einen Clouddienst zurückgreifen zu müssen. <ABB> zeigt den Aufbau und Zusammenhänge der Datenbasis. Aus der Abbildung gehen die Klassen Animal, User, Diary und Diaryentry, sowie die Verbindungsklasse DiaryDiaryentry hervor.

<Hier Bild einfügen>

Die Klasse Animal speichert Informationen zu den Tieren, die virtuell auf der Brille eingeblendet werden sollen. Darunter fallen die Informationen ID, Name und Lebensraum des Tieres, sowie die Pfade zum animierten 3D-Objekt und zum realistischen Foto[[13]](#footnote-13), das für das Tagebuch relevant ist. 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 müssen, damit sie angezeigt werden können, werden statt Bilder, Animationen oder Objekten lediglich die Pfade zu den realen Ablageorten gespeichert.

Die Klasse User enthält Daten wie User-ID, Name, Username, Mail und Passwort. Diese Daten liegen als Text vor und können später ebenfalls problemlos in eine reale, relationale Datenbank überführt werden. Das Speichern von Passwort und Mail als Klartext kann für eine produktive Anwendung eine Sicherheitslücke bedeuten. Daher ist es sinnvoll in einem weiteren Schritt einen externen Dienst, wie zum Beispiel Google Firebase, einzubinden, damit die Authentifizierung darüber abgewickelt wird und sensible Daten dort gespeichert werden können.

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

Die Klasse Diaryentry kann vorerst als Verbindungsklasse zwischen Tagebuch und Tier angesehen werden, da sie aktuell lediglich die Informationen ID und ID des zugehörigen Tieres enthält. Allerdings kann ein Tagebucheintrag als abhängige Klasse der Klasse Tagebuch angesehen werden, stellt allerdings eine eigene Entität dar (ein Tagebuch besteht aus Tagebucheinträgen und nicht aus 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 Tagebuch vorgesehen und es ist auch nur eine Verbindung zwischen einem Tagebuch und einem Eintrag vorgesehen, allerdings können Einträge in mehreren Tagebüchern verwendet werden.

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 für jedes Tier ein Eintrag für jeden User angelegt werden muss. Dies ist zudem eine Möglichkeit den benötigten Speicherplatz zu reduzieren.

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 spätere Skalierbarkeit und Migration unterstützt.

## Backend

Das Backend wurde mit C# implementiert und weiter wurden in den Repositories LINQ-Ausdrücke, sowie in den Controllern ASP .NET verwendet.

Alle Repositories sind gleich aufgebaut und implementieren zu Beginn eine Verbindung zu den jeweiligen CSV-Dateien. Dafür wird die Datei geladen und jede Zeile repräsentiert die Informationen einer Instanz. Die jeweiligen Spalten repräsentieren die Attribute der Instanz. So werden verwertbare Objekte erstellt, indem die entsprechenden Informationen einer Zelle auf die Attribute eines C#-Objekts gemappt werden, die Informationen also „eingetragen“ werden. Entsprechend werden Änderungen gespeichert, indem Attributwerte eines C#-Objekts in entsprechende Zellen der CSV-Datei geschrieben werden. Welche Daten genau aus der Datenbasis geladen werden, wird über LINQ-Ausdrücke definiert. LINQ ermöglicht es, SQL-ähnliche Abfragen an eine Datenbasis zu senden und bietet dabei eine objektorientierte Sicht auf die Daten. So ist es beispielsweise möglich alle Objekte einer Tabelle zu laden, oder nur solche, die eine bestimmte Bedingung erfüllen. Mit dem Ausdruck „FirstOrDefault“ kann beispielsweise das erste Tier aus der Datenbank gelesen werden, 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 APIs** (*What Is ASP.NET Core?*, o. J.).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 **HTTP-Protokoll**, also das gleiche Protokoll, das auch dem Aufruf von Webseiten zugrunde liegt. ASP.NET Core unterstützt die Umsetzung solcher Schnittstellen in Form sogenannter **REST-APIs**, wodurch ein modularer Aufbau und eine klare Trennung zwischen Frontend und Backend ermöglicht wird. Durch die Einhaltung etablierter Webstandards ist die API unabhängig vom verwendeten Frontend nutzbar, was langfristig zur Wartbarkeit und Skalierbarkeit beiträgt. 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, die über den Endpunkt alle in der Datenbank verfügbaren Objekte einer Tabelle ausgibt. Aber auch Endpunkte mit komplexerer Logik sind implementiert. So steht für die Klasse Tier im Animal-Controller der Endpunkt „Create“ zur Verfügung, wodurch das Frontend die Erstellung eines neuen Tieres anfragen kann. Der Controller erstellt hierfür eine neue ID, indem über das Repository alle Tiere und damit alle bestehenden IDs ausgelesen werden und die nächste freie Nummer verwendet wird. Als zweites erfolgt eine Kontrolle, ob bereits ein Tier mit der vergebenen ID existiert und ist dies nicht der Fall, wird das neue Tier mit der neuen ID über das Repository (und damit, wie bereits beschrieben, in Verbindung mit dem Model) in die Datenbank geschrieben. Parallel wird allerdings auch mit der neuen 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. Bei einer Löschanfrage für ein Tier hingegen, werden zuerst alle Tagebucheinträge über das Diaryentry-Repositoy aus der Datenbank abgefragt, die über die Tier-ID mit dem Tier verbunden sind, und anschließend gelöscht. 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 6 sind alle verwendeten Models, Repositories und Controller mit zugehörigen Attributen und Methoden aufgeführt.

Abbildung 6 - 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

Für die Entwicklung des Frontend wurde C# verwendet, ergänzt um spezielle Befehle für Unity.

Die Frontend-Models spiegeln die Backend-Datenstrukturen wider und enthalten neben der Beschreibung der Eigenschaften hauptsächlich 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. Der Inspector zeigt Eigenschaften eines ausgewählten Elements an und bietet die Möglichkeit für Verknüpfung von UI-Elementen und beispielsweise Variablen.

Die Models werden für die Erstellung von Objekten verwendet und dies geschieht in den Services. Ein zentraler Service initialisiert einen HTTP-Client, der für die Kommunikation mit den verschiedenen API-Endpunkten 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. Damit diese genutzt werden können werden Objekte instanziiert und in eine serialisierbare Liste eingefügt. 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 beziehungsweise Service notwendig ist. Dabei ist die verwendete Sprache C#, ohne direkte Abhängigkeit zu Unity, was die Wiederverwendbarkeit und Testbarkeit erhöht. Das ViewModel ist beispielsweise dafür zuständig, die API-Abfrage in den Services anzustoßen. Erhält das ViewModel diese Daten, schreibt es sie in sogenannte „ObservableCollections“, damit Änderungen automatisch erkannt und UI-Elemente aktualisiert werden können. Für die Integration in Unity wird dies mit einem PropertyChanged-Mechanismus kombiniert. Auch das ViewModel ist nicht einem einzelnen Service oder einer einzelnen View zugeordnet, sondern kann von unterschiedlichen Views und Services angesprochen werden. Neben dem Laden, Aktualisierung und Löschen von Daten, ist das ViewModel aber auch für speziellere Logiken zuständig. Das UserViewModel setzt zum Beispiel automatische Bindung um, was im weiteren Verlauf dieses Kapitels noch ausführlicher betrachtet wird. Das DiaryViewModel hingegen ist für die Zusammenführung von Tagebuch, den Einträgen, den 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. In diesem speziellen Fall ist aufgrund der unterschiedlichen Implementierung für unterschiedliche Tieres und der damit einhergehenden Komplexität befindet sich die konkrete Implementierung in der Library-Schicht.

Die Library ist die neu hinzugefügte 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. Diese Brücke nimmt das jeweils entsprechende Objekt aus dem ViewModel entgegen und erstellt daraus ein unitykompatibles Objekt, sodass darauf die entsprechenden Funktionen ausgeführt werden können. Diese Brücke kann 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 deshalb getrennt in der Library.

Die View-Schicht ist für die Verknüpfung von UI-Elementen und den dahinterliegenden Funktionen zuständig und damit die Einzige, die vollständig unityspezifisch ist. 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. Was allerdings bei Drücken eines Buttons passiert, wird über den Inspector festgelegt. So kann über den Inspector der Button mit einem Event, einer Variablen oder einer Funktion der entsprechenden View verknüpft werden. Diese Funktionen können beispielsweise 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 zuvor erläuterten Basisklasse MonoBehaviour, die Lebenszyklusmethoden wie Start() bereitstellt. Damit haben sie eine Funktion Start(), die eine Verbindung zu den benötigten ViewModels herstellt. Dort wird auch auf die Daten des aktuellen Users zugegriffen, sodass 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 Bild, das Text, Diagramm, Plan, parallel enthält.

Automatisch generierte BeschreibungEin Überblick über diesen Aufbau und die dabei entstandenen Verknüpfungen zeigt Abbildung 7.

Abbildung 7 - 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 5.1.3 Frontend erläutert, sorgt es dafür, dass Änderungen an sogenannten Properties im ViewModel automatisch in der UI erscheinen und umgekehrt Benutzereingaben direkt in das ViewModel übernommen werden. Nachfolgend wird die Umsetzung von Data Binding in Unity am Beispiel des Anwendungsfalls „Passwort ändern“ erläutert, da dies elementar für die Funktion der weiterentwickelten Architektur ist.

Für den Prototypen wurde das Data Binding mit Properties und Events umgesetzt. In dieser Unity-Umgebung wird das Data Binding über das 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 kann. Für die Synchronisierung zwischen View und ViewModel verwenden die ViewModels ObsrvableCollections und implementieren das Interface INotifyBindablePropertyChanged. Über dieses Interface können Änderungen an Properties sofort als Event gemeldet und 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-Listener, sodass bei einer Eingabe der eingegebene Wert 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 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.

### Weiteres

**Animationen** Da für die virtuelle Tierbeobachtung ein annähernd 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 eigenständig, sodass längere Bewegungsabläufe mit unterschiedlichen Animationen zusammengesetzt werden müssen. 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.

Damit ein Tier in Unity erscheint, 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. Dafür wird diesem Objekt ein Animator hinzugefügt und der AnimatorController[[14]](#footnote-14) 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 implementiert. So werden einzelne Animationen aneinandergefügt, indem bestimmte Parameter (zum Beispiel bools, floats, trigger) gesetzt werden. Das ermöglicht zum Beispiel den Ablauf „RunAway“, bei dem auf ein vorwärts rennen zuerst ein nach rechts drehen und dann nach rechts rennen erfolgt. Damit einzelne Animationen aufeinanderfolgen können, müssen sogenannte Transitionen erstellt 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 so nur dann erfolgen, wenn der Parameter „isRunning“ auf zum Beispiel „true“ gesetzt wird. Die 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, die unter Umständen schreibgeschützt sind, sodass trotzt 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) 12](https://d.docs.live.net/2f16c520d8c484c7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc205560849)

[Abbildung 2 - Augmentierter Wolf (WWF, 2024; eigene Aufnahme) 12](https://d.docs.live.net/2f16c520d8c484c7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc205560850)

[Abbildung 3 - Unterscheidung MVC und MVVM (MVVM Pattern on Android | WOXAPP, o. J.) 20](https://d.docs.live.net/2f16c520d8c484c7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc205560851)

[Abbildung 4 - MVVM-3D (eigene Darstellung) 29](https://d.docs.live.net/2f16c520d8c484c7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc205560852)

[Abbildung 5 - Ablauf Kollaboration (eigene Darstellung) 31](https://d.docs.live.net/2f16c520d8c484c7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc205560853)

[Abbildung 6 - UML-Diagramm Backend (eigene Darstellung) 35](https://d.docs.live.net/2f16c520d8c484c7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc205560854)

[Abbildung 7 - UML-Diagramm Frontend (eigene Darstellung) 38](https://d.docs.live.net/2f16c520d8c484c7/Dokumente/Studium/MIT/2_Semester/Forschungsprojekt/forschungsprojekt/doku_lokal.docx#_Toc205560855)

Tabellenverzeichnis

Normen

Software Tools

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

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/

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

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

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

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/

Wong, N. (2025, April 24). *Unity 6.1: Supported Update with Enhanced Performance*. Unity. https://unity.com/blog/unity-6-1-is-now-available

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-1)
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-2)
3. Hochauflösend ist an dieser Stelle ein Werbeversprechen und muss nicht mit der subjektiven Wahrnehmung zusammenfallen. [↑](#footnote-ref-3)
4. Spiele mit hohem Budget und meist hochwertiger Grafik und aufwendigen Spielmechanismen. Meist von großen Entwicklerstudios produziert. [↑](#footnote-ref-4)
5. Eine Möglichkeit sind beispielsweise virtuelle Maschinen oder Container, diese Technologien allerdings auszuführen ginge für diese Arbeit zu weit, da die derzeitige Rechenleistung für einen Prototypen ausreicht. [↑](#footnote-ref-5)
6. 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)
7. 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)
8. Gemeint ist damit, dass Tiere bei der Unterschreitung einer gewissen Distanz vor einer Person weglaufen [↑](#footnote-ref-8)
9. Event: Änderungssignal [↑](#footnote-ref-9)
10. Observer-Pattern: Objekt „lauscht“ auf Änderungen eines anderen Objekt, erfährt Änderungen sofort [↑](#footnote-ref-10)
11. Reactive-Pattern: Systematische Beobachtung von Datenflüssen und Reaktion darauf [↑](#footnote-ref-11)
12. 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-12)
13. 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-13)
14. 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 Architetkurmodell. [↑](#footnote-ref-14)